Sybase® Adaptive Server™ Enterprise
Reference Manual
Volume 2: Procedures

Document ID: 32402-01-1150

September 1997

Copyright Information

Sybase, Inc., 6475 Christie Avenue, Emeryville, CA 94608.

Table of Contents

Chapters 1 and 2 are in Volume 1 of the *Adaptive Server Reference Manual*.

About This Book

3. System Procedures

4. Catalog Stored Procedures

5. System Extended Stored Procedures

6. *dbcc* Stored Procedures

List of Figures

List of Tables

About This Book

The *Adaptive Server Reference Manual* is a three-volume guide to Sybase® Adaptive Server™ Enterprise and the Transact-SQL® language. This volume includes information about system procedures, catalog stored procedures, system extended stored procedures, and **dbcc** stored procedures. Volume 1, *Commands and Functions*, contains information about Transact-SQL datatypes, commands, and built-in functions. Volume 3, *Datatypes and System Tables*, contains information about datatypes, the system tables, expressions and identifiers, SQLSTATE errors, reserved words, and the index for all three volumes.

For information about the intended audience of this book, related documents, other sources of information, conventions used in this manual, and help, see "About This Book" in Volume 1.

How to Use This Book

This manual consists of the following chapters:

- Chapter 3, "System Procedures," contains reference pages for Adaptive Server system procedures.
- Chapter 4, "Catalog Stored Procedures," contains reference pages for Adaptive Server catalog stored procedures.
- Chapter 5, "System Extended Stored Procedures," contains reference pages for Adaptive Server system extended stored procedures.
- Chapter 6, "dbcc Stored Procedures," contains reference pages for Adaptive Server **dbcc** stored procedures.

System Procedures

3 System Procedures

This chapter describes the system procedures, which are Sybase-supplied stored procedures used for updating and getting reports from system tables. Table 3-1 lists the system procedures discussed in this chapter.

Table 3-1: System procedures

Procedure	Description
sp_activeroles	Displays all active roles granted to a user's login.
sp_addalias	Allows an Adaptive Server user to be known in a database as another user.
sp_addauditrecord	Allows users to enter user-defined audit records (comments) into the audit trail.
sp_addaudittable	Adds another system audit table after auditing is installed.
sp_addengine	Adds an engine to an existing engine group or, if the group does not exist, creates an engine group and adds the engine.
sp_addexeclass	Creates or updates a user-defined execution class that you can bind to client applications, logins, and stored procedures.
sp_addextendedproc	Creates an extended stored procedure (ESP) in the *master* database.
sp_addexternlogin	(Component Integration Services only) Assigns an alternate login name and password to be used when communicating with a remote server.
sp_addgroup	Adds a group to a database. Groups are used as collective names in granting and revoking privileges.
sp_addlanguage	Defines the names of the months and days, and the date format, for an alternate language.
sp_addlogin	Adds a new user account to Adaptive Server.
sp_addmessage	Adds user-defined messages to *sysusermessages* for use by stored procedure **print** and **raiserror** calls and by **sp_bindmsg**.
sp_addobjectdef	(Component Integration Services only) Specifies the mapping between a local table and an external storage location.
sp_addremotelogin	Authorizes a new remote server user by adding an entry to *master.dbo.sysremotelogins*.
sp_add_resource_limit	Creates a limit on the amount of server resources that a login or application can use to execute a query, query batch, or transaction.

Table 3-1: System procedures (continued)

Procedure	Description
sp_addsegment	Defines a segment on a database device in the current database.
sp_addserver	Defines a remote server or defines the name of the local server.
sp_addthreshold	Creates a threshold to monitor space on a database segment. When free space on the segment falls below the specified level, Adaptive Server executes the associated stored procedure.
sp_add_time_range	Adds a named time range to Adaptive Server.
sp_addtype	Creates a user-defined datatype.
sp_addumpdevice	Adds a dump device to Adaptive Server.
sp_adduser	Adds a new user to the current database.
sp_altermessage	Enables and disables the logging of a specific system-defined or user-defined message in the Adaptive Server error log.
sp_audit	Allows a System Security Officer to configure auditing options.
sp_autoconnect	(Component Integration Services only) Defines a passthrough connection to a remote server for a specific user, which allows the named user to enter passthrough mode automatically at login.
sp_bindcache	Binds a database, table, index, *text* object, or *image* object to a data cache.
sp_bindefault	Binds a user-defined default to a column or user-defined datatype.
sp_bindexeclass	Associates an execution class with a client application, login, or stored procedure.
sp_bindmsg	Binds a user message to a referential integrity constraint or check constraint.
sp_bindrule	Binds a rule to a column or user-defined datatype.
sp_cacheconfig	Creates, configures, reconfigures, drops, and provides information about data caches.
sp_cachestrategy	Enables or disables prefetching (large I/O) and MRU cache replacement strategy for a table, index, *text* object, or *image* object.
sp_changedbowner	Changes the owner of a database.
sp_changegroup	Changes a user's group.
sp_checknames	Checks the current database for names that contain characters not in the 7-bit ASCII set.

Table 3-1: System procedures (continued)

Procedure	Description
sp_checkreswords	Detects and displays identifiers that are Transact-SQL reserved words. Checks server names, device names, database names, segment names, user-defined datatypes, object names, column names, user names, login names, and remote login names.
sp_chgattribute	Changes the **max_rows_per_page** value for future space allocations of a table or index.
sp_clearpsexe	Clears the execution attributes of the client application, login, or stored procedure that was set by **sp_setpsexe**.
sp_clearstats	Initiates a new accounting period for all server users or for a specified user. Prints statistics for the previous period by executing **sp_reportstats**.
sp_commonkey	Defines a common key—columns that are frequently joined—between two tables or views.
sp_configure	Displays or changes configuration parameters.
sp_countmetadata	Displays the number of user indexes, objects, or databases in a server.
sp_cursorinfo	Reports information about a specific cursor or all cursors that are active for your session.
sp_dboption	Displays or changes database options.
sp_dbremap	Forces Adaptive Server to recognize changes made by **alter database**. Run this procedure only when instructed to do so by an Adaptive Server message.
sp_defaultloc	(Component Integration Services only) Defines a default storage location for objects in a local database.
sp_depends	Displays information about database object dependencies—the view(s), trigger(s), and procedure(s) that depend on a specified table or view, and the table(s) and view(s) that the specified view, trigger, or procedure depends on.
sp_diskdefault	Specifies whether or not a database device can be used for database storage if the user does not specify a database device or specifies **default** with the **create database** or **alter database** commands.
sp_displaylevel	Sets or shows which Adaptive Server configuration parameters appear in **sp_configure** output.
sp_displaylogin	Displays information about a login account.
sp_displayroles	Displays all roles granted to another role, or displays the entire hierarchy tree of roles in table format.

Table 3-1: System procedures (continued)

Procedure	Description
sp_dropalias	Removes the alias user name identity established with **sp_addalias**.
sp_dropdevice	Drops an Adaptive Server database device or dump device.
sp_dropengine	Drops an engine from a specified engine group or, if the engine is the last one in the group, drops the engine group.
sp_dropexeclass	Drops a user-defined execution class.
sp_dropextendedproc	Removes an ESP from the master database.
sp_dropexternlogin	(Component Integration Services only) Drops the definition of a remote login previously defined to the local server by **sp_addexternlogin**.
sp_dropglockpromote	Removes lock promotion values from a table or database.
sp_dropgroup	Drops a group from a database.
sp_dropkey	Removes a key defined with **sp_primarykey**, **sp_foreignkey**, or **sp_commonkey** from the *syskeys* table.
sp_droplanguage	Drops an alternate language from the server and removes its row from *master.dbo.syslanguages*.
sp_droplogin	Drops an Adaptive Server user login by deleting the user's entry in *master.dbo.syslogins*.
sp_dropmessage	Drops user-defined messages from *sysusermessages*.
sp_dropobjectdef	(Component Integration Services only) Deletes the external storage mapping provided for a local object.
sp_dropremotelogin	Drops a remote user login.
sp_drop_resource_limit	Removes one or more resource limits from Adaptive Server.
sp_dropsegment	Drops a segment from a database or unmaps a segment from a particular database device.
sp_dropserver	Drops a server from the list of known servers.
sp_dropthreshold	Removes a free-space threshold from a segment.
sp_drop_time_range	Removes a user-defined time range from Adaptive Server.
sp_droptype	Drops a user-defined datatype.
sp_dropuser	Drops a user from the current database.
sp_estspace	Estimates the amount of space required for a table and its indexes, and the time needed to create the index.
sp_extendsegment	Extends the range of a segment to another database device.

Table 3-1: System procedures (continued)

Procedure	Description
sp_familylock	Reports information about all the locks held by a family (coordinating process and its worker processes) executing a statement in parallel.
sp_forceonline_db	Provides access to all the pages in a database that were previously taken offline by recovery.
sp_forceonline_page	Provides access to pages previously taken offline by recovery.
sp_foreignkey	Defines a foreign key on a table or view in the current database.
sp_freedll	Unloads a dynamic link library (DLL) that was previously loaded into XP Server memory to support the execution of an ESP.
sp_getmessage	Retrieves stored message strings from *sysmessages* and *sysusermessages* for **print** and **raiserror** statements.
sp_grantlogin	(Windows NT only) When Integrated Security mode or Mixed mode (with Named Pipes) is active, assigns Adaptive Server roles or **default** permissions to Windows NT users and groups.
sp_help	Reports information about a database object (any object listed in *sysobjects*) and about Adaptive Server-supplied or user-defined datatypes.
sp_helpartition	Lists the first page and the control page for each partition in a partitioned table.
sp_helpcache	Displays information about the objects that are bound to a data cache or the amount of overhead required for a specified cache size.
sp_helpconfig	Reports help information about configuration parameters, such as how much memory is needed if the parameter is set to a certain value. **sp_helpconfig** displays the current setting, the amount of memory used for that setting, the default value, and the minimum and maximum settings.
sp_helpconstraint	Reports information about any integrity constraints specified for a table. This information includes the constraint name and the definition of the bound default, unique or primary key constraint, referential constraint, or check constraint. **sp_helpconfig** also displays the number of references used by the specified tables.
sp_helpdb	Reports information about a particular database or about all databases.
sp_helpdevice	Reports information about a particular device or about all Adaptive Server database devices and dump devices.

Table 3-1: System procedures (continued)

Procedure	Description
sp_helpextendedproc	Displays ESPs registered in the current database, along with their associated DLL files.
sp_helpexternlogin	(Component Integration Services only) Reports information about external login names.
sp_helpgroup	Reports information about a particular group or about all groups in the current database.
sp_helpindex	Reports information about the indexes created on a table.
sp_helpjoins	Lists the columns in two tables or views that are likely join candidates.
sp_helpkey	Reports information about a primary, foreign, or common key of a particular table or view, or about all keys in the current database.
sp_helplanguage	Reports information about a particular alternate language or about all languages.
sp_helplog	Reports the name of the device that contains the first page of the transaction log.
sp_helpobjectdef	(Component Integration Services only) Reports information about remote object definitions. Shows owners, objects, type, and definition.
sp_helpremotelogin	Reports information about a particular remote server's logins or about all remote servers' logins.
sp_help_resource_limit	Reports information about all resource limits, limits for a given login or application, limits in effect at a given time or day of the week, or limits with a given scope or action.
sp_helprotect	Reports information about permissions for database objects, users, groups, or roles.
sp_helpsegment	Reports information about a particular segment or about all segments in the current database.
sp_helpserver	Reports information about a particular remote server or about all remote servers.
sp_helpsort	Displays Adaptive Server's default sort order and character set.
sp_helptext	Prints the text of a system procedure, trigger, view, default, rule, or integrity check constraint.
sp_helpthreshold	Reports the segment, free-space value, status, and stored procedure associated with all thresholds in the current database or all thresholds for a particular segment.
sp_helpuser	Reports information about a particular user or about all users in the current database.

Table 3-1: System procedures (continued)

Procedure	Description
sp_indsuspect	Checks user tables for indexes marked as suspect during recovery following a sort order change.
sp_listsuspect_db	Lists all databases that have offline pages because of corruption detected on recovery.
sp_listsuspect_page	Lists all pages that are currently offline because of corruption detected on recovery.
sp_lock	Reports information about processes that currently hold locks.
sp_locklogin	Locks an Adaptive Server account so that the user cannot log in, or displays a list of all locked accounts.
sp_logdevice	Moves the transaction log of a database with log and data on the same device to a separate database device.
sp_loginconfig	(Windows NT only) Displays the value of one or all integrated security parameters.
sp_logininfo	(Windows NT only) Displays all roles granted to Windows NT users and groups with **sp_grantlogin**.
sp_logiosize	Changes the log I/O size used by Adaptive Server to a different memory pool when it is doing I/O for the transaction log of the current database.
sp_modifylogin	Modifies the default database, default language, default role activation, or full name for an Adaptive Server login account.
sp_modify_resource_limit	Changes a resource limit by specifying a new limit value or the action to take when the limit is exceeded, or both.
sp_modifythreshold	Modifies a threshold by associating it with a different threshold procedure, free-space level, or segment name. You **cannot** use **sp_modifythreshold** to change the amount of free space or the segment name for the last-chance threshold.
sp_modify_time_range	Changes the start day, start time, end day, and/or end time associated with a named time range.
sp_monitor	Displays statistics about Adaptive Server.
sp_monitorconfig	Displays cache usage statistics regarding metadata descriptors for indexes, objects, and databases, such as the number of metadata descriptors currently in use by Adaptive Server. Also reports the number of auxiliary scan descriptors in use.
sp_passthru	(Component Integration Services only) Allows the user to pass a SQL command buffer to a remote server.
sp_password	Adds or changes a password for an Adaptive Server login account.

Table 3-1: System procedures (continued)

Procedure	Description
sp_placeobject	Puts future space allocations for a table or an index on a particular segment.
sp_poolconfig	Creates, drops, resizes, and provides information about memory pools within data caches.
sp_primarykey	Defines a primary key on a table or view.
sp_processmail	(Windows NT only) Reads, processes, sends, and deletes messages in the Adaptive Server message inbox.
sp_procqmode	Displays the query processing mode of a stored procedure, view, or trigger.
sp_procxmode	Displays or changes the transaction modes associated with stored procedures.
sp_recompile	Causes each stored procedure and trigger that uses the named table to be recompiled the next time it runs.
sp_remap	Remaps a stored procedure, trigger, rule, default, or view from releases later than 4.8 and earlier than 10.0 to be compatible with releases 10.0 and later. Use **sp_remap** on pre-release 11.0 objects that the release 11.0 upgrade procedure failed to remap.
sp_remoteoption	Displays or changes remote login options.
sp_remotesql	(Component Integration Services only) Establishes a connection to a remote server, passes a query buffer to the remote server from the client, and relays the results back to the client.
sp_rename	Changes the name of a user-created object or user-defined datatype in the current database.
sp_renamedb	Changes the name of a database. You **cannot** rename system databases or databases with external referential integrity constraints.
sp_reportstats	Reports statistics on system usage.
sp_revokelogin	(Windows NT only) When Integrated Security mode or Mixed mode (with Named Pipes) is active, revokes Adaptive Server roles and default permissions from Windows NT users and groups.
sp_role	Grants or revokes system roles to an Adaptive Server login account.
sp_serveroption	Displays or changes remote server options.
sp_setlangalias	Assigns or changes the alias for an alternate language.
sp_setpglockpromote	Sets or changes the lock promotion thresholds for a database, for a table, or for Adaptive Server.

Table 3-1: System procedures (continued)

Procedure	Description
sp_setpsexe	Sets custom execution attributes "on the fly" for an active client application, login, or stored procedure.
sp_setsuspect_granularity	Displays and sets the recovery fault isolation mode.
sp_setsuspect_threshold	On recovery, sets the maximum number of suspect pages that Adaptive Server will allow in the specified database before taking the entire database offline.
sp_showcontrolinfo	Displays information about engine group assignments, bound client applications, logins, and stored procedures.
sp_showexeclass	Displays the execution class attributes and the engines in any engine group associated with the specified execution class.
sp_showplan	Displays the query plan for any user connection for the current SQL statement (or a previous statement in the same batch). The query plan is displayed in **showplan** format.
sp_showpsexe	Displays execution class, current priority, and affinity for all processes running on Adaptive Server.
sp_spaceused	Displays estimates of the number of rows, the number of data pages, and the space used by one table or by all tables in the current database.
sp_syntax	Displays the syntax of Transact-SQL statements, system procedures, utilities, and other routines, depending on which products and corresponding **sp_syntax** scripts exist on Adaptive Server.
sp_sysmon	Displays performance information.
sp_thresholdaction	Executes automatically when the number of free pages on the log segment falls below the last-chance threshold, unless the threshold is associated with a different procedure. Sybase does not provide this procedure.
sp_unbindcache	Unbinds a database, table, index, *text* object, or *image* object from a data cache.
sp_unbindcache_all	Unbinds all objects that are bound to a cache.
sp_unbindefault	Unbinds a created default value from a column or from a user-defined datatype.
sp_unbindexeclass	Unbinds a database, table, index, *text* object, or *image* object from a data cache.
sp_unbindmsg	Unbinds a user-defined message from a constraint.
sp_unbindrule	Unbinds a rule from a column or from a user-defined datatype.
sp_volchanged	Notifies the Backup Server™ that the operator performed the requested volume handling during a dump or load.

Table 3-1: System procedures (continued)

Procedure	Description
sp_who	Reports information about all current Adaptive Server users and processes or about a particular user or process.

Introduction to System Procedures

The system procedures, created by **installmaster** at installation, are located in the *sybsystemprocs* database and are owned by the System Administrator, but many of them can be run in any database.

You can create your own system procedures that can be executed from any database. (See Chapter 1, "Overview of System Administration," in the *System Administration Guide* for more information.)

All system procedures execute at isolation level 1.

All system procedures report a return status. For example:

```
return status = 0
```

means that the procedure executed successfully. The examples in this book do not include the return status.

Permissions on System Procedures

Since system procedures are located in the *sybsystemprocs* database, their permissions are also set there.

Some system procedures can be run only by Database Owners. These procedures make sure that the user executing the procedure is the owner of the database from which they are being executed.

Other system procedures (for example, all the **sp_help** procedures) can be executed by any user who has been granted permission, but this permission must be granted in *sybsystemprocs*. In other words, a user must have permission to execute a system procedure either in all databases or in none of them.

A user who is not listed in *sybsystemprocs..sysusers* is treated as a"guest" user in *sybsystemprocs* and is automatically granted permission on many of the system procedures. To deny a user permission on a system procedure, the System Administrator must add the user to *sybsystemprocs..sysusers* and write a **revoke** statement that applies to that procedure. The owner of a user database cannot

directly control permissions on the system procedures within his or her own database.

Executing System Procedures

If a system procedure is executed in a database other than *sybsystemprocs*, it operates on the system tables in the database in which it was executed. For example, if the Database Owner of *pubs2* runs **sp_adduser** in *pubs2*, the new user is added to *pubs2..sysusers*.

To run a system procedure in a specific database, either open that database with the **use** command and execute the procedure, or qualify the procedure name with the database name.

For example, the user-defined system procedure *sp_foo*, which executes the **db_name()** system function, returns the name of the database in which it is executed. When executed in the *pubs2* database, it returns the value "pubs2":

```
exec pubs2..sp_foo

------------------------------
pubs2

(1 row affected, return status = 0)
```

When executed in *sybsystemprocs*, it returns the value "sybsystemprocs":

```
exec sybsystemprocs..sp_foo

------------------------------
sybsystemprocs

(1 row affected, return status = 0)
```

Values for Parameters

If a parameter value for a system procedure contains punctuation or embedded blanks, or is a reserved word, you must enclose it in single or double quotes. If the parameter is an object name qualified by a database name or owner name, enclose the entire name in single or double quotes.

➤ ***Note***

Do not use delimited identifiers as system procedure parameters; they may produce unexpected results.

If a procedure has multiple optional parameters, you can supply parameters in the form:

```
@parametername = value
```

instead of supplying all the parameters. The parameter names in the syntax statements match the parameter names defined by the procedures.

For example, the syntax for **sp_addlogin** is:

```
sp_addlogin login_name, password [, defdb
    [, deflanguage [, fullname]]]
```

To use **sp_addlogin** to create a login for "susan" with a password of "wonderful", a full name of Susan B. Anthony, and the server's default database and language, you can use:

```
sp_addlogin susan, wonderful,
        @fullname="Susan B. Anthony"
```

This provides the same information as the command with all the parameters specified:

```
sp_addlogin susan, wonderful, public_db,
    us_english, "Susan B. Anthony"
```

You can also use "null" as a placeholder:

```
sp_addlogin susan, wonderful, null, null,
        "Susan B. Anthony"
```

Do not enclose "null" in quotes.

Remember that 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. If you issue a system procedure followed by a command, Adaptive Server attempts to execute the system procedure and then the command. For example, if you execute the following command:

```
sp_help checkpoint
```

Adaptive Server returns the output from **sp_help** and runs the **checkpoint** command.

If you specify more parameters than the number of parameters expected by the system procedure, the extra parameters are ignored by Adaptive Server.

System Procedure Messages

System procedures return informational and error messages, which are listed with each procedure in this book. System procedure error messages start at error number 17000.

Error messages from the functions and commands included in a procedure are documented in *Error Messages*.

System Procedure Tables

Several **system procedure tables** in the *master* database, such as *spt_values*, *spt_committab*, *spt_monitor*, and *spt_limit_types*, are used by system procedures to convert internal system values (for example, status bits) into human-readable format.

spt_values is never updated. To see how it is used, execute **sp_helptext** to look at the text for one of the system procedures that references it.

In addition, some system procedures create and then drop temporary tables.

sp_activeroles

Function

Displays all active roles.

Syntax

```
sp_activeroles [expand_down]
```

Parameters

expand_down – shows the hierarchy tree of all active roles contained by your roles.

Examples

1. **sp_activeroles**

```
Role Name
------------------------------
sa_role
sso_role
oper_role
replication_role
```

2. **sp_activeroles expand_down**

```
Role Name   Parent Role Name               Level
------------------------------ --------------------
sa_role     NULL                             1
doctor_role NULL                             1
oper_role   NULL                             1
```

Comments

- **sp_activeroles** displays all your active roles and all roles contained by those roles.
- For more information on roles, see Chapter 4, "Administering Roles," in the *Security Administration Guide.*

Messages

```
An invalid argument was entered. Usage:
sp_activeroles [expand_down]
```

Permissions

Any user can use **sp_activeroles**.

Tables Used

master.dbo.sysattributes, master.dbo.syssrvroles, master.dbo.sysloginroles

See Also

Commands	**alter role**, **create role**, **drop role**, **grant**, **revoke**, **set**
Functions	**mut_excl_roles**, **proc_role**, **role_contain**, **role_name**
System procedures	**sp_displayroles**

sp_addalias

Function

Allows an Adaptive Server user to be known in a database as another user.

Syntax

```
sp_addalias loginame, name_in_db
```

Parameters

loginame – is the *master.dbo.syslogins* name of the user who wants an alternate identity in the current database.

name_in_db – is the database user name to alias *loginame* to. The name must exist in both *master.dbo.syslogins* and in the *sysusers* table of the current database.

Examples

```
1. sp_addalias victoria, albert
```

There is a user named "albert" in the database's *sysusers* table and a login for a user named "victoria" in *master.dbo.syslogins*. This command allows "victoria" to use the current database by assuming the name "albert".

Comments

- Executing **sp_addalias** maps one user to another in the current database. The mapping is shown in *sysalternates*, where the two users' *suids* (system user IDs) are connected.
- A user can be aliased to only one database user at a time.
- A report on any users mapped to a specified user can be generated with **sp_helpuser**, giving the specified user's name as an argument.
- When a user tries to use a database, Adaptive Server checks *sysusers* to see if the user is listed there. If the user is not listed there, Adaptive Server then checks *sysalternates*. If the user's *suid* is listed in *sysalternates*, mapped to a database user's *suid*, Adaptive Server treats the first user as the second user while using the database.

  If the user named in *loginame* is in the database's *sysusers* table, Adaptive Server does not use the user's alias identity, since it

checks *sysusers* and finds the *loginame* before checking *sysalternates*, where the alias is listed.

Messages

- `Alias user added.`

  The procedure was successful. Now *loginame* can use the current database. While doing so, the user is known as *name_in_db*.

- `'loginame' is already a user in the current database.`

  A user with a login in the current database cannot be aliased to another login in that database.

- `No login with the specified name exists.`

  There is no entry in *master.dbo.syslogins* for *loginame*. Everyone using Adaptive Server, whether aliased or not, must have a login.

- `No user with the specified name exists in the current database.`

  Since *name_in_db* is not a user in the database, *loginame* cannot be aliased to it.

- `The specified user name is already aliased.`

  The *loginame* is already aliased to a user in the current database. A *loginame* can be aliased to only one database user at a time. To change an alias, first drop the current alias using **sp_dropalias**, and then add the new alias.

Permissions

Only the Database Owner or a System Administrator can execute **sp_addalias**.

Tables Used

master.dbo.syslogins, *sysalternates*, *sysobjects*, *sysusers*

See Also

Commands	**use**
System procedures	**sp_addlogin**, **sp_adduser**, **sp_dropalias**, **sp_helpuser**

sp_addauditrecord

Function

Allows users to enter user-defined audit records (comments) into the audit trail.

Syntax

```
sp_addauditrecord [text [, db_name [, obj_name
   [, owner_name [, dbid [, objid]]]]]]
```

Parameters

text – is the text of the message to add to the current audit table. The text is inserted into the *extrainfo* field of the table.

db_name – is the name of the database referred to in the record. The name is inserted into the *dbname* field of the current audit table.

obj_name – is the name of the object referred to in the record. The name is inserted into the *objname* field of the current audit table.

owner_name – is the owner of the object referred to in the record. The name is inserted into the *objowner* field of the current audit table.

dbid – is the database ID number of *db_name*. Do not enclose this integer value in quotes. *dbid* is inserted into the *dbid* field of the current audit table.

objid – is the object ID number of *obj_name*. Do not enclose this integer value in quotes. *objid* is inserted into the *objid* field of the current audit table.

Examples

1. **`sp_addauditrecord "I gave A. Smith permission to view the payroll table in the corporate database. This permission was in effect from 3:10 to 3:30 pm on 9/22/92.", "corporate", "payroll", "dbo", 10, 1004738270`**

   Adds “I gave A. Smith permission to view the payroll table in the corporate database. This permission was in effect from 3:10 to 3:30 pm on 9/22/92.” to the *extrainfo* field; “corporate” to the *dbname* field; “payroll” to the *objname* field; “dbo” to the *objowner* field; “10” to the *dbid* field, and “1004738270” to the *objid* field of the current audit table.

2. **sp_addauditrecord @text="I am disabling auditing briefly while we reconfigure the system", @db_name="corporate"**

   Adds this record to the audit trail. This example uses parameter names with the @ prefix, which allows you to leave some fields empty.

Comments

- Adaptive Server writes all audit records to the current audit table. The current audit table is determined by the value of the **current audit table** configuration parameter, set with **sp_configure**. An installation can have up to eight system audit tables, named *sysaudits_01*, *sysaudits_02*, and so forth, through *sysaudits_08*.

➤ ***Note***

The records actually are first stored in the in-memory audit queue, and the audit process later writes the records from the audit queue to the current audit table. Therefore, you cannot count on an audit record being stored immediately in the audit table.

- You can use **sp_addauditrecord** if:
  - You have been granted execute permission on **sp_addauditrecord**. (No special role is required.)
  - Auditing is enabled. (A System Security Officer used **sp_configure** to turn on the **auditing** configuration parameter.)
  - The **adhoc** option of **sp_audit** is set to **on**.

Messages

None.

Permissions

By default, only the System Security Officer can execute **sp_addauditrecord**. The Database Owner of *sybsecurity* (who must also be a System Security Officer) can grant execute permission to other users.

Tables Used

sybsecurity.dbo.sysaudits_01...sysaudits_08

See Also

System procedures	**sp_audit**

sp_addaudittable

Function

Adds another system audit table after auditing is installed.

Syntax

sp_addaudittable *devname*

Parameters

devname – is the name of the device for the audit table. Specify a device name or specify "default". If you specify "default", Adaptive Server creates the audit table on the same device as the *sybsecurity* database. Otherwise, Adaptive Server creates the table on the device you specify.

Examples

1. **sp_addaudittable auditdev2**

   Creates a system audit table on *auditdev2*. If only one system audit table (*sysaudits_01*) exists when you execute the procedure, Adaptive Server names the new audit table *sysaudits_02* and places it on its own segment, called *aud_seg_02*, on *auditdev2*.

2. **sp_addaudittable default**

   Creates a system audit table on the same device as the *sybsecurity* database. If two system audit tables (*sysaudits_01* and *sysaudits_02*) exist when you execute the procedure, Adaptive Server names the new audit table *sysaudits_03* and places it on its own segment, called *aud_seg_03*, on the same device as the *sybsecurity* database.

Comments

- Auditing must already be installed when you run **sp_addaudittable**. Follow this procedure to add a system audit table:

  a. Create the device for the audit table, using **disk init**. For example, run a command like this for UNIX:

  ```
  disk init name = "auditdev2",
  physname = "/dev/rxyla",
  vdevno = 2, size = 5120
  ```

b. Add the device to the *sybsecurity* database with the **alter database** command. For example, to add *auditdev2* to the *sybsecurity* database, use this command:

```
alter database sybsecurity on auditdev2
```

c. Execute **sp_addaudittable** to create the table.

- Adaptive Server names the new system audit table and the new segment according to how many audit tables are already defined. For example, if five audit tables are defined before you execute the procedure, Adaptive Server names the new audit table *sysaudits_06* and the new segment *aud_seg_06*. If you specify "default", Adaptive Server places the segment on the same device as the *sybsecurity* database. Otherwise, Adaptive Server places the segment on the device you name.
- A maximum of eight audit tables is allowed. If you already have eight audit tables, and you attempt to execute **sp_addaudittable** to add another one, Adaptive Server displays an error message.
- For information about how to install auditing initially, see the installation documentation for your platform. For a discussion about how to use auditing, see the *Security Administration Guide.*

Messages

- `Can't run sp_addaudittable from within a transaction.`

  sp_addaudittable modifies system tables, so it cannot be run within a transaction.

- `Creating segment` *`segname`* `on` *`devname`* `device.`

  sp_addaudittable created the device for the audit table.

- `Creating table sysaudits_0`*`N`* `on` *`segname`* `segment.`

  sp_addaudittable created the audit table.

- `Syntax: sp_addaudittable <device_name> | 'default'`

  You must specify a device name.

- `You cannot create a total of more than 8 audit tables.`

  Eight audit tables already exist.

- `You must be in the sybsecurity database to run this procedure.`

  Issue the **use** command to change to the *sybsecurity* database.

Permissions

You must be both a System Administrator and a System Security Officer to execute **sp_addaudittable.**

Tables Used

sybsecurity..sysobjects

See Also

System procedures	**sp_audit**

sp_addengine

Function

Adds an engine to an existing engine group or, if the group does not exist, creates an engine group and adds the engine.

Syntax

sp_addengine *engine_number, engine_group*

Parameters

engine_number – is the number of the engine you are adding to the group. Legal values are between 0 and a maximum equal to the number of configured online engines minus one.

engine_group – is the name of the engine group to which you are adding the engine. If *engine_group* does not exist, Adaptive Server creates it and adds the engine to it. Engine group names must conform to the rules for identifiers (see "Identifiers" in Appendix A, "Expressions, Identifiers, and Wildcard Characters").

Examples

1. sp_addengine 2, DS_GROUP

If no engine group is called *DS_GROUP*, this statement establishes the group. If *DS_GROUP* already exists, this statement adds engine number 2 to that group.

Comments

- **sp_addengine** creates a new engine group if the value of *engine_group* does not already exist.
- The engine groups *ANYENGINE* and *LASTONLINE* are predefined. *ANYENGINE* includes all existing engines. *LASTONLINE* specifies the engine with highest engine number. A System Administrator can create additional engine groups. You cannot modify predefined engine groups.
- As soon as you use **sp_bindexeclass** to bind applications or logins to an execution class associated with *engine_group*, the associated process may start running on *engine_number*.
- Prior to making engine affinity assignments, study the environment and consider the number of non-preferred applications and the number of Adaptive Server engines

available. For more information about non-preferred applications, see "Assigning Execution Classes" in Chapter 22, "Distributing Engine Resources Between Tasks" of the *Performance and Tuning Guide*.

Messages

- `Cannot modify pre-defined engine groups.`

  You cannot add engines to or remove engines from the predefined engine groups *ANYENGINE* and *LASTONLINE*.

- `Can't run sp_addengine from within a transaction.`

  sp_addengine modifies system tables, so it cannot be run within a transaction.

- `Failed to add engine` *`engine_number`* `to engine group` *`engine_group`*`.`

  Your attempt to modify the engine group resulted in an error. Please check the error log.

- `Name` *`engine_group`* `is not valid.`

  The name you used is not a valid identifier. You used more than 30 characters or illegal characters.

- `Validation of engine group modification failed.`

  The parameter combination you used is invalid. Check the error log.

Permissions

Only a System Administrator can execute **sp_addengine**.

Tables Used

master..sysattributes

See Also

System procedures	**sp_addexeclass, sp_bindexeclass, sp_clearpsexe, sp_dropengine, sp_setpsexe, sp_showcontrolinfo, sp_showexeclass, sp_showpsexe, sp_unbindexeclass**

sp_addexeclass

Function

Creates or updates a user-defined execution class that you can bind to client applications, logins, and stored procedures.

Syntax

```
sp_addexeclass classname, priority, timeslice,
   engine_group
```

Parameters

classname – is the name of the new execution class.

priority – is the priority value with which to run the client application, login, or stored procedure after it is associated with this execution class. Legal values are **HIGH**, **LOW**, and **MEDIUM**.

timeslice – is the time unit assigned to processes associated with this class. Adaptive Server currently ignores this parameter.

engine_group – identifies an existing group of engines on which processes associated with this class can run.

Examples

```
1. sp_addexeclass "DS", "LOW", 0, "DS_GROUP"
```

This statement defines a new execution class called *DS* with a *priority* value of **LOW** and associates it with the engine group *DS_GROUP*.

Comments

- **sp_addexeclass** creates or updates a user-defined execution class that you can bind to client applications, logins, and stored procedures. If the class already exists, the class attribute values are updated with the values supplied by the user.
- Use the predefined engine group parameter *ANYENGINE* if you do not want to restrict the execution object to an engine group.
- Use **sp_addengine** to define engine groups. Use **sp_showexeclass** to display execution class attributes and the engines in any engine group associated with the specified execution class. **sp_showcontrolinfo** lists the existing engine groups.

Messages

- `Can't run sp_addexeclass from within a transaction.`

  sp_addexeclass modifies system tables, so it cannot be run within a transaction.

- `Execution Class classname is a system defined class. This class cannot be created or changed.`

  You cannot create or modify a system-defined execution class.

- `Failed to update attributes of objects assigned to class classname. Check server errorlog for any additional information.`

  The parameter combination you used is invalid. Check the error log.

- `Name classname is not valid.`

  classname does not conform to the rules for identifiers.

- `No specification for engine group engine_group exists.`

  engine_group does not exist. Use **sp_showcontrolinfo** to see which engine groups exist.

- `Priority value priority is not valid.`

  The *priority* value must be **HIGH**, **LOW**, or **MEDIUM**.

Permissions

Only a System Administrator can execute **sp_addexeclass.**

Tables Used

master..sysattributes

See Also

System procedures	**sp_addengine, sp_bindexeclass, sp_clearpsexe, sp_dropengine, sp_dropexeclass, sp_setpsexe, sp_showcontrolinfo, sp_showexeclass, sp_showpsexe, sp_unbindexeclass**

sp_addextendedproc

Function

Creates an extended stored procedure (ESP) in the *master* database.

Syntax

sp_addextendedproc *esp_name, dll_name*

Parameters

esp_name – is the name of the extended stored procedure. This name must be identical to the name of the procedural language function that implements the ESP. *esp_name* must be a valid Adaptive Server identifier.

dll_name – is the name of the dynamic link library (DLL) file containing the function specified by *esp_name*. The *dll_name* can be specified with no extension or with its platform-specific extension, such as *.dll* on Windows NT or *.so* on Sun Solaris. If an extension is specified, the *dll_name* must be enclosed in quotation marks.

Examples

1. `sp_addextendedproc xp_echo, "sqlsrvdll.dll"`

Registers an ESP for the function named *xp_echo*, which is in the *sqlsrvdll.dll* file. The name of the resulting ESP database object is also *xp_echo*.

Comments

- Execute **sp_addextendedproc** from the *master* database.
- The *esp_name* is case sensitive. It must match the name of the function in the DLL.
- The DLL represented by *dll_name* must reside on the server machine on which the ESP is being created and the DLL directory must be in the *$PATH* on Windows NT, the *$LD_LIBRARY_PATH* on Digital UNIX, or the *$SH_LIBRARY_PATH* on HP. If the file is not found, the search mechanism also searches *$SYBASE/dll* on Windows NT and *$SYBASE/lib* on other platforms.
- On Windows NT, an ESP function should not call a C run-time signal routine. This can cause XP Server to fail, because Open Server™ does not support signal handling on Windows NT.

Messages

- `Can't run sp_addextendedproc from within a transaction.`

  sp_addextendedproc modifies system tables, so it cannot be run within a transaction.

- *`esp_name`* `is not a valid name.`

  The *esp_name* must be a valid Adaptive Server identifier. You may need to change the name of the function implementing the ESP so that it is in compliance, since the ESP and its supporting function must have identical names.

- `sp_addextendedproc failed. Check the SQL Server error log file.`

  Make sure that the *dll_name* exists and contains a function named *esp_nam*e.

- `You must be in the master database in order to run sp_addextendedproc.`

  Change to the *master* database.

Permissions

Only the System Administrator can execute **sp_addextendedproc**.

Tables Used

master.dbo.syscomments, *sysobjects*

See Also

Commands	**create procedure**
System procedures	**sp_dropextendedproc**, **sp_helpextendedproc**

sp_addexternlogin

(Component Integration Services only)

Function

Creates an alternate login account and password to use when communicating with a remote server through Component Integration Services.

Syntax

```
sp_addexternlogin server, loginname, externname
   [, externpassword]
```

Parameters

server – is the name of the remote server that has been added to the local server with **sp_addserver**.

loginname – is the name of the Adaptive Server login account for which to create an alternate login account.

externname – is the name of an account on the remote server *server*. This account is used when logging into *server*.

externpassword – is the password for *externname*.

Examples

1. `sp_addexternlogin JOBSERV, sa, system, sys_pass`

   Allows the local server to gain access to the remote server JOBSERV using the remote login "system" and the remote password "sys_pass" on behalf of user "sa".

2. `sp_addexternlogin CIS1012, bobj, jordan, hitchpost`

   When the user "bobj" logs into the remote server CIS1012, he connects using the remote server login name "jordan" and the password "hitchpost".

Comments

- **sp_addexternlogin** assigns an alternate login name and password to be used when communicating with a remote server. It stores the password internally in encrypted form.

  You can use **sp_addexternlogin** only when Component Integration Services is installed and configured.

- Before running **sp_addexternlogin**, add the remote server to Adaptive Server with **sp_addserver**.
- *externname* and *externpassword* must be a valid user and password combination on the node where the *server* runs.
- Sites with automatic password expiration need to plan for periodic updates of passwords for external logins.
- Use **sp_dropexternlogin** to remove the definition of the external login.
- **sp_addexternlogin** cannot be used from within a transaction.
- The "sa" account and the *loginname* account are the only users who can modify remote access for a given local user.

Messages

- `Can't run sp_addexternlogin from within a transaction.`

  sp_addexternlogin modifies system tables, so it cannot be run within a transaction.

- *`loginname`* `is not a local user -- request denied.`

  Check the spelling of *loginname*. Use **sp_helpuser** to list logins; add logins with **sp_addlogin**.

- `Only the System Administrator may update another's external login.`

  Only the "sa" account can use **sp_addexternlogin** for an account that you do not own.

- *`server`* `is the local server - external login not applicable.`

  You specified the name of the local server for the *server* parameter.

- `Sysattributes row updated.`

  This message confirms that if an *externname* is already defined for a *server*, the entry was updated.

- `There is not a server named` *`server`*`.`

  The *server* specified is not defined to the local server. Check the spelling of the remote server name or use **sp_helpserver** to list the known remote servers. Add remote servers with **sp_addserver**.

- User *loginname* will be known as *externname* to remote server *server*.

  sp_addexternlogin was successful. The local user *loginname* now has access to *server* as *externname*.

Permissions

Only the *loginname*, the System Administrator, and the System Security Officer can add or modify an external login for *loginname*.

Tables Used

master.dbo.syslogins, *master.dbo.sysattributes*, *master.dbo.sysservers*

See Also

System procedures	**sp_addserver**, **sp_dropexternlogin**, **sp_helpserver**

sp_addgroup

Function

Adds a group to a database. Groups are used as collective names in granting and revoking privileges.

Syntax

sp_addgroup *grpname*

Parameters

grpname – is the name of the group. Group names must conform to the rules for identifiers.

Examples

1. sp_addgroup accounting

Creates a group named *accounting* in the current database.

Comments

- **sp_addgroup** adds the new group to a database's *sysusers* table. Each group's user ID (*uid*) is 16384 or larger (except "public," which is always 0).
- A group and a user cannot have the same name.
- Once a group has been created, add new users with **sp_adduser**. To add an existing user to a group, use **sp_changegroup**.
- Every database is created with a group named "public". Every user is automatically a member of "public". Each user can be a member of one additional group.

Messages

- `A group with the specified name already exists.`

  The group name you supplied is being used as a group name. Choose another name.

- `A user with the specified group name already exists.`

  The group name you supplied is being used as a user name. Choose another name.

- *`grpname`* `is not a valid name.`

  Group names must conform to the rules for identifiers.

- `New group added.`

  The group has been added to the current database's *sysusers* table.

Permissions

Only the Database Owner, a System Administrator, or a System Security Officer can execute **sp_addgroup**.

Tables Used

sysobjects, *sysusers*

See Also

Commands	**grant**, **revoke**
System procedures	**sp_adduser**, **sp_changegroup**, **sp_dropgroup**, **sp_helpgroup**

sp_addlanguage

Function

Defines the names of the months and days for an alternate language and its date format.

Syntax

```
sp_addlanguage language, alias, months, shortmons,
   days, datefmt, datefirst
```

Parameters

language – is the official language name for the language, entered in 7-bit ASCII characters only.

alias – substitutes for the alternate language's official name. Enter either "null", to make the alias the same as the official language name, or a name you prefer. You can use 8-bit ASCII characters in an alias—"français", for example—if your terminal supports them.

months – is a list of the full names of the 12 months, ordered from January through December, separated only by commas (no spaces allowed). Month names can be up to 20 characters long and can contain 8-bit ASCII characters.

shortmons – is a list of the abbreviated names of the 12 months, ordered from January through December, separated only by commas (no spaces allowed). Month abbreviations can be up to 9 characters long and can contain 8-bit ASCII characters.

days – is a list of the full names of the seven days, ordered from Monday through Sunday, separated only by commas (no spaces allowed). Day names can be up to 30 characters long and can contain 8-bit ASCII characters.

datefmt – is the date order of the date parts *month/day/year* for entering *datetime* or *smalldatetime* data. Valid arguments are *mdy*, *dmy*, *ymd*, *ydm*, *myd*, or *dym*. "dmy" indicates that dates are in day / month / year order.

datefirst – sets the number of the first weekday for date calculations. For example, Monday is 1, Tuesday is 2, and so on.

Examples

```
1. sp_addlanguage french, null,
       "janvier,fevrier,mars,avril,mai,juin,juillet,
        aout,septembre,octobre,novembre,decembre",
        "jan,fev,mars,avr,mai,juin,jui,aout,sept,oct,
        nov,dec",
        "lundi,mardi,mercredi,jeudi,vendredi,samedi,
        dimanche",
        dmy, 1
```

This stored procedure adds French to the languages available on the server. "null" makes the alias the same as the official name, "french". Date order is "dmy"—day / month / year. "1" specifies that lundi, the first item in the *days* list, is the first weekday. Because the French do not capitalize the names of the days and months except when they appear at the beginning of a sentence, this example shows them being added in lowercase.

Comments

- Usually, you add alternate languages from one of Adaptive Server's Language Modules using the **langinstall** utility or the Adaptive Server installation program. A Language Module supplies the names of the dates and translated error messages for that language. However, if a Language Module is not provided with your server, use **sp_addlanguage** to define the date names and format.
- Use **sp_modifylogin** to change a user's default language. If you set a user's default language to a language added with **sp_addlanguage**, and there are no localization files for the language, the users receive an informational message when they log in, indicating that their client software could not open the localization files.

System Table Changes

- **sp_addlanguage** creates an entry in *master.dbo.syslanguages*, inserting a unique numeric value in the *langid* column for each alternate language. *langid* 0 is reserved for U.S. English.
- The *language* parameter becomes the official language name, stored in the *name* column of *master.dbo.syslanguages*. Language names must be unique. Use **sp_helplanguage** to display a list of the alternate languages available on Adaptive Server.
- **sp_addlanguage** sets the *alias* column in *master.dbo.syslanguages* to the official language name if NULL is entered for *alias*, but System

Administrators can change the value of *syslanguage.alias* with **sp_setlangalias.**

- **sp_addlanguage** sets the *upgrade* column in *master.dbo.syslanguages* to 0.

Dates for Languages added with *sp_addlanguage*

- For alternate languages added with Language Modules, Adaptive Server sends date values to clients as *datetime* datatype, and the clients use localization files to display the dates in the user's current language. For date strings added with **sp_addlanguage**, use the **convert** function to convert the dates to character data in the server:

  ```
  select convert(char, pubdate) from table
  ```

  where *pubdate* is *datetime* data and *table* is any table.

- When users perform data entry on date values and need to use date names created with **sp_addlanguage**, the client must have these values input as character data, and sent to the server as character data.

Messages

- `'language' already exists in syslanguages.`

  This language already exists on the server. To change only the language alias, use **sp_setlangalias.** To change *months*, *shortmons*, *days*, *datefmt*, or *datefirst*, drop the language with **sp_droplanguage**, and then add it again with your new specifications.

- `List of full month names contains spaces, which are not allowed.`

  Separate month names with commas only; no spaces are allowed.

- `List of full month names contains name(s) which have iso_1 non-alphabetic characters.`

  Month names cannot contain non-alphabetic characters, such as punctuation.

- `List of full month names has too few names.`

  The months list must have exactly 12 names separated by exactly 11 commas.

- `List of full month names has too many names.`

  The months list must have exactly 12 names separated by exactly 11 commas.

- `List of full month names has name(s) which are too long.`

  One or more names in the list of full month names is more than 20 characters long.

- `List of short month names contains spaces, which are not allowed.`

  Short month names cannot contain non-alphabetic characters, such as spaces.

- `List of short month names contains name(s) which have iso_1 non-alphabetic characters.`

  Short month names cannot contain non-alphabetic characters, such as punctuation.

- `List of short month names has too few names.`

  The months list must have exactly 12 names separated by exactly 11 commas.

- `List of short month names has too many names.`

  The months list must have exactly 12 names separated by exactly 11 commas.

- `List of short month names has name(s) which are too long.`

  One or more names in the list of short month names is more than 9 characters long.

- `List of day names contains spaces, which are not allowed.`

  Day names cannot contain non-alphabetic characters, such as spaces.

- `List of day names contains name(s) which have iso_1 non-alphabetic characters.`

  Day names cannot contain non-alphabetic characters, such as punctuation.

- `List of day names has too few names.`

  The days list must have exactly 7 names separated by exactly 6 commas.

- `List of day names has too many names.`

  The days list must have exactly 7 names separated by exactly 6 commas.

- `List of day names has name(s) which are too long.`

  One or more names in the list of day names is more than 30 characters long.

- `'datefmt' is not a valid date order.`

  datefmt must be in one of the following six orders: “mdy”, “myd”, “dmy,” “dym”, “ydm”, “ymd”.

- `'datefirst' is not a valid first day.`

  The first day of a week must be 1 for Monday through 7 for Sunday.

- `'alias' alias already exists in syslanguages.`

  The name given as an alias is already in use as an alias in the table *master.dbo.syslanguages*. If *alias* was specified as NULL, then the official language name for this new language is already in use as the alias for another language.

- `Language not inserted.`

  An error occurred while you were adding this language to *master.dbo.syslanguages*, so the language was not added. The Adaptive Server error message that appeared before this message gives more specific information about the error.

- `New language inserted.`

  A new alternate language was added to *master.dbo.syslanguages*.

Permissions

Only a System Administrator can execute **sp_addlanguage**.

Tables Used

master.dbo.syslanguages, *sysobjects*

See Also

Commands	**set**
System procedures	**sp_droplanguage**, **sp_helplanguage**, **sp_setlangalias**, **sp_modifylogin**

sp_addlogin

Function

Adds a new user account to Adaptive Server.

Syntax

```
sp_addlogin loginame, passwd [, defdb [, deflanguage
   [, fullname]]]
```

Parameters

loginame – is the user's login name. Login names must conform to the rules for identifiers.

passwd – is the user's password. Passwords must be at least 6 characters long. If you specify a shorter password, **sp_addlogin** returns an error message and exits. Enclose passwords that include characters besides A-Z, a-z, or 0-9 in quotation marks. Also enclose passwords that **begin** with 0-9 in quotation marks.

defdb – is the name of the default database assigned when a user logs into Adaptive Server. If you do not specify *defdb*, the default, *master*, is used.

deflanguage – is the official name of the default language assigned when a user logs into Adaptive Server. The Adaptive Server default language, defined by the **default language id** configuration parameter, is used if you do not specify *deflanguage*.

fullname – is the full name of the user who owns the login account. This can be used for documentation and identification purposes.

Examples

1. **`sp_addlogin albert, longer1, corporate`**

   Creates an Adaptive Server login for "albert" with the password "longer1" and the default database *corporate*.

2. **`sp_addlogin claire, bleurouge, public_db, french`**

   Creates an Adaptive Server login for "claire". Her password is "bleurouge", her default database is *public_db*, and her default language is French.

3. **sp_addlogin robertw, terrible2, public_db, null, "Robert Willis"**

   Creates an Adaptive Server login for "robertw". His password is "terrible2", his default database is *public_db*, and his full name is "Robert Willis". Do not enclose **null** in quotes.

4. **sp_addlogin susan, wonderful, null, null, "Susan B. Anthony"**

   Creates a login for "susan" with a password of "wonderful", a full name of "Susan B. Anthony", and the server's default database and language. Do not enclose **null** in quotes.

5. **sp_addlogin susan, wonderful, @fullname="Susan B. Anthony"**

   An alternative way of creating the login shown in example 4.

Comments

- For ease of management, it is strongly recommended that all users' Adaptive Server login names be the same as their operating system login names. This makes it easier to correlate audit data between the operating system and Adaptive Server. Otherwise, keep a record of the correspondence between operating system and server login names.
- After assigning a default database to a user with **sp_addlogin**, the Database Owner or System Administrator must provide access to the database by executing **sp_adduser** or **sp_addalias**.
- Although a user can use **sp_modifylogin** to change his or her own default database at any time, a database cannot be used without permission from the Database Owner.
- A user can use **sp_password** at any time to change his or her own password. A System Security Officer can use **sp_password** to change any user's password.
- A user can use **sp_modifylogin** to change his or her own default language. A System Administrator can use **sp_modifylogin** to change any user's default language.
- A user can use **sp_modifylogin** to change his or her own *fullname*. A System Administrator can use **sp_modifylogin** to change any user's *fullname*.

Messages

- `'loginame' is not a valid name.`

  loginame must conform to the rules for identifiers.

- `'deflanguage' is not an official language name from syslanguages.`

  Use **sp_helplanguage** to determine the alternate languages available. Add an alternate language with **langinstall**, or specify us_english.

- `Can't run sp_addlogin from within a transaction.`

  sp_addlogin modifies system tables, so it cannot be run within a transaction.

- `A user with the specified login name already exists.`

  Choose another *loginame*. If you want to change only the user's password, default database, or default language, use **sp_password** or **sp_modifylogin**.

- `Database name not valid -- login not added.`

  The default database you specified does not exist. Create the database first or choose a database that already exists.

- `New login created.`

Permissions

Only a System Administrator or a System Security Officer can execute **sp_addlogin**.

Tables Used

master.dbo.sysdatabases, master.dbo.syslogins, sysobjects

See Also

System procedures	**sp_addalias, sp_adduser, sp_droplogin, sp_locklogin, sp_modifylogin, sp_password, sp_role**

sp_addmessage

Function

Adds user-defined messages to *sysusermessages* for use by stored procedure **print** and **raiserror** calls and by **sp_bindmsg**.

Syntax

```
sp_addmessage message_num, message_text [, language
   [, with_log [, replace]]]
```

Parameters

message_num – is the message number of the message to add. The message number for a user-defined message must be 20000 or greater.

message_text – is the text of the message to add.The maximum length is 255 bytes.

language – is the language of the message to add. This must be a valid language name in the *syslanguages* table. If this parameter is missing, Adaptive Server assumes that messages are in the default session language indicated by *@@langid*.

with_log – specifies whether the message is logged in the Adaptive Server error log as well as in the Windows NT Event Log on Windows NT servers, if logging is enabled. If *with_log* is TRUE, the message is logged, regardless of the severity of the error. If *with_log* is FALSE, the message may or may not be logged, depending on the severity of the error. If you do not specify a value for *with_log*, the default is FALSE.

replace – specifies whether to overwrite an existing message of the same number and *languid*. If *replace* is TRUE, the existing message is overwritten; if *replace* is FALSE, it is not. If you do not specify a value for *replace*, the default, FALSE, is used.

Examples

1.
```
sp_addmessage 20001, "The table '%1!' is not owned
by the user '%2!'."
```

   Adds a message with the number 20001 to *sysusermessages*.

2. **sp_addmessage 20002, "The procedure'%1!' is not owned by the user '%2!'.", NULL, TRUE, TRUE**

   Adds a message with the number 20002 to *sysusermessages*. This message is logged in the Adaptive Server error log, as well as in the Windows NT Event Log on Windows NT servers, if event logging is enabled. If a message numbered 20002 exists in the default session language, this message overwrites the old message.

Comments

- **sp_addmessage** does not overwrite an existing message of the same number and *langid* unless you specify **@replace = TRUE**.
- **print**, **raiserror**, and **sp_bindmsg** recognize placeholders in the message text to print out. A single message can contain up to 20 unique placeholders in any order. These placeholders are replaced with the formatted contents of any arguments that follow the message when the text of the message is sent to the client.

  The placeholders are numbered to allow reordering of the arguments when Adaptive Server is translating a message to a language with a different grammatical structure. A placeholder for an argument appears as "%*nn*!", a percent sign (%), followed by an integer from 1 to 20, followed by an exclamation point (!). The integer represents the argument number in the string in the argument list. "%1!" is the first argument in the original version, "%2!" is the second argument, and so on.

Messages

- `'language' is not an official language name from syslanguages.`

  Use **sp_helplanguage** to display the list of official language names available on this Adaptive Server.

- `Message number must be at least 20000.`

  User-defined messages must have a message number of 20000 or greater.

- `Cannot add message until sysusermessages system table is created properly by Upgrade.`

  sysusermessages was added to Adaptive Server in release 4.9. This Adaptive Server has not been properly upgraded. See the installation documentation for your platform for information on upgrading Adaptive Server.

- `A message with number message_number in the specified language already exists. Drop the old message first if you still wish to add this one.`

  To insert a message with a number and language that already exists in *sysusermessages*, specify a *replace* value of TRUE.

- `The message has not been inserted.`

  sp_addmessage failed. *sysusermessages* is unchanged.

- `The message has been inserted.`

  sp_addmessage succeeded.

Permissions

Any user can execute **sp_addmessage**.

Tables Used

master.dbo.syslanguages, *sysobjects*, *sysusermessages*

See Also

Commands	**print**, **raiserror**
System procedures	**sp_altermessage**, **sp_dropmessage**, **sp_getmessage**

sp_addobjectdef

(Component Integration Services only)

Function

Specifies the mapping between a local table and an external storage location.

Syntax

```
sp_addobjectdef tablename, "objectdef" ["objecttype"]
```

Parameters

tablename – is the name of the object as it is defined in a local table. The *tablename* can be in any of the following forms:

- *dbname.owner.object*
- *dbname..object*
- *owner.object*
- *object*

dbname and *owner* are optional. *object* is required. If you do not specify an *owner*, the default (current user name) is used. If you specify a *dbname*, it must be the current database name, and you must specify *owner* or mark the owner with a placeholder in the format *dbname..object*. Enclose any multipart *tablename* values in quotes.

objectdef – is a string naming the external storage location of the object. The *objecttype* at *objectdef* can be a table, view, or read-only remote procedure call (RPC) result set accessible to a remote server. A table, view, or RPC uses the following format for *objectdef*:

server_name.dbname.owner.object

server_name and *object* are required. *dbname* and *owner* are optional, but if they are not supplied, a placeholder in the format *dbname..object*, is required.

For more information, see “Server Classes” in the *Component Integration Services User’s Guide*.

objecttype – is one of the values that specify the format of the object named by *objectdef*. Table 3-2 describes the valid values. Enclose the *objecttype* value in quotes.

Table 3-2: Allowable values for objecttype

Value	Description
table	Indicates that the object named by *objectdef* is a table accessible to a remote server. This value is the default for *objecttype*.
view	Indicates that the object named by *objectdef* is a view managed by a remote server and processed as a table.
rpc	Indicates that the object named by *objectdef* is an RPC managed by a remote server. Adaptive Server processes the result set from the RPC as a read-only table.

Table 3-3 summarizes how each *objecttype* is used:

Table 3-3: Summary of objecttype uses

objecttype	*create table*	*create existing table*	Write to Table	Read from Table
table	Yes	Yes	Yes	Yes
view	No	Yes	Yes	Yes
rpc	No	Yes	No	Yes

Examples

1. **`sp_addobjectdef "finance.dbo.accounts", "SYBASE.pubs.dbo.accounts", "table"`**

   Maps the local table *accounts* in the database *finance* to the remote object *pubs.dbo.accounts* in the remote server named SYBASE. The current database must be *finance*. A subsequent **create table** creates a table in the *pubs* database. If *pubs.dbo.accounts* is an existing table, a **create existing table** statement populates the table *finance.dbo.accounts* with information about the remote table.

2. **`sp_addobjectdef stockcheck, "NEWYORK.wallstreet.kelly.stockcheck", "rpc"`**

   Maps the local table *stockcheck* to an RPC named *stockcheck* on remote server NEWYORK in the database *wallstreet* with owner

"kelly". The result set from RPC *stockcheck* is seen as a read-only table. Typically, the next operation would be a **create existing table** statement for the object *stockcheck*.

Comments

- **sp_addobjectdef** specifies the mapping between a local table and an external storage location. It identifies the format of the object at that location.

  You can use **sp_addobjectdef** only when Component Integration Services is installed and configured.

- **sp_addobjectdef** replaces the **sp_addtabledef** command. **sp_addtabledef** allows existing scripts to run without modification. Internally, **sp_addtabledef** invokes **sp_addobjectdef.**

- Only the System Administrator can provide the name of another user as a table owner.

- When *objecttype* is **table**, **view**, or **rpc**, the *objectdef* parameter takes the following form:

  "*server_name.database.owner.tablename*"

  - *server_name* represents a server that has already been added to *sysservers* by **sp_addserver.**
  - *database* may not be required. Some server classes do not support it.
  - *owner* should always be provided, to avoid ambiguity. If you do not specify *owner*, the remote object referenced may vary, depending on whether or not the external login corresponds to the remote object owner.
  - *tablename* is the name of a remote server table.

- Use **sp_addobjectdef** before issuing any **create table** or **create existing table** commands. **create table** is valid only for the *objecttype* values **table** and **file**. When either **create table** or **create existing table** is used, Adaptive Server checks *sysattributes* to determine whether any table mapping has been specified for the object. Follow the *objecttype* values **view** and **rpc** with **create existing table** statements.

- After the table has been created, all future references to the local table name (by **select, insert, delete** and **update**) are mapped to the correct location.

- For information about RMS, see the *Component Integration Services User's Guide.*

Messages

- `Database dbname is not your current database.`

  dbname is optional in *tablename*. If supplied, it must be the current database.

- `tablename is not a valid name.`

  The syntax for *tablename* is not valid. One or more of the elements in *dbname.owner.object* does not conform to the rules for identifiers.

- `object is not a valid name`

  The format of the name supplied for the *object* component in *tablename* is not valid.

- `Object objectdef has invalid format.`

  For a **table**, **view**, or **rpc** object type, the format of the *objectdef* parameter is invalid.

- `Object tablename has already been defined.`

  An entry for *tablename* already exists in *sysattributes*. The definition must be dropped using **sp_dropobjectdef** before it can be defined again with this name.

- `owner is not a valid user in dbname database.`

  The value specified for *owner* in the *tablename* is not a defined user in the current database.

- `Server name is not permitted in the local tablename.`

  The *tablename* contains a server name.

- `Server name server_name does not exist in sysservers.`

  The remote server has not been identified to the server by an **sp_addserver** command. Use **sp_helpserver** to list all remote servers.

- `Table tablename has already been created.`

  The **create table** or **create existing table** command has already been issued against this object name. To add an object definition with this object name, the object must first be dropped using **drop table**.

- `The server name in objectdef must be a remote server; server_name is the local server.`

  sp_addobjectdef does not allow mapping to the local server.

- `Unknown object type objecttype.`

  The value for the *objecttype* parameter must be either **table**, **view**, or **rpc**.

- `You must be the System Administrator (sa) or Database Owner (dbo) to add a definition for another user's table.`

  If the *owner* parameter is supplied, the value must be either the owner of the object or the System Administrator.

Permissions

Any user can issue **sp_addobjectdef**.

Tables Used

sysobjects, *sysattributes*, *sysservers*

See Also

Commands	**create existing table**, **create table**, **drop table**
System procedures	**sp_addlogin**, **sp_addserver**, **sp_defaultloc**, **sp_dropobjectdef**, **sp_helpserver**

sp_addremotelogin

Function

Authorizes a new remote server user by adding an entry to *master.dbo.sysremotelogins*.

Syntax

```
sp_addremotelogin remoteserver [, loginame
   [, remotename] ]
```

Parameters

remoteserver – is the name of the remote server to which the remote login applies. This server must be known to the local server by an entry in the *master.dbo.sysservers* table, which was created with **sp_addserver**.

➤ ***Note***

This manual page uses the term "local server" to refer to the server that is executing the remote procedures run from a "remote server".

loginame – is the login name of the user on the local server. *loginame* must already exist in the *master.dbo.syslogins* table.

remotename – is the name used by the remote server when logging into the local server. All *remotename*s that are not explicitly matched to a local *loginame* are automatically matched to a local name. In example 1, the local name is the remote name that is used to log in. In example 2, the local name is “albert”.

Examples

1. `sp_addremotelogin GATEWAY`

Creates an entry in the *sysremotelogins* table for the remote server GATEWAY, for purposes of login validation. This is a simple way to map remote names to local names when the local and remote servers have the same users.

This example results in a value of -1 for the *suid* column and a value of NULL for the *remoteusername* in a row of *sysremotelogins*.

2. **`sp_addremotelogin GATEWAY, albert`**

   Creates an entry that maps all logins from the remote server GATEWAY to the local user name "albert". Adaptive Server adds a row to *sysremotelogins* with Albert's server user ID in the *suid* column and a null value for the *remoteusername*.

   For these logins to be able to run RPCs on the local server, they must specify a password for the RPC connection when they log into the local server, or they must be "trusted" on the local server. To define these logins as "trusted", use **sp_remoteoption**.

3. **`sp_addremotelogin GATEWAY, ralph, pogo`**

   Maps a remote login from the remote user "pogo" on the remote server GATEWAY to the local user "ralph". Adaptive Server adds a row to *sysremotelogins* with Ralph's server user ID in the *suid* column and "pogo" in the *remoteusername* column.

Comments

- When a remote login is received, the local server tries to map the remote user to a local user in three different ways:
  - First, the local server looks for a row in *sysremotelogins* that matches the remote server name and the remote user name. If the local server finds a matching row, the local server user ID for that row is used to log in the remote user. This applies to mappings from a specified remote user.
  - If no matching row is found, the local server searches for a row that has a null remote name and a local server user ID other than -1. If such a row is found, the remote user is mapped to the local server user ID in that row. This applies to mappings from any remote user from the remote server to a specific local name.
  - Finally, if the previous attempts failed, the local server checks the *sysremotelogins* table for an entry that has a null remote name and a local server user ID of -1. If such a row is found, the local server uses the remote name supplied by the remote server to look for a local server user ID in the *syslogins* table. This applies when login names from the remote server and the local server are the same.
- The name of the local user may be different on the remote server.
- If you use **sp_addremotelogin** to map all users from a remote server to the same local name, use **sp_remoteoption** to specify the "trusted" option for those users. For example, if all users from the server

GOODSRV that are mapped to “albert” are to be “trusted”, use **sp_remoteoption** as follows:

```
sp_remoteoption GOODSRV, albert, NULL, trusted
  true
```

Logins that are not specified as “trusted” cannot execute RPCs on the local server unless they specify passwords for the local server when they log into the remote server. In Open Client™ Client-Library™, the user can use the **ct_remote_pwd** routine to specify a password for server-to-server connections. **isql** and **bcp** do not permit users to specify a password for RPC connections.

If users are logged into the remote server using “unified login”, these logins are already authenticated by a security mechanism. These logins must also be trusted on the local server, or the users must specify passwords for the server when they log into the remote server.

- For more information about setting up servers for remote procedure calls and for using “unified login”, see the *Security Administration Guide*.
- Every remote login entry has a status. The default status for the **trusted** option is **false** (not trusted). This means that when a remote login comes in using that entry, the password is checked. If you do not want the password to be checked, change the status of the **trusted** option to **true** with **sp_remoteoption**.

Messages

- `Can’t run sp_addremotelogin from within a transaction.`

  sp_addremotelogin modifies system tables, so it cannot be run within a transaction.

- `'`*`loginame`*`' isn’t a local user -- remote login denied.`

  The *loginame* is not in the *master..syslogins* table. If you supply a local name, it must exist as a user on the local server.

- `New remote login created.`

  A remote login was created.

- `'`*`remoteserver`*`' is the local server - remote login not applicable.`

  You specified the name of the local server as the remote server.

- `There is already a default-name mapping of a remote login from remote server '`*`remoteserver`*`'.`

  You tried to add a remote login entry that already exists. See example 1 and example 2. Use **sp_helpremotelogin** to see the remote logins for the *remoteserver*.

- `There is already a remote user named '`*`remotename`*`' for remote server '`*`remoteserver`*`'.`

  A user with the remote login name you specified for the remote server already exists. Drop the remote user before choosing another *remotename*.

- `There is not a server named '`*`server`*`'.`

  The remote server you specified does not exist. Use **sp_helpserver** to get a list of the existing remote servers.

- `Usage:sp_addremotelogin remoteserver [, loginame [, remotename]]`

  The parameter you specified is incorrect.

Permissions

Only a System Administrator can execute **sp_addremotelogin**.

Tables Used

master.dbo.syslogins, *master.dbo.sysremotelogins*, *master.dbo.sysservers*, *sysobjects*

See Also

System procedures	**sp_addlogin**, **sp_addserver**, **sp_dropremotelogin**, **sp_helpremotelogin**, **sp_helprotect**, **sp_helpserver**, **sp_remoteoption**

sp_add_resource_limit

Function

Creates a limit on the number of server resources that can be used by an Adaptive Server login and / or an application to execute a query, query batch, or transaction.

Syntax

```
sp_add_resource_limit name, appname, rangename,
   limittype, limitvalue [, enforced [, action
   [, scope ]]]
```

Parameters

name – is the Adaptive Server login to which the limit applies. You must specify either a *name* or an *appname* or both. To create a limit that applies to all users of a particular application, specify a *name* of NULL.

appname – is the name of the application to which the limit applies. You must specify either a *name* or an *appname* or both. To create a limit that applies to all applications used by an Adaptive Server login, specify an *appname* of **null**. To create a limit that applies to a particular application, specify the application name that the client program passes to the Adaptive Server in the login packet.

rangename – is the time range during which the limit is enforced. The time range must exist in the *systimeranges* system table of the *master* database at the time you create the limit.

limittype – is the type of resource to limit. This must be one of the following:

Limit Type	Description
row_count	Limits the number of rows a query can return
elapsed_time	Limits the number of seconds, in wall-clock time, that a query batch or transaction can run
io_cost	Limits either the actual cost or the optimizer's cost estimate for processing a query

limitvalue – is the maximum amount of the server resource (I / O cost, elapsed time in seconds, or row count) that can be used by the

login or application before Adaptive Server enforces the limit. This must be a positive, nonzero integer that is less than or equal to 2^{31}. The following table indicates what value to specify for each limit type:

Limit Type	Limit Value
row_count	The maximum number of rows that can be returned by a query before the limit is enforced.
elapsed_time	The number of seconds, in wall-clock time, that a query batch or transaction can run before the limit is enforced.
io_cost	A unitless measure derived from the optimizer's costing formula.

enforced – determines whether the limit is enforced prior to or during query execution. The following table lists the valid values for each limit type:

enforced Code	Description	Limit Type
1	Action is taken when the estimated I/O cost of execution exceeds the specified limit.	**io_cost**
2	Action is taken when the actual row count, elapsed time, or I/O cost of execution exceeds the specified limit.	**row_count** **elapsed_time** **io_cost**
3	Action is taken when either the estimated cost or the actual cost exceeds the specified limit.	**io_cost**

If you specify an *enforced* value of 3, Adaptive Server performs a logical "or" of 1 and 2. For example, assume *enforced* is set to 3. If you run a query whose **io_cost** exceeds the estimated cost, the specified *action* is executed. If the query is within the limits specified for estimated cost but exceeds the actual cost, the specified *action* is also executed.

If you do not specify an *enforced* value, Adaptive Server enforces limit 2 for **row_count** and **elapsed_time** and limit 3 for **io_cost.** In other words, if the limit type is **io_cost**, the specified action is executed if the query exceeds either the estimated or actual cost.

action – is the action to take when the limit is exceeded. The following action codes are valid for all limit types:

action Code	Description
1	Issues a warning
2	Aborts the query batch
3	Aborts the transaction
4	Kills the session

If you do not specify an *action* value, Adaptive Server uses a default value of 2 (abort the query batch).

scope – is the scope of the limit. Specify one of the following codes appropriate to the type of limit:

scope Code	Description	Limit Type
1	Query	**io_cost** **row_count**
2	Query batch (one or more SQL statements sent by the client to the server)	**elapsed_time**
4	Transaction	**elapsed_time**
6	Query batch **and** transaction	**elapsed_time**

If you do not specify a *scope* value, the limit applies to all possible scopes for the limit type.

Examples

1. **`sp_add_resource_limit NULL, payroll, early_morning, elapsed_time, 120, 2, 1, 2`**

   Creates a resource limit that applies to all users of the **payroll** application during the *early_morning* time range. If the query batch takes more than 120 seconds to execute, Adaptive Server issues a warning.

2. **`sp_add_resource_limit joe_user, NULL, midday, row_count, 5000, 2, 3, 1`**

   Creates a resource limit that applies to all ad hoc queries and applications run by "joe_user" during the *midday* time range.

When a query returns more than 5000 rows, Adaptive Server aborts the transaction.

3. **`sp_add_resource_limit joe_user, NULL, midday, io_cost, 650, 1, 3, 1`**

   Creates a resource limit that applies to all ad hoc queries and applications run by "joe_user" during the *midday* time range. When the optimizer estimates that the I/O cost would exceed 650, Adaptive Server aborts the transaction.

Comments

- Multiple resource limits can exist for a given user, application, limit type, scope, and enforcement time, as long as their time ranges do not overlap.
- All limits for the currently active named time ranges and the "at all times" range for a login and/or application name are bound to the user's session at login time. Therefore, if a user logs into Adaptive Server independently of a given application, resource limits that restrict the user in combination with that application do not apply. To guarantee restrictions on that user, create a resource limit that is specific to the user and independent of any application.
- Since either the user login name or application name, or both, are used to identify a resource limit, Adaptive Server observes a predefined search precedence while scanning the *sysresourcelimits* table for applicable limits for a login session. The following table describes the precedence of matching ordered pairs of login name and application name:

Level	Login Name	Application Name
1	"joe_user"	**payroll**
2	NULL	**payroll**
3	"joe_user"	NULL

  If one or more matches are found for a given precedence level, no further levels are searched. This prevents conflicts regarding similar limits for different login/application combinations.

  If no match is found at any level, no limit is imposed on the session.
- When you add, delete, or modify resource limits, Adaptive Server rebinds the limits for each session for that login and/or application at the beginning of the next query batch for that session.

- Adaptive Server also rebinds limits for the session when you change the currently active time ranges. This rebinding occurs at the beginning of the next query batch.
- You cannot associate the limits for a particular login, application, or login / application combination with named time ranges that overlap (except for limits that share the same time range).

  For example, if a user is limited to retrieving 50 rows between 9:00 a.m. and 1:00 p.m., you cannot create a second resource limit for the same user that limits him to retrieving 100 rows between 10:00 a.m. and 12:00 noon. However, you can create a resource hierarchy by assigning the 100-row limit to the **user** between 10:00 a.m. and 12:00 noon and assigning the 50-row limit to an **application**, like **isql**, between 9:00 a.m. and 1:00 p.m.
- For more information on resource limits, see Chapter 12, "Limiting Access to Server Resources," in the *System Administration Guide*.

Messages

- `At least one of login or application name must be non-NULL.`

  You must specify either an Adaptive Server login or an application name, or both, to which the resource limit applies.
- `Can't run sp_add_resource_limit from within a transaction.`

  Because **sp_add_resource_limit** modifies system tables, it cannot be run from within a transaction.
- `Illegal action` *`action.`*

  You must specify an *action* of **1**, **2**, **3**, **4**, or null.
- `Illegal enforcement-time value` *`enforced`* `for this limit type.`

  The enforcement time must be **1**, **2**, **3**, or null and must be compatible with the limit type.
- `Illegal limit value` *`limitvalue`*`. Value must be non-negative.`

  The limit value must be a positive integer greater than zero and less than or equal to 2^{31}.
- `Illegal scope value` *`scope`* `for this limit type.`

  The scope must be **1**, **2**, **4**, **6**, or null and must be compatible with the limit type.

- `Limit type must be non-NULL.`

  You must specify a limit type of **row_count**, **elapsed_time**, or **io_cost** for the resource limit.

- `New limit would overlap with limits on range` *`rangename`* `for this user-application combination.`

  The time range for this limit overlaps that of another resource limit for this user and application.

- `New resource limit created.`

  The resource limit was successfully added to the Adaptive Server.

- `No login with the specified name exists.`

  The login you specified does not exist in the *syslogins* system table of the *master* database.

- `Only the System Administrator (SA) may execute this procedure.`

  You must be a System Administrator to run **sp_add_resource_limit**.

- `This user/application can have only one limit for each distinct combination of time range, limit type, enforcement time and scope.`

  The time range for this limit overlaps that of another resource limit for this user, application, limit type, scope, and enforcement time.

- `Timerange name must be non-NULL.`

  You must specify a time range for the resource limit.

- `Unknown ending time value` *`endtime`* `found in systimeranges.`

  The time range you specified for this limit does not have an *endtime*. Use **sp_modify_time_range** to specify an *endtime*.

- `Unknown limit type` *`limittype.`*

  The limit type you specified does not exist in the *spt_limit_types* system procedure table in the *master* database. You must specify a limit type of **row_count**, **elapsed_time**, or **io_cost**.

- `Unknown starting time value` *`starttime`* `found in the systimeranges.`

  The time range you specified for this limit does not have a *starttime*. Use **sp_modify_time_range** to specify a *starttime*.

- `Unknown timerange name` *`rangename`*`.`

  The time range you specified does not exist in the *systimeranges* system table of the *master* database.

Permissions

Only System Administrators can execute **sp_add_resource_limit.**

Tables Used

master..sysresourcelimits, master..systimeranges

See Also

System procedures	**sp_configure**, **sp_drop_resource_limit**, **sp_help_resource_limit**, **sp_modify_resource_limit**

sp_addsegment

Function

Defines a segment on a database device in a database.

Syntax

sp_addsegment *segname*, *dbname*, *devname*

Parameters

segname – is the name of the new segment to add to the *syssegments* table of the database. Segment names are unique in each database.

dbname – specifies the name of the database in which to define the segment. *dbname* must be the name of the current database or match the database name qualifying **sp_addsegment**.

devname – is the name of the database device in which to locate *segname*. A database device can have more than one segment associated with it.

Examples

1. sp_addsegment indexes, pubs2, dev1

Creates a segment named *indexes* for the database *pubs2* on the database device named *dev1*.

2.
```
disk init
     name = "pubs2_dev",
     physname = "/dev/pubs_2_dev",
     vdevno = 9, size = 5120
go
alter database pubs2 on pubs2_dev = 2
go
pubs2..sp_addsegment indexes, pubs2, dev1
```

Creates a segment named *indexes* for the database *pubs2* on the database device named *dev1*.

Comments

- **sp_addsegment** defines segment names for database devices created with **disk init** and assigned to a specific database with an **alter database** or **create database** command.

- After defining a segment, use it in **create table** and **create index** commands and in the **sp_placeobject** procedure to place a table or index on the segment.

  When a table or index is created on a particular segment, all subsequent data for the table or index is located on the segment.
- Use the system procedure **sp_extendsegment** to extend the range of a segment to another database device used by the same database.
- If a database is extended with **alter database** on a device used by that database, the segments mapped to that device are also extended.
- The *system* and *default* segments are mapped to each database device included in a **create database** or **alter database** command. The *logsegment* is also mapped to each device, unless you place it on a separate device with the **log on** extension to **create database** or with **sp_logdevice**. See Chapter 17, "Creating and Using Segments," in the *System Administration Guide* for more information.
- If you attempt to use **sp_addsegment** in a database that has both data and the log on the same device, Adaptive Server returns an error message.

Messages

- `Can't run sp_addsegment from within a transaction.`

  sp_addsegment modifies system tables, so it cannot be run within a transaction.
- `'devname' is reserved exclusively as a log device.`

  You cannot create a segment on a disk device that is dedicated to the database log.
- `No such device exists -- run sp_helpdb to list the devices for the current database.`

  The named device does not exist in *sysdevices*.
- `Segment created.`

  The procedure was successful; *segname* is now in the current database.
- `'segname' is not a valid identifier.`

  Segment names must conform to the rules for identifiers. They must begin with a letter, an underscore (_), or a pound sign (#). After the first character, identifiers can include letters, underscores, pound signs, or dollar signs ($).

- `The maximum number of segments for the current database are already defined.`

  A database can have no more than 31 segments. You can drop a segment with **sp_dropsegment** and replace it with a new one.

- `The specified device is not used by the database.`

  Although the device named as the *devname* parameter exists in *master.dbo.sysdevices*, it is not used by the specified database, and a segment cannot be added to it. Segments can be defined only on database devices used by the database. Use the **alter database** command to extend the database on the device.

- `The specified device is not a database device.`

  Although the device named as the *devname* parameter exists in *master.dbo.sysdevices*, it is not a database device. It may be a dump device.

- `There is already a segment named 'segname'.`

  Segment names must be unique in each database.

- `You must execute this procedure from the database in which you wish to add a segment. Please execute 'use database_name' and try again.`

  sp_addsegment can add segments only in the database from which it is issued. Either issue the **use** *database_name* command to open the database in which you want to add a segment, or qualify **sp_addsegment** with the database name as shown in example 2.

Permissions

Only the Database Owner or a System Administrator can execute **sp_addsegment.**

Tables Used

master.dbo.sysdevices, *master.dbo.sysusages*, *sysobjects*, *syssegments*

See Also

Commands	**alter database**, **create index**, **create table**, **disk init**
System procedures	**sp_dropsegment**, **sp_extendsegment**, **sp_helpdb**, **sp_helpdevice**, **sp_placeobject**

sp_addserver

Function

Defines a remote server, or defines the name of the local server.

Syntax

```
sp_addserver lname [, class [, pname]]
```

Parameters

lname – is the name used to address the server on your system. **sp_addserver** adds a row to the *sysservers* table if there is no entry already present for *lname*. Server names must be unique and must conform to the rules for identifiers.

class – identifies the category of server being added. Table 3-4 lists allowable values for the *class* parameter:

Table 3-4: Allowable values for server_class parameter

class Parameter Value	Description
access_server	Server coded to the DirectConnect™ specification (Component Integration Services only)
db2	Server accessible by Net-Gateway™ or MDI™ Database Gateway (Component Integration Services only)
direct_connect	Functionally the same as **access_server** (Component Integration Services only)
generic	Server coded to the Generic Access Module specification (Component Integration Services only)
local	Local server (there can be only one) used only once after start-up, or after restarting Adaptive Server, to identify the local server name so that it can appear in messages printed by Adaptive Server
null	Remote server with no category defined
sql_server	Another Adaptive Server or Omni server (this is the default value)

pname – is the name in the interfaces file for the server named *lname*. This enables you to establish local aliases for other Adaptive

Servers or Backup Servers that you may need to communicate with. If you do not specify a *pname*, *lname* is used.

Examples

1. **`sp_addserver GATEWAY`**

   Adds an entry for a remote server named GATEWAY in *master.dbo.sysservers*. The *pname* is also GATEWAY.

2. **`sp_addserver GATEWAY, null, VIOLET`**

   Adds an entry for a remote server named GATEWAY in *master.dbo.sysservers*. The *pname* is VIOLET. If there is already a *sysservers* entry for GATEWAY with a different *pname*, the *pname* of server GATEWAY changes to VIOLET.

3. **`sp_addserver PRODUCTION, local`**

   Adds an entry for the local server named PRODUCTION.

4. **`sp_addserver SQLSRV10, sql_server, SS_MOSS`**

   Adds an entry for a remote server known to the local server as SQLSRV10. The remote server is of server class **sql_server**. The *network_name* for SQLSRV10 is SS_MOSS.

5. **`sp_addserver RDBAM_ALPHA, generic, rdbam_alpha`**

   Adds an entry for a remote server known to the local server as *RDBAM_ALPHA*. The remote server *RDBAM_ALPHA* is written to the Generic Access Module specification, which requires server class **generic**.

Comments

- The *sysservers* table identifies the name of the local server and its options, and any remote servers that the local server can communicate with.

  To execute a remote procedure call on a remote server, the remote server must exist in the *sysservers* table.

- If *lname* already exists as a server name in the *sysservers* table, **sp_addserver** changes the remote server's *srvnetname* to the name specified by *pname*. When it does this, **sp_addserver** reports which server it changed, what the old network name was, and what the new network name is.

- The installation or upgrade process for your server adds an entry in *sysservers* for a Backup Server. If you remove this entry, you cannot back up your databases.

- Adaptive Server requires that the Backup Server have an *lname* of SYB_BACKUP. If you do not want to use that as the name of your Backup Server, or if you have more than one Backup Server running on your system, modify the *pname* for server SYB_BACKUP with **sp_addserver** so that Adaptive Server can communicate with Backup Server for database dumps and loads.
- If you specify an *lname*, *pname* and *class* that already exist in *sysservers*, **sp_addserver** prints an error message and does not update *sysservers*.
- Use **sp_serveroption** to set or clear server options.
- For information on using Component Integration Services, see the *Component Integration Services User's Guide*.

Messages

- `Adding server 'lname', physical name 'pname'`

  The procedure was successful; *lname* is now known to the local Adaptive Server and can access the physical device *pname*.

- `Can't run sp_addserver from within a transaction.`

  sp_addserver modifies the system table *master.dbo.sysservers*, so it cannot be run within a transaction.

- `Changing physical name of server 'lname' from 'old_netname' to 'pname'`

  The server known to your Adaptive Server as *lname* now accesses the physical device *pname*, instead of *old_netname*.

- `Changing server class of server name from class to class.`

  This confirmation message indicates that the server *name*, which was defined with one *class* value, is now defined with a different *class* value.

- `'lname' is not a valid name.`

  lname does not conform to the rules for identifiers.

- `Server added.`

  You have successfully added a new server.

- `There is already a local server.`

  Although there may be many remote servers, there can be only one local server. **sp_addserver** with a value of **local** for the *class* parameter defines the name of the local server. If it already exists, the request is rejected.

- `There is already a server named 'lname', physical name 'pname'.`

  You have specified an *lname* and a *pname* that already exist in *sysservers*. Nothing changed.

- `Usage: sp_addserver lname [, {local | null | server_class} [, pname]]`

  If you specify a *pname*, you must specify a *class*.

Permissions

Only a System Security Officer can execute **sp_addserver**.

Tables Used

master.dbo.sysservers, *sysobjects*

See Also

System procedures	**sp_addremotelogin**, **sp_dropremotelogin**, **sp_dropserver**, **sp_helpremotelogin**, **sp_helpserver**, **sp_serveroption**

sp_addthreshold

Function

Creates a threshold to monitor space on a database segment. When free space on the segment falls below the specified level, Adaptive Server executes the associated stored procedure.

Syntax

sp_addthreshold *dbname*, *segname*, *free_space*, *proc_name*

Parameters

dbname – is the database for which to add the threshold. This must be the name of the current database.

segname – is the segment for which to monitor free space. Use quotes when specifying the "default" segment.

free_space – is the number of free pages at which the threshold is crossed. When free space in the segment falls below this level, Adaptive Server executes the associated stored procedure.

proc_name – is the stored procedure to be executed when the amount of free space on *segname* drops below *free_space*. The procedure can be located in any database on the current Adaptive Server or on an Open Server. Thresholds cannot execute procedures on remote Adaptive Servers.

Examples

1. **sp_addthreshold mydb, segment1, 200, pr_warning**

   Creates a threshold for *segment1*. When the free space on *segment1* drops below 200 pages, Adaptive Server executes the procedure **pr_warning**.

2. **sp_addthreshold userdb, user_data, 100,**
   o_server...mail_me

   Creates a threshold for the *user_data* segment. When the free space on *user_data* falls below 100 pages, Adaptive Server executes a remote procedure call to the Open Server **mail_me** procedure.

```
3. pubs2..sp_addsegment pubs2, indexes, 100,
   pr_warning
```

Creates a threshold on the *indexes* segment of the *pubs2* database. You can issue this command from any database.

Comments

- See Chapter 23, "Managing Free Space with Thresholds," in the *System Administration Guide* for more information about using thresholds.

Crossing a Threshold

- When a threshold is crossed, Adaptive Server executes the associated stored procedure. Adaptive Server uses the following search path for the threshold procedure:
  - If the procedure name does not specify a database, Adaptive Server looks in the database in which the threshold was crossed.
  - If the procedure is not found in this database, and the procedure name begins with "sp_", Adaptive Server looks in the *sybsystemprocs* database.

  If the procedure is not found in either database, Adaptive Server sends an error message to the error log.
- Adaptive Server uses a **hysteresis value**, the global variable *@@thresh_hysteresis*, to determine how sensitive thresholds are to variations in free space. Once a threshold executes its procedure, it is deactivated. The threshold remains inactive until the amount of free space in the segment rises to *@@thresh_hysteresis* pages above the threshold. This prevents thresholds from executing their procedures repeatedly in response to minor fluctuations in free space.

The Last-Chance Threshold

- By default, Adaptive Server monitors the free space on the segment where the log resides and executes **sp_thresholdaction** when the amount of free space is less than that required to permit a successful dump of the transaction log. This amount of free space, called the **last-chance threshold**, is calculated by Adaptive Server and cannot be changed by users.
- If the last-chance threshold is crossed before a transaction is logged, Adaptive Server suspends the transaction until log space

is freed. Use **sp_dboption** to change this behavior for a particular database. **sp_dboption "abort tran on log full", true** causes Adaptive Server to roll back all transactions that have not yet been logged when the last-chance threshold is crossed.

Creating Additional Thresholds

- Each database can have up to 256 thresholds, including the last-chance threshold.
- When you add a threshold, it must be at least 2 times *@@thresh_hysteresis* pages from the closest threshold.

Creating Threshold Procedures

- Any user with **create procedure** permission can create a threshold procedure in a database. Usually, a System Administrator creates **sp_thresholdaction** in the *sybsystemprocs* database, and the Database Owners create threshold procedures in user databases.
- **sp_addthreshold** does not verify that the specified procedure exists. It is possible to add a threshold before creating the procedure it executes.
- **sp_addthreshold** checks to ensure that the user adding the threshold procedure has been directly granted the “sa_role”. All system roles active when the threshold procedure is created are entered in *systhresholds* as valid roles for the user writing the procedure. However, only directly granted system roles are activated when the threshold fires. Indirectly granted system roles and user-defined roles are not activated.
- Adaptive Server passes four parameters to a threshold procedure:
  - *@dbname*, *varchar*(30), which identifies the database
  - *@segmentname*, *varchar*(30), which identifies the segment
  - *@space_left*, *int*, which indicates the number of free pages associated with the threshold
  - *@status*, *int*, which has a value of 1 for last-chance thresholds and 0 for other thresholds

  These parameters are passed by position rather than by name; your threshold procedure can use other names for them, but it must declare them in the order shown and with the correct datatypes.

- It is not necessary to create a different procedure for each threshold. To minimize maintenance, you can create a single threshold procedure in the *sybsystemprocs* database that is executed for all thresholds in Adaptive Server.
- Include **print** and **raiserror** statements in the threshold procedure to send output to the error log.

Executing Threshold Procedures

- Tasks initiated when a threshold is crossed execute as background tasks. These tasks do not have an associated terminal or user session. If you execute **sp_who** while these tasks are running, the *status* column shows "background".
- Adaptive Server executes the threshold procedure with the permissions the user had at the time he or she added the threshold, minus any permissions that have since been revoked.
- Each threshold procedure uses one user connection, for as long as it takes for the procedure to execute.

Changing or Deleting Thresholds

- Use **sp_helpthreshold** for information about existing thresholds.
- Use **sp_modifythreshold** to associate a threshold with a new threshold procedure, free-space value, or segment. (You cannot change the free-space value or segment name associated with the last-chance threshold.)

  Each time a user modifies a threshold, that user becomes the threshold owner. When the threshold is crossed, Adaptive Server executes the threshold with the permissions the owner had at the time he or she modified the threshold, minus any permissions that have since been revoked.
- Use **sp_dropthreshold** to drop a threshold from a segment.

Disabling Free-Space Accounting

- Use the **no free space acctg** option of **sp_dboption** to disable free-space accounting on non-log segments.
- You cannot disable free-space accounting on log segments.

◆ ***WARNING!***

System procedures cannot provide accurate information about space allocation when free-space accounting is disabled.

Creating Last-Chance Thresholds for Pre-Release 11.0 Databases

- Databases do not automatically acquire a last-chance threshold when upgraded from a release prior to 10.0. Use the **lct_admin** system function to create a last-chance threshold in a pre-10.0 database upgraded to the current release.
- Only databases that store their logs on a separate segment can have a last-chance threshold. Use **sp_logdevice** to move the transaction log to a separate device.

Messages

- `Adding threshold for segment 'segname' at 'pageno' pages.`

  The **sp_addthreshold** command succeeded.

- `Table 'systhresholds' does not exist in database 'dbname'--cannot add thresholds.`

  The *systhresholds* table is missing. This table is created when the database is created (or an upgrade to release 11.0 is performed), and must not be removed.

- `There is no segment named 'segname'.`

  Run **sp_helpsegment** to see a list of segment names.

- `This threshold is too close to one or more existing thresholds. Thresholds must be no closer than 128 pages to each other.`

  Execute **sp_helpthreshold** to see a list of existing thresholds and sizes.

- `A threshold at pageno pages is logically impossible for segment 'segname'. Choose a value between value1 and value2 pages.`

  A threshold must be at least 2 times *@@thresh_hysteresis* pages from the closest threshold.

- `This procedure can only affect thresholds in the current database. Say 'use database_name' then run this procedure again.`

sp_addthreshold can create thresholds only in the current database. Issue the **use** *database_name* command to open the database in which you want to add a threshold, or qualify **sp_addthreshold** with the database name as shown in example 3.

Permissions

Only the Database Owner or a System Administrator can execute **sp_addthreshold.**

Tables Used

master.dbo.sysusages, *sysobjects*, *syssegments*, *systhresholds*

See Also

Commands	**create procedure**, **dump transaction**
Functions	**lct_admin**
System procedures	**sp_dboption**, **sp_dropthreshold**, **sp_helpthreshold**, **sp_modifythreshold**, **sp_thresholdaction**

sp_add_time_range

Function

Adds a named time range to an Adaptive Server.

Syntax

```
sp_add_time_range name, startday, endday,
   starttime, endtime
```

Parameters

name – is the name of the time range. Time range names must be 30 characters or fewer. The name cannot already exist in the *systimeranges* system table of the *master* database.

startday – is the day of the week on which the time range begins. This must be the full weekday name for the default server language, as stored in the *syslanguages* system table of the *master* database.

endday – is the day of the week on which the time range ends. This must be the full weekday name for the default server language, as stored in the *syslanguages* system table of the *master* database. The *endday* can fall either earlier or later in the week than the *startday* or can be the same day as the *startday*.

starttime – is the time of day when the time range begins. Specify the *starttime* in terms of a 24-hour clock, with a value between "00:00" (midnight) and "23:59" (11:59 p.m.). Use the following form:

```
"HH:MM"
```

endtime – is the time of day when the time range ends. Specify the *endtime* in terms of a 24-hour clock, with a value between "00:00" (midnight) and "23:59" (11:59 p.m.). Use the following form:

```
"HH:MM"
```

➤ ***Note***

To create a time range that spans the entire day, specify both a start time and an end time of "00:00".

The *endtime* must occur later in the day than the *starttime*, unless *endtime* is "00:00".

Examples

1. **`sp_add_time_range business_hours, monday, Friday, "09:00", "17:00"`**

   Creates the *business_hours* time range, which is active Monday through Friday, from 9:00 a.m. to 5:00 p.m.

2. **`sp_add_time_range before_hours, Monday, Friday, "00:00", "09:00"`**

   `sp_add_time_range after_hours, Monday, Friday, "18:00", "00:00"`

   Creates two time ranges, *before_hours* and *after_hours*, that, together, span all non-business hours Monday through Friday. The *before_hours* time range covers the period from 12:00 midnight to 9:00 a.m., Monday through Friday. The *after_hours* time range covers the period from 6:00 p.m. through 12:00 midnight, Monday through Friday.

3. **`sp_add_time_range weekends, Saturday, Sunday, "00:00", "00:00"`**

   Creates the *weekends* time range, which is 12:00 midnight Saturday to 12:00 midnight Sunday.

4. **`sp_add_time_range Fri_thru_Mon, Friday, Monday, "09:00", "17:00"`**

   Creates the *Fri_thru_Mon* time range, which is 9:00 a.m. to 5:00 p.m., Friday, Saturday, Sunday, and Monday.

5. **`sp_add_time_range Wednesday_night, Wednesday, Wednesday, "17:00", "00:00"`**

   Creates the *Wednesday_night* time range, which is Wednesday from 5:00 p.m. to 12:00 midnight.

Comments

- Adaptive Server includes one named time range, the “at all times” time range. This time range covers all times, from the first day through the last of the week, from 00:00 through 23:59. It cannot be modified or deleted.
- Adaptive Server generates a unique ID number for each named time range and inserts it into the *systimeranges* system table,
- When storing a time range in the *systimeranges* system table, Adaptive Server converts its *startday* and *endday* values into integers. For servers with a default language of us_english, the week begins on Monday (day 1) and ends on Sunday (day 7).

- It is possible to create a time range that overlaps with one or more other time ranges.
- Range days are contiguous, so the days of the week can wrap around the end to the beginning of the week. In other words, Sunday and Monday are contiguous days, as are Tuesday and Wednesday.
- The active time ranges are bound to a session at the beginning of each query batch. A change in the server's active time ranges due to a change in actual time has no effect on a session during the processing of a query batch. In other words, if a resource limit restricts a query batch during a given time range but a query batch begins before that time range becomes active, the query batch that is already running is not affected by the resource limit.
- The addition, modification, and deletion of time ranges using the system procedures does not affect the active time ranges for sessions currently in progress.
- If a resource limit has a transaction as its scope, and a change occurs in the server's active time ranges while a transaction is running, the newly active time range does not affect the transaction currently in progress.
- Changes to a resource limit that has a transaction as its scope does not affect any transactions currently in progress.
- For more information on time ranges, see Chapter 12, "Limiting Access to Server Resources," in the *System Administration Guide*.

Messages

- `Can't run sp_add_time_range from within a transaction.`

  sp_add_time_range modifies system tables, so it cannot be executed from within a transaction.

- `Ending day must be non-NULL.`

  You must specify an *endday* for the time range.

- `Ending time must be later in the day than starting time.`

  The *starttime* cannot be later than the *endtime*.

- `Ending time must be non-NULL.`

  You must specify an *endtime* for the time range.

- `New time range created. ID number is` *`rangeid.`*

  The time range was successfully added to the server.

- `Only the System Administrator (SA) may execute this procedure.`

  You must be a System Administrator to use **sp_add_time_range**.

- `Range` *`rangename`* `already exists.`

  A time range with this name already exists in the *systimeranges* system table in the *master* database.

- `Starting day must be non-NULL.`

  You must specify a *startday* for the time range.

- `Starting time must be non-NULL.`

  You must specify a *starttime* for the time range.

- `Timerange name must be non-NULL.`

  You must specify a name for the time range.

- `Unknown endday` *`endday`*`.`

  You must specify the full name of the day of the week, as stored in the *syslanguages* system table in the *master* database.

- `Unknown ending time value` *`endtime`*`.`

  endtime must be a valid time between 00:00 and 23:59.

- `Unknown startday` *`startday`*`.`

  You must specify the full name of the day of the week, as stored in the *syslanguages* system table in the *master* database.

- `Unknown starting time value` *`starttime`*`.`

  starttime must be a valid time between 00:00 and 23:59.

Permissions

Only a System Administrator can execute **sp_add_time_range**.

Tables Used

master..systimeranges, master..syslanguages

See Also

System procedures	**sp_add_resource_limit**, **sp_drop_time_range**, **sp_modify_time_range**

sp_addtype

Function

Creates a user-defined datatype.

Syntax

```
sp_addtype typename,
   phystype [(length) | (precision [, scale])]
   [, "identity" | nulltype]
```

Parameters

typename – is the name of the user-defined datatype. Type names must conform to the rules for identifiers and must be unique in each database.

phystype – is the physical or Adaptive Server-supplied datatype on which to base the user-defined datatype. You can specify any Adaptive Server datatype except *timestamp*.

The *char*, *varchar*, *nchar*, *nvarchar*, *binary*, and *varbinary* datatypes expect a *length* in parentheses. If you do not supply one, Adaptive Server uses the default length of 1 character.

The *float* datatype expects a binary *precision* in parentheses. If you do not supply one, Adaptive Server uses the default precision for your platform.

The *numeric* and *decimal* datatypes expect a decimal *precision* and *scale*, in parentheses and separated by a comma. If you do not supply them, Adaptive Server uses a default precision of 18 and a scale of 0.

Enclose physical types that include punctuation, such as parentheses or commas, within single or double quotes.

identity – indicates that the user-defined datatype has the IDENTITY property. Enclose the **identity** keyword within single or double quotes. You can specify the IDENTITY property only for *numeric* datatypes with a scale of 0.

IDENTITY columns store sequential numbers, such as invoice numbers or employee numbers, that are generated by Adaptive Server. The value of the IDENTITY column uniquely identifies each row in a table. IDENTITY columns are not updatable and do not allow null values.

nulltype – indicates how the user-defined datatype handles null value entries. Acceptable values for this parameter are **null**, **NULL**, **nonull**, **NONULL**, **"not null"**, and **"NOT NULL"**. Any *nulltype* that includes a blank space must be enclosed in single or double quotes.

If you omit both the IDENTITY property and the *nulltype*, Adaptive Server creates the datatype using the null mode defined for the database. By default, datatypes for which no *nulltype* is specified are created NOT NULL (that is, null values are not allowed and explicit entries are required). For compliance to the SQL standards, use the **sp_dboption** system procedure to set the **allow nulls by default** option to **true**. This changes the database's null mode to NULL.

Examples

1. **`sp_addtype ssn, "varchar(11)"`**

   Creates a user-defined datatype called *ssn* to be used for columns that hold social security numbers. Since the *nulltype* parameter is not specified, Adaptive Server creates the datatype using the database's default null mode. Notice that *varchar*(11) is enclosed in quotation marks, because it contains punctuation (parentheses).

2. **`sp_addtype birthday, "datetime", null`**

   Creates a user-defined datatype called *birthday* that allows null values.

3. **`sp_addtype temp52, "numeric(5,2)"`**

   Creates a user-defined datatype called *temp52* used to store temperatures of up to 5 significant digits with 2 places to the right of the decimal point.

4. **`sp_addtype "row_id", "numeric(10,0)", "identity"`**

   Creates a user-defined datatype called *row_id* with the IDENTITY property, to be used as a unique row identifier. Columns created with this datatype store system-generated values of up to 10 digits in length.

5. **`sp_addtype systype, sysname`**

   Creates a user-defined datatype with an underlying type of *sysname*. Although you cannot use the *sysname* datatype in a **create table**, **alter table**, or **create procedure** statement, you can use a user-defined datatype that is based on *sysname*.

Comments

- **sp_addtype** creates a user-defined datatype and adds it to the *systypes* system table. Once a user-defined datatype is created, you can use it in **create table** and **alter table** statements and bind defaults and rules to it.
- Build each user-defined datatype in terms of one of the Adaptive Server-supplied datatypes, specifying the length or the precision and scale, as appropriate. You cannot override the length, precision, or scale in a **create table** or **alter table** statement.
- A user-defined datatype name must be unique in the database, but user-defined datatypes with different names can have the same definitions.
- If *nchar* or *nvarchar* is specified as the *phystype*, the maximum length of columns created with the new type is the length specified in **sp_addtype** multiplied by the value of *@@ncharsize* at the time the type was added.
- Each system type has a **hierarchy**, stored in the *systypes* system table. User-defined datatypes have the same datatype hierarchy as the physical types on which they are based. In a mixed-mode expression, all types are converted to a common type, the type with the lowest hierarchy.

  Use the following query to list the hierarchy for each system-supplied and user-defined type in your database:

  ```
  select name, hierarchy
  from systypes
  order by hierarchy
  ```

Types with the IDENTITY Property

- If a user-defined datatype is defined with the IDENTITY property, all columns created from it are IDENTITY columns. You can specify either IDENTITY or NOT NULL—or neither one—in the **create** or **alter table** statement. Following are three different ways to create an IDENTITY column from a user-defined datatype with the IDENTITY property:

  ```
  create table new_table (id_col IdentType)
  create table new_table (id_col IdentType identity)
  create table new_table (id_col IdentType not null)
  ```

- When you create a column with the **create table** or **alter table** statement, you can override the null type specified with the **sp_addtype** system procedure:
  - Types specified as NOT NULL can be used to create NULL or IDENTITY columns.
  - Types specified as NULL can be used to create NOT NULL columns, but not to create IDENTITY columns.

➤ ***Note***

If you try to create a null column from an IDENTITY type, the **create** or **alter table** statement fails.

Messages

- `A type with the specified name already exists.`

  Choose a different *typename*.
- `Illegal length specified–must be between 1 and 255.`

  The length of a datatype must be between 1 and 255.
- `Illegal precision specified -- must be between 1 and 38.`

  The precision of a *numeric* or *decimal* datatype must be between 1 and 38.
- `Illegal precision specified -- must be between 1 and 48.`

  The precision of a *float* or *double precision* datatype must be between 1 and 48.
- `Illegal scale specified -- must be less than precision.`

  The scale of a *numeric* or *decimal* datatype must be between 0 and the datatype's precision.
- `Physical datatype does not allow nulls.`

  You tried to allow null values with the *bit* datatype, which does not allow null values.
- `Physical datatype does not exist.`

  The *phystype* is not an Adaptive Server datatype.

- `Physical type is fixed length. You cannot specify the length.`

  The physical datatypes that take length specifications are *char*, *nchar*, *varchar*, *nvarchar*, *binary*, and *varbinary*. You cannot change the fixed lengths of other physical datatypes.

- `Type added.`

  The **sp_addtype** command succeeded. You created a user-defined datatype that can be used in **create table** statements or to bind rules and defaults.

- `'typename' is not a valid type name.`

  typename must conform to the rules for identifiers and be unique for each owner in each database.

- `User-defined datatypes based on the 'timestamp' datatype are not allowed.`

  The *timestamp* datatype is based on *varbinary*(8), which you can use instead.

- `Usage: sp_addtype name, 'datatype' [, null | nonull | identity]`

  The third parameter can specify either a null type ("null", "NULL", "nonull", "NONULL", not null", or "NOT NULL") or the IDENTITY property.

- `User types with the identity property must be numeric with a scale of 0.`

- `You must specify a length with this physical type.`

  You used a *phystype—char, nchar, varchar, nvarchar, binary,* or *varbinary*—that requires a length. For example, "char(10)" is acceptable, but "char" is not.

Permissions

Any user can execute **sp_addtype**.

Tables Used

master.dbo.spt_values, *master.dbo.sysdatabases*, *sysobjects*, *systypes*

See Also

Commands	**create default**, **create rule**, **create table**
Datatypes	User-Defined Datatypes
System procedures	**sp_bindefault**, **sp_bindrule**, **sp_dboption**, **sp_droptype**, **sp_rename**, **sp_unbindefault**, **sp_unbindrule**

sp_addumpdevice

Function

Adds a dump device to Adaptive Server.

Syntax

```
sp_addumpdevice {"tape" | "disk"}, logicalname,
   physicalname [, tapesize]
```

Parameters

"tape" – for tape drives. Enclose **tape** in quotes.

"disk" – is for a disk or a file device. Enclose **disk** in quotes.

logicalname – is the "logical" dump device name. It must be a valid identifier. Once you add a dump device to *sysdevices*, you can specify its logical name in the **load** and **dump** commands.

physicalname – is the physical name of the device. You can specify either an absolute path name or a relative path name. During dumps and loads, the Backup Server resolves relative path names by looking in Adaptive Server's current working directory. Enclose names containing non-alphanumeric characters in quotation marks. For UNIX platforms, specify a non-rewinding tape device name.

tapesize – is the capacity of the tape dump device, specified in megabytes. OpenVMS systems ignore the *tapesize* parameter if specified. Other platforms require this parameter for tape devices but ignore it for disk devices. The *tapesize* should be at least five database pages (each page requires 2048 bytes). Sybase recommends that you specify a capacity that is slightly below the rated capacity for your device.

Examples

1. ```
   sp_addumpdevice "tape", mytapedump, "/dev/nrmt8",
   40
   ```

   Adds a 40MB tape device. Dump and load commands can reference the device by its physical name, */dev/nrmt8*, or its logical name, *mytapedump*.
   ```

2. **sp_addumpdevice "disk", mydiskdump, "/dev/rxy1d/dump.dat"**

   Adds a disk device named *mydiskdump*. Specify an absolute or relative path name and a file name.

Comments

- **sp_addumpdevice** adds a dump device to the *master.dbo.sysdevices* table. Tape devices are assigned a *cntrltype* of 3; disk devices are assigned a *cntrltype* of 2.
- To use an operating system file as a dump device, specify a device of type **disk** and an absolute or relative path name for the *physicalname*. Omit the *tapesize* parameter. If you specify a relative path name, dumps are made to—or loaded from—the current Adaptive Server working directory at the time the dump or load command executes.
- Ownership and permission problems can interfere with the use of disk or file dump devices. **sp_addumpdevice** adds the device to the *sysdevices* table, but does not guarantee that you can create a file as a dump device or that users can dump to a particular device.
- The **with capacity** = *megabytes* clause of the **dump database** and **dump transaction** commands can override the *tapesize* specified with **sp_addumpdevice**. On platforms that do not reliably detect the end-of-tape marker, the Backup Server issues a volume change request after the specified number of megabytes have been dumped.
- When a dump device fails, use **sp_dropdevice** to drop it from *sysdevices*. After replacing the device, use **sp_addumpdevice** to associate the logical device name with the new physical device. This avoids updating backup scripts and threshold procedures each time a dump device fails.
- To add database devices to *sysdevices*, use the **disk init** command.

Messages

- `Can't run sp_addumpdevice from within a transaction.`

  sp_addumpdevice modifies the system table *master.dbo.sysdevices*, so it cannot be run within a transaction.

- `'logicalname' is not a valid name.`

  The value for *logicalname* must conform to the rules for identifiers.

- *logicalname* may not be NULL.

  You must specify a device name.

- Device with same logical name already exists.

  All dump devices must have unique logical names. There is already a device with the name supplied for the *logicalname* parameter.

- 'Disk' device added.

  The disk dump device was added successfully.

- physicalname may not be NULL.

  You must specify a physical dump device name.

- Please specify media capacity in megabytes (1 MB minimum).

  You must specify a tape capacity, in megabytes, for tape devices. The minimum capacity is 1MB. There is no default.

- 'Tape' device added.

  The tape dump device was added successfully.

- WARNING: physical device name '*physicalname*' is not unique.

  You attempted to create a new dump device that has the same physical name as an existing dump device.

- WARNING: specified size parameter is not used for the disk device type.

- Unknown device type. Use 'disk' or 'tape'.

  The value supplied for the first parameter is not a known device type.

Permissions

Only a System Administrator can execute **sp_addumpdevice**.

Tables Used

master.dbo.sysdevices, *sysobjects*

See Also

Commands	**disk init**, **dump database**, **dump transaction**, **load database**, **load transaction**
System procedures	**sp_dropdevice**, **sp_helpdevice**

sp_adduser

Function

Adds a new user to the current database.

Syntax

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

Parameters

loginame – is the user's name in *master.dbo.syslogins*.

name_in_db – is a new name for the user in the current database.

grpname – adds the user to an existing group in the database.

Examples

1. `sp_adduser margaret`

   Adds "margaret" to the database. Her database user name is the same as her Adaptive Server login name, and she belongs to the default group, "public".

2. `sp_adduser haroldq, harold, fort_mudge`

   Adds "haroldq" to the database. When "haroldq" uses the current database, his name is "harold." He belongs to the *fort_mudge* group, as well as to the default group "public".

Comments

- The Database Owner executes **sp_adduser** to add a user name to the *sysusers* table of the current database, enabling the user to access the current database under his or her own name.
- Specifying a *name_in_db* parameter gives the new user a name in the database that is different from his or her login name in Adaptive Server. The ability to assign a user a different name is provided as a convenience. It is not an alias, as provided by **sp_addalias**, since it is not mapped to the identity and privileges of another user.
- A user and a group cannot have the same name.
- A user can be a member of only one group other than the default group, "public". Every user is a member of the default group, "public". Use **sp_changegroup** to change a user's group.

- In order to access a database, a user must either be listed in *sysusers* (with **sp_adduser**) or mapped to another user in *sysalternates* (with **sp_addalias**), or there must be a "guest" entry in *sysusers*.

Messages

- `A user with the same name already exists in the database.`

  The *name_in_db* is already a user in the database. Choose another name.

- `All user ids have been assigned.`

  The database has reached the maximum number of user IDs.

- `'name_in_db' is not a valid name.`

  The *name_in_db* specified does not follow the rules for identifiers.

- `New user added.`

  The **sp_adduser** command succeeded. The user is now known in the current database.

- `No group with the specified name exists.`

  The group name does not exist in this database. Either omit the *grpname* parameter or create the group with **sp_addgroup**.

- `No login with the specified name exists.`

  The *loginame* is unknown to Adaptive Server. Each user must have a login on Adaptive Server before being added to a database.

- `User already has a login under a different name.`

  The user with the *loginame* you specified is listed in the current database's *sysusers* table with a name that is different from the one you specified as the *name_in_db* parameter.

- `User already has alias access to the database.`

  The *loginame* is already known to the database by an alias. To add the user, drop the alias with **sp_dropalias**, and then execute **sp_adduser** again.

Permissions

Only the Database Owner, a System Administrator, or a System Security Officer can execute **sp_adduser**.

Tables Used

master.dbo.syslogins, *master.dbo.syssrvroles*, *sysalternates*, *sysobjects*, *sysusers*

See Also

Commands	**grant**, **revoke**, **use**
System procedures	**sp_addalias**, **sp_addgroup**, **sp_changegroup**, **sp_dropalias**, **sp_dropgroup**, **sp_helpuser**

sp_altermessage

Function

Enables and disables the logging of a system-defined or user-defined message in the Adaptive Server error log.

Syntax

sp_altermessage *message_id, parameter, parameter_value*

Parameters

message_id – is the message number of the message to be altered. This is the number of the message as it is recorded in the *error* column in the *sysmessages* or *sysusermessages* system table.

parameter – is the message parameter to be altered. The maximum length is 255 bytes. The only valid parameter is **with_log**.

parameter_value – is the new value for the parameter specified in *parameter*. The maximum length is 255 bytes. Values are **true** and **false**.

Examples

1. `sp_altermessage 2000, 'with_log', 'TRUE'`

Specifies that message number 2000 in *sysmessages* should be logged in the Adaptive Server error log and also in the Windows NT Event Log (if logging is enabled).

Comments

- If the *parameter_value* is **true**, the specified message is always logged. If it is **false**, the default logging behavior is used; the message may or may not be logged, depending on the severity of the error and other factors. Setting the *parameter_value* to **false** produces the same behavior that would occur if **sp_altermessage** had not been called.
- On Windows NT servers, **sp_altermessage** also enables and disables logging in the Windows NT Event Log.

Messages

- *Message_id* does not exist.

  The message number you specified is not recorded in *sysmessages* or *sysusermessages*.

- Only the System Administrator (SA) or the database Owner (dbo) may execute this stored procedure.

  You do not have the correct permissions to execute **sp_altermessage**.

- The only valid parameter value is 'with_log'.

  Change the *parameter* to **with_log**.

- The only valid parameter_value values are TRUE or FALSE.

  Change the *parameter_value* to **true** or **false**.

Permissions

Only the Database Owner or System Administrator can execute **sp_altermessage**.

Tables Used

sysmessages, *sysusermessages*

See Also

System procedures	**sp_addmessage**, **sp_dropmessage**

sp_audit

Function

Allows a System Security Officer to configure auditing options.

Syntax

```
sp_audit option, login_name, object_name [,setting]
```

Parameters

option – is the name of the auditing option to set. Table 3-5 lists the valid auditing options.

Table 3-5: Auditing options

Option	Description
adhoc	Allows users to use **sp_addauditrecord** to add their own user-defined audit records to the audit trail
all	Audits all actions performed by a particular user or by users with a particular role **Note:** Specifying all actions does not affect whether users can add ad hoc audit records.
alter	Audits the execution of the **alter table** or **alter database** commands
bcp	Audits the execution of the **bcp in** utility
bind	Audits the execution of **sp_bindefault**, **sp_bindmsg**, and **sp_bindrule** system procedures
cmdtext	Audits the entry of all text entered by the user while logged into the server.
create	Audits the creation of database objects
dbaccess	Audits access to the current database from another database
dbcc	Audits the execution of **dbcc**
delete	Audits the deletion of rows from a table or view
disk	Audits the execution of **disk init**, **disk refit**, **disk reinit**, **disk mirror**, **disk unmirror**, and **disk remirror**
drop	Audits the dropping of database objects
dump	Audits the execution of **dump database** or **dump transaction** commands
errors	Audits errors, whether fatal or not
exec_procedure	Audits the execution of a stored procedure

Table 3-5: Auditing options (continued)

Option	Description
exec_trigger	Audits the execution of a trigger
func_dbaccess	Audits access to a database via a Transact-SQL function
func_obj_access	Audits access to a database object via a Transact-SQL function
grant	Audits the execution of the **grant** command
insert	Audits the insertion of rows into a table or view
load	Audits the execution of the **load database** or **load transaction** commands
login	Audits all login attempts into Adaptive Server
logout	Audits all logout attempts from Adaptive Server
reference	Audits references between tables.
revoke	Audits the execution of the **revoke** command
rpc	Audits the execution of remote procedure calls
security	Audits the following security-relevant events: • Starting up or shutting down the server • Activating or deactivating a role • Issuing any of the following commands: **connect** **kill** **online database** **set proxy** **set session authorization** **sp_configure** • Using any of the following functions: **valid_user** **proc_role** (from within a system procedure) • Regenerating the SSO passwords
select	Audits the execution of the **select** command
setuser	Audits the execution of the **setuser** command
table_access	Audits access to any table by a specific user
truncate	Audits the execution of the **truncate table** command
unbind	Audits the execution of the **sp_unbindrule**, **sp_unbindmsg**, and **sp_unbindefault** system procedures

Table 3-5: Auditing options (continued)

Option	Description
update	Audits updates to rows in a table or view
view_access	Audits access to any view by a specific user

login_name – is the name of a specific login to be audited. To audit all logins, specify **all** for the *option* parameter. If you specify **all**, you can set the *login_name* parameter to a specific system role to audit all actions by users with that system role active. You cannot specify a user-defined role for *login_name*. For more information on the *login_name* values that are valid with each *option* value, see "Auditing Options: Their Types and Requirements" later in the **sp_audit** discussion.

object_name – is the name of the object to be audited. Valid values, depending on the value you specified for *option*, are:

- The object name, including the owner's name if you do not own the object. For example, to audit a table named *inventory* that is owned by Joe, you would specify *joe.inventory* for *object_name*.
- **all** for all objects.
- **default table**, **default view**, **default procedure**, or **default trigger** to audit access to any new table, view, procedure, or trigger.

  default table and **default view** are valid values for *object_name* when you specify **delete**, **insert**, **select**, or **update** for the *option* parameter. **default procedure** is valid when you specify the **exec_procedure** option. **default trigger** is valid when you specify the **exec_trigger** option.

For more information about the *object_name* values that are valid with each *option* value, see "Auditing Options: Their Types and Requirements" later in the **sp_audit** discussion.

setting – is the level of auditing. If you do not specify a value for *setting*, Adaptive Server displays the current auditing setting for

the option. Valid values for the *setting* parameter are described in the following table:

setting value	Description
on	Activates auditing for the specified option. Adaptive Server generates audit records for events controlled by this option, whether the event passes or fails permission checks.
off	Deactivates auditing for the specified option.
pass	Activates auditing for events that pass permission checks.
fail	Deactivates auditing for events that fail permission checks.

If you specify **pass** for an option and later specify **fail** for the same option, or vice versa, the result is equivalent to specifying **on**. Adaptive Server generates audit records regardless of whether events pass or fail permission checks. Settings of **on** or **off** apply to all auditing options. Settings of **pass** and **fail** apply to all options except **errors** and **adhoc**. For these options, only **on** or **off** applies. The initial, default value of all options is **off**.

Examples

1. `sp_audit "security", "all", "all", "on"`

   Initiates auditing for security-relevant events. Both successful and failed events are audited.

2. `sp_audit "security", "all", "all"`

   Displays the setting of the **security** auditing option.

3. `sp_audit "create", "all", master, "on"`

   Initiates auditing for the creation of objects in the *master* database, including **create database**.

4. `sp_audit "create", "all", db1, "on"`

   Initiates auditing for the creation of all objects in the *db1* database.

5. `sp_audit "all", "sa_role", "all", "fail"`

   Initiates auditing for all failed executions by a System Administrator.

6. `sp_audit "update", "all", "default table", "on"`

Initiates auditing for all updates to future tables in the current database. For example, if the current database is *utility*, all new tables created in *utility* will be audited for updates. The auditing for existing tables is not affected.

Comments

- **sp_audit** determines what will be audited when auditing is enabled. No actual auditing takes place until you use **sp_configure** to set the **auditing** parameter to **on**. Then, all auditing options that have been configured with **sp_audit** take effect. See **sp_configure** for more information.
- If you are not the owner of the object being specified, qualify the *object_name* parameter value with the owner's name, in the following format:

  "*ownername.objname*"
- You cannot activate default auditing for the following options in the *tempdb* database:
  - **delete**
  - **insert**
  - **select**
  - **update**
  - **exec_procedure**
  - **exec_trigger**
- You can use threshold procedures to switch between audit tables so that no audit data is lost when an audit table fills up. For information about handling audit tables, see the *Security Administration Guide*.
- Table 3-6 lists the configuration parameters that control auditing.

Table 3-6: Configuration parameters that control auditing

Configuration Parameter	Effect
auditing	Enables or disables auditing for the server.
audit_queue_size	Establishes the size of the audit queue.
current_audit_table	Sets the current audit table. Adaptive Server writes all audit records to that table.
suspend_auditing_when_full	Controls the behavior of the audit process when an audit device becomes full.

The **auditing**, **current_audit_table**, and **suspend_auditing_when_full** configuration parameters are dynamic and take effect immediately. Because **audit_queue_size** affects memory allocation, the parameter is static and does not take effect until Adaptive Server is restarted.

See **sp_configure** and "Auditing" in the *Security Administration Guide* for more information about configuring Adaptive Server for auditing.

Auditing Options: Their Types and Requirements

- The values you can specify for the *login_name* and *object_name* parameters depend on the type of auditing option you specify:
  - Global options apply to commands that affect the entire server. Examples of global events are booting the server, disk commands, and allowing ad hoc, user-defined audit records. Option settings for global events are stored in the *sybsecurity..sysauditoptions* system table.
  - Database-specific options apply to a database. Examples of database-specific events are altering a database, bulk copy (**bcp in**) of data into a database, granting or revoking access to objects in a database, and creating objects in a database. Option settings for database-specific events are stored in the *master..sysdatabases* system table.
  - Object-specific options apply to a specific object. Examples of object-specific events are selecting, inserting, updating, or deleting rows of a particular table or view and the execution of a particular trigger or procedure. Option settings for object-specific events are stored in the *sysobjects* system table in the relevant database.
  - User-specific options apply to a specific user or system role. Examples of user-specific events are all accesses by a particular user to any table or view or all actions performed when a particular system role, such as **sa_role**, is active. Option settings for individual users are stored in *master..syslogins*. The settings for system roles are stored in *master..sysauditoptions*.
- Table 3-7 shows:
  - Valid values for the *option* and the type of each option – global, database-specific, object-specific, or user-specific
  - Valid values for the *login_name* and *object_name* parameters for each option

- The database to be in when you use **sp_audit** to set the auditing option
- The command or access that is audited when you set the option
- An example for each option. Some examples use parameter names in the invocation, such as *@option* = "adhoc", and some specify values for the options in the required order: *option*, *login_name*, *object_name*, *setting*.

The default value of all options is **off**. For information about audit records and event codes that result from using an auditing option, see "sysaudits_01, sysaudits_02...sysaudits_08" in Chapter 8, "System Tables."

Table 3-7: Auditing options, requirements, and examples

Option (Option Type)	*login_name*	*object_name*	Database to Be In To Set the Option	Command or Access Being Audited
adhoc (user-specific)	**all**	**all**	Any	This option sets ad hoc auditing on. It allows users to use **sp_addauditrecord**.
Example:	`sp_audit @option="adhoc", @login_name="all", @object_name="all", @setting = "on"` (Enables auditing for user-defined auditing records.)			
all (user-specific)	A login name or a role	**all**	Any	All actions of a particular user or by users with a particular role active.
Example	`sp_audit "all", "sa_role", "all", "on"` (Turns on auditing for all actions in which the **sa_role** is active.)			
alter (database-specific)	**all**	Name of the database to be audited	Any	**alter database**, **alter table**
Example	`sp_audit @option = "alter", @login_name = "all", @object_name = "master", @setting = "on"` (Turns on auditing for all executions of **alter database** and **alter table** in the *master* database.)			

Table 3-7: Auditing options, requirements, and examples (continued)

Option (Option Type)	*login_name*	*object_name*	Database to Be In To Set the Option	Command or Access Being Audited
bcp (database-specific)	**all**	Name of the database to be audited	Any	**bcp in**
Example	**sp_audit "bcp", "all", "pubs2"** (Returns the status of **bcp** auditing in the *pubs2* database. If you do not specify a value for *setting*, Adaptive Server returns the status of auditing for the option you specify.			
bind (database-specific)	**all**	Name of the database to be audited	Any	**sp_bindefault**, **sp_bindmsg**, **sp_bindrule**
Example	**sp_audit "bind", "all", "planning", "off"** (Turns off bind auditing for the *planning* database.)			
cmdtext (user-specific)	A login name or a role	**all**	Any	All actions of a particular user or by users with a particular role active.
Example	**sp_audit "cmdtext", "dbo", "off"** (Turns off cmdtext auditing for Database Owners.)			
create (database-specific)	**all**	Name of the database to be audited Specify *master* for *object_name* if you want to audit the **create database** command. You will also be auditing the creation of other objects in *master*.	Any	**create database**, **create table**, **create procedure**, **create trigger**, **create rule**, **create default**, **sp_addmessage**, **create view**
Example:	**sp_audit "create", "all", "planning", "pass"** (Turns on auditing of successful object creations in the *planning* database. The current status of auditing **create database** is not affected because you did not specify the *master* database.)			
dbaccess (database-specific)	**all**	Name of the database to be audited	Any	Any access to the database from another database
Example:	**sp_audit "dbaccess", "all", "project", "on"** (Audits all external accesses to the *project* database.)			

Table 3-7: Auditing options, requirements, and examples (continued)

Option (Option Type)	*login_name*	*object_name*	Database to Be In To Set the Option	Command or Access Being Audited
dbcc (global)	**all**	**all**	Any	**dbcc**
Example:	`sp_audit "dbcc", "all", "all", "on"`			
	(Audits all executions of the **dbcc** command.)			
delete (object-specific)	**all**	Name of a table or view, **default table**, or **default view**	The database of the table or view (except *tempdb*)	**delete** from a table, **delete** from a view
Example:	`sp_audit "delete", "all", "default table", "on"`			
	(Audits all delete actions for all future tables in the current database.)			
disk (global)	**all**	**all**	Any	**disk init, disk refit, disk reinit, disk mirror, disk unmirror, disk remirror**
Example:	`sp_audit "disk", "all", "all", "on"`			
	(Audits all disk actions for the server.)			
drop (database-specific)	**all**	Name of the database to be audited	Any	**drop database**, **drop table**, **drop procedure**, **drop trigger**, **drop rule**, **drop default**, **sp_dropmessage**, **drop view**
Example:	`sp_audit "drop", "all", "financial", "fail"`			
	(Audits all drop commands in the *financial* database that fail permission checks.)			
dump (database-specific)	**all**	Name of the database to be audited	Any	**dump database**, **dump transaction**
Example:	`sp_audit "dump", "all", "pubs2", "on"`			
	(Audits dump commands in the *pubs2* database.)			
errors (global)	**all**	**all**	Any	Fatal error, non-fatal error
Example:	`sp_audit "errors", "all", "all", "on"`			
	(Audits errors throughout the server.)			

Table 3-7: Auditing options, requirements, and examples (continued)

Option (Option Type)	*login_name*	*object_name*	Database to Be In To Set the Option	Command or Access Being Audited
exec_procedure (object-specific)	**all**	Name of the procedure or **default procedure**	The database of the procedure (except *tempdb*)	**execute**
Example:	**sp_audit "exec_procedure", "all", "default procedure", "off"** (Turns off automatic auditing of new procedures in the current database.)			
exec_trigger (object-specific)	**all**	Name of the trigger or **default trigger**	The database of the trigger (except *tempdb*)	Any command that fires the trigger
Example:	**sp_audit "exec_trigger", "all", "trig_fix_plan", "fail"** (Audits all failed executions of the **trig_fix_plan** trigger in the current database.)			
func_dbaccess (database-specific)	**all**	Name of the database	Any	Access to the database via Transact-SQL built-in functions
Example:	**sp_audit @option="func_dbaccess", @login_name="all", @object_name = "strategy", @setting = "on"** (Audits accesses to the strategy database via built-in functions.)			
func_obj_access (object-specific)	**all**	Name of the object	Any	Access to an object via Transact-SQL built-in functions
Example:	**sp_audit @option="func_obj_access", @login_name="all", @object_name = "customer", @setting = "on"** (Audits accesses to the *customer* table via built-in functions.)			
grant (database-specific)	**all**	Name of the database to be audited	Any	**grant**
Example:	**sp_audit @option="grant", @login_name="all", @object_name = "planning", @setting = "on"** (Audits all grants in the *planning* database.)			
insert (object-specific)	**all**	Name of a table or view, **default table**, or **default view**	The database of the object (except *tempdb*)	**insert** into a table, **insert** into a view
Example:	**sp_audit "insert", "all", "dpt_101_view", "on"** (Audits all inserts into the *dpt_101_view* view in the current database..)			

Table 3-7: Auditing options, requirements, and examples (continued)

Option (Option Type)	*login_name*	*object_name*	Database to Be In To Set the Option	Command or Access Being Audited
load (database-specific)	**all**	Name of the database to be audited	Any	**load database**, **load transaction**
Example:	**sp_audit "load", "all", "projects_db", "fail"** (Audits all failed executions of database and transaction loads in the *projects_db* database.)			
login (global)	**all**	**all**	Any	Any login to Adaptive Server
Example:	**sp_audit "login", "all", "all", "fail"** (Audits all failed attempts to log into the server.)			
logout (global)	**all**	**all**	Any	Any logout from Adaptive Server
Example:	**sp_audit "logout", "all", "all", "off"** (Turns off auditing of logouts from the server.)			
reference (object-specific)	**all**	Name of the table to be audited	Any	Creation of a reference between tables
Example:	**sp_audit "reference", "all", "titles", "off"** (Turns off auditing of the creation of references between the *titles* table and other tables.)			
revoke (database-specific)	**all**	Name of the database to be audited	Any	**revoke**
Example:	**sp_audit "revoke", "all", "payments_db", "off"** (Turns off auditing of the execution of **revoke** in the *payments_db* database.)			
rpc (global)	**all**	**all**	Any	Remote procedure calls (either in or out)
Example:	**sp_audit "rpc", "all", "all", "on"** (Audits all remote procedure calls out of or into the server.)			
security (global)	**all**	**all**	Any	Server-wide security-relevant events. See the "security" option in Table 3-5.
Example:	**sp_audit "security", "all", "all", "on"** (Audits server-wide security-relevant events in the server.)			

Table 3-7: Auditing options, requirements, and examples (continued)

Option (Option Type)	*login_name*	*object_name*	Database to Be In To Set the Option	Command or Access Being Audited
select (object-specific)	**all**	Name of a table or view, **default table**, or **default view**	The database of the object (except *tempdb*)	**select** from a table, **select** from a view
Example:	**sp_audit "select", "all", "customer", "fail"** (Audits all failed selects from the *customer* table in the current database.)			
setuser (database-specific)	**all**	**all**	Any	**setuser**
Example:	**sp_audit "setuser", "all", "projdb", "on"** (Audits all executions of **setuser** in the *projdb* database.)			
table_access (user-specific)	Login name	**all**	Any	**select**, **delete**, **update**, or **insert** access in a table
Example:	**sp_audit "table_access", "smithson", "all", "on"** (Audits all table accesses by the login named “smithson”.)			
truncate (database-specific)	**all**	Name of the database to be audited	Any	**truncate table**
Example:	**sp_audit "truncate", "all", "customer", "on"** (Audits all table truncations in the *customer* database.)			
unbind (database-specific)	**all**	Name of the database to be audited	Any	**sp_unbindefault**, **sp_unbindrule**, **sp_unbindmsg**
Example:	**sp_audit "unbind", "all", "master", "fail"** (Audits all failed attempts of unbinding in the *master* database.)			
update (object-specific)	**all**	Name of a table, view, **default table**, or **default view**	The database of the object (except *tempdb*)	**update** to a table, **update** to a view
Example:	**sp_audit "update", "all", "projects", "on"** (Audits all attempts by users to update the *projects* table in the current database.)			
view_access (user-specific)	Login name	**all**	Any	**select**, **delete**, **insert**, or **update** to a view
Example:	**sp_audit "view_access", "joe", "all", "off"** (Turns off view auditing of user “joe”.)			

Messages

- `Audit option has been changed and has taken effect immediately.`

  Indicates that the specified auditing option has been activated.

- `Can't run sp_audit from within a transaction.`

  sp_audit modifies system tables, so it cannot be run from within a transaction.

- `Error with spt_values table. This is a system error. Contact an SA-role user.`

  Adaptive Server could not find a value in the *spt_values* table. Contact a System Administrator.

- `Error updating the audit flags in the system catalogs. This is a system error. Contact an SA-role user.`

  Adaptive Server encountered an error. Contact a System Administrator.

- `Login name must be 'all'`

  The *option* you specified requires a *login_name* of **all**.

- *`login_name`* `is not a valid login name or role name`

  Check the spelling of *login_name*. Use **sp_helpuser** to list valid users. You cannot specify a user-defined role for *login_name*.

- *`login_name`* `is not a valid user name`

  Check the spelling of *login_name*. Use **sp_helpuser** to list valid users.

- *`object_name`* `is not in the current database`

  Adaptive Server could not find the specified object in the current database. Use **sp_help** to list objects, or issue **sp_audit** from the database where the object resides.

- `Object name must be 'all'`

  The *option* you specified requires an *object_name* of **all**.

- *`object_name`* `is not a valid database name.`

  Check the spelling of the database name. Use **sp_helpdb** to list databases.

- *`object_name`* `is not a valid object name.`

  Check the spelling of the object name. Use **sp_help** to list objects.

- *`option`* `is not a valid audit option.`

Check the spelling of the option name.

- `Server-wide auditing 'option' is 'setting'.`

  Indicates the setting of the auditing option.

Permissions

Only a System Security Officer can execute **sp_audit**.

Tables Used

master..sysdatabases, *sysobjects*, *sybsecurity..sysauditoptions*, *sybsecurity..sysaudits_01...sybsecurity..sysaudits_08*

See Also

System procedures	**sp_addauditrecord**, **sp_configure**

sp_autoconnect

(Component Integration Services only)

Function

Defines a passthrough connection to a remote server for a specific user, which allows the named user to enter passthrough mode automatically at login.

Syntax

```
sp_autoconnect server, {true|false}
   [, loginame]
```

Parameters

server – is the name of a server to which an automatic passthrough connection is made. *server* must be the name of a remote server already added by **sp_addserver**. This server cannot be the local server.

true | **false** – determines whether the automatic passthrough connection is enabled or disabled for *server*. **true** enables the automatic connection. **false** disables it.

loginame – specifies the name of the user for which automatic connection is required. If no *loginame* is supplied, the autoconnect status is modified for the current user.

Examples

1. `sp_autoconnect SYBASE, true`

   The current user is automatically connected to the server SYBASE the next time that user logs in. The user's connection is placed in passthrough mode.

2. `sp_autoconnect SYBASE, false, steve`

   Disables the autoconnect feature for the user "steve".

Comments

- **sp_autoconnect** defines a passthrough connection to a remote server for a specific user, which allows the named user to enter passthrough mode automatically at login.
- The **connect to** permission must be explicitly granted by the System Administrator.

- You can use **sp_autoconnect** only when Component Integration Services is installed and configured.
- Do not change the autoconnect status of the "sa" login account.
- Changing the autoconnect status does not occur immediately for users who are currently connected. They must disconnect from the local server and then reconnect before the change is made.
- Use **disconnect** to exit passthrough mode.

Messages

- `Login` *`loginame`* `does not exist in syslogins.`

  The *loginame* specification is not in the *syslogins* table. Add the login with **sp_addlogin**.

- `Only the System Administrator can modify another's autoconnect capability`

  You must be a System Administrator to change another user's autoconnect status.

- `Server name` *`server`* `not found in sysservers table.`

  Add the remote server with **sp_addserver**.

- `Unrecognized` *`flag`* `value; must be either 'true' or 'false'.`

  The second parameter for **sp_autoconnect** must be either **true** or **false**.

Permissions

Only a System Administrator can enable or disable another user's autoconnect status. The System Administrator must grant **connect to** permission to the login prior to executing **sp_autoconnect**.

Tables Used

sysservers, *syslogins*

See Also

Commands	**connect to...disconnect**, **grant**
System procedures	**sp_addlogin**, **sp_passthru**, **sp_remotesql**, **sp_addserver**

sp_bindcache

Function

Binds a database, table, index, *text* object, or *image* object to a data cache.

Syntax

```
sp_bindcache cachename, dbname
   [, [ownername.]tablename
   [, indexname | "text only"]]
```

Parameters

cachename – is the name of an active data cache.

dbname – is the name of the database to be bound to the cache or the name of the database containing the table, index, *text* or *image* object to be bound to the cache.

ownername – is the name of the table's owner. If the table is owned by "dbo", the owner name is optional.

tablename – is the name of the table to be bound to the cache, or the name of the table whose index, *text* object, or *image* object is to be bound to the cache.

indexname – is the name of the index to be bound to the cache.

text only – binds *text* or *image* objects to a cache. When this parameter is used, you cannot give an index name at the same time.

Examples

1. `sp_bindcache pub_cache, pubs2, titles`

   Binds the *titles* table to the cache named *pub_cache*.

2. `sp_bindcache pub_ix_cache, pubs2, titles, title_id_cix`

   Binds the clustered index *titles.title_id_cix* to the *pub_ix_cache*.

3. `sp_bindcache tempdb_cache, tempdb`

   Binds *tempdb* to the *tempdb_cache*.

4. `sp_bindcache logcache, pubs2, syslogs`

   Binds the *pubs2* transaction log, *syslogs*, to the cache named *logcache*.

5. **`sp_bindcache pub_cache, pubs2, au_pix, "text only"`**

   Binds the *image* chain for the *au_pix* table to the cache named *pub_cache*.

Comments

- A database or database object can be bound to only one cache. You can bind a database to one cache and bind individual tables, indexes, *text* objects, or *image* objects in the database to other caches. The database binding serves as the default binding for all objects in the database that have no other binding. The data cache hierarchy for a table or index is as follows:
  - If the object is bound to a cache, the object binding is used.
  - If the object is not bound to a cache, but the object's database is bound to a cache, the database binding is used.
  - If neither the object nor its database is bound to a cache, the default data cache is used.
- The cache and the object or database being bound to it must exist before you can execute **sp_bindcache**. Create a cache with **sp_cacheconfig** and restart Adaptive Server before binding objects to the cache.
- Cache bindings take effect immediately, and do not require a restart of the server. When you bind an object to a data cache:
  - Any pages for the object that are currently in memory are cleared.
  - When the object is used in queries, its pages are read into the bound cache.
- You can bind an index to a different cache than the table it references. If you bind a clustered index to a cache, the binding affects only the root and intermediate pages of the index. It does not affect the data pages (which are, by definition, the leaf pages of the index).
- To bind a database, you must be using the *master* database. To bind tables, indexes, *text* objects, or *image* objects, you must be using the database where the objects are stored.
- To bind any system tables in a database, you must be using the database and the database must be in single-user mode. Use the command:

  `sp_dboption` ***`db_name`*****`, "single user", true`**

  See **sp_dboption** for more information.

- You do not have to unbind objects or databases in order to bind them to a different cache. Issuing **sp_bindcache** on an object that is already bound drops the old binding and creates the new one.
- **sp_bindcache** needs to acquire an exclusive table lock when you are binding a table or its indexes to a cache so that no pages can be read while the binding is taking place. If a user holds locks on a table, and you issue **sp_bindcache** on that object, the task doing the binding sleeps until the locks are released.
- When you bind or unbind an object, all stored procedures that reference the object are recompiled the next time they are executed. When you change the binding for a database, all stored procedures that reference objects in the bound database are recompiled the next time they are executed.
- When you drop a table, index, or database, all associated cache bindings are dropped. If you re-create the table, index, or database, you must use **sp_bindcache** again if you want it bound to a cache.
- If a database or a database object is bound to a cache, and the cache is dropped, the cache bindings are marked invalid, but remain stored in the *sysattributes* system table(s). Warnings are printed in the error log when Adaptive Server is restarted. If a cache of the same name is created, the bindings become valid when Adaptive Server is restarted.
- The following procedures provide information about the bindings for their respective objects: **sp_helpdb** for databases, **sp_help** for tables, and **sp_helpindex** for indexes. **sp_helpcache** provides information about all objects bound to a particular cache.
- Use **sp_spaceused** to see the current size of tables and indexes, and **sp_estspace** to estimate the size of tables that you expect to grow. Use **sp_cacheconfig** to see information about cache size and status, and to configure and reconfigure caches.

Restrictions

- The *master* database, the system tables in *master*, and the indexes on the system tables in *master* cannot be bound to a cache. You can bind non-system tables from *master*, and their indexes, to caches.
- You cannot bind a database or an object to a cache if:
  - Isolation level 0 reads are active on the table

  - The task doing the binding currently has a cursor open on the table

- If a cache has the type **log only**, you can bind a *syslogs* table only to that cache. Use **sp_cacheconfig** to see a cache's type.

Messages

- `Can't run sp_bindcache from within a transaction.`

  You are currently in a transaction. You must roll back or commit the transaction before you can execute **sp_bindcache**.

- `Command Failed: Database '5' must be in single user mode to bind target object.`

  You tried to bind a system table to a cache. You must use **sp_dboption** to put the database in single-user mode before you can bind system tables (including *syslogs*) to a cache.

- `Individual tables in 'tempdb' cannot be bound to named caches. However, all of 'tempdb' may be bound.`

  All tables in *tempdb* are dropped when the server is restarted. You cannot bind individual tables in *tempdb* to a cache.

- `Only logs may be bound to this cache.`

  The cache has the type "log only". Only *syslogs* tables can be bound to caches with this type.

- `Specified named cache does not exist.`

  There is no cache with the name you specified. Use **sp_cacheconfig** with no parameters to see the names of existing caches.

- `Specified named cache is not active yet. The SQL Server must be rebooted to activate the named cache.`

  Adaptive Server has not been restarted since the cache was created. You must restart Adaptive Server to activate a cache after it is configured.

- `The 'master' database cannot be bound to a named cache.`

  You tried to bind *master* to a cache.

- `The target database does not exist.`

  The database name you specified does not exist. To see the names of all databases, execute **sp_helpdb**.

- `The target index does not exist.`

  The index name you specified does not exist. To see the names of indexes on a table, execute **sp_helpindex** *tablename*.

- `The target object does not exist.`

  The table name you specified does not exist. You must be using a database to bind any objects in that database. To see the names of tables in a database, execute **sp_help**.

- `You must be in Master to bind or unbind a database.`

  Database binding can take place only from the *master* database. Issue the command **use master**, and execute the command again.

Permissions

Only a System Administrator can execute **sp_bindcache**.

Tables Used

master..sysattributes, master..sysdatabases, sysindexes, sysobjects

See Also

System procedures	**sp_cacheconfig**, **sp_configure**, **sp_help**, **sp_helpcache**, **sp_helpdb**, **sp_helpindex**, **sp_poolconfig**, **sp_unbindcache**, **sp_unbindcache_all**

sp_bindefault

Function

Binds a user-defined default to a column or user-defined datatype.

Syntax

```
sp_bindefault defname, objname [, futureonly]
```

Parameters

defname – is the name of a default created with **create default** statements to bind to specific columns or user-defined datatypes.

objname – is the name of the table and column, or user-defined datatype, to which the default is to be bound. If the *objname* parameter is not of the form *"table.column"*, it is assumed to be a user-defined datatype. If the object name includes embedded blanks or punctuation, or is a reserved word, enclose it in quotation marks.

Existing columns of the user-defined datatype inherit the default *defname*, unless you specify **futureonly.**

futureonly – prevents existing columns of a user-defined datatype from acquiring the new default. This parameter is optional when you are binding a default to a user-defined datatype. It is never used to bind a default to a column.

Examples

1. `sp_bindefault today, "employees.startdate"`

   Assuming that a default named *today* has been defined in the current database with **create default**, this command binds it to the *startdate* column of the *employees* table. Each new row added to the *employees* table has the value of the *today* default in the *startdate* column, unless another value is supplied.

2. `sp_bindefault def_ssn, ssn`

   Assuming that a default named *def_ssn* and a user-defined datatype named *ssn* exist, this command binds *def_ssn* to *ssn*. The default is inherited by all columns that are assigned the user-defined datatype *ssn* when a table is created. Existing columns of type *ssn* also inherit the default *def_ssn*, unless you specify **futureonly** (which prevents existing columns of that user-defined datatype from inheriting the default), or unless the column's

default has previously been changed (in which case the changed default is maintained).

3. `sp_bindefault def_ssn, ssn, futureonly`

   Binds the default *def_ssn* to the user-defined datatype *ssn*. Because the **futureonly** parameter is included, no existing columns of type *ssn* are affected.

Comments

- You can create column defaults in two ways: by declaring the default as a column constraint in the **create table** or **alter table** statement or by creating the default using the **create default** statement and binding it to a column using **sp_bindefault**. Using **create default**, you can bind that default to more than one column in the database.
- You cannot bind a default to an Adaptive Server-supplied datatype.
- You cannot bind a default to a system table.
- Defaults bound to a column or user-defined datatype with the IDENTITY property have no effect on column values. Each time you insert a row into the table, Adaptive Server assigns the next sequential number to the IDENTITY column.
- If binding a default to a column, give the *objname* argument in the form "*table.column*". Any other format is assumed to be the name of a user-defined datatype.
- If a default already exists on a column, you must remove it before binding a new default. Use **sp_unbindefault** to remove defaults created with **sp_bindefault**. To remove defaults created with **create table** or **alter table**, use **alter table** to replace the default with NULL.
- Existing columns of the user-defined datatype inherit the new default unless you specify **futureonly**. New columns of the user-defined datatype always inherit the default. Binding a default to a user-defined datatype overrides defaults bound to columns of that type; to restore column bindings, unbind and rebind the column default.
- Statements that use a default cannot be in the same batch as their **sp_bindefault** statement.

Messages

- `Default and table or usertype must be in current database.`

  The *objname* parameter supplied with the procedure contained a reference to another database. Defaults can be bound to objects in the current database only.

- `Default bound to column.`

  The default was successfully bound to the specified column in the specified table.

- `Default bound to datatype.`

  The default was successfully bound to the specified user-defined datatype.

- `No such default exists. You must create the default first.`

  First, create the default in the current database with **create default**. Then, execute **sp_bindefault**.

- `The column already has a default. Bind disallowed.`

  Execute **sp_unbindefault** to unbind the existing default.

- `The new default has been bound to column(s) of the specified user datatype.`

  The command succeeded. Existing columns of the user-defined datatype specified now have the new default bound to them (unless their defaults were previously changed).

- `Usage: sp_bindefault defname, objname [, 'futureonly']`

  You incorrectly specified a parameter to **sp_bindefault**.

- `You cannot bind a declared default. The default must be created using create default.`

  First, create the default in the current database with **create default**. Then, execute **sp_bindefault**.

- `You can’t bind a default to a timestamp datatype column.`

  The value in a *timestamp* column represents an Adaptive Server-supplied sequence identifier. You cannot supply a default value for a timestamp.

- `You do not own a column of that name.`

  Only the owner of a table can bind a default to any of its columns. You are not the owner, or the object does not exist.

- `You do not own a datatype of that name.`

  Only the owner of a user-defined datatype can bind a default to that datatype. You are not the owner.

Permissions

Only the object owner can execute **sp_bindefault**.

Tables Used

syscolumns, *sysobjects*, *sysprocedures*, *systypes*

See Also

Commands	**create default**, **create table**, **drop default**
System procedures	**sp_unbindefault**

sp_bindexeclass

Function

Associates an execution class with a client application, login, or stored procedure.

Syntax

```
sp_bindexeclass "object_name", "object_type",
   "scope", "classname"
```

Parameters

object_name – is the name of the client application, login, or stored procedure to be associated with the execution class, *classname*.

object_type – identifies the type of *object_name*. Use **ap** for application, **lg** for login, or **pr** for stored procedure.

scope – is the name of a client application or login, or it can be NULL for **ap** and **lg** objects. It is the name of the stored procedure owner (user name) for objects. When the object with *object_name* interacts with the application or login, *classname* attributes apply for the scope you set.

classname – specifies the type of class to associate with *object_name*. Values are:

- *EC1*, *EC2*, or *EC3*
- The name of a user-defined execution class
- *ANYENGINE*

ExamplesAP

1. `sp_bindexeclass 'isql', 'ap', NULL, 'EC3'`

   This statement specifies that Transact-SQL applications will execute with *EC3* attributes for any login or application process (because the value of *scope* is NULL) that invokes **isql**, unless the login or application is bound to a higher execution class.

2. `sp_bindexeclass 'sa', 'lg', 'isql', 'EC1'`

   This statement specifies that when a login with the System Administrator role executes Transact-SQL applications, the login process executes with *EC1* attributes. If you have already executed the statement in the first example, then any other login

or client application that invokes **isql** will execute with *EC3* attributes.

1. `sp_bindexeclass 'my_proc', 'PR', 'kundu, 'EC3'`

   This statement assigns *EC3* attributes to the stored procedure named *my_proc* owned by user *kundu*.

Comments

- **sp_bindexeclass** associates an execution class with a client application, login, or stored procedure. Create execution classes with **sp_addexeclass**.
- When *scope* is NULL, *object_name* has no scope. *classname's* execution attributes apply to all of its interactions. For example, if *object_name* is an application name, the attributes apply to any login process that invokes the application. If *object_name* is a login name, the attributes apply to a particular login process for any application invoked by the login process.
- When binding a stored procedure to an execution class, you must use the name of the stored procedure owner (user name) for the *scope* parameter. This narrows the identity of a stored procedure when there are multiple invokations of it in the same database.
- Due to precedence and scoping rules, the execution class being bound may or may not have been in effect for the object called *object_name*. The object automatically binds itself to another execution class, depending on other binding specifications, precedence, and scoping rules. If no other binding is applicable, the object binds to the default execution class, *EC2*.
- Binding fails when you attempt to bind an active process to an engine group with no online engines.
- Adaptive Server creates a row in the *sysattributes* table containing the object ID and user ID in the row that stores data for the binding.
- A stored procedure must exist before it can be bound.
- Stored procedure bindings must be done in the database in which the stored procedure resides. Therefore, when binding system procedures, execute **sp_bindexeclass** from within the *sybsystemprocs* database.
- Only the "priority attribute" of the execution class is used when you bind the class to a stored procedure.

- The name of the owner of a stored procedure must be supplied as the *scope* parameter when you are binding a stored procedure to an execution class. This helps to uniquely identify a stored procedure when multiple stored procedures with the same name (but different owners) exist in the database.

Messages

- `Can't run sp_bindexeclass from within a transaction.`

  sp_bindexeclass modifies system tables, so it cannot be run within a transaction.

- `Failed to bind` *`object_name`* `to execution class` *`classname`*`. Check server errorlog for any additional information.`

  Binding the object to the execution class resulted in an error. Please check the error log.

- `No definition for classname` *`classname`* `exists`

  You must define the execution class before binding an object to it.

- `No login with the specified name` *`object_name`* `exists.`

  object_name is not a valid Adaptive Server login.

- *`object_name`* `cannot be NULL.`

  You must supply the name of the object to which the execution class will be bound.

- *`object_type`* `is not a valid object type for this stored procedure.`

  Use **PR** for *object_type* to bind an execution class to a stored procedure.

- *`object_type`* `is not a valid user of this database.`

  Use **sp_who** to check valid users for the current database.

- `Permission denied:` *`sybase_role`* `is required for binding with execution class 'EC0'.`

  Check that you have the proper security role.

- `Scope` *`scope`* `is not a valid name.`

  The scope supplied does not follow the rules for identifiers.

- `Stored procedure *object_name*, owned by *scope*, does not exist in this database.`

  Check the spelling of the stored procedure and the owner's name. Use **sp_help** to list objects and their owners.

- `Validation of execution class binding failed. Check server errorlog for any additional information.`

  The supplied parameter combination is invalid. Please check the error log.

Permissions

Only a System Administrator can execute **sp_bindexeclass**.

Tables Used

sysattributes, syslogins

See Also

System procedures	**sp_addengine, sp_addexeclass, sp_clearpsexe, sp_dropengine, sp_dropexeclass, sp_setpsexe, sp_showcontrolinfo, sp_showexeclass, sp_showpsexe, sp_unbindexeclass**

sp_bindmsg

Function

Binds a user message to a referential integrity constraint or check constraint.

Syntax

sp_bindmsg *constrname*, *msgid*

Parameters

constrname – is the name of the integrity constraint to which you are binding a message. Use the **constraint** clause of the **create table** command, or the **add constraint** clause of the **alter table** command to create and name constraints.

msgid – is the number of the user message to be bound to an integrity constraint. The message must exist in the *sysusermessages* table in the local database prior to calling **sp_bindmsg**.

Examples

1. sp_bindmsg positive_balance, 20100

Binds user message number 20100 to the *positive_balance* constraint.

Comments

- **sp_bindmsg** binds a user message to an integrity constraint by adding the message number to the constraint row in the *sysconstraints* table.
- Only one message can be bound to a constraint. To change the message for a constraint, just bind a new message. The new message number replaces the old message number in the *sysconstraints* table.
- You cannot bind a message to a unique constraint because a unique constraint does not have a constraint row in *sysconstraints* (a unique constraint is a unique index).
- Use the **sp_addmessage** procedure to insert user messages into the *sysusermessages* table.
- The **sp_getmessage** procedure retrieves message text from the *sysusermessages* table.

- **sp_help** *tablename* displays all constraint names declared on *tablename*.

Messages

- `Binding message failed unexpectedly. Please try again.`

  An error occurred while binding this message. Reissue the command.

- `Constraint name must be in 'current' database.`

  You can bind messages only to constraints that are defined in the current database.

- `Constraint name must belong to the current user.`

  You cannot bind a message to a constraint created by another user.

- `Message bound to constraint`

  You successfully bound the message to the constraint.

- `Message id must be a user defined message.`

  User-defined messages must have a number greater than 20,000. Only user-defined messages can be bound to constraints.

- `No such constraint exists. Please create the constraint first using CREATE/ALTER TABLE command.`

  Use **create table** or **alter table** to create the constraint before binding a message to it. You can see a list of all existing constraints on a table by using **sp_help** *tablename*.

- `No such message exists. Please create the message first using sp_addmessage.`

  The message must exist in the *sysusermessages* table before you can bind it to a constraint. Use **sp_addmessage** to create the message.

- `No such referential or check constraint exists. Please check whether the constraint name is correct.`

  You can see a list of all existing constraints on a table by using **sp_help** *tablename*.

Permissions

Only the object owner can execute **sp_bindmsg**.

Tables Used

sysconstraints, *sysobjects*, *sysusermessages*

See Also

Commands	**alter table**, **create table**
System procedures	**sp_addmessage**, **sp_getmessage**, **sp_unbindmsg**

sp_bindrule

Function

Binds a rule to a column or user-defined datatype.

Syntax

```
sp_bindrule rulename, objname [, futureonly]
```

Parameters

rulename – is the name of a rule. Create rules with **create rule** statements and bind rules to specific columns or user-defined datatypes with **sp_bindrule**.

objname – is the name of the table and column, or user-defined datatype, to which the rule is to be bound. If *objname* is not of the form *"table.column"*, it is assumed to be a user-defined datatype. If the object name has embedded blanks or punctuation, or is a reserved word, enclose it in quotation marks.

futureonly – prevents existing columns of a user-defined datatype from inheriting the new rule. This parameter is optional when you bind a rule to a user-defined datatype. It is meaningless when you bind a rule to a column.

Examples

1. `sp_bindrule today, "employees.startdate"`

   Assuming that a rule named *today* has been created in the current database with **create rule**, this command binds it to the *startdate* column of the *employees* table. When a row is added to *employees*, the data for the *startdate* column is checked against the rule *today*.

2. `sp_bindrule rule_ssn, ssn`

   Assuming the existence of a rule named *rule_ssn* and a user-defined datatype named *ssn*, this command binds *rule_ssn* to *ssn*. In a **create table** statement, columns of type *ssn* inherit the rule *rule_ssn*. Existing columns of type *ssn* also inherit the rule *rule_ssn*, unless *ssn*'s rule was previously changed (in which case the changed rule is maintained in the future only).

3. sp_bindrule rule_ssn, ssn, futureonly

The rule *rule_ssn* is bound to the user-defined datatype *ssn*, but no existing columns of type *ssn* are affected. **futureonly** prevents existing columns of type *ssn* from inheriting the rule.

Comments

- Create a rule using the **create rule** statement. Then execute **sp_bindrule** to bind it to a column or user-defined datatype in the current database.
- Rules are enforced when an **insert** is attempted, not when **sp_bindrule** is executed. You can bind a character rule to a column with an exact or approximate numeric datatype, even though such an **insert** is illegal.
- You cannot use **sp_bindrule** to bind a check constraint for a column in a **create table** statement.
- You cannot bind a rule to an Adaptive Server-supplied datatype or to a *text* or an *image* column.
- You cannot bind a rule to a system table.
- If you are binding to a column, the *objname* argument must be of the form *"table.column"*. Any other format is assumed to be the name of a user-defined datatype.
- Statements that use a rule cannot be in the same batch as their **sp_bindrule** statement.
- You can bind a rule to a column or user-defined datatype without unbinding an existing rule. Rules bound to columns always take precedence over rules bound to user-defined datatypes. Binding a rule to a column will replace a rule bound to the user-defined datatype of that column, but binding a rule to a datatype will not replace a rule bound to a column of that user-defined datatype. Table 3-8 indicates the precedence when binding rules to columns and user-defined datatypes where rules already exist:

Table 3-8: Precedence of new and old bound rules

	Old Rule Bound To:	
New Rule Bound To:	**User-Defined Datatype**	**Column**
user-defined datatype	Replaces old rule	No change
column	Replaces old rule	Replaces old rule

- Existing columns of the user-defined datatype inherit the new rule unless their rule was previously changed, or the value of the optional third parameter is **futureonly**. New columns of the user-defined datatype always inherit the rule.

Messages

- `No such rule exists. You must create the rule first.`

  First create the rule in the current database with **create rule**. Then execute **sp_bindrule**.

- `Rule and table or usertype must be in current database.`

  The *objname* parameter contained a reference to another database. Rules can only be bound to objects in the current database.

- `Rule bound to datatype.`

  The rule was successfully bound to the specified user-defined datatype.

- `Rule bound to table column.`

  The rule was successfully bound to the specified column in the specified table.

- `The new rule has been bound to column(s) of the specified user datatype.`

  Existing columns of the specified user-defined datatype now have the new rule bound to them (unless their rules were previously changed).

- `Usage: sp_bindrule rulename, objname [,futureonly]`

  Syntax summary. You incorrectly specified a parameter to **sp_bindrule**.

- `You can't bind a rule to a text, image, or timestamp datatype column.`

  The column you specified was a *text*, *image*, or *timestamp* column. Rules cannot be applied to *text*, *image*, or *timestamp* datatypes.

- `You can't bind a rule to a text, image, or timestamp datatype.`

  The datatype you specified was a *text*, *image*, or *timestamp* datatype. Rules cannot be applied to *text*, *image*, or *timestamp* datatypes.

- `You cannot bind a declared constraint. The rule must be created using create rule.`

  First create the rule in the current database with **create rule**. Then execute **sp_bindrule**.

- `You do not own a column of that name.`

  Only the owner of a table can bind a rule to any of its columns. You are not the owner, or the object does not exist.

- `You do not own a datatype of that name.`

  Only the owner of a user-defined datatype can bind a rule to it. You are not the owner.

Permissions

Only the object owner can execute **sp_bindrule**.

Tables Used

syscolumns, *sysconstraints*, *sysobjects*, *sysprocedures*, *systypes*

See Also

Commands	**create rule**, **drop rule**
System procedures	**sp_unbindrule**

sp_cacheconfig

Function

Creates, configures, reconfigures, and drops data caches, and provides information about them.

Syntax

```
sp_cacheconfig [cachename [ ,"cache_size[P|K|M|G]" ]
   [,logonly | mixed ] [,strict | relaxed ] ]
```

Parameters

cachename – is the name of the data cache to be created or configured. Cache names must be unique, and can be up to 30 characters long. A cache name does not have to be a valid Adaptive Server identifier, that is, it can contain spaces and other special characters.

cache_size – is the size of the data cache to be created or, if the cache already exists, the new size of the data cache. The minimum size of a cache is 512K. Size units can be specified with **P** for pages, **K** for kilobytes, **M** for megabytes, or **G** for gigabytes. The default is **K**. For megabytes and gigabytes, you can specify floating-point values.

logonly | mixed – specifies the type of cache.

strict | relaxed – specifies the cache replacement policy.

Examples

1. `sp_cacheconfig pub_cache, "10M"`

   Creates the data cache *pub_cache* with 10MB of space. All space is in the default 2K memory pool.

2. `sp_cacheconfig pub_cache`

   Reports the current configuration of *pub_cache* and any memory pools in the cache.

3. `sp_cacheconfig pub_cache, "0"`

   Drops *pub_cache* at the next start of Adaptive Server.

4. `sp_cacheconfig pub_log_cache, "2000K", logonly`

   Creates *pub_log_cache* and sets its type to **logonly** in a single step.

```
5. sp_cacheconfig pub_log_cache, "2000K"
   sp_cacheconfig pub_log_cache, logonly
```

The first command creates the cache *pub_log_cache* with the default type **mixed**. The second command changes its status to **logonly**. The resulting configuration is the same as that in example 4.

Comments

- Creating data caches divides Adaptive Server's single default data cache into smaller caches. You can then configure pools within the data cache to allow Adaptive Server to perform large I/O using **sp_poolconfig**. You can bind tables, indexes, databases, and *text* or *image* chains to a specific cache using **sp_bindcache**.
- When you first create a data cache:
  - All space is allocated to the 2K memory pool.
  - The default type is **mixed**.
- Figure 3-1 shows a data cache with two user-defined data caches configured and the following pools:
  - The default data cache with a 2K pool and a 16K pool
  - A user cache with a 2K pool and a 16K pool
  - A log cache with a 2K pool and a 4K pool

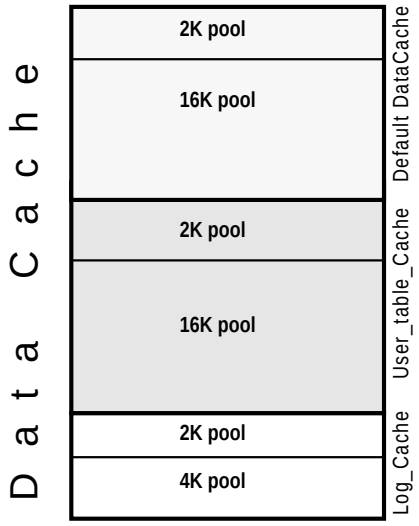

Figure 3-1: Data cache with default and user-defined caches

- Creating, dropping, and changing the size or cache policy of data caches requires a restart of Adaptive Server for the configuration to take effect. You cannot configure pools or bind objects to caches until the cache is active, that is, until the server has been restarted.

  Other changes to data caches take effect without a restart, including changing the type, creating, dropping, and resizing memory pools with **sp_poolconfig**, changing the wash percentage of the pools, and binding and unbinding objects.
- The default data cache must always have the type **default**, and no other cache can have the type **default**.
- The Adaptive Server housekeeper task does not do any buffer washing in caches with a type of **logonly** or in caches with a relaxed LRU replacement policy.
- The following commands perform only 2K I/O: **disk init**, some **dbcc** commands, and **drop table**. The **dbcc checkdb** and **dbcc checktable** commands can perform large I/O for tables, but perform 2K I/O

on indexes. Table 3-9 shows cache usage, depending on the binding of the database or object.

Table 3-9: Cache usage for Transact-SQL commands

Command	Database Bound	Table or Index Is Bound	Database or Object Not Bound
create index	Bound cache	N / A	Default data cache
disk init	N / A	N / A	Default data cache
dbcc checkdb	Bound cache	N / A	Default data cache
dbcc checktable, **indexalloc**, **tablealloc**	Bound cache	Bound cache	Default data cache
drop table	Bound cache	Bound cache	Default data cache

- Recovery uses only the 2K pool of the default data cache. All pages for all transactions that must be rolled back or rolled forward are read into and changed in this pool. Be sure that your default 2K pool is large enough for these transactions.
- When you use **sp_cacheconfig** with no parameters, it reports information about all of the caches on the server. If you specify only a cache name, it reports information about only the specified cache. If you use a fragment of a cache name, it reports information for all names matching "%*fragment*%".

  All reports include a block of information that reports information about caches, and a separate block of data for each cache that provides information about the pools within the cache.

  The output below shows the configuration for:

  - The default data cache with two pools: a 2K pool and a 16K pool
  - *pubs_cache* with two pools: 2K and 16K
  - *pubs_log*, with the type set to **logonly**, with a 2K pool and a 4K pool

```
Cache Name          Status    Type     Config Value Run Value
------------------- --------- -------- ------------ ------------
default data cache  Active    Default       0.00 Mb     22.27 Mb
pubs_cache          Active    Mixed        10.00 Mb     10.00 Mb
pubs_log            Active    Log Only      2.40 Mb      2.40 Mb
                                        ------------ ------------
                                Total      12.40 Mb     34.67 Mb
============================================================================
Cache: default data cache,   Status: Active,   Type: Default
    Config Size: 0.00 Mb,   Run Size: 22.27 Mb
    Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    512 Kb      0.00 Mb     14.27 Mb     10
   16 Kb   4096 Kb      8.00 Mb      8.00 Mb     10
============================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
    Config Size: 10.00 Mb,   Run Size: 10.00 Mb
    Config Replacement: strict LRU,   Run Replacement: strict LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    512 Kb      0.00 Mb      6.00 Mb     10
   16 Kb    816 Kb      4.00 Mb      4.00 Mb     10
============================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
   Config Size: 2.40 Mb,   Run Size: 2.40 Mb
   Config Replacement: relaxed LRU,  Run Replacement: relaxed LRU

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    512 Kb      0.00 Mb      1.00 Mb     10
    4 Kb    284 Kb      1.40 Mb      1.40 Mb     10
```

Table 3-10 lists the meaning of the columns in the output:

Table 3-10: sp_cacheconfig output

Column	Meaning
Cache Name	The name of the cache.
Status	One of the following: • "Active" • "Pend / Act" • "Pend / Del" These are explained following this table.
Type	"Mixed" or "Log Only" for user-defined caches, "Default" for the default data cache.
I / O Size	The size of I / O for a memory pool. This column is blank on the line that shows that cache configuration.
Wash Size	The size of the wash area for the pool. As pages enter the wash area of the cache, they are written to disk. This column is blank on the line that shows the cache configuration.
Config Value or Config Size	The size that the cache or pool will have after the next time Adaptive Server is restarted. These are the values that take effect the next time Adaptive Server is restarted. If the value is 0, the size has not been explicitly configured, and a default value will be used.
Run Value or Run Size	The size of the cache or pool now in use on Adaptive Server.
Config / Run Replacement	The cache policy (strict or relaxed) that will be used for the cache after the next restart, and the current replacement policy. These will be different only if the policy has been changed since the last replacement.
APF Percent	The percentage of buffers in the pool that can hold buffers that have been fetched by asynchronous prefetch, but have not been used.
Total	The total size of data cache, if the report covers all caches, or the current size of the particular cache, if you specify a cache name.

The status "Pend" is short for pending. It always occurs in combination with either "Act" for Active or "Del" for Delete. It indicates that a configuration action has taken place, but that the server must be restarted in order for the changes to take effect.

When you first create a new cache, but have not yet restarted Adaptive Server, the status is "Pend / Act", meaning that the cache has just been configured and will be active after a restart. If you set the size of a cache to 0 to delete it, the status changes from "Active" to "Pend / Del", meaning that the cache still exists, and still functions, but that it will be deleted at the next restart.

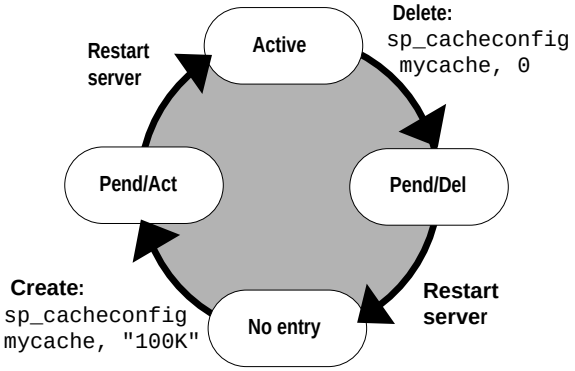

Figure 3-2: Effects of restarts and sp_cacheconfig on cache status

- You can also configure caches and pools by editing the configuration file. See Chapter 9, "Configuring Data Caches," in the *System Administration Guide*.

Data Cache Memory

- When Adaptive Server is first installed, all data cache memory is assigned to the 2K pool of the cache named *default data cache*. The default data cache is used by all objects that are not explicitly bound to a data cache with **sp_bindcache** or whose databases are not bound to a cache.
- When you create data caches, the memory allocation comes from the default data cache. Memory for caches is allocated out of the memory allocated to Adaptive Server with the **total memory** configuration parameter. To increase the amount of space available for caches, increase **total memory**, or decrease other configuration settings that use memory. If you need to decrease the size of **total memory**, the space must be available in the default data cache.

  You cannot reduce the size of the default data cache to less than 512K. In most cases, the default cache should be much larger than the minimum. This cache is used for all objects, including

system tables, that are not bound to another cache, and is the only cache used during recovery. See Chapter 9, "Configuring Data Caches," in the *System Administration Guide*.

- A data cache requires a small percentage of overhead for structures that manage the cache. All cache overhead is taken from the default data cache. To see the amount of overhead required for a specific size of cache, use the system procedure **sp_helpcache**, giving the size:

  ```
  sp_helpcache "200M"

  10.38Mb of overhead memory will be needed to
  manage a cache of size 200M
  ```

Changing Existing Caches

- To change the size of an existing cache, specify the cache's name and the new size.
  - If you increase the size of an existing cache, all of the added space is placed in the 2K pool.
  - To reduce the size of an existing cache, all of the space must be available in the 2K pool. You may need to use **sp_poolconfig** to move space from other pools to the 2K pool.
- If you have a database or any nonlog objects bound to a cache, you cannot change its type to **logonly**.

Dropping Caches

- To drop or delete a data cache, change its size to 0, as shown in example 3. When you set a cache's size to 0, the cache is marked for deletion, but it is not dropped until the next restart of the server. The cache remains active, and all objects that are bound to that cache continue to use it.

  You cannot drop the default data cache.
- If you drop a cache that has objects bound to it, all of the object bindings for the cache are marked invalid the next time you restart Adaptive Server. A message is printed to the error log on restart, giving the database ID, object ID and index ID:

  ```
  00:95/11/05 18:20:39.42 server  Cache binding for
  database '6', object '8', index '0' is being
  marked invalid in Sysattributes.
  ```

  If you subsequently create a cache of the same name, bindings are marked valid when the cache is activated.

Messages

- `Attempt to delete the default data cache. The default cache may not be deleted.`

  You cannot delete the default data cache.

- `Can't run sp_cacheconfig from within a transaction.`

  sp_cacheconfig modifies system tables, so it cannot be run within a transaction.

- `Cannot modify a cache type to be 'log only' when non-log objects are bound to it. Use sp_helpcache to print out bound objects and sp_unbindcache to delete the cache bindings.`

  You tried to change a cache's type to **logonly**, but there are tables, indexes, or other objects bound to the cache.

- `Invalid configuration for the default 2k pool in cache` *`cachename`*`. The default 2k pool must be a minimum of 512k.`

  You specified a size of less than 512K for the cache.

- `The cache type can be either 'logonly' or 'mixed' only.`

  The third parameter to **sp_cacheconfig** specifies the cache type. It must be **logonly** or **mixed**.

- `The cache type can only be specified once.`

  You specified a cache type in both the second and third parameters.

- `The change is completed. The SQL Server must be rebooted for the change to take effect.`

  The **sp_cacheconfig** command was successful. You cannot bind objects to the cache or configure pools in it until you restart Adaptive Server.

- `The specified named cache '`*`cachename`*`' does not exist.`

  The name of the cache you specified does not exist. You see this message only when you use **sp_cacheconfig** to display information about a particular cache.

- `The replacement policy can only be specified once.`

  You included the **strict** or **relaxed** cache replacement policy more than once in your command.

- `The specified named cache '`*`cachename`*`' does not exist.`

  The name of the cache you specified does not exist. You only see this message when you use **sp_cacheconfig** to display information about a particular cache.

- `Usage: sp_cacheconfig [ cachename [, 'cache_size[K|P|M|G]' ] [, logonly | mixed ]]`

  You typed a parameter incorrectly.

- `You must have the following role(s) to execute this command/procedure: 'sa_role'. Please contact a user with the appropriate role for help.`

  Only a System Administrator can change cache configurations; other users can only view cache configurations.

Permissions

Only a System Administrator can change cache configurations with **sp_cacheconfig**. All users can use **sp_cacheconfig** to view cache configurations.

Tables Used

master..sysconfigures

See Also

System procedures	**sp_bindcache**, **sp_configure**, **sp_help**, **sp_helpcache**, **sp_helpdb**, **sp_helpindex**, **sp_poolconfig**, **sp_unbindcache**, **sp_unbindcache_all**

sp_cachestrategy

Function

Enables or disables prefetching (large I/O) and MRU cache replacement strategy for a table, index, *text* object, or *image* object.

Syntax

```
sp_cachestrategy dbname, [ownername.]tablename
   [, indexname | "text only" | "table only"
   [, { prefetch | mru }, { "on" | "off"}]]
```

Parameters

dbname – is the name of the database where the object is stored.

ownername – is the name of the table's owner. If the table is owned by "dbo", the owner name is optional.

tablename – is the name of the table.

indexname – is the name of the index on the table.

text only – changes the cache strategy for a *text* or *image* object.

table only – changes the cache strategy for a table.

prefetch | **mru** – is **prefetch** or **mru**, and specifies which setting to change.

on | off – specifies the setting, **"on"** or **"off"**, enclosed in quotes.

Examples

1. **sp_cachestrategy pubs2, titles**

   ```
   object name        index name    large IO MRU
   ------------------ ------------- -------- --------
   dbo.titles           titleidind     ON        ON
   ```

   Displays information about cache strategies for the *titles* table.

2. **sp_cachestrategy pubs2, titles, titleind**

   Displays information about cache strategies for the *titleind* index.

3. **sp_cachestrategy pubs2, titles, titleind, prefetch, "off"**

   Disables prefetch on the *titleind* index of the *titles* table.

4. **sp_cachestrategy pubs2, authors, "table only", mru, "on"**

   Reenables MRU replacement strategy on the *authors* table.

5. **sp_cachestrategy pubs2, blurbs, "text only", prefetch, "on"**

   Reenables prefetching on the text pages of the *blurbs* table.

Comments

- If memory pools for large I/O are configured for the cache used by a table or an index, the optimizer can choose to prefetch data or index pages by performing large I/Os of up to eight data pages at a time. This **prefetch** strategy can be used on the data pages of a table or on the leaf-level pages of a nonclustered index. By default, prefetching is enabled for all tables, indexes, and *text* or *image* objects. Setting the **prefetch** option to "**off**" disables prefetch for the specified object.
- The optimizer can choose to use **MRU replacement strategy** to fetch and discard buffers in cache for table scans and index scans for I/O of any size. By default, this strategy is enabled for all objects. Setting **mru** to "**off**" disables this strategy. If you turn **mru** off for an object, all pages are read into the MRU/LRU chain in cache, and they remain in the cache until they are flushed by additional I/O. See Chapter 3, "Data Storage" in the *Performance and Tuning Guide* for more information on cache strategies.
- You can change the cache strategy only for objects in the current database.
- When you use **sp_cachestrategy** without specifying the strategy and setting, it reports the current settings for the object, as shown in example 1.
- To see the size, status and I/O size of all data caches on the server, use the system procedure **sp_cacheconfig**.
- Setting **prefetch "on"** has no effect on tables or indexes that are read into a cache that allows only 2K I/O. The **mru** strategy can be used in all caches, regardless of available I/O size.

Overrides

- If prefetching is turned on for a table or an index, you can override the prefetching for a session with **set prefetch "off"**. If prefetching is turned off for an object, you cannot override that setting.

- The **prefetch**, **lru**, and **mru** options to the **select**, **delete** and **update** commands suggest the I/O size and cache strategy for individual statements. If prefetching or MRU strategy is enabled for a table or an index, you can override it for a query by specifying 2K I/O for **prefetch**, and by specifying **lru** strategy. For example, the following command forces LRU strategy, 2K I/O, and a table scan of the *titles* table:

  ```
  select avg(advance)
  from titles (index titles prefetch 2 lru)
  ```

  If you request a prefetch size, and the object's cache is not configured for I/O of the requested size, the optimizer chooses the best available I/O size.

- If prefetching is enabled for an object with **sp_cachestrategy**, using a **prefetch** specification of 2K in a **select**, **update** or **delete** command overrides an earlier **set prefetch "on"** statement. Specifying a larger I/O size in a **select**, **update** or **delete** command does not override a **set prefetch "off"** command.

Messages

- '*argument*' is not a valid argument.

  The valid arguments are: **lru**, **mru**, and **prefetch**.

- No such object or user exists in the database.

  The table does not exist in the database, or the owner name is not correct.

- Object must be in the current database.

  You cannot change the cache strategy for an object that is not in the current database.

- Only the System Administrator (SA) or the Object Owner may execute this stored Procedure.

  Only the System Administrator or the object owner can change the cache strategy.

- The target index does not exist.

  The name you gave for an index is not a valid index on the table. Use **sp_helpindex** *tablename* to see the index names.

- The target object does not exist.

  You issued **sp_cachestrategy** with the name of a table that does not exist in the current database. Run **sp_help** to see a list of the objects in the current database.

- `Usage: sp_cachestrategy dbname, [ownername.]tablename [, indexname | 'text only' | 'table only' [, { prefetch | mru }, { 'on'|'off'}]]`

  You made a syntax error when you executed the procedure. Only the literal values provided in the usage statement are valid.

Permissions

Only a System Administrator or the object owner can view and change cache strategies.

Tables Used

master..sysattributes, *master..sysdatabases*, *sysattributes*, *sysindexes*, *sysobjects*

See Also

Commands	**delete**, **select**, **set**, **update**
Stored procedures	**sp_cacheconfig**, **sp_poolconfig**

sp_changedbowner

Function

Changes the owner of a user database.

Syntax

```
sp_changedbowner loginame [, true ]
```

Parameters

loginame – is the login name of the new owner of the current database.

true – transfers aliases and their permissions to the new database owner. Values are "**true**" and "**TRUE**".

Examples

```
1. sp_changedbowner albert
```

Makes the user "albert" the owner of the current database.

Comments

- The new owner must not already be known as either a user or alias (that is, the new owner must not already be listed in *sysusers* or *sysalternates*). Executing **sp_changedbowner** with the single parameter *loginame* changes the database ownership to *loginame* and drops aliases of users who could act as the old "dbo."
- After executing **sp_changedbowner**, the new owner is known as the Database Owner inside the database.
- **sp_changedbowner** cannot transfer ownership of the system databases.
- The new owner must already have a login name in Adaptive Server, but must **not** have a database user name or alias name in the database. To assign database ownership to such a user, drop the user name or alias entry before executing **sp_changedbowner.**
- To grant permissions to the new owner, a System Administrator must grant them to the Database Owner, since the user is no longer known inside the database under any other name.

Messages

- `Can't change the owner of the master database.`

  No one can change the owner of the *master* database.

- `Database owner changed.`

  The **sp_changedbowner** command succeeded and the Database Owner has been changed.

- `Only the System Administrator (SA) or the Database Owner (dbo) can change the owner of a database.`

  You must be a System Administrator or the Database Owner to execute **sp_changedbowner.**

- `The dependent aliases were mapped to the new dbo.`

  You used the optional parameter "**true**". Aliases and their permissions were transferred to the new Database Owner.

- `The dependent aliases were dropped.`

  You did not use the optional parameter "**true**". Aliases and their permissions have been dropped.

- `No login with the specified name exists.`

  The proposed new Database Owner must have a login on Adaptive Server.

- `The proposed new db owner already is a user in the database.`

  The specified *loginame* is already a user in the current database. To make the user the Database Owner, drop the user entry from the current database's *sysusers* table.

- `The proposed new db owner already is aliased in the database.`

  The specified *loginame* is already aliased in the current database. To make the user the Database Owner, drop the user alias entry from the current database's *sysalternates* table.

Permissions

Only a System Administrator can execute **sp_changedbowner.**

Tables Used

master..syslogins, sysalternates, sysobjects, sysusers

See Also

Commands	**create database**
System procedures	**sp_addlogin**, **sp_dropalias**, **sp_dropuser**, **sp_helpdb**

sp_changegroup

Function

Changes a user's group.

Syntax

sp_changegroup ***grpname, username***

Parameters

grpname – is the name of the group. The group must already exist in the current database. If you use "public" as the *grpname*, enclose it in quotes, because it is a keyword.

username – is the name of the user to be added to the group. The user must already exist in the current database.

Examples

1. **sp_changegroup fort_mudge, albert**

   The user "albert" is now a member of the "fort_mudge" group. It doesn't matter what group "albert" belonged to before.

2. **sp_changegroup "public", albert**

   Removes "albert" from the group he belonged to without making him a member of a new group (all users are always members of "public".)

Comments

- Executing **sp_changegroup** adds the specified user to the specified group. The user is dropped from the group he or she previously belonged to and is added to the one specified by *grpname*.
- New database users can be added to groups at the same time they are given access to the database with **sp_adduser.**
- Groups are used as a collective name for granting and revoking privileges. Every user is always a member of the default group, "public", and can belong to only one other group.
- To remove someone from a group without making that user a member of a new group, use **sp_changegroup** to change the user's group to "public", as shown above in example 2.

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

Messages

- `Group changed.`

  The user now belongs to the specified group.

- `No group with the specified name exists.`

  The specified group does not exist in the current database.

- `No user with the specified name exists in the current database.`

  The specified user does not exist in the current database.

Permissions

Only the Database Owner, a System Administrator, or a System Security Officer can execute **sp_changegroup**.

Tables Used

master..syssrvroles, *syscolumns*, *sysobjects*, *sysprotects*, *sysusers*

See Also

Commands	**grant**, **revoke**
System procedures	**sp_addgroup**, **sp_adduser**, **sp_dropgroup**, **sp_helpgroup**

sp_checknames

Function

Checks the current database for names that contain characters not in the 7-bit ASCII set.

Syntax

sp_checknames

Parameters

None.

Examples

1. sp_checknames

```
Looking for non 7-bit ASCII characters in the system tables
of database:
"master"


============================================================
Table.Column name: "syslogins.password"

The following logins have passwords that contain non 7-bit
ASCII characters.  If you wish to change them use "sp_password";
Remember, only the sa and the login itself may examine or change
the syslogins.password column:

 suid   name
 ------ ------------------------------
      1 sa
      2 probe
      3 bogususer
```

Comments

- **sp_checknames** examines the names of all objects, columns, indexes, user names, group names, and other elements in the current database for characters outside of the 7-bit ASCII set. It reports illegal names and gives instructions to make them compatible with the 7-bit ASCII set.
- Run **sp_checknames** in every database on your server after upgrading from a SQL Server of release 4.0.x or 4.2.x, and after using a default character set that was not 7-bit ASCII.

- Follow the instructions in the **sp_checknames** report to correct all non-ASCII names.

Messages

- `Good news! Database "`*`db_name`*`" has no obj/user/etc. names that contain non 7-bit ASCII characters.`

  If **sp_checknames** finds any names that are not fully 7-bit ASCII, appropriate messages and remedial instructions appear.

Permissions

Any user can execute **sp_checknames**.

Tables Used

sp_checknames uses the following tables when it is executed in any database:

dbo.syscolumns, *dbo.sysindexes*, *dbo.sysobjects*, *dbo.syssegments*, *dbo.systypes*, *dbo.sysusers*

sp_checknames uses the following tables when it is executed in the *master* database:

master..sysdatabases, *master..sysdevices*, *master..syslogins*, *master..sysremotelogins*, *master..sysservers*

See Also

Commands	**update**
System procedures	**sp_password**, **sp_rename**, **sp_renamedb**

sp_checkreswords

Function

Detects and displays identifiers that are Transact-SQL reserved words. Checks server names, device names, database names, segment names, user-defined datatypes, object names, column names, user names, login names, and remote login names.

Syntax

sp_checkreswords [*user_name_param*]

Parameters

user_name_param – is the name of a user in the current database. If you supply *user_name_param*, **sp_checkreswords** checks only for objects that are owned by the specified user.

Examples

1. sp_checkreswords *(executed in master database)*

```
Reserved Words Used as Database Object Names for Database master

Upgrade renames sysobjects.schema to sysobjects.schemacnt.

Owner
------------------------------
dbo

Table                          Reserved Word Column Names
------------------------------ ------------------------------
authorization                  cascade

Object Type                    Reserved Word Object Names
------------------------------ ------------------------------
rule                           constraint
stored procedure               check
user table                     arith_overflow
user table                     authorization

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

Owner
------------------------------
lemur
```

```
Table                          Reserved Word Column Names
------------------------------ ------------------------------
key                            close

Table                          Reserved Word Index Names
------------------------------ ------------------------------
key                            isolation

Object Type                    Reserved Word Object Names
------------------------------ ------------------------------
default                        isolation
rule                           level
stored procedure               mirror
user table                     key

Reserved Word Datatype Names
------------------------------
identity


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

Database-wide Objects
---------------------


Reserved Word User Names
------------------------------
at
identity

Reserved Word Login Names
------------------------------
at
identity

Reserved Word as Database Names
------------------------------
work

Reserved Word as Language Names
------------------------------
national
```

```
Reserved Word as Server Names
------------------------------
mirror
primary

Reserved Word ServerNetNames
--------------------------------
mirror
primary
```

2. sp_checkreswords *(executed in user database)*

```
Reserved Words Used as Database Object Names for Database user_db

Upgrade renames sysobjects schema to sysobjects.schemacnt.

Owner
------------------------------
tamarin

Table                          Reserved Word Column Names
------------------------------ ------------------------------
cursor                         current
endtran                        current
key                            identity
key                            varying
schema                         primary
schema                         references
schema                         role
schema                         some
schema                         user
schema                         work

Table                          Reserved Word Index Names
------------------------------ ------------------------------
key                            double

Object Type                    Reserved Word Object Names
------------------------------ ------------------------------
default                        escape
rule                           fetch
stored procedure               foreign
user table                     cursor
user table                     key
user table                     schema
```

```
view                            endtran


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

Database-wide Objects
---------------------

Found no reserved words used as names for database-wide objects.
```

Comments

- **sp_checkreswords** reports the names of existing objects that are reserved words. Transact-SQL does not allow words that are part of any command syntax to be used as identifiers, unless you are using delimited identifiers. Reserved words are pieces of SQL syntax, and they have special meaning when you use them as part of a command. For example, in pre-release 10.0 SQL Server, you could have a table called *work*, and select data from it with this query:

  ```
  select * from work
  ```

  work was a new reserved word in SQL Sercer release 10.0, part of the command **commit work**. Issuing the same **select** statement in release 10.0 or later causes a syntax error. **sp_checkreswords** finds identifiers that would cause these problems.

- **sp_checkreswords** also finds reserved words, used as identifiers, that were created using the **set quoted_identifier** option.

- Use **sp_checkreswords** before or immediately after upgrading to a new release of Adaptive Server. See the installation documentation for your platform for information on installing and running this procedure before performing the upgrade.

  Run **sp_checkreswords** in the *master* database and in each user database. Also run it in *model* and *sybsystemprocs*, if you have added users or objects to those databases.

- The return status indicates the number of items found.

- If you supply a user name, **sp_checkreswords** checks for all of the objects that can be owned by a user tables, indexes, views, procedures, triggers, rules, defaults, and user-defined datatypes. It reports all identifiers that are reserved words.

- If your current database is not the *master* database, and you do not provide a user name, **sp_checkreswords** checks for all of the objects above, with a separate section in the report for each user

name. It also checks *sysusers* and *syssegments* for user names and segment names that are reserved words. You only need to check *model* and *sybsystemprocs* if you have added objects, users, or user-defined datatypes.

- If your current database is *master*, and you do not provide a user name, **sp_checkreswords** performs all of the checks above and also checks *sysdatabases*, *syslogins*, *syscharsets*, *sysservers*, *sysremotelogins*, *sysdevices*, and *syslanguages* for reserved words used as the names of databases, local or remote logins, local and remote servers, character sets, and languages.

Handling Reported Instances of Reserved Words

- If **sp_checkreswords** reports that reserved words are used as identifiers, you have two options:
  - Use **sp_rename**, **sp_renamedb**, or update the system tables to change the name of the identifier.
  - Use **set quoted_identifier on** if the reserved word is a table name, view name, or column name. If most of your applications use stored procedures, you can drop and re-create these procedures with **set quoted_identifier on**, and quote all identifiers. All users will be able to run the procedures, without having to use **set quoted_identifier on** for their session. You can use **set quoted_identifier on**, create views that give alternative names to tables or columns, and change your applications to reference the view instead.

    The following example provides alternatives for the new reserved words “key”, “level”, and “work”:

    ```
    create view keyview
    as
    select lvl = "level", wrk = "work"
    from "key"
    ```

    The syntax for the **set** command is:

    ```
    set quoted_identifier on
    ```

- If you do not either change the identifiers or use delimited identifiers, any query that uses the reserved words as identifiers reports an error, usually a syntax error. For example:

  ```
  select level, work from key
  ```

  ```
  Msg 156, Level 15, State 1:
  Server 'rosie', Line 1:
  Incorrect syntax near the keyword 'level'.
  ```

➤ ***Note***

The quoted identifier option is a SQL92 option and may not be supported by many client products that support other Adaptive Server features. For example, you cannot use **bcp** on tables whose names are reserved words. Before choosing the quoted identifier option, perform a test on various objects using all the tools you will use to access Adaptive Server. Use **set quoted_identifier on**, create a table with a reserved word for a name and reserved words for column names. If the client product generates SQL code, it must enclose identifiers in double quotes (if they are reserved words) and character constants in single quotes.

- Procedures, triggers, and views that depend on objects whose names have been changed may continue to work for some time after the name change, and then suddenly stop working when the query plan is recompiled. Recompilation takes place for many reasons, without notification to the user. Change the names of objects in procedures, triggers, and views immediately after you change the object name.
- Whether you change the object names or use delimited identifiers, you must change all stored procedures, views, triggers, and applications that include the reserved word. If you change object names, you must change identifiers; if you use delimited identifiers, you must add the **set quoted_identifier** option and quotation marks.
- If you do not have the text of your procedures, triggers, views, rules, and defaults saved in operating system files, you can use **defncopy** to copy the definitions from the server to files. See **defncopy** in the *Utility Programs* manual for your platform.

Changing Identifiers

- If you change the names of the items reported by **sp_checkreswords**, you must change the names in all procedures, triggers, views, and applications that reference the object using the reserved word.
- Dump your database before changing identifier names. After you change the identifier names, run **dbcc** to determine that there are no problems, and dump the database again.
- If you are changing identifiers on an active production database:

  - Perform the changes when the system is least busy, so that you will disrupt as few users as possible.
  - Prepare carefully by finding all Open Client DB-Library™ programs, windowing applications, stored procedures, triggers, and scripts that use a particular identifier. This way, you can make the edits needed in the source code, and then change the identifiers and replace the procedures and code as quickly as possible.

- The procedure **sp_depends** can help find procedures, views, and triggers that use table and view names.

Using *sp_rename* to Change Identifiers

- The system procedure **sp_rename** renames tables, indexes, views, procedures, triggers, rule, defaults, user-defined datatypes, and columns. Use **sp_renamedb** to rename databases.
- Table 3-11 shows the types of identifiers that you can change with **sp_rename** and lists other changes that may have to be made on the server and in your application programs.

Table 3-11: sp_rename and changing identifiers

Identifier	Remember To
Table name	• Drop all procedures, triggers and views that reference the table, and re-create them with the new name. Use **sp_depends** to find the objects that depend on the table. • Change all applications or SQL source scripts that reference the table to use the new table name. • Change **dbcc** scripts that perform table-level checks using table names.
Index name	• Drop any stored procedures that create or drop the index, and re-create them with the new name. • Change all applications or SQL source scripts that create or drop the index. • Change **dbcc** scripts that perform index-level checks using index names.
View name	• Drop all procedures, triggers, and views that reference the view, and re-create them with the new name. Use **sp_depends** to find the objects that depend on the view. • Change all applications or SQL source scripts that reference the view to use the new view name.

Table 3-11: sp_rename and changing identifiers (continued)

Identifier	Remember To
Procedure name	• Drop and re-create with the new procedure name all procedures and triggers that reference the procedure. • Change all applications or SQL source scripts that execute the procedure to use the new name. • If another server remotely calls the procedure, change applications on the remote server to use the new procedure name.
Trigger name	• Change any SQL source scripts that create the trigger.
Rule name	• Change any SQL source scripts that create the rule.
Default name	• Change any SQL source scripts that create the default.
User-defined datatype name	• Drop all procedures that create tables with user-defined datatypes, and re-create them with the new name. • Change any applications that create tables with user-defined datatypes.

Column name	• Drop all procedures, triggers and views that reference the column, and re-create them with the new column name.
	• **sp_depends** cannot find column name references. The following query displays the names of procedures, triggers, and views that reference a column named "key":
	`select distinct sysobjects.name`
	`from sysobjects, syscomments`
	`where sysobjects.id = syscomments.id`
	`and syscomments.text like "%key%"`
	• Change all applications and SQL source scripts that reference the column by name.

The following command changes the name of the view *isolation* to *isolated*:

```
sp_rename "isolation", isolated
```

The following command changes the name of a column in the renamed view *isolated*:

```
sp_rename "isolated.key", keyname
```

- Use **sp_depends** to get a list of all views, procedures, and triggers that reference a view, procedure, or table that will be renamed. To use **sp_depends** after renaming an object, give the new name. For example:

  ```
  sp_depends new_name
  ```

Renaming Databases with *sp_renamedb*

- To change the name of a database, use **sp_renamedb**. The database must be in single-user mode. Drop and re-create any procedures, triggers, and views that explicitly reference the database name. See **sp_renamedb** for more information.

Changing Other Identifiers

- To change user names, login names, device names, remote server names, remote server user names, segment names, and character set and language names, first determine if you can drop the object or user and then add or create it again. If you cannot do that, use **sp_configure "allow updates to system tables", 1** to allow direct updates to system tables. Only a System Security Officer can set the **allow updates to system tables** configuration parameter.

  Since errors during direct updates to system tables can create severe problems in Adaptive Server, refer to Table 3-12 to determine whether you can drop the objects or user , and then re-create them. *Table 3-14: Considerations when changing identifiers* on page 1-161 shows possible dependencies on this set of identifiers. Refer to this table for possible dependencies, whether you choose to upgrade by dropping and recreating objects, by using delimited identifiers, or by performing direct updates to system tables.

Table 3-12: Alternatives to direct system tables updates when changing identifiers

Identifier Type	Suggested Actions to Avoid Updates to System Tables
User names and login names	To change the name of a user with no objects, first use **sp_helprotect** *username* in each database to record the user's permissions. Then, drop the user from all of the databases (**sp_dropuser**), and drop the login (**sp_droplogin**). Finally, add the new login name (**sp_addlogin**), add the new user name to the databases (**sp_adduser**), and restore the user's permissions with **grant**.
Device names	If this device is completely allocated, you will not need to use its name in a **create database** command, so you can leave the name unchanged.
Remote server names	Unless there are large numbers of remote login names from the remote server, drop the remote server (**sp_dropserver**) and add it with a new name (**sp_addserver**).
Remote server logins	Drop the remote login with **sp_dropremotelogin**, add it with a new name using **sp_addremotelogin**, and restore the user's permission to execute procedures with **grant**.
Segment names	These are rarely used, once objects have been created on the segments.

Table 3-12: Alternatives to direct system tables updates when changing identifiers (continued)

Identifier Type	Suggested Actions to Avoid Updates to System Tables
Character set and language names	Languages and character sets have reserved words as identifiers only if a System Administrator has created alternative languages with **sp_addlanguage**. Drop the language with **sp_droplanguage**, and add it with a new name.

◆ ***WARNING!***

Direct updates to system tables can be very dangerous. You can make mistakes that make it impossible for Adaptive Server to run or make it impossible to access objects in your databases. Undertake this effort when you are calm and collected, and when little or no production activity is taking place on the server. If possible, use the alternative methods described Table 3-12.

- The following example shows a "safe" procedure for updating a user name, with all data modification preceded by a **begin transaction** command:

  The System Security Officer executes the following command:

  ```
  sp_configure "allow updates to system tables", 1
  ```

  Then you can execute the following:

  ```
  begin transaction
  update sysusers
  set name = "workerbee"
  where name = "work"
  ```

  At this point, run the query, and check to be sure that the command affected only the row that you intended to change. The only identifier change that affects more than one row is changing the *language* name in *syslogins*.

  - If the query affected only the correct row, use **commit transaction**.
  - If the query affected more than one row, or the incorrect row, use **rollback transaction**, determine the source of the problem, and execute the command correctly.

  When you are finished, the System Security Officer turns off the **allow updates to system tables** configuration parameter with this command:

  ```
  sp_configure "allow updates to system tables", 0
  ```

◆ ***WARNING!***

Only update system tables in a single database in each user defined transaction. Do not issue a begin transaction command, and then update tables in several databases. Such actions can make recovery extremely difficult.

Table 3-13 shows the system tables and columns that you should update to change reserved words. The tables preceded by "*master.dbo.*" occur only in the *master* database. All other tables occur in *master* and in user databases. Be certain you are using the correct database before you attempt the update. You can check for the current database name with this command:

```
select db_name()
```

Table 3-13: System table columns to update when changing identifiers

Type of Identifier	Table to Update	Column Name
User name	*sysusers*	*name*
Login names	*master.dbo.syslogins*	*name*
Segment names	*syssegments*	*name*
Device name	*sysdevices*	*name*
Remote server name	*sysservers*	*srvname*
Remote server network name	*sysservers*	*srvnetname*
Character set names	*master.dbo.syscharsets*	*name*
Language name	*master.dbo.syslanguages* *master.dbo.syslogins*	*name* *language*

Table 3-14 shows other changes that may have to be made on the server and in your application programs:

Table 3-14: Considerations when changing identifiers

Identifier	Remember To
Login name	Change the user name in each database where this person is a user.
User name	Drop, edit, and re-create all procedures, triggers, and views that use qualified (*owner_name.object_name*) references to objects owned by this user. Change all applications and SQL source scripts that use qualified object names to use the new user name. You do not have to drop the objects themselves; *sysusers* is linked to *sysobjects* by the column that stores the user's ID, not the user's name.
Device name	Change any SQL source scripts or applications that reference the device name to use the new name.
Remote server name	Change the name on the remote server. If the name that **sp_checkreswords** reports is the name of the local server, you must restart the server before you can issue or receive remote procedure calls.
Remote server network name	Change the server's name in the interfaces files.
Remote server login name	Change the name on the remote server.
Segment name	Drop and re-create all procedures that create tables or indexes on the segment name. Change all applications that create objects on segments to use the new segment name.
Character set name	None.
Language name	Change both *master.dbo.syslanguages* and *master.dbo.syslogins*. The update to *syslogins* may involve many rows. Also, change the names of your localization files.

Using Delimited Identifiers

- You can use delimited identifiers for table names, column names, and view names. You cannot use delimited identifiers for other object names.
- If you choose to use delimited identifiers, use **set quoted_identifier on**, and drop and re-create all the procedures, triggers, and views that use the identifier. Edit the text for those objects, enclosing the

reserved words in double quotes and enclosing all character strings in single quotes.

The following example shows the changes to make to queries in order to use delimited identifiers. This example updates a table named *work*, with columns named *key* and *level*. Here is the pre-release 10.0 query, which encloses character literals in double quotes, and the edited version of the query for use with delimited identifiers:

```
/* pre-release 10.0 version of query */
update work set level = "novice"
    where key = "19-732"

/* 10.0 or later version of query, using
** the quoted identifier option
*/
update "work" set "level" = 'novice'
    where "key" = '19-732'
```

- All applications that use the reserved word as an identifier must be changed as follows:
  - The application must set the quoted identifier option on.
  - All uses of the reserved word must be enclosed in double quotes.
  - All character literals used by the application while the quoted identifier option is turned on must be enclosed in single quotes. Otherwise, Adaptive Server attempts to interpret them as object names.

  For example, the following query results in an error message:

  ```
  set quoted_identifier on

  select * from titles where title_id like "BU%"
  ```

  Here is the correct query:

  ```
  select * from titles where title_id like 'BU%'
  ```

- Stored procedures that you create while the delimited identifiers are in effect can be run without turning on the option. (The **allow updates to system tables** option also works this way.) This means that you can turn on quoted identifier mode, drop a stored procedure, edit it to insert quotation marks around reserved words used as identifiers, and re-create the procedure. All users can execute the procedure without using **set quoted_identifier.**

Messages

- `Found no reserved words used as database object names.`

  No tables, views, procedures, triggers, rules or defaults in the current database use reserved words as names.

- `Found no reserved words used as names for database-wide objects.`

  No items such as segments or user names use reserved words as names.

- `No user with the specified name exists in the current database.`

  The user name you specified is not a user in the current database. Be sure you spelled the name correctly, and be sure you are using the correct database.

Permissions

Only a System Administrator can execute **sp_checkreswords**.

Tables Used

master..syscharsets, *master..sysdatabases*, *master..sysdevices*, *master..syslanguages*, *master..syslogins*, *master..sysremotelogins*, *master..sysservers*, *master..sysmessages*, *syscolumns*, *sysindexes*, *sysobjects*, *syssegments*, *systypes*, *sysusers*

See Also

Commands	**set**
System procedures	**sp_configure**, **sp_depends**, **sp_rename**, **sp_renamedb**

sp_checksource

Function

Checks for the existence of the **source text** of the **compiled object**.

Syntax

```
sp_checksource [objname [, tabname [, username]]]
```

Parameters

objname – is the compiled object to be checked for the existence of its source text.

tabname – is the name of the table or view to be checked for the existence of all check constraints, defaults, and triggers defined on it.

username – is the name of the user who owns the compiled objects to be checked for the existence of the source text.

Examples

1. **`sp_checksource`**

   Checks for the existence of the source text of all compiled objects in the current database.

2. **`sp_checksource titleview`**

   Checks for the existence of the source text of the view named *titleview.*

3. **`sp_checksource title_vu, @username = Mary`**

   Checks for the existence of the source text of the view named *titls_vu* that is owned by Mary.

4. **`sp_checksource list_phone_proc`**

   Checks for the existence of the source text of the custom stored procedure *list_phone_proc.*

5. **`sp_checksource @tabname = "my_tab"`**

   Checks for the existence of the source text of all the check constraints, triggers, and declarative defaults defined on the table named *my_tab.*

6. **sp_checksource @objname = "my_vu", @tabname = "my_tab"**

   Checks for the existence of the source text of the view *my_vu* and all check constraints, triggers, and defaults defined on the table *my_tab*.

7. **sp_checksource @username = "Tom"**

   Checks for the existence of the source text of all compiled objects owned by Tom.

Comments

- **sp_checksource** checks for the existence of the source text of the specified compiled object. If the source text exists for the specified object, **sp_checksource** returns 0. If the source text does not exist for the specified object, **sp_checksource** returns 1.
- If you do not provide any parameters, **sp_checksource** checks the existence of the source text for all compiled objects in the current database.
- To use **sp_checksource** with no parameters, you must be the Database Owner or System Administrator.

Messages

- `No user with the specified name exists in the current database.`

  You entered an invalid user name. Use **sp_helpuser** to lists user names.

- `Object does not exist in this database.`

  You entered an invalid name for the compiled object or table. Use **sp_help** to list objects.

- `Source text for compiled object` *`objname`* `exists.`

  sp_checksource found the source text for the compiled object.

- `Source text for compiled object` *`objname`* `does not exist.`

  sp_checksource did not find the source text of the compiled object.

- `You must have the following role(s) to execute this command/procedure: 'sa_role'. Please contact a user with the appropriate role for help.`

  Only a Database Owner or System Administrator can check the existence of the source text for compiled objects that are owned by another user.

Permissions

Any user can use **sp_checksource** to check for the existence of the source text for his or her own compiled objects. Only a Database Owner or System Administrator can check for the existence of the source text of compiled objects that are owned by another user.

Tables Used

syscolumns, syscomments, sysconstraints, sysobjects, sysprocedures

See Also

System procedures	**sp_hidetext**

sp_chgattribute

Function

Changes the **max_rows_per_page** value for future space allocations of a table or index.

Syntax

```
sp_chgattribute objname, optname, optvalue
```

Parameters

objname – is the name of the table or index for which future space allocations are to be changed.

optname – is **max_rows_per_page**. You must use quotes because **max_rows_per_page** is a Transact-SQL reserved word.

optvalue – is the new **max_rows_per_page** value. For tables and clustered indexes, the value can be between 0 and 256. For nonclustered indexes, the range of values depends on the size of the key on the leaf page. A value of 0 instructs Adaptive Server not to limit the number of rows on a page.

Examples

1. ```
   sp_chgattribute authors, "max_rows_per_page", 1
   ```

   Sets the **max_rows_per_page** to 1 for the *authors* table for all future space allocations.

2. ```
   sp_chgattribute "titles.titleidind",
   "max_rows_per_page", 4
   ```

   Sets the **max_rows_per_page** to 4 for the *titleidind* index for all future space allocations.

Comments

- **sp_chgattribute** changes the **max_rows_per_page** value for future space allocations of *objname*. It does not affect the space allocations of existing data pages. You can change the **max_rows_per_page** value of an object in the current database only.
- Setting **max_rows_per_page** to 0 tells Adaptive Server to fill the data or index pages and not to limit the number of rows (this is the default behavior of Adaptive Server if **max_rows_per_page** is not set).

- Low values for *optvalue* may cause page splits. Page splits occur when new data or index rows need to be added to a page, and there is not enough room for the new row. Usually, the data on the existing page is split fairly evenly between the newly allocated page and the existing page.
- To approximate the maximum value for a nonclustered index, subtract 32 from the page size and divide the resulting number by the index key size. The following statement calculates the maximum value of **max_rows_per_page** for the nonclustered index *titleind*:

```
select
    (select @@pagesize - 32) / minlen
     from sysindexes where name = "titleind"

-----------
        288
```

If you specify too high a value for *optvalue*, Adaptive Server returns an error message specifying the highest value allowed.

Messages

- `Can't run sp_chgattribute from within a transaction.`

  sp_chgattribute modifies system tables, so it cannot be run within a transaction.

- `Object must be in the current database.`

  You cannot qualify *objname* with a reference to another database. Qualify **sp_chgattribute** with the database name or issue the **use** *database_name* command to open the database in which the table or index resides, and run **sp_chgattribute** again.

- `You do not own a table, column or index of that name in the current database.`

  The specified table or index does not exist in the current database. Be sure you spelled the name correctly, and be sure you are using the correct database.

- `Unrecognized change attribute option.`

  optname must be **max_rows_per_page**.

- '*optname*' attribute of object '*objname*' changed to *optvalue*

  The procedure succeeded.

Permissions

Only the object owner can execute **sp_chgattribute.**

Tables Used

sysindexes, sysobjects

See Also

Commands	**alter table**, **create index**, **create table**
System procedures	**sp_helpindex**

sp_clearpsexe

Function

Clears the execution attributes of an Adaptive Server session that was set by **sp_setpsexe**.

Syntax

```
sp_clearpsexe spid, exeattr
```

Parameters

spid – is the process ID of the session for which execution attributes are to be cleared.

exeattr – identifies the execution attributes to be cleared. Values for *exeattr* are "**priority**" and **"enginegroup"**.

Examples

1. `sp_clearpsexe 12, 'enginegroup'`

Drops the engine group entry for process 12.

Comments

- **sp_clearpsexe** clears the execution attributes of the session that was set by **sp_setpsexe**. For more information, see Chapter 22, "Distributing Engine Resources Between Tasks," in the *Performance and Tuning Guide*.
- If the execution attributes are not cleared during the lifetime of the session, they are cleared when the session exits or terminates abnormally.
- **sp_clearpsexe** fails if there are no online engines in the associated engine group.
- When you drop an engine group entry, the session executes on an engine group determined by a class definition or by the default class.
- Use **sp_who** to list process IDs (*spid*s).

Messages

- `A non-SA user cannot modify attributes of another task.`

  Only a System Administrator can modify execution attributes.

- `A non-SA user can only modify its priority value.`

  Users who are not System Administrators can only lower the *priority* value.

- `Can't run sp_setpsexe from within a transaction.`

  sp_setpsexe modifies system tables, so it cannot be run within a transaction.

- *`exeattr`* `is not a valid execution attribute.`

  The attribute name is not valid. Specify either **priority** or **enginegroup**.

- `Failed to clear value of attribute` *`spid`* `for spid` *`exattr`*`. Check server errorlog for any additional information.`

  The execution attribute value could not be cleared for the specified Adaptive Server process. Check the error log.

- `No SQL Server process with the specified ID exists.`

  To see a list of valid *spid*s, run **sp_who**.

- `No specification for the specified process exists.`

  The *spid* used is not valid.

Permissions

All users can execute **sp_clearpsexe** to clear the priority attributes of tasks that user owns. Only a System Administrator can execute **sp_clearpsexe** to clear priority attributes for all users.

Tables Used

sysattributes, *sysprocesses*

See Also

System procedures	**sp_addengine, sp_addexeclass, sp_bindexeclass, sp_dropengine, sp_dropexeclass, sp_setpsexe, sp_showcontrolinfo, sp_showexeclass, sp_showpsexe, sp_unbindexeclass**

sp_clearstats

Function

Initiates a new accounting period for all server users or for a specified user. Prints statistics for the previous period by executing **sp_reportstats**.

Syntax

sp_clearstats [*loginame*]

Parameters

loginame – is the user's login name.

Examples

1. sp_clearstats

```
Name    Since           CPU  Percent CPU     I/O    Percent I/O
------  --------     ------  ----------- -------    -------------
probe   Jun 19 1990       0           0%       0         0%
julie   Jun 19 1990   10000     24.9962%    5000         24.325%
jason   Jun 19 1990   10002     25.0013%    5321         25.8866%
ken     Jun 19 1990   10001     24.9987%    5123         24.9234%
kathy   Jun 19 1990   10003     25.0038%    5111         24.865%
(5 rows affected)
             Total CPU   Total I/O
             ---------      ---------
             40006       20555

             5 login accounts cleared.
```

Initiates a new accounting period for all users.

2. sp_clearstats kathy

```
Name  Since         CPU     Percent CPU     I/O     Percent I/O
----- -----------   -----   ------------    -----   -----------
KATHY Jul 24 1990   498     49.8998%        483924  9.1829%
(1 row affected)
             Total CPU   Total I/O
             ---------   ----------
             998         98392
             1 login account cleared.
```

Initiates a new accounting period for the user "kathy."

Comments

- **sp_clearstats** creates an accounting period and should be run only at the end of a period.
- Because **sp_clearstats** clears out the accounting statistics, you must record the statistics **before** running the procedure.
- **sp_clearstats** updates the *syslogins* field *accdate* and clears the *syslogins* fields *totcpu* and *totio*.

Messages

- `No login with the specified name exists.`

  The *loginame* you specified does not exist in this database.
- *number* `login account(s) cleared.`

  The **sp_clearstats** command initiated a new accounting period for *number* users.

Permissions

Only a System Administrator can execute **sp_clearstats**.

Tables Used

master..syslogins, sysobjects

See Also

System procedures	**sp_reportstats**

sp_commonkey

Function

Defines a common key—columns that are frequently joined—between two tables or views.

Syntax

```
sp_commonkey tabaname, tabbname, col1a, col1b
     [, col2a, col2b, ..., col8a, col8b]
```

Parameters

tabaname – is the name of the first table or view to be joined.

tabbname – is the name of the second table or view to be joined.

col1a – is the name of the first column in the table or view *tabaname* that makes up the common key. Specify at least one pair of columns (one column from the first table or view and one from the second table or view).

col1b – is the name of the partner column in the table or view *tabbname* that is joined with *col1a* in the table or view *tabaname*.

Examples

1. **`sp_commonkey titles, titleauthor, title_id, title_id`**

   Defines a common key on *titles.titleid* and *titleauthor.titleid*.

2. **`sp_commonkey projects, departments, empid, empid`**

   Assumes two tables, *projects* and *departments*, each with a column named *empid*. This statement defines a frequently used join on the two columns.

Comments

- Common keys are created in order to make explicit a logical relationship that is implicit in your database design. The information can be used by an application. **sp_commonkey** does not enforce referential integrity constraints; use the **primary key** and **foreign key** clauses of the **create table** or **alter table** command to enforce key relationships.

- Executing **sp_commonkey** adds the key to the *syskeys* system table. To display a report on the common keys that have been defined, use **sp_helpkey**.
- You must be the owner of at least one of the two tables or views in order to define a common key between them.
- The number of columns from the first table or view must be the same as the number of columns from the second table or view. Up to eight columns from each table or view can participate in the common key. The datatypes of the common columns must also agree. For columns that take a length specification, the lengths can differ. The null types of the common columns need not agree.
- The installation process runs **sp_commonkey** on appropriate columns of the system tables.

Messages

- `First table in the common key doesn't exist.`

  The table or view you gave as *tabaname* does not exist in the current database.

- `New common key added.`

  The common key between the specified tables or views has been added to *syskeys*.

- `Only the table owner may define its common keys.`

  You are not the owner of either *tabaname* or *tabbname*.

- `Second table in the common key doesn't exist.`

  The table or view you gave as *tabbname* does not exist in the current database.

- `Table or view name must be in current database.`

  Either the column pair that you specified does not exist, or the columns in the pair are different types.

- `The tables have no such` *n*`th column or the columns are of different types.`

  Either the column pair that you specified does not exist, or the columns in the pair are different types.

Permissions

Only the owner of *tabaname* or *tabbname* can execute **sp_commonkey**.

Tables Used

syscolumns, *syskeys*, *sysobjects*

See Also

Commands	**alter table**, **create table**, **create trigger**
System procedures	**sp_dropkey**, **sp_foreignkey**, **sp_helpjoins**, **sp_helpkey**, **sp_primarykey**

sp_configure

Function

Displays or changes configuration parameters.

Syntax

```
sp_configure [configname [, configvalue] | group_name
   | non_unique_parameter_fragment]
sp_configure "configuration file", 0, {"write" |
   "read" | "verify" | "restore"} "file_name"
```

Parameters

Syntax	Effect
sp_configure	Displays configuration parameters by group, their current values, their default values, the value to which they have most recently been set, and the amount of memory used by this setting. Displays only the parameters whose display level is the same as or below that of the user.
sp_configure *configname*	Displays the current value, default value, most recently changed value, and amount of memory used by the setting for all parameters matching *parameter*.
sp_configure *configname*, *configvalue*	Resets *configname* to *configvalue* and displaysthe current value, default value, configured value, and amount of memory used by *configname*.
sp_configure *configname*, 0, "default"	Resets *configname* to its default value and displays current value, default value, configured value, and amount of memory used by *configname*.
sp_configure *group_name*	Displays all configuration parameters in *group_name*, their current values, their default values, the value (if applicable) to which they have most recently been set, and the amount of memory used by this setting.
sp_configure *non_unique_parameter_fragment*	Displays all parameter names that match *non_unique_parameter_fragment*, their current values, default values, configured values, and the amount of memory used.
sp_configure "configuration file", 0, "write", "*file_name***"**	Creates *file_name* from the current configuration. If *file_name* already exists, a message is written to the error log and the existing file is renamed using the convention *file_name.001*, *file_name.002*, and so on. If you have changed a static parameter but have not restarted your server, **"write"** gives you the currently running value for that parameter.

Syntax	Effect
sp_configure "configuration file", 0, "read", "*file_name***"**	Performs validation checking on values contained in *file_name* and reads those values that pass validation into the server. If any parameters are missing from *file_name*, the current running values for those parameters are used.
sp_configure "configuration file", 0, "verify", "*file_name***"**	Performs validation checking on the values in *file_name*.
sp_configure "configuration file", 0, "restore", "*file_name***"**	Creates *file_name* with the values in *sysconfigures*. This is useful if all copies of the configuration file have been lost and you need to generate a new copy.

Examples

1. **sp_configure**

   Displays all configuration parameters by group, their current values, their default values, the value (if applicable) to which they have most recently been set, and the amount of memory used by this setting.

2. **sp_configure "identity"**

```
Configuration option is not unique.

Parameter Name              Default   Memory Used Config Value Run Value
--------------------------- --------- ----------- ------------ ---------
identity burning set factor      5000           0         5000      5000
identity grab size                  1           0            1         1
size of auto identity column       10           0           10        10
```

   Displays all configuration parameters that include the word "identity."

3. **sp_configure "recovery interval in minutes", 3**

```
Parameter Name              Default Memory Used Config Value Run Value
--------------------------- ------- ----------- ------------ -----------
recovery interval in minutes      5           0            3           3
Configuration option changed. The SQL Server need not be rebooted
since the option is dynamic.
```

   Sets the system **recovery interval in minutes** to 3 minutes.

4. **sp_configure "number of device", 0, "default"**

   Resets the value for **number of devices** to the Adaptive Server default.

Comments

- Any user can execute **sp_configure** to display information about parameters and their current values, but not to modify parameters. System Administrators can execute **sp_configure** to change the values of most configuration parameters. Only System Security Officers can execute certain parameters. These are listed under "Permissions" in this section.
- When you execute **sp_configure** to modify a dynamic parameter:
  - The configuration and run values are updated.
  - The configuration file is updated.
  - The change takes effect immediately.
- When you execute **sp_configure** to modify a static parameter:
  - The configuration value is updated.
  - The configuration file is updated.
  - The change takes effect only when you restart Adaptive Server.
- When issued with no parameters, **sp_configure** displays a report of all configuration parameters by group, their current values, their default values, the value (if applicable) to which they have most recently been set, and the amount of memory used by this setting:
  - The *default* column in the report displays the value Adaptive Server is shipped with. If you do not explicitly reconfigure a parameter, it retains its default value.
  - The *memory used* column displays the amount of memory used by the parameter at its current value. Some related parameters draw from the same memory pool. For instance, the memory used for **stack size** and **stack guard size** is already accounted for in the memory used for **number of user connections**. If you added the memory used by each of these parameters separately, it would total more than the amount actually used. In the *memory used* column, parameters that "share" memory with other parameters are marked with a hash mark (#).
  - The *config_value* column displays the most recent value to which the configuration parameter has been set with **sp_configure**.
  - The *run_value* column displays the value being used by Adaptive Server. It changes after you modify a parameter's value with **sp_configure** and, for static parameters, after you

restart Adaptive Server. This is the value stored in in *syscurconfigs.value*.

➤ ***Note***

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

- Each configuration parameter has an associated display level. There are three display levels:
  - The "basic" level displays only the most basic parameters. It is appropriate for very general server tuning.
  - The "intermediate" level displays parameters that are somewhat more complex, as well as showing you all the "basic" parameters. This level is appropriate for a moderately complex level of server tuning.
  - The "comprehensive" level displays all parameters, including the most complex ones. This level is appropriate for users who do highly detailed server tuning.

  The default display level is "comprehensive". Setting one of the other display levels lets you work with a subset of the configuration parameter, shortening the amount of information displayed by **sp_configure**.

  The syntax for showing your current display level is:

  sp_displaylevel
- For information on the individual configuration parameters, see Chapter 11, "Setting Configuration Parameters," in the *System Administration Guide*.

Messages

- '*command*' is an invalid file command. The valid commands are 'verify', 'read', 'write', and 'restore'.

  Check the spelling.
- Can't run sp_configure from within a transaction.

  sp_configure modifies system tables, so it cannot be run within a transaction.

- `Configuration option changed. The SQL Server must be rebooted before the change in effect since the option is static.`

  The procedure completed sucessfully. Restart Adaptive Server for the new value to take effect.

- `Configuration option changed. The SQL Server need not be rebooted since the option is dynamic.`

  The procedure completed successfully.

- `Configuration option doesn't exist.`

  Check the spelling of the name you supplied as the *configname* parameter.

- `Configuration option is not unique.`

  The name you supplied as the *configname* parameter is not unique. No configuration parameter was changed. For example, two of the configuration parameters are **recovery interval in minutes** and **print recovery information**. Using **recovery** for the *configname* parameter generates this message because it matches both names. The complete names that match the string supplied are printed out so that you can see how to make the *configname* more specific.

- `Configuration option value is not legal.`

  The *configvalue* you supplied is not in the range of permissible values for the specified configuration parameter. For a display of the range of permissible values, rerun **sp_configure** with the name of the configuration parameter as the only parameter.

- `Maximum file descriptors or FILLM process quota too low to support requested number of user connections. Configuration variable 'user connections' will not be modified.`

  Use this command:

  ```
  select @@max_connections
  ```

  to find the maximum value to which **user connections** can be configured.

- `Must provide the parameter '`*`filename`*`'.`

  Provide a file name for the configuration file.

- `No matching configuration options.  Here is a listing of groups:`

If the *non_unique_parameter_fragment* does not match any configuration options, Adaptive Server lists all configuration groups. No configuration parameter was changed.

- `The character set, 'charset', is invalid since it is not defined in Syscharsets.`

  Check the spelling of the character set.

- `The sortorder, 'sortorder', is invalid since it is not defined in Syscharsets.`

  Check the spelling of the sort order name.

- `The value of the 'number of devices' must be greater than the highest VDEVNO, 'number', defined in sysdevices.`

  Use **sp_helpdevice** to see a list of devices defined for this server.

- `You can't set the default language to a language ID that is not defined in syslanguages.`

  Use **sp_helplanguage** to see the list of official language names available.

Permissions

Any user can execute **sp_configure** to display information about parameters and their current values.

Only System Administrators and System Security Officers can execute **sp_configure** to modify configuration parameters.

Only System Security Officers can execute **sp_configure** to modify values for:

allow procedure grouping
allow select on syscomments.text
allow updates
audit queue size
auditing
current audit table
remote access
suspend auditing when full
systemwide password expiration

System Administrators can modify all other parameters.

Tables Used

master..sysdevices, *master..sysconfigures*, *master..syscurconfigs*, *master..sysdatabases*, *master..sysdevices*, *master..sysindexes*,

master..syslanguages, *master..sysmessages*, *master..sysobjects*, *master..sysservers*

See Also

Commands	**set**
System procedures	**sp_addlanguage**,**sp_audit**, **sp_dboption**, **sp_displaylevel**, **sp_droplanguage**, **sp_helpconfig**, **sp_modifylogin**, **sp_monitorconfig**

sp_countmetadata

Function

Displays the number of indexes, objects, or databases in Adaptive Server.

Syntax

sp_countmetadata "*configname*" [, *dbname*]

Parameters

configname – is either **"open indexes"**, **"open objects"**, or **"open databases"**.

dbname – is the name of the database on which to run **sp_countmetadata**. If no database name is given, **sp_countmetadata** provides a total count for all databases.

Examples

1. sp_countmetadata "open objects"

```
There are 283 user objects in all database(s),
requiring 117.180 Kbytes of memory. The 'open
objects' configuration parameter is currently set
to a run value of 500.
```

Reports on the number of user objects in Adaptive Server. Use this value to set the number of objects allowed in the database, plus space for additional objects and temporary tables. For example:

sp_configure "number of open objects", 310

2. sp_countmetadata "open indexes", pubs2

```
There are 21 user indexes in pubs2 database(s),
requiring 8.613 kbytes of memory. The 'open
indexes' configuration parameter is currently set
to 600.
```

Reports on the number of indexes in SQL Server.

Comments

- **sp_countmetadata** displays the number of indexes, objects, or databases in Adaptive Server, including the number of system databases such as *model* and *tempdb*.

- Avoid running **sp_countmetadata** during Adaptive Server peak times. It can cause contention on the *sysindexes*, *sysobjects*, and *sysdatabases* system tables.
- You can run **sp_countmetadata** on a specified database if you want information on a particular database. However, when configuring caches for indexes, objects, or databases, run **sp_countmetadata** without the *database_name* option.
- The information on memory returned by **sp_countmetadata** can vary by platform. For example, a database on Adaptive Server for Windows NT could have a different **sp_countmetadata** result than the same database on Sun Solaris. Information on the number of user indexes, objects, or databases should be consistent, however.
- **sp_countmetadata** does not include temporary tables in its calculation. Add 5 percent to the **open objects** value and 10 percent to the **open indexes** value to accommodate temporary tables.
- If you specify a nonunique fragment of **"open indexes"**, **"open objects"**, or **"open databases"** for *configname*, **sp_countmetadata** returns a list of matching configuration parameter names with their configured values and current values. For example:

  sp_countmetadata "open"

```
Configuration option is not unique.
 option_name                    config_value run_value
 ------------------------------ ------------ -----------
 curread change w/ open cursors            1           1
 number of open databases                 12          12
 number of open indexes                  500         500
 number of open objects                  500         500
 open index hash spinlock ratio          100         100
 open index spinlock ratio               100         100
 open object spinlock ratio              100         100
```

Messages

- `Can't run sp_countmetadata from within a transaction.`

  sp_countmetadata cannot be run within a transaction because it modifies system tables.
- `Configuration parameter` *`configname`* `is not supported in this system stored procedure.`

  Use **"open indexes"**, **"open objects"**, or **"open databases"** for *configname*.

- There are *number* databases, requiring *number* Kbytes of memory. The ‘open databases’ configuration parameter is currently set to *value*.
- There are *number* user indexes in *number* database(s), requiring *number* Kbytes of memory. The ‘open indexes’ configuration parameter is currently set to *value*.
- There are *nn* user objects in *mm* database(s), requiring *number* Kbytes of memory. The 'open objects' configuration parameter is currently set to *value*.

Permissions

Only a System Administrator or the Database Owner can use **sp_countmetadata.**

Tables Used

master..sysdatabases, sysindexes, sysobjects

See Also

System procedures	**sp_configure**, **sp_helpconfig, sp_monitorconfig**

sp_cursorinfo

Function

Reports information about a specific cursor or all cursors that are active for your session.

Syntax

```
sp_cursorinfo [{cursor_level | null}] [, cursor_name]
```

Parameters

cursor_level | **null** – is the level at which Adaptive Server returns information for the cursors. You can specify the following for *cursor_level*:

Level	Types of Cursors
N	Any cursors declared inside stored procedures at a specific procedure nesting level. You can specify any positive number for its level.
0	Any cursors declared outside stored procedures.
-1	Any cursors from either of the above. You can substitute any negative number for this level.

If you want information about cursors with a specific *cursor_name*, regardless of cursor level, specify **null** for this parameter.

cursor_name – is the specific name for the cursor. Adaptive Server reports information about all active cursors that use this name at the *cursor_level* you specify. If you omit this parameter, Adaptive Server reports information about all the cursors at that level.

Examples

1. sp_cursorinfo 0, authors_crsr

```
Cursor name 'authors_crsr' is declared at nesting level '0'.
The cursor id is 327681
The cursor has been successfully opened 1 times.
The cursor was compiled at isolation level 0.
The cursor is not open.
The cursor will remain open when a transaction is commited or
rolled back.
The number of rows returned for each FETCH is 1.
The cursor is read only.
There are 3 columns returned by this cursor.
The result columns are:
Name = 'au_id', Table = 'authors', Type = ID,
     Length = 11 (read only)
Name = 'au_lname', Table = 'authors', Type = VARCHAR,
Length = 40 (read only)
Name = 'au_fname', Table = 'authors', Type = VARCHAR,
Length = 20 (read only)
```

Displays the information about the cursor named *authors_crsr* at level 0.

2. sp_cursorinfo null, author_sales

```
Cursor name 'author_sales' is declared on procedure 'au_sales'.
Cursor name 'author_sales' is declared at nesting level '1'.
The cursor id is 327682
The cursor has been successfully opened 1 times.
The cursor was compiled at isolation level 1.
The cursor is currently scanning at a nonzero isolation level.
The cursor is positioned after the last row.
The cursor will be closed when a transaction is commited or
rolled back.
The number of rows returned for each FETCH is 1.
The cursor is updatable.
There are 3 columns returned by this cursor.
The result columns are:
Name = 'title_id', Table = 'titleauthor', Type = ID,
      Length = 11 (updatable)
Name = 'title', Table = 'titles', Type = VARCHAR,
      Length = 80 (updatable)
Name = 'total_sales', Table = 'titles', Type = INT (updatable)
```

Displays the information about any cursors named *author_sales* declared by a user across all levels.

Comments

- If you do not specify either *cursor_level* or *cursor_name*, Adaptive Server displays information about all active cursors. Active cursors are those declared by you and allocated by Adaptive Server.
- Adaptive Server reports the following information about each cursor:
  - The cursor name, its nesting level, its cursor ID, and the procedure name (if it is declared in a stored procedure).
  - The number of times the cursor has been opened.
  - The isolation level (0, 1, or 3) in which it was compiled and in which it is currently scanning (if open).
  - Whether the cursor is open or closed. If the cursor is open, it indicates the current cursor position and the number of rows fetched.
  - Whether the open cursor will be closed if the cursor's current position is deleted.
  - Whether the cursor will remain open or be closed if the cursor's current transaction is committed or rolled back.
  - The number of rows returned for each fetch of that cursor.
  - Whether the cursor is updatable or read-only.
  - The number of columns returned by the cursor. For each column, it displays the column name, the table name or expression result, and whether it is updatable.

  In addition to the above, **sp_cursorinfo** displays the **showplan** output for the cursor. See Chapter 9, "Understanding Query Plans," in the *Performance and Tuning Guide* for more information about **showplan**. The output from **sp_cursorinfo** varies, depending on the status of the cursor.

Messages

- `There are no active cursors.`

  Adaptive Server cannot find any declared cursors.
- `There are no active cursors that match the search criteria.`

  Adaptive Server cannot find any declared cursors that match the values you specified for *cursor_level* and *cursor_name*.

Permissions

Any user can execute **sp_cursorinfo.**

Tables Used

sysobjects

See Also

Commands	**declare cursor**, **set**

sp_dboption

Function

Displays or changes database options.

Syntax

```
sp_dboption [dbname, optname, {true | false}]
```

Parameters

dbname – is the name of the database in which the option is to be set. You must be using *master* to execute **sp_dboption** with parameters (that is, in order to change a database option). You cannot, however, change option settings in the *master* database.

optname – is the name of the option to be set. Adaptive Server understands any unique string that is part of the option name. Use quotes around the option name if it is a keyword or includes embedded blanks or punctuation.

{**true** | **false**} – **true** to turn the option on, **false** to turn it off.

Examples

1. sp_dboption

```
Settable database options

 database_options
 ------------------------
 abort tran on log full
 allow nulls by default
 auto identity
 dbo use only
 ddl in tran
 identity in nonunique index
 no chkpt on recovery
 no free space acctg
 read only
 select into/bulkcopy/pllsort
 single user
 trunc log on chkpt
 trunc. log on chkpt.
 unique auto_identity index
```

Displays a list of the database options.

2.
```
use pubs2
go
master..sp_dboption pubs2, "read", true
go
checkpoint
go
```

   Makes the database *pubs2* read only. The **read** string uniquely identifies the **read only** option from among all available database options. Note the use of quotes around the keyword **read**.

3.
```
pubs2..sp_dboption pubs2, "read", false
go
checkpoint
go
```

   Makes the database *pubs2* writable again.

4.
```
use pubs2
go
master..sp_dboption pubs2, "select into", true
go
checkpoint
go
```

   Allows **select into**, **bcp** and parallel sort operations on tables in the *pubs2* database. The **select into** string uniquely identifies the **select into/ bulkcopy** option from among all available database options. Note that quotes are required around the option because of the embedded space.

5.
```
use mydb
go
master..sp_dboption mydb, "auto identity", true
go
checkpoint
go
```

   Automatically defines 10-digit IDENTITY columns in new tables created in *mydb*. The IDENTITY column, *SYB_IDENTITY_COL*, is defined in each new table that is created without specifying either a primary key, a **unique** constraint, or an IDENTITY column.

```
6. use master
   go
   sp_dboption mydb, "identity in nonunique index",
   true
   go
   use mydb
   go
   checkpoint
   go
```

Automatically includes an IDENTITY column in the *mydb* tables' index keys, provided these tables already have an IDENTITY column. All indexes created on the tables will be internally unique.

```
7. use master
   go
   sp_dboption pubs2, "unique auto_identity index",
   true
   go
   use pubs2
   go
   checkpoint
   go
```

Automatically includes an IDENTITY column with a unique, nonclustered index for new tables in the *pubs2* database.

Comments

- The *master* database option settings cannot be changed.
- To display a list of database options, execute **sp_dboption** with no parameters from inside the *master* database.
- For a report on which database options are set in a particular database, execute **sp_helpdb**.
- The Database Owner or System Administrator can set or unset particular database options for all new databases by executing **sp_dboption** on *model*.
- After **sp_dboption** has been executed, the change does not take effect until the **checkpoint** command is issued in the database for which the option was changed.

Database Options

- The **abort tran on log full** option determines the fate of a transaction that is running when the last-chance threshold is crossed in the log segment of the specified database. The default value is **false**,

meaning that the transaction is suspended and is awakened only when space has been freed. If you change the setting to **true**, all user queries that need to write to the transaction log are killed until space in the log has been freed.

- Setting the **allow nulls by default** option to **true** changes the default value of a column from **not null** to **null**, in compliance with the SQL standards. The Transact-SQL default value for a column is **not null**, meaning that null values are not allowed in a column unless **null** is specified in the **create table** or **alter table** column definition. **allow nulls by default true** reverses this.
- While the **auto identity** option is set to **true** (on), a 10-digit IDENTITY column is defined in each new table that is created without specifying either a **primary** key, a **unique** constraint, or an IDENTITY column. The column is not visible when you select all columns with the **select *** statement. To retrieve it, you must explicitly mention the column name, *SYB_IDENTITY_COL*, in the select list.

  To set the precision of the automatic IDENTITY column, use the **size of auto identity column** configuration parameter.

  Though you can set **auto identity** to **true** in *tempdb*, it is not recognized or used, and temporary tables created there do not automatically include an IDENTITY column.

  For a report on indexes in a particular table that includes the IDENTITY column, execute **sp_helpindex**.
- While the **dbo use only** option is set to **true** (on), only the database's owner can use the database.
- When the **ddl in tran option** is set to **true** (on), you can use certain data definition language commands in transactions. If **ddl in tran** is **true** in a particular database, commands such as **create table**, **grant**, and **alter table** are allowed inside transactions in that database. If **ddl in tran** is **true** in the *model* database, the commands are allowed inside transactions in all databases created after **ddl in tran** was set in *model*.

◆ ***WARNING!***

Data definition language commands hold locks on system tables such as *sysobjects*. Avoid using them inside transactions; if you must use them, keep the transactions short.

Using any data definition language commands on *tempdb* within transactions may cause your system to grind to a halt. Always leave ddl in tran set to false in *tempdb*.

- Table 3-15 lists the commands that can be used inside a user-defined transaction when the **ddl in tran** option is set to **true**:

Table 3-15: DDL commands allowed in transactions

alter table (clauses other than **partition** and **unpartition** are allowed)	**create default**	**drop default**	**grant**
	create index	**drop index**	**revoke**
	create procedure	**drop procedure**	
	create rule	**drop rule**	
	create schema	**drop table**	
	create table	**drop trigger**	
	create trigger	**drop view**	
	create view		

- Table 3-16 lists the commands that cannot be used inside a user-defined transaction under any circumstances:

Table 3-16: DDL commands not allowed in transactions

alter database	**dump database**	**select into**
alter table...partition	**dump transaction**	**truncate table**
alter table...unpartition	**drop database**	**update statistics**
create database	**load transaction**	
disk init	**load database**	

  In addition, system procedures that create temporary tables or change the *master* database cannot be used inside user-defined transactions.

- The **identity in nonunique index** option automatically includes an IDENTITY column in a table's index keys, so that all indexes created on the table are unique. This database option makes logically nonunique indexes internally unique, and allows these indexes to be used to process updatable cursors and isolation level 0 reads.

The table must already have an IDENTITY column for the **identity in nonunique index** option to work, either from a **create table** statement or by setting the **auto identity** database option to **true** before creating the table.

Use **identity in nonunique index** if you plan to use cursors and isolation level 0 reads on tables with nonunique indexes. A unique index ensures that the cursor will be positioned at the correct row the next time a **fetch** is performed on that cursor. If you plan to use cursors on tables with unique indexes and any isolation level, you may want to use the **unique auto_identity index** option.

For a report on indexes in a particular table that includes the IDENTITY column, execute **sp_helpindex**.

- The **no free space acctg** option suppresses free-space accounting and execution of threshold actions for the non-log segments. This speeds recovery time because the free-space counts are not recomputed for those segments.
- The **no chkpt on recovery** option is set to **true** (on) when an up-to-date copy of a database is kept. In these situations, there is a "primary" and a "secondary" database. Initially, the primary database is dumped and loaded into the secondary database. Then, at intervals, the transaction log of the primary database is dumped and loaded into the secondary database.

  If this option is set to **false** (off), the default condition, a checkpoint record is added to a database after it is recovered when you restart Adaptive Server. This checkpoint, which ensures that the recovery mechanism will not be rerun unnecessarily, changes the sequence number and causes a subsequent load of the transaction log from the primary database to fail.

  Setting this option to **true** (on) for the secondary database causes it not to get a checkpoint from the recovery process so that subsequent transaction log dumps from the primary database can be loaded into it.
- The **read only** option means that users can retrieve data from the database, but cannot modify any data.
- Setting the **select into/bulkcopy/pllsort** option to **true** (on) enables the use of **writetext**, **select into** a permanent table, "fast" bulk copy into a table that has no indexes or triggers, using **bcp** or the bulk copy library routines, and parallel sort. A transaction log dump cannot recover these minimally logged operations, so **dump transaction** to a

dump device is prohibited. After non-logged operations are completed, set **select into/bulk copy/pllsort** to **false** (off) and issue **dump database**.

Issuing the **dump transaction** statement after unlogged changes have been made to the database with **select into**, bulk copy, or parallel sort produces an error message instructing you to use **dump database** instead. (The **writetext** command does not have this protection.)

You do not have to set the **select into/bulkcopy/pllsort** option to **true** in order to **select into** a temporary table, since *tempdb* is never recovered. The option need not be set to **true** in order to run **bcp** on a table that has indexes, because tables with indexes are always copied with the slower version of bulk copy and are logged.

- When **single user** is set to **true**, only one user at a time can access the database (single-user mode).

  You cannot set **single user** to **true** in a user database from within a stored procedure or while users have the database open. You cannot set **single user** to **true** for *tempdb*.

- The **trunc log on chkpt** option means that if the transaction log has more than 50 rows of committed transactions, the transaction log is truncated (the committed transactions are removed) every time the **checkpoint** checking process occurs (usually more than once per minute). When the Database Owner runs **checkpoint** manually, however, the log is **not** truncated. It may be useful to turn this option on while doing development work, to prevent the log from growing.

  While the **trunc log on chkpt** option is on, **dump transaction** to a dump device is prohibited, since dumps from the truncated transaction log cannot be used to recover from a media failure. Issuing the **dump transaction** statement produces an error message instructing you to use **dump database** instead.

- When the **unique auto_identity index** option is set to **true**, it adds an IDENTITY column with a unique, nonclustered index to new tables. By default, the IDENTITY column is a 10-digit numeric datatype, but you can change this default with the **size of auto identity column** configuration parameter. As with **auto identity**, the IDENTITYcolumn is not visible when you select all columns with the **select *** statement. To retrieve it, you must explicitly mention the column name, *SYB_IDENTITY_COL*, in the select list.

If you need to use cursors or isolation level 0 reads with nonunique indexes, use the **identity in nonunique index** option.

Though you can set **unique auto_identity index** to **true** in *tempdb*, it is not recognized or used, and temporary tables created there do not automatically include an IDENTITY column with a unique index.

- See Chapter 16, "Setting Database Options," in the *System Administration Guide* for additional information on database options.

Messages

- `Can't run sp_dboption from within a transaction.`

  sp_dboption modifies system tables, so it cannot be run within a transaction.

- `Database option 'option_name' turned [OFF | ON] for database 'dbname'.`

  The **sp_dboption** command succeeded. This message reports on the option you have just set.

- `Database option doesn't exist or can't be set by user.`

  Either the option does not exist or the user does not have permission to set it. Run **sp_dboption** with no parameters to display a list of the options.

- `Database option is not unique.`

  The name supplied as the *optname* parameter is not unique. No database option value was changed. For example, two of the database options are **dbo use only** and **read only.** Using **only** for the *optname* parameter generates this message because it matches both names. **sp_dboption** prints out the complete names that match the string so you can see how to make the *optname* more specific.

- `No such database–run sp_helpdb to list databases.`

  No database with the supplied name exists. Run **sp_helpdb** to get a list of databases.

- `Run the CHECKPOINT command in the database that was changed.`

  The change in the database option takes effect only after the **checkpoint** command is run.

- `Settable database options.`

  Executing **sp_dboption** with no parameters displays a list of database options that can be set by the System Administrator or Database Owner.

- `The database is currently in use -- 'read only' option disallowed.`

  You must wait until no one is using the database before issuing this command. Use **sp_who** to monitor usage.

- `The 'master' database's options cannot be changed.`

  No one can change any of the *master*'s database option settings.

- `Usage: sp_dboption [dbname, optname, {true | false}]`

  Either the *optname* parameter was omitted or the third parameter was something other than TRUE or FALSE.

- `You must be in the 'master' database in order to change database options.`

  In order to change a database option (of any database other than *master*), execute the **sp_dboption** procedure, with the appropriate parameters, while using *master*.

- `Run sp_dboption with no parameters to see options.`

  The command failed. Check the spelling of the options and run **sp_dboption** again.

Permissions

Any user can view the database options by executing **sp_dboption** with no parameters. Only a System Administrator or the Database Owner can change options by executing **sp_dboption** with parameters. A user aliased to the Database Owner cannot change options with **sp_dboption.**

Tables Used

master.dbo.sysdatabases, *master.dbo.sysmessages*, *master.dbo.sysprocesses*, *sysobjects*

See Also

Commands	**checkpoint**, **select**
System procedures	**sp_configure**, **sp_helpdb**, **sp_helpindex**, **sp_helpjoins**

sp_dbremap

Function

Forces Adaptive Server to recognize changes made by **alter database**. Run this procedure only when instructed to do so by an Adaptive Server message.

Syntax

```
sp_dbremap dbname
```

Parameters

dbname – is the name of the database in which the **alter database** command was interrupted.

Examples

1. `sp_dbremap sample_db`

An **alter database** command changed the database *sample_db*. This command makes the changes visible to Adaptive Server.

Comments

- If an **alter database** statement issued on a database that is in the process of being dumped is interrupted, Adaptive Server prints a message instructing the user to execute **sp_dbremap**.
- Any changes to *sysusages* during a database or transaction dump are not copied into active memory until the dump completes, to ensure that database mapping does not change during the dump. Running **alter database** makes changes to system tables on the disk immediately. In-memory allocations cannot be changed until a dump completes. This is why **alter database** pauses.

  When you execute **sp_dbremap**, it must wait until the dump process completes.
- If you are instructed to run **sp_dbremap**, but do not do it, the space you have allocated with **alter database** does not become available to Adaptive Server until the next restart.

Messages

- `Can't run sp_dbremap from within a transaction.`

  sp_dbremap modifies system tables, so it cannot be run within a transaction.

- '*dbname*' is not a valid identifier.

  The database name you specified is not a valid identifier.
- The specified database does not exist

  The database name you specified is not the name of a database on this server.

Permissions

Only a System Administrator or Database Owner can execute **sp_dbremap**.

Tables Used

master.dbo.sysdatabases, sysobjects

See Also

Commands	**alter database**, **dump database**, **dump transaction**

sp_defaultloc

(Component Integration Services only)

Function

Defines a default storage location for objects in a local database.

Syntax

```
sp_defaultloc dbname, {"defaultloc"| NULL}
   [, "defaulttype"]
```

Parameters

dbname – is the name of a database being mapped to a remote storage location. The database must already have been defined by a **create database** statement. You cannot map system databases to a remote location.

defaultloc – is the remote storage location to which the database is being mapped. To direct the server to delete an existing default mapping for a database, supply NULL for this parameter. The value of *defaultloc* must end in a period (.), as follows:

```
server.dbname.owner.
```

defaulttype – is one of the values that specify the format of the object named by *object_loc*. Table 3-17 describes the valid values. Enclose the *defaulttype* value in quotes.

Table 3-17: Allowable values for defaulttype

Value	Description
table	Indicates that the object named by *object_loc* is a table accessible to a remote server. This value is the default for *defaulttype*.
view	Indicates that the object named by *object_loc* is a view managed by a remote server, processed as a table.
rpc	Indicates that the object named by *object_loc* is an RPC managed by a remote server; processes the result set from the RPC as a read-only table.

Examples

1. **`sp_defaultloc pubs, "SYBASE.pubs.dbo.", "table"`**
   `create table pubs.dbo.book1 (bridges char(15))`

   sp_defaultloc defines the remote storage location *pubs.dbo.* in the remote server named SYBASE. It maps the database *pubs* to the remote location. A "**create table** book1" statement would create a table named *book1* at the remote location. A **create existing table** statement for *bookN* would require that *pubs.dbo.bookN* already exist at the remote location, and information about table *bookN* would be stored in the local table *bookN*.

2. **`sp_defaultloc pubs, NULL`**

   Removes the mapping of the database *pubs* to a remote location.

3. **`sp_defaultloc ticktape,"wallst.nasdaq.dbo.", "rpc"`**
   `create existing table sybase (bestbuy integer)`

   Identifies the remote storage location *wallst.nasdaq.dbo* where "wallst" is the value provided for *server_name*, "nasdaq" is provided for *database*, and "dbo" is provided for *owner*. The RPC *sybase* must already exist at the remote location. A "**create existing table** sybase" statement would store information about the result set from RPC *sybase* in local table *ticktape*. The result set from RPC *sybase* is regarded as a read-only table. Inserts, updates and deletes are not supported for RPCs.

Comments

- **sp_defaultloc** defines a default storage location for tables in a local database. It maps table names in a database to a remote location. It permits the user to establish a default for an entire database, rather than issue an **sp_addobjectdef** command before every **create table** and **create existing table** command.
- When *defaulttype* is **table**, **view**, or **rpc**, the *defaultloc* parameter takes the form:

  `server_name.dbname.owner.`

  - Note that the *defaultloc* specification ends in a period (.).
  - *server_name* represents a server already added to *sysservers* by **sp_addserver**. The *server_name* parameter is required.
  - *dbname* might not be required. Some server classes do not support it.
  - *owner* should always be provided to avoid ambiguity. If it is not provided, the remote object actually referenced could vary,

depending on whether the external login corresponds to the remote object owner.

- Issue **sp_defaultloc** before any **create table** or **create existing table** statement. When either statement is used, the server uses the *sysattributes* table to determine whether any table mapping has been specified for the object about to be created or defined. If the mapping has been specified, a **create table** statement directs the table to be created at the location specified by *object_loc*. A **create existing table** statement stores information about the existing remote object in the local table.
- If you issue **sp_defaultloc** on *defaulttype* **view**, and then issue **create table**, Component Integration Services creates a new table, not a view, on the remote server.
- Changing the default location for a database does not affect tables that have previously been mapped to a different default location.
- After tables in the database have been created, all future references to tables in *dbname* (by **select**, **insert**, **delete,** and **update**) are mapped to the correct location.

Messages

- `An RMS filespec for a directory must be specified.`

  For an *defaulttype* of **file**, the *defaultloc* must be an RMS directory specification ending in either a right bracket (]) or a colon (:).

- `Created default location for database` *`dbname`* `at` *`defaultloc`*`.`

  The requested default storage location has been defined for the database.

- `Database` *`dbname`* `already has a default location` *`olddefaultloc`*`.`

  The command failed. The database already has a default storage location definition in *sysdatabases*. To delete an existing definition, execute **sp_defaultloc** with NULL for *defaultloc*.

- `Default storage location deleted for database` *`dbname`*`.`

  The value for *defaultloc* is NULL, the *dbname* is valid, and the requested deletion of a default storage location is complete.

- `No default storage location in effect for` *`dbname`*`.`

  The command failed. The value specified for *defaultloc* is NULL, but there is no entry in *sysdatabases* naming a default storage

location for the database specified in *dbname*. There is nothing to nullify.

- `Storage location syntax must end in a '.'.`

  For an *defaulttype* of **table**, **view**, or **rpc**, the *defaultloc* must be syntax for a remote storage location, ending in a period (.).

- `There is not a database named dbname.`

  The *dbname* specification must identify a database that has been created with a **create database** statement. Use **sp_helpdb** for a list of defined databases.

- `There is not an object type named defaulttype.`

  The value for *defaulttype* must be **table**, **view**, or **rpc**. For details on these values, see Table 3-17.

- `You cannot re-map objects in the master, model, or tempdb databases.`

  You cannot define a default storage location for the *master*, *model*, or *tempdb* system databases.

Permissions

sp_defaultloc permission defaults to "public."

Tables Used

master.dbo.sysservers, *master.dbo.spt_values*, *master.dbo.sysattributes*

See Also

Commands	**create existing table**, **create table**
System procedures	**sp_addobjectdef**,**sp_addserver**, **sp_help**, **sp_helpserver**

sp_depends

Function

Displays information about database object dependencies—the view(s), trigger(s), and procedure(s) in the database that depend on a specified table or view, and the table(s) and view(s) in the database on which the specified view, trigger, or procedure depends.

Syntax

sp_depends *objname*

Parameters

objname – is the name of the table, view, stored procedure, or trigger to be examined for dependencies. You cannot specify a database name. Use owner names if the object owner is not the user running the command and is not the Database Owner.

Examples

1. sp_depends sysobjects

Lists the database objects that depend on the table *sysobjects*.

2. sp_depends titleview

```
Things that the object references in the current
database.

object          type        updated selected
--------------  ----------- ------- -----
dbo.authors     user table  no      no
dbo.titleauthor user table  no      no
dbo.titles      user table  no      no

Things inside the current database that reference
the object.

object          type
------------    ---------------
dbo.tview2      view
```

Lists the database objects that depend on the *titleview* view, and the database objects on which the *titleview* view depends.

3. sp_depends "mary.titles"

Lists the database objects that depend on the *titles* table owned by the user “mary”. The quotes are needed, since the period is a special character.

Comments

- Executing **sp_depends** lists all objects in the current database that depend on *objname*, and on which *objname* depends. For example, views depend on one or more tables and can have procedures or other views that depend on them. An object that references another object is dependent on that object. References to objects outside the current database are not reported.
- The **sp_depends** procedure determines the dependencies by looking at the *sysdepends* table.

  If the objects were created out of order (for example, if a procedure that uses a view was created before the view was created), no rows exist in *sysdepends* for the dependencies, and **sp_depends** does not report the dependencies.
- The *updated* and *selected* columns in the report from **sp_depends** are meaningful if the object being reported on is a stored procedure or trigger. The values for the *updated* column indicate whether the stored procedure updates the object. The *selected* column indicates whether the object is being used for a read cursor or a data modification statement.
- **sp_depends** follows these Adaptive Server rules for finding objects:
  - If the user does not specify an owner name, and the user executing the command owns an object with the specified name, that object is used.
  - If the user does not specify an owner name, and the user does not own an object of that name, but the Database Owner does, the Database Owner's object is used.
  - If neither the user nor the Database Owner owns an object of that name, the command reports an error condition, even if an object exists in the database with that object name, but with a different owner.
  - If both the user and the Database Owner own objects with the specified name, and the user wants to access the Database Owner's object, the name must be specified, as in *dbo.objectname*.
- Objects owned by database users other than the user executing a command and the Database Owner must always be qualified with the owner's name, as in example 3.

Messages

- `Object does not exist in this database.`

  The object name supplied for the *objname* parameter does not exist in the current database.

- `Object doesn't reference any object and no objects reference it.`

  Nothing depends upon *objname*, and *objname* does not reference any objects.

- `Object must be in the current database.`

  You cannot reference an object that is not in your current database.

Permissions

All users can execute **sp_depends**.

Tables Used

master.dbo.spt_values, *master.dbo.sysmessages*, *sysdepends*, *sysobjects*, *sysusers*

See Also

Commands	**create procedure**, **create table**, **create view**, **execute**
System procedures	**sp_help**

sp_diskdefault

Function

Specifies whether or not a database device can be used for database storage if the user does not specify a database device or specifies **default** with the **create database** or **alter database** commands.

Syntax

```
sp_diskdefault logicalname, {defaulton | defaultoff}
```

Parameters

logicalname – is the logical name of the device as given in *master.dbo.sysdevices.name*. The device must be a database device rather than a dump device.

defaulton | **defaultoff** – **defaulton** designates the database device as a default database device; **defaultoff** designates that the specified database device is not a default database device.

Use **defaulton** after adding a database device to the system with **disk init**. Use **defaultoff** to change the default status of the master device (which is designated as a default device when Adaptive Server is first installed).

Examples

```
1. sp_diskdefault master, defaultoff
```

The master device is no longer used by **create database** or **alter database** for default storage of a database.

Comments

- A default database device is one that is used for database storage by **create database** or **alter database** if the user does not specify a database device name or specifies the keyword **default**.
- You can have multiple default devices. They are used in the order they appear in the *master.dbo.sysdevices* table (that is, alphabetical order). When the first default device is filled, the second default device is used, and so on.
- When you first install Adaptive Server, the master device is the only default database device.

➤ ***Note***

Once you initialize devices to store user databases, use **sp_diskdefault** to turn off the master device's default status. This prevents users from accidentally creating databases on the master device and simplifies recovery of the *master* database.

- To find out which database devices are default database devices, execute **sp_helpdevice**.

Messages

- `Can't run sp_diskdefault from within a transaction.`

  sp_diskdefault modifies system tables, so it cannot be run within a transaction.

- `No such device exists -- run sp_helpdevice to list the Adaptive Server devices.`

  The device name supplied for the *logicalname* parameter does not exist. Run **sp_helpdevice** without a parameter to see a list of all devices. To add a new database device to the system, use the **disk init** command.

- `The device name supplied is not a database disk.`

  The device name supplied for the *logicalname* parameter is in *sysdevices*, but it is a dump device rather than a database device. Run **sp_helpdevice** without a parameter to see a list of all devices. To add a new database device to the system, use the **disk init** command.

- `Usage: sp_diskdefault logicalname {defaulton | defaultoff}.`

  The second parameter must be either **defaulton** or **defaultoff**.

Permissions

Only a System Administrator can execute **sp_diskdefault**.

Tables Used

master.dbo.sysdevices, sysobjects

See Also

Commands	**alter database**, **create database**, **disk init**
System procedures	**sp_helpdevice**

sp_displayaudit

Function

Displays the status of audit options.

Syntax

```
sp_displayaudit ["procedure" | "object" | "login" |
   "database" | "global" | "default_object" |
   "default_procedure" [, "name"]]
```

Parameters

procedure – displays the status of audit options for the specified stored procedure or trigger. If you do not specify a value for *name*, **sp_displayaudit** displays the active audit options for all procedures and triggers in the current database.

object – displays the status of audit options for the specified table or view. If you do not specify a value for *name*, **sp_displayaudit** displays the active audit options for all tables and views in the current database.

login – displays the status of audit options for the specified user login. If you do not specify a value for *name*, **sp_displayaudit** displays the active audit options for all logins in the *master* database.

database – displays the status of audit options for the specified database. If you do not specify a value for *name*, **sp_displayaudit** displays the active audit options for all databases on the server.

global – displays the status of the specified global audit option. If you do not specify a value for *name*, **sp_displayaudit** displays the active audit options for all procedures and triggers in the current database.

default_object – displays the default audit options that will be used for any new table or view created on the specified database. If you do not specify a value for *name*, **sp_displayaudit** displays the default audit options for all databases with active default audit settings.

default_procedure – displays the default audit options that will be used for any new procedure or trigger created on the specified database. If you do not specify a value for *name*, **sp_displayaudit** displays the default audit options for all databases with active default audit settings.

name – is the information for the specified parameter, as described in the following table:

Parameter	Value for *name*
procedure	Procedure or trigger name
object	Table or view name
login	Uuser login
database	Database name
global	Global audit option
default_object	Database name
default_procedure	Database name

Examples

1. sp_displayaudit

```
Procedure/Trigger    Audit Option   Value Database
 -----------------   -------------- ----- ---------------------
 dbo.sp_altermessage exec_procedure on    sybsystemprocs
 dbo.sp_help         exec_procedure on    sybsystemprocs
 dbo.sp_who          exec_procedure on    sybsystemprocs
No databases currently have default sproc/trigger auditing
enabled.
No objects currently have auditing enabled.
No databases currently have default table/view auditing enabled.
No logins currently have auditing enabled.
No databases currently have auditing enabled.

Option Name                    Value
------------------------------ ------------------------------
adhoc                          off
dbcc                           off
disk                           off
errors                         off
login                          off
logout                         off
navigator_role                 off
oper_role                      off
replication_role               off
rpc                            off
sa_role                        off
security                       off
sso_role                       off
```

When no parameter is specified, the status of each category and all auditing options is displayed.

2. sp_displayaudit "procedure"

```
Procedure/Trigger    Audit Option   Value Database
 -----------------   -------------- ----- ---------------------
 dbo.sp_altermessage exec_procedure on    sybsystemprocs
 dbo.sp_help         exec_procedure on    sybsystemprocs
 dbo.sp_who          exec_procedure on    sybsystemprocs
```

When no procedure name is specified, the status of all procedure audit options is displayed.

3. sp_displayaudit "procedure", "sp_who"

```
Procedure/Trigger Audit Option    Value Database
----------------- -------------- ----- ---------------------
dbo.sp_who        exec_procedure  on    sybsystemprocs
```

When you specify a name for the procedure, only the status of that procedure is displayed.

4. sp_displayaudit "global"

```
Option Name                    Value
------------------------------ ------------------------------
adhoc                          off
dbcc                           off
disk                           off
errors                         off
login                          off
logout                         off
navigator_role                 off
oper_role                      off
replication_role               off
rpc                            off
sa_role                        off
security                       off
sso_role                       off
```

When no global audit option is specified, the status of all global audit options is displayed.

Comments

- **sp_displayaudit** displays the status of audit options.

- The following table shows the valid auditing options for each parameter:

Object Type Parameter	Valid Auditing Options
procedure	exec_procedure, exec_trigger
object	delete, func_obj_access, insert, reference, select, update
login	all, cmdtext, table_access, view_access
database	alter, bcp, bind, create, dbaccess, drop, dump, func_dbaccess, grant, load, revoke, setuser, truncate, unbind
global	adhoc, dbcc, disk, errors, login, logout, navigator_role, oper_role, replication_role, rpc, sa_role, security, sso_role
default_object	delete, func_obj_access, insert, reference, select, update
default_procedure	exec_procedure, exec_trigger

- You cannot specify a value for name unless you first specify an object type parameter.
- For information on setting up auditing, see Chapter 8, "Auditing," in the *Security Administration Guide*.

Messages

- `'name' does not exist.`

  The database or login name supplied for the *name* parameter does not exist on the server.

- `'name' is not in the current database.`

  The procedure, trigger, table, or view name supplied for the *name* parameter does not exist in the current database.

- `Audit option 'option' does not exist. Valid options are:`

  The option name supplied for the *name* parameter does not exist.

- `Audit option option is ambiguous. Ambiguous options are:`

  If the *non_unique_parameter_fragment* does not match any audit options, Adaptive Server lists all audit options.

- `No databases currently have auditing enabled.`

  Auditing is not enabled for any databases in the server.
- `No databases currently have default sproc/trigger auditing enabled.`

  Auditing is not enabled for any default stored procedures or default triggers.
- `No databases currently have` *`option`* `auditing enabled.`

  Auditing is not enabled for the specified auditing option.
- `No logins currently have auditing enabled.`

  Auditing is not enabled for any user logins.
- `No objects currently have auditing enabled.`

  Auditing is not enabled for any tables or views.
- `No procs/triggers currently have auditing enabled.`

  Auditing is not enabled for any stored procedures or triggers.
- `Object does not exist in this database.`

  The object name supplied for the *name* parameter does not exist in the current database.

Permissions

Only the System Security Officer can use **sp_displayaudit**.

Tables Used

sysauditoptions, sysdatabases, syslogins, sysobjects

See Also

System procedures	**sp_audit**

sp_displaylevel

Function

Sets or shows which Adaptive Server configuration parameters appear in **sp_configure** output.

Syntax

```
sp_displaylevel [loginame [, level]]
```

Parameters

loginame – is the Adaptive Server login of the user for whom you want to set or show the display level.

level – sets the display level. Can be **basic**, **intermediate**, or **comprehensive**.

"basic" display level shows just the most basic configuration parameters. This level is appropriate for very general server tuning.

"intermediate" display level shows configuration parameters that are somewhat more complex, as well as all the "basic" level parameters. This level is appropriate for moderately complex server tuning.

"comprehensive" display level shows all configuration parameters, including the most complex ones. This level is appropriate for highly detailed server tuning.

Examples

1. **sp_displaylevel**

   ```
   The current display level for login 'sa' is
   'comprehensive'.
   ```

   Shows the current display level for the user who invoked **sp_displaylevel**.

2. **sp_displaylevel jerry**

   ```
   The current display level for login 'jerry' is
   'intermediate'.
   ```

   Shows the current display level for the user "jerry".

3. **sp_displaylevel jerry, comprehensive**

   ```
   The display level for login 'jerry' has been
   changed to 'comprehensive'.
   ```

Sets the display level to “comprehensive” for the user “jerry”.

Messages

- `Can’t run sp_displaylevel from within a transaction.`

  sp_displaylevel modifies system tables, so it cannot be run within a transaction.

- `The login '`*`loginame`*`' does not exist.`

  Check the spelling of the user’s name and reissue the command.

- `The current display level for login '`*`loginame`*`' is '`*`level`*`'.`

  The **sp_displaylevel** command succeeded. This message reports on the user’s current display level.

- `Invalid display level. The valid values are 'comprehensive', 'basic', or 'intermediate'.`

  Check the spelling of the display level and reissue the command.

- `The display level for login '`*`loginame`*`' has been changed to '`*`level`*`'.`

  The **sp_displaylevel** command succeeded. This message reports on the option you have just set.

Permissions

Any user can set and show his or her own display level. Only System Administrators can set the display level for another user.

Tables Used

master..sysattributes

See Also

System procedures	**sp_configure**

sp_displaylogin

Function

Displays information about a login account. Also displays information about the hierarchy tree above or below the login account when you so specify.

Syntax

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

Parameters

loginame – is the user login account about which you want information if it is other than your own. You must be a System Security Officer or System Administrator to get information about someone else's login account.

expand_up – specifies that Adaptive Server display all roles in the role hierarchy that contain the *loginame* role.

expand_down – specifies that Adaptive Server display all roles in the role hierarchy that are contained by the *loginame* role.

Examples

1. sp_displaylogin

```
Suid: 1
Loginame: sa
Fullname:
Default Database: master
Default Language:
Configured Authorization:
    sa_role (default ON)
    sso_role (default ON)
    oper_role (default ON)
Locked: NO
Date of Last Password Change: Nov 16 1994 10:08AM
```

Displays information about your server login account.

2. **sp_displaylogin susanne**

```
Suid: 12
Loginame: susanne
Fullname:
Default Database: pubs2
Default Language:
Configured Authorization:
    supervisor (default OFF)
Locked: NO
Date of Last Password Change: May 12 1997 11:09AM
```

   Displays information about the login account "susanne". The information displayed varies, depending on the role of the user executing **sp_displaylogin**.

3. **sp_displaylogin pillai, expand_up**

   Displays information about all roles containing the role of the login account "pillai". The information displayed varies, depending on the role of the user executing **sp_displaylogin**.

Comments

- **sp_displaylogin** displays configured roles, so even if you have made a role inactive with the **set** command, it is displayed.
- When you use **sp_displaylogin** to get information about your own account, you do not need to use the *loginame* parameter. **sp_displaylogin** displays your server user ID, login name, full name, any roles that have been granted to you, date of last password change, default database, default language, and whether your account is locked.
- If you are a System Security Officer or System Administrator, you can use the *loginame* parameter to access information about any account.

Messages

- `No login with the specified name exists.`

  You specified an incorrect *loginame*.

Permissions

Any user can execute **sp_displaylogin** to get information about his or her own login account. System Security Officers and System Administrators can use **sp_displaylogin** with the *loginame* and **expand** parameters to get information about other users' login accounts.

Tables Used

master.dbo.sysloginroles, master.dbo.syslogins, master.dbo.syssrvroles, sysobjects

See Also

Stored procedures	**sp_activeroles**, **sp_displayroles**, **sp_helprotect**, **sp_modifylogin**

sp_displayroles

Function

Displays all roles granted to another role, or displays the entire hierarchy tree of roles in table format.

Syntax

```
sp_displayroles [grantee_name [, mode]]
```

Parameters

grantee_name – is the login name of a user whose roles you want information about, or the name of a role you want information about.

mode – is **expand_up** or **expand_down**. **expand_up** shows the role hierarchy tree for the parent levels and **expand_down** shows the role hierarchy tree for the child levels.

Examples

1. **sp_displayroles**

```
Role Name
------------------------------
supervisor_role
```

Displays all roles granted to the user issuing the command.

2. **sp_displayroles "supervisor_role"**

```
Role Name
------------------------------
clerk
```

Displays all roles granted to *supervisor_role*.

3. **sp_displayroles susanne, expand_down**

```
Role Name         Parent Role Name       Level
----------------- ---------------------- ------
supervisor_role   NULL                        1
clerk_role       supervisor_role             2
```

Displays the active roles granted to login “susanne” and the roles above it in the hierarchy.

4. **sp_displayroles "intern_role", expand_up**

Displays the active roles granted to *intern_role* and the roles above it in the hierarchy.

Comments

- When you specify the optional parameter **expand_up** or **expand_down** all directly granted roles contained by or containing the specified role name are displayed.

Messages

- `Invalid name 'role_name'. This role or user does not exist in this SQL Server.`

  You specified an invalid role name or user name.

- `Invalid mode 'mode'. Mode should either be 'expand_up' or 'expand_down'.`

  You specified an invalid mode. Specify one of the modes shown in the message.

- `The user name 'user_name' should correspond to the current user or you must possess System Security Officer (SSO) authorization to execute 'procedure_name' for another user.`

  You do not possess the correct role to execute the specified procedure for the specified user.

Permissions

Any user can use **sp_displayroles** to see his or her own active roles. Only a System Administrator or System Security Officer can display information on roles activated by any other user.

Tables Used

master.dbo.sysattributes, master.dbo.syssrvroles, master.dbo.syslogins, master.dbo.sysloginroles

See Also

Commands	**alter role, create role, drop role, grant, revoke, set**
System procedures	**sp_activeroles, sp_displaylogin, sp_helprotect, sp_modifylogin**

sp_dropalias

Function

Removes the alias user name identity established with **sp_addalias**.

Syntax

```
sp_dropalias loginame
```

Parameters

loginame – is the name (in *master.dbo.syslogins*) of the user who was aliased to another user.

Examples

```
1. sp_dropalias victoria
```

Assuming that "victoria" was aliased (for example, to the Database Owner) in the current database, this statement drops "victoria" as an aliased user from the database.

Comments

- Executing the **sp_dropalias** procedure deletes an alternate *suid* mapping for a user from the *sysalternates* table.
- When a user's alias is dropped, he or she no longer has access to the database for which the alias was created.

Messages

- `Alias user dropped.`

  The user is no longer aliased to another user in the current database. The user cannot use the database until he or she is reinstated by the Database Owner with **sp_adduser** or **sp_addalias**.

- `No alias for specified user exists.`

  The named user does not have an alias in the current database.

- `No login with the specified name exists.`

  The *loginame* you specified has no account on Adaptive Server. No action was taken.

Permissions

Only the Database Owner or a System Administrator can execute **sp_dropalias**.

Tables Used

sysalternates, sysobjects

See Also

Commands	**use**
System procedures	**sp_addalias**, **sp_adduser**, **sp_changedbowner**, **sp_droplogin**, **sp_dropuser**, **sp_helpuser**

sp_dropdevice

Function

Drops an Adaptive Server database device or dump device.

Syntax

sp_dropdevice *logicalname*

Parameters

logicalname – is the name of the device as listed in *master.dbo.sysdevices.name*.

Examples

1. **sp_dropdevice tape5**

   Drops the device named *tape5* from Adaptive Server.

2. **sp_dropdevice fredsdata**

   Drops the database device named *fredsdata* from Adaptive Server. The device must not be in use by any database.

Comments

- The **sp_dropdevice** procedure drops a device from Adaptive Server, deleting the device entry from *master.dbo.sysdevices*.
- **sp_dropdevice** does not remove a file that is being dropped as a database device; it makes the file inaccessible to Adaptive Server. Use operating system commands to delete a file after using **sp_dropdevice**.

Messages

- `Can't run sp_dropdevice from within a transaction.`

  sp_dropdevice modifies system tables, so it cannot be run within a transaction.

- `Device dropped.`

  The device was dropped from the *master.dbo.sysdevices* table.

- `Device is being used by a database. You can't drop it.`

  Only database devices that are not in use can be dropped. You must drop all the databases associated with the device before dropping the device.

- `No such device exists -- run sp_helpdevice to list the Adaptive Server devices.`

  You tried to drop a device that does not exist on Adaptive Server.

Permissions

Only a System Administrator can execute **sp_dropdevice**.

Tables Used

master.dbo.sysdatabases, *master.dbo.sysdevices*, *master.dbo.sysusages*, *sysobjects*

See Also

Commands	**drop database**
System procedures	**sp_addumpdevice**, **sp_helpdb**, **sp_helpdevice**

sp_dropengine

Function

Drops an engine from a specified engine group or, if the engine is the last one in the group, drops the engine group.

Syntax

sp_dropengine *engine_number, engine_group*

Parameters

engine_number – is the number of the engine you are dropping from the group. Values are between 0 and a maximum equal to the number of configured online engines, minus one.

engine_group – is the name of the engine group from which to drop the engine.

Examples

1. sp_dropengine 2, DS_GROUP

This statement drops engine number 2 from the group called *DS_GROUP*. If it is the last engine in the group, the group is also dropped.

Comments

- **sp_dropengine** can be invoked only from the *master* database.
- If *engine_number* is the last engine in *engine_group*, Adaptive Server also drops *engine_group*.
- The *engine_number* you specify must exist in *engine_group*.

Messages

- `Cannot drop engine group bound to a SQL Server process.`

  You must unbind the process from the engine group before you can drop the engine group.

- `Cannot drop engine group used in class definition.`

  You cannot drop an engine group currently used in an execution class definition.

- `Cannot drop pre-defined engine groups.`

  You cannot drop the system-defined engine groups *ANYENGINE* and *LASTONLINE*.

- `Cannot modify pre-defined engine groups.`

  You cannot modify the system-defined engine groups *ANYENGINE* and *LASTONLINE*.

- `Can't run sp_dropengine from within a transaction.`

  sp_dropengine modifies system tables, so it cannot be run within a transaction.

- `Failed to drop engine` *`engine_number`* `from engine group` *`engine_group.`*

  Modifying *engine_group* resulted in error. Please check the error log.

- `Internal Error: Wrong engine list format in sysattributes.`

  Failed to process list of engines in *sysattributes* table.

- `Validation of engine group modification failed.`

  The parameter combination you supplied is invalid. Check the error log.

Permissions

Only a System Administrator can execute **sp_dropengine**.

Tables Used

sysattributes

See Also

System procedures	**sp_addengine, sp_addexeclass, sp_bindexeclass, sp_clearpsexe, sp_dropexeclass, sp_setpsexe, sp_showcontrolinfo, sp_showexeclass, sp_showpsexe**

sp_dropexeclass

Function

Drops a user-defined execution class.

Syntax

```
sp_dropexeclass classname
```

Parameters

classname – is the name of the user-defined execution class to be dropped.

Examples

```
1. sp_dropexeclass 'DECISION'
```

This statement drops the user-defined execution class *DECISION*.

Comments

- An execution class helps define the execution precedence used by Adaptive Server to process tasks. For more information on execution classes and execution attributes, see Chapter 22, "Distributing Engine Resources Between Tasks," in the *Performance and Tuning Guide*.
- *classname* must not be bound to any client application, login, or stored procedure. Unbind the execution class first, using **sp_unbindexeclass**, and then drop the execution class, using **sp_dropexeclass**.
- You cannot drop system-defined execution classes.

Messages

- `Can't run sp_dropexeclass from within a transaction.`

  sp_dropexeclass modifies system tables, so it cannot be run within a transaction.

- *`classname`* `is being used so it cannot be dropped.`

  The execution class is currently bound to an object. Use **sp_showpsexe** to find out which objects the execution class is bound to, use **sp_unbindexeclass** to unbind the objects from the

execution class, and then use **sp_dropexeclass** to drop the execution class.

- *classname* is not a valid class name.

  To see a list of valid class names, run **sp_showexeclass**.

Permissions

Only a System Administrator can execute **sp_dropexeclass**.

Tables Used

sysattributes

See Also

System procedures	**sp_addengine, sp_addexeclass, sp_bindexeclass, sp_clearpsexe, sp_dropengine, sp_setpsexe, sp_showcontrolinfo, sp_showexeclass, sp_showpsexe, sp_unbindexeclass**

sp_dropextendedproc

Function

Removes an extended stored procedure (ESP).

Syntax

sp_dropextendedproc *esp_name*

Parameters

esp_name – is the name of the extended stored procedure to be dropped.

Examples

1. sp_dropextendedproc xp_echo

Removes *xp_echo*.

Comments

- **sp_dropextendedproc** must be executed from the *master* database.
- The *esp_name* is case sensitive. It must precisely match the name with which the ESP was created.

Messages

- `Can't run sp_dropextendedproc from within a transaction.`

  sp_dropextendedproc modifies system tables, so it cannot be run within a transaction.

- *`esp_name`* `is not a valid name.`

  The *esp_name* you tried to drop does not exist. Use **sp_helpextendedproc** to list valid ESP names.

- `sp_dropextendedproc failed. Check the SQL Server error log file.`

  Make sure that the ESP exists and that you are using its correct name.

- `You must be in the master database in order to run sp_dropextendedproc.`

  Change to the *master* database.

Permissions

Only the System Administrator can execute **sp_dropextendedproc**.

Tables Used

master.dbo.syscomments, sysobjects

See Also

Commands	**drop procedure**
System procedures	**sp_addextendedproc, sp_freedll, sp_helpextendedproc**

sp_dropexternlogin

(Component Integration Services only)

Function

Drops the definition of a remote login previously defined by **sp_addexternlogin.**

Syntax

```
sp_dropexternlogin remote_server [, login_name]
```

Parameters

remote_server – is the name of the remote server from which the local server is dropping account access. The *remote_server* is known to the local server by an entry in the *master.dbo.sysservers* table.

login_name – is a login account known to the local server. If *login_name* is not specified, the current account is used. *login_name* must exist in the *master.dbo.syslogins* table. .

Examples

1. `sp_dropexternlogin JOBSERV, sa`

   Drops the definition of an external login to the remote server JOBSERV from *login_name* "sa".

2. `sp_dropexternlogin CIS1012, bobj`

   Drops the definition of an external login to the remote server CIS1012 from "bobj". Only the "bobj" account and the "sa" account can add or modify a remote login for "bobj".

Comments

- **sp_dropexternlogin** drops the definition of a remote login previously defined to the local server by **sp_addexternlogin.**
- You cannot execute **sp_dropexternlogin** from within a transaction.
- The *remote_server* must be defined to the local server by **sp_addserver.**
- To add and drop local server users, use the system procedures **sp_addlogin** and **sp_droplogin.**

Messages

- `Can't run sp_dropexternlogin from within a transaction.`

  sp_dropexternlogin cannot be run within a transaction.

- `Only the System Administrator may drop another's external login.`

  Only *login_name* or the "sa" account can drop or modify an external login for *login_name*.

- `Remote login/alias dropped.`

  This message confirms that the requested action was taken.

- *`remote_server`* `is the local server - external login not applicable.`

  You specified the name of the local server for the *remote_server* parameter.

- `Server name` *`remote_server`* `has not been added to sysservers.`

  remote_server is not defined to the local server. Use **sp_helpserver** to list defined servers. Add remote servers with **sp_addserver**.

- `There is no external login for server` *`remote_server`*`.`

  The *remote_server* specification is valid, but an **sp_addexternlogin** has not been issued for the server; therefore, the external login cannot be dropped.

Permissions

Only *login_name* or the "sa" account can drop or modify an external login for *login_name*.

Tables Used

master.dbo.sysattributes, master.dbo.sysservers

See Also

System procedures	**sp_addlogin**, **sp_addexternlogin**, **sp_addobjectdef**, **sp_addserver**,**sp_droplogin**, **sp_dropobjectdef**, **sp_helpdb**, **sp_helpserver**

sp_dropglockpromote

Function

Removes lock promotion values from a table or database.

Syntax

```
sp_dropglockpromote {"database" | "table"}, objname
```

Parameters

database | **table** – specifies whether to remove the lock promotion thresholds from a database or table. The quotes are required because these are Transact-SQL keywords.

objname – is the name of the table or database from which to remove the lock promotion thresholds.

Examples

```
1. sp_dropglockpromote table, titles
```

Removes the lock promotion values from *titles*. Lock promotion for *titles* now uses the database or server-wide values.

Comments

- Use **sp_dropglockpromote** to drop lock promotion values set with **sp_setpglockpromote.**
- When you drop a database's lock promotion thresholds, tables that do not have lock promotion thresholds configured will use the server-wide values.
- When a table's values are dropped, Adaptive Server uses the database's lock promotion thresholds if they are configured or the server-wide values if they are not.
- Server-wide values can be changed with **sp_setpglockpromote**, but cannot be dropped.

Messages

- `Can't run sp_dropglockpromote from within a transaction.`

  sp_dropglockpromote updates system tables, so it cannot be run from within a transaction.

- `No such database -- run sp_helpdb to list databases.`

  The database does not exist. Check the spelling.

- `Object must be in the current database.`

  sp_dropglockpromote can remove lock promotion values only from tables in the current database. Qualify **sp_dropglockpromote** with the database name or issue the **use** *database_name* command to open the database in which the table resides, and then issue **sp_dropglockpromote** again.

- `The target object does not exist.`

  The table specified with the *objname* parameter does not exist in the current database, or the database does not exist.

- `You must be in 'master' to add, change or drop lock promotion attributes for a database.`

  Qualify **sp_dropglockpromote** with *master* or issue the **use** *database_name* command to open the *master* database, and then issue **sp_dropglockpromote** again.

- *`objname`* `is a system table. This stored procedure cannot be used on system tables.`

  You cannot configure lock promotion thresholds for system tables.

- `Lock promotion attribute does not exist for` *`scope`*`, '`*`objname`*`'. Cannot delete it!`

  Lock promotion is not configured for the database or object you specified.

- `Lock promotion attribute of object` *`objname`* `has been dropped!`

  sp_dropglockpromote succeeded.

- `Invalid value` *`value`*`, specified for 'scope' parameter. Valid values are 'DATABASE' or 'TABLE'.`

  Specify **"database"** or **"table"**. The quotes are required because these are Transact-SQL keywords.

- `Server-wide lock promotion values cannot be dropped. Use sp_configure to restore server-wide defaults.`

Permissions

Only a System Administrator can execute **sp_dropglockpromote**.

Tables Used

master.dbo.sysattributes, *sysobjects*

See Also

System procedures	**sp_configure**, **sp_setpglockpromote**

sp_dropgroup

Function

Drops a group from a database.

Syntax

sp_dropgroup *grpname*

Parameters

grpname – is the name of a group in the current database.

Examples

1. **sp_changegroup accounting, martha**
   sp_changegroup "public", george
   sp_dropgroup purchasing

   The "purchasing" group has merged with the "accounting" group. These commands move "martha" and "george", members of the "purchasing" group, to other groups before dropping the group. The group name "public" is quoted because "public" is a reserved word.

Comments

- Executing **sp_dropgroup** drops a group name from a database's *sysusers* table.
- You cannot drop a group if it has members. You must execute **sp_changegroup** for each member before you can drop the group.

Messages

- `Can't drop the group 'public'.`

  The "public" group exists in every database. It is the group that all users belong to by default, and it cannot be dropped.

- `Group has been dropped.`

  The command succeeded. The group no longer exists in the current database.

- `Group has members. It must be empty before it can be dropped.`

  Groups with members cannot be dropped. Reassign the members of the group to another group using **sp_changegroup**. A list of the group members appears after this message.

- `No group with the specified name exists.`

  The specified group does not exist.

Permissions

Only the Database Owner, a System Administrator, or a System Security Officer can execute **sp_dropgroup**.

Tables Used

master.dbo.syssrvroles, *sysobjects*, *sysprotects*, *sysusers*

See Also

Commands	**grant**, **revoke**, **use**
System procedures	**sp_addgroup**, **sp_adduser**, **sp_changegroup**, **sp_dropuser**, **sp_helpgroup**

sp_dropkey

Function

Removes from the *syskeys* table a key that had been defined using **sp_primarykey**, **sp_foreignkey**, or **sp_commonkey**.

Syntax

```
sp_dropkey keytype, tabname [, deptabname]
```

Parameters

keytype – is the type of key to be dropped. The *keytype* must be **primary**, **foreign**, or **common**.

tabname – is the name of the key table or view that contains the key to be dropped.

deptabname – specifies the name of the second table in the relationship, if the *keytype* is **foreign** or **common**. If the *keytype* is **primary**, this parameter is not needed, since **primary** keys have no dependent tables. If the *keytype* is **foreign**, this is the name of the primary key table. If the *keytype* is **common**, give the two table names in the order in which they appear with **sp_helpkey**.

Examples

1. `sp_dropkey primary, employees`

   Drops the primary key for the *employees* table. Any foreign keys that were dependent on the primary key for *employees* are also dropped.

2. `sp_dropkey common, employees, projects`

   Drops the common keys between the *employees* and *projects* tables.

3. `sp_dropkey foreign, titleauthor, titles`

   Drops the foreign key between the *titleauthor* and *titles* tables.

Comments

- Executing **sp_dropkey** deletes the specified key from *syskeys*. Only the owner of a table can drop a key from that table.
- Keys are created to make explicit a logical relationship that is implicit in your database design. This information can be used by an application.

- Dropping a primary key automatically drops any foreign keys associated with it. Dropping a foreign key has no effect on a primary key specified on that table.
- Executing **sp_commonkey, sp_primarykey,** or **sp_foreignkey** adds the key to the *syskeys* system table. To display a report on the keys that have been defined, execute **sp_helpkey.**

Messages

- `Common keys dropped.`

  The **sp_dropkey** command succeeded, deleting the common key definition from *syskeys.*
- `Dependent foreign keys were also dropped.`

  When a primary key is dropped, any foreign keys that depend on it are also dropped.
- `Foreign key dropped.`

  The **sp_dropkey** command succeeded, deleting the foreign key definition from *syskeys.*
- `No common keys exist between the two tables or views supplied.`

  There are no common keys between the *tabname* and *deptabname* tables, or the table names were given in the wrong order. No action was taken. Use **sp_helpkey** to see the keys and the order in which to give the arguments.
- `No foreign key for the table or view exists.`

  tabname has no foreign key defined.
- `No primary key for the table or view exists.`

  tabname has no primary key defined.
- `Primary key for the table or view dropped.`

  The **sp_dropkey** command succeeded, dropping the primary key and deleting it from *syskeys.*
- `Table or view name must be in current database.`

  You cannot drop keys on tables or views in other databases.
- `The dependent table or view doesn't exist in the current database.`

  The name you supplied for the *deptabname* parameter is not a table or a view in the current database.

- `The table or view named doesn't exist in the current database.`

  The *tabname* supplied is not a table or a view in the current database.

- `Usage: sp_dropkey {primary | foreign | common}, tabname [, deptabname]. Type must be 'primary', 'foreign', or 'common'.`

  The *keytype* parameter should specify the type of key to be dropped.

- `You must be the owner of the table or view to drop its key.`

  You are not the owner of the table, so you cannot drop the key.

- `You must supply the dependent table or view as the third parameter.`

  When dropping a foreign or common key, you must name both the *tabname* and the *deptabname* tables.

Permissions

Only the owner of *tabname* can use **sp_dropkey**.

Tables Used

syskeys, *sysobjects*

See Also

System procedures	**sp_commonkey**, **sp_foreignkey**, **sp_helpkey**, **sp_primarykey**

sp_droplanguage

Function

Drops an alternate language from the server and removes its row from *master.dbo.syslanguages*.

Syntax

```
sp_droplanguage language [, dropmessages]
```

Parameters

language – is the official name of the language to be dropped.

dropmessages – drops all Adaptive Server system messages in *language*. You cannot drop a language with associated system messages without also dropping its messages.

Examples

1. `sp_droplanguage french`

   This command drops French from the available alternate languages, if there are no associated messages.

2. `sp_droplanguage french, dropmessages`

   This command drops French from the available alternate languages, if there are associated messages.

Comments

- Executing **sp_droplanguage** drops a language from a list of alternate languages by deleting its entry from the *master.dbo.syslanguages* table.
- If you try to drop a language that has system messages, the request fails unless you supply the **dropmessages** parameter.

Messages

- *language* is not an official language name from syslanguages.

  Use **sp_helplanguage** to see the list of official languages available on this Adaptive Server.

- `Can't drop 'language' because there are associated entries in master.dbo.sysmessages. Run sp_droplanguage with 'dropmessages' flag.`

  You cannot drop a language for which the *master* database contains associated system messages. Rerun **sp_droplanguage** with the **dropmessages** option to drop the language and all associated system messages.

- `The only legal value for the second parameter is 'dropmessages'.`

  You cannot specify an option other than **dropmessages**.

- `Language deleted.`

  The language is deleted from *master.dbo.syslanguages*. Error messages associated with this language are deleted from *master.dbo.sysmessages*.

Permissions

Only a System Administrator can issue **sp_droplanguage**.

Tables Used

master.dbo.syslanguages, *master.dbo.sysmessages*, *sysobjects*

See Also

System procedures	**sp_addlanguage**, **sp_helplanguage**

sp_droplogin

Function

Drops an Adaptive Server user login by deleting the user's entry from *master.dbo.syslogins*.

Syntax

```
sp_droplogin loginame
```

Parameters

loginame – is the name of the user, as listed in *master.dbo.syslogins*.

Examples

1. `sp_droplogin victoria`

   Drops the "victoria" login from Adaptive Server.

Comments

- Executing **sp_droplogin** drops a user login from Adaptive Server, deleting the user's entry from *master.dbo.syslogins*.
- Adaptive Server reuses a dropped login's server user ID, which compromises accountability. You can avoid dropping accounts entirely and, instead, use **sp_locklogin** to lock any accounts that will no longer be used.

  If you need to drop logins, be sure to audit these events (using **sp_audit**) so that you have a record of them.
- **sp_droplogin** deletes all resource limits associated with the dropped login.
- **sp_droplogin** fails if the login to be dropped is a user in any database on the server. Use **sp_dropuser** to drop the user from a database. You cannot drop a user from a database if that user owns any objects in the database.
- If the login to be dropped is a System Security Officer, **sp_droplogin** verifies that at least one other unlocked System Security Officer's account exists. If not, **sp_droplogin** fails. Similarly, **sp_droplogin** ensures that there is always at least one unlocked System Administrator account.

Messages

- `Can't run sp_droplogin from within a transaction.`

  sp_droplogin modifies system tables, so it cannot be run within a transaction.

- `Login dropped.`

  The user's entry in *master.dbo.syslogins* has been deleted. The user no longer has access to Adaptive Server.

- `No such account -- nothing changed.`

  The specified login name does not exist.

- `User exists or is an alias in at least one database. Drop user/alias before dropping login.`

  You cannot drop a login that is a user in any database on the server or a user who has an alias in a database. Use **sp_dropuser** to drop a user from a database or **sp_dropalias** to drop the alias from the databases.

- `Warning: the specified account is currently active. Nothing changed.`

  You cannot drop an account if it is active. Run the command again when the user has logged off. You may be able to use **kill** to end the user's Adaptive Server session.

Permissions

Only a System Administrator or a System Security Officer can execute **sp_droplogin**.

Tables Used

master.dbo.sysloginroles, master.dbo.syslogins, master.dbo.sysprocesses, sysobjects

See Also

System procedures	**sp_addlogin**, **sp_audit**, **sp_changedbowner**, **sp_dropalias**, **sp_dropuser**, **sp_helpuser**, **sp_locklogin**

sp_dropmessage

Function

Drops user-defined messages from *sysusermessages*.

Syntax

```
sp_dropmessage message_num [, language]
```

Parameters

message_num – is the message number of the message to be dropped. Message numbers must have a value of 20000 or higher.

language – is the language of the message to be dropped.

Examples

1. `sp_dropmessage 20002, french`

   Removes the French version of the message with the number 20002 from *sysusermessages*.

Comments

- The *language* parameter is optional. If included, only the message with the indicated *message_num* in the indicated language is dropped. If you do not specify a *language*, all messages with the indicated *message_num* are dropped.

Messages

- *language* is not an official language name from syslanguages.

  The *language* you supplied is not a valid name in the *syslanguages* table.

- Message number must be at least 20000.

  Only user-defined messages, which have message numbers of 20000 or higher, can be deleted.

- Message number *message_num* does not exist.

  No message with the given message number exists in *sysusermessages*.

- `Message number message_num does not exist in the language language.`

  A message with the given message number does not exist in the *language* given.

- `Message deleted.`

  The message has been dropped.

- `User user_name does not have permission to drop message number message_num.`

  Only a System Administrator, the Database Owner, and the user who created the message being dropped can delete a message.

- `User user_name does not have permission to drop message number message_num in the language language.`

  Only a System Administrator, the Database Owner, and the user who created the message being dropped can delete a message.

Permissions

Only a System Administrator, the Database Owner, and the user who created the message being dropped can execute **sp_dropmessage**.

Tables Used

master.dbo.syslanguages, *sysobjects*, *sysusermessages*

See Also

System procedures	**sp_addmessage**, **sp_getmessage**

sp_dropobjectdef

(Component Integration Services only)

Function

Deletes the external storage mapping provided for a local object.

Syntax

```
sp_dropobjectdef "object_name"
```

Parameters

object_name has the form *dbname.owner.object*, where:

dbname is the name of the database containing the object whose storage location you are dropping. *dbname* is optional; if present, it must be the current database, and the *owner* or a placeholder is required.

owner is the name of the owner of the object whose storage location you are dropping. *owner* is optional; it is required if *dbname* is specified.

object is the name of the local table for which external storage mapping is to be dropped.

Examples

1. `sp_dropobjectdef "personnel.dbo.colleges"`

   Deletes the entry from *sysattributes* that provided the external storage mapping for a table known to the server as the *colleges* table in database *personnel*.

2. `sp_dropobjectdef "andrea.fishbone"`

   Deletes the entry from *sysattributes* that provided the external storage mapping for the *andrea.fishbone* object, where *andrea* is the *owner* and the local table name is *fishbone*.

Comments

- **sp_dropobjectdef** deletes the external storage mapping provided for a local object. It replaces **sp_droptabledef**.
- Use **sp_dropobjectdef** after dropping a remote table with **drop table**.
- Dropping a table does not remove the mapping information from the *sysattributes* table if it was added using **sp_addobjectdef**. It must be explicitly removed using **sp_dropobjectdef**.

- The *object_name* can be in any of these forms:
  - *object*
  - *owner.object*
  - *dbname..object*
  - *dbname.owner.object*

Messages

- `Database dbname is not the current database.`

  If the *dbname* parameter is supplied, it must be the name of the current database.

- `owner is not a valid user in dbname database.`

  The name of the *owner* is not defined in the current database. If the *owner* parameter is supplied, it must a valid user of the current database.

- `Server name is not permitted in the local object_name.`

  Do not specify the server name in *object_name*.

- `Table has not been defined.`

  No entry for this table exists in *sysattributes*. Before the external mapping for the table can be dropped, it must be defined using **sp_addtabledef** or **sp_addobjectdef**.

- `You must be the System Administrator (sa) or Database Owner (dbo) to drop a definition for another user's table.`

  If the *owner* parameter is supplied, the name supplied must be either the owner of the object or the System Administrator.

- `You cannot remove a table definition for a table that has been created. Drop the table first.`

  Drop the table with **drop table**.

Permissions

sp_dropobjectdef permission defaults to the Database Owner and System Administrator. Only the System Administrator can drop the storage mapping for another user's object.

Tables Used

sysobjects, sysattributes, sysusers

See Also

Commands	**create existing table**, **create table**, **drop table**
System procedures	**sp_addobjectdef**

sp_dropremotelogin

Function

Drops a remote user login.

Syntax

```
sp_dropremotelogin remoteserver [, loginame
   [, remotename] ]
```

Parameters

remoteserver – is the name of the server that has the remote login to be dropped.

loginame – is the local server's user name that is associated with the remote server in the *sysremotelogins* table.

remotename – is the remote user name that gets mapped to *loginame* when logging in from the remote server.

Examples

1. `sp_dropremotelogin GATEWAY`

   Drops the entry for the remote server named GATEWAY.

2. `sp_dropremotelogin GATEWAY, churchy`

   Drops the entry for mapping remote logins from the remote server GATEWAY to the local user named "churchy".

3. `sp_dropremotelogin GATEWAY, churchy, pogo`

   Drops the login for the remote user "pogo" on the remote server GATEWAY that was mapped to the local user named "churchy".

Comments

- Executing **sp_dropremotelogin** drops a user login from a remote server, deleting the user's entry from *master.dbo.sysremotelogins*.
- For a more complete discussion on remote logins, see **sp_addremotelogin**.
- To add and drop local server users, use the system procedures **sp_addlogin** and **sp_droplogin**.

Messages

- `Can't run sp_dropremotelogin from within a transaction.`

  sp_dropremotelogin modifies system tables, so it cannot be run within a transaction.

- `Remote login dropped.`

  The remote user's entry in *master.dbo.sysremotelogins* has been deleted. The remote user no longer has access to this server.

- `There is no remote user '`*`remotename`*`' mapped to local user '`*`loginame`*`' from the remote server '`*`remoteserver`*`'.`

  The specified remote login name does not exist for the named server.

Permissions

Only the System Administrator can execute **sp_dropremotelogin.**

Tables Used

master.dbo.sysremotelogins, master.dbo.sysservers, sysobjects

See Also

System procedures	**sp_addlogin**, **sp_addremotelogin**, **sp_addserver**, **sp_droplogin**, **sp_helpremotelogin**, **sp_helpserver**

sp_drop_resource_limit

Function

Removes one or more resource limits from Adaptive Server.

Syntax

```
sp_drop_resource_limit {name, appname }
   [, rangename, limittype, enforced, action, scope]
```

Parameters

name – is the Adaptive Server login to which the limit applies. To drop resource limits that apply to all users of a particular application, specify the *appname* and a *name* of NULL.

appname – is the application to which the limit applies. To drop resource limits that apply to all applications used by the specified login, specify the login name and an *appname* of NULL. To drop a limit that applies to a particular application, specify the application name that the client program passes to the Adaptive Server in the login packet.

rangename – is the time range during which the limit is enforced. This must be an existing time range stored in the *systimeranges* system table or NULL to delete all resource limits for the specified *name*, *appname*, *limittype*, *action*, and *scop*e, without regard to *rangename*.

limittype – is the type of resource being limited. This must be one of the following:

Limit Type	Description
row_count	Drops only limits that restrict the number of rows a query can return.
elapsed_time	Drops only limits that restrict the number of seconds that a query batch or transaction can run.
io_cost	Drops only limits that restrict actual or estimated query processing cost.
NULL	Drops all resource limits with the specified *name, appname, rangename,* enforcement time, *action,* and *scope,* without regard to *limittype*.

enforced – determines whether the limit is enforced prior to or during query execution. The following table lists the valid values for each limit type:

Enforced Code	Description	Limit Type
1	Drops only limits for which action is taken when the estimated cost of execution exceeds the specified limit.	**io_cost**
2	Drops only limits for which action is taken when the actual row count, elapsed time, or cost of execution exceeds the specified limit.	**row_count** **elapsed_time** **io_cost**
3	Drops only limits for which action is taken when either the estimated cost (1) or the actual cost (2) exceeds the specified limit.	**io_cost**
NULL	Drops all resource limits with the specified *name*, *appname*, *rangename*, *limittype*, and *scope*, without regard to when the *action* is enforced.	

action – is the action taken when the limit is exceeded. This must be one of the following:

Action Code	Description
1	Drops only limits that issue a warning.
2	Drops only limits that abort the query batch.
3	Drops only limits that abort the transaction.
4	Drops only limits that kill the session.
NULL	Drops all resource limits with the specified *name*, *appname*, *rangename*, *limittype*, enforcement time, and *scope*, without regard to the *action* they take.

scope – is the scope of the limit. This must be one of the following:

Scope Code	Description
1	Drops only limits that apply to queries.
2	Drops only limits that apply to query batches.

Scope Code	Description
4	Drops only limits that apply to transactions.
6	Drops only limits that apply to both query batches and transactions.
NULL	Drops all resource limits with the specified *name*, *appname*, *rangename*, *limittype*, enforcement time, and *action*, without regard to their *scope*.

Examples

1. **`sp_drop_resource_limit joe, payroll, friday_afternoon, io_cost, 2, 4, 1`**

   Drops the single resource limit that kills the session whenever joe's use of the *payroll* application runs a query during the *friday_afternoon* time range that results in excessive execution-time I/O cost.

➤ ***Note***

If no resource limit matches these selection criteria, **sp_drop_resource_limit** returns without error.

2. **`sp_drop_resource_limit joe, payroll`**

   Drops all limits that apply to joe's use of the *payroll* application.

3. **`sp_drop_resource_limit joe`**

   Drops all limits that apply to the user "joe".

4. **`sp_drop_resource_limit NULL, payroll`**

   Drops all resource limits that apply to the *payroll* application.

5. **`sp_drop-resource_limit NULL, payroll, NULL, NULL, NULL, 4, NULL`**

   Drops all resource limits on the *payroll* application whose action is to kill the session.

Comments

- Use the **sp_help_resource_limit** system procedure to determine which resource limits apply to a given user, application, or time of day.
- When you use **sp_droplogin** to drop an Adaptive Server login, all resource limits associated with that login are also dropped.

- The deletion of a resource limit causes the limits for each session for that login and / or application to be rebound at the beginning of the next query batch for that session.
- For more information on resource limits, see Chapter 12, "Limiting Access to Server Resources," in the *System Administration Guide.*

Messages

- `At least one of the login or application name must be non-NULL.`

  You must specify the Adaptive Server login or application name, or both, to which the limit applies.

- `Can't run sp_drop_resource_limit from within a transaction.`

  Because **sp_drop_resource_limit** modifies system tables, it cannot be executed from within a transaction.

- `Illegal action` *`action.`*

  You must specify an action of **1**, **2**, **3**, **4**, or **NULL**.

- `Illegal scope value` *`scope`* `for this limit type.`

  The scope must be **1**, **2**, **3**, **4**, **6**, or **NULL** and must be compatible with the limit type.

- `No login with the specified name exists.`

  The login you specified does not exist. Use **sp_helpuser** to list login names.

- `Only the System Administrator (SA) may execute this procedure.`

  You must be a System Administrator to run **sp_drop_resource_limit**.

- `Resource limit dropped.`

  You successfully removed the resource limit.

- `Unknown limit type` *`limittype.`*

  The specified limit type does not exist. Use **sp_help_resource_limit** to list resource limits.

- `Unknown time range name` *`rangename.`*

  The time range you specified does not exist.

Permissions

Only a System Administrator can execute **sp_drop_resource_limit**.

Tables Used

master..sysresourcelimits, master..systimeranges, master..spt_limit_types

See Also

System procedures	**sp_add_resource_limit**, **sp_help_resource_limit**, **sp_modify_resource_limit**

sp_dropsegment

Function

Drops a segment from a database or unmaps a segment from a particular database device.

Syntax

```
sp_dropsegment segname, dbname [, device]
```

Parameters

segname – is the name of the segment to be dropped.

dbname – is the name of the database from which the segment is to be dropped.

device – is the name of the database device from which the segment *segname* is to be dropped. This parameter is optional, except when the system segment *system*, *default*, or *logsegment* is being dropped from a database device.

Examples

1. `sp_dropsegment indexes, pubs2`

   This command drops the segment *indexes* from the *pubs2* database.

2. `sp_dropsegment indexes, pubs2, dev1`

   This command unmaps the segment *indexes* from the database device *dev1*.

Comments

- You can drop a segment if it is not referenced by any table or index in the specified database.
- If you do not supply the optional argument *device*, the segment is dropped from the specified database. If you do supply a *device* name, the segment is no longer mapped to the named database device, but the segment is not dropped.
- Dropping a segment drops all thresholds associated with that segment.
- When you unmap a segment from one or more devices, Adaptive Server drops any thresholds that exceed the total space on the

segment. When you unmap the *logsegment* from one or more devices, Adaptive Server recalculates the last-chance threshold.

- **sp_placeobject** changes future space allocations for a table or index from one segment to another, and removes the references from the original segment. After using **sp_placeobject**, you can drop the original segment name with **sp_dropsegment**.
- For the system segments *system*, *default*, and *logsegment*, you must specify the device name from which you want the segments dropped.

Messages

- `Can't drop the 'segname' segment completely.`

  You did not specify the device from which you want the segment dropped.

- `Can't run sp_dropsegment from within a transaction.`

  sp_dropsegment modifies system tables, so it cannot be run within a transaction.

- `Segment dropped.`

  The procedure was successful. There is no longer a segment named *segname* in the specified database.

- `Segment reference to device dropped.`

  The procedure was successful. The segment *segname* no longer refers to database device *device*.

- `Segment 'segname' does not reference device 'device'.`

  The segment you tried to drop from *device* is not referenced by *segname*. Run **sp_helpsegment** *segname* to list the devices referenced by *segname*.

- `The specified device is not used by the database.`

  The specified database does not use device *device*. Use **sp_helpsegment** to see which devices are referenced by *segname*.

- `The segment 'segname' is being used.`

  You cannot drop a segment that is referenced by a table or an index. If you still want to drop the segment, you must redefine the segment for the affected tables or indexes by using the system procedure **sp_placeobject**.

- `There is no such segment as 'segname'.`

The segment you tried to drop does not exist. All segments for a database are listed in the *syssegments* table.

- `There is only one device mapping for the segment 'segname' -- use sp_dropsegment with no device argument.`

  The *device* you tried to drop is the last device reference for *segname*. It is illegal to drop the last device reference for a segment.

- `WARNING: There are no longer any segments referencing device 'device'. This device will no longer be used for space allocation.`

  The procedure was successful, but the device is now unassigned and cannot be used for storing data or log information.

- `WARNING: There are no longer any segments referencing devices 'device'. These devices will no longer be used for space allocation.`

  The procedure was successful, but the devices are now unassigned and cannot be used for storing data or log information.

- `You must execute this procedure from the database in which you wish to add a segment. Please execute 'use database_name' and try again.`

  sp_dropsegment can drop segments only in the database the command is issued from. Qualify **sp_dropsegment** with the database name or issue the **use** *database_name* command to open the database from which you want to drop a segment, and then run **sp_dropsegment** again.

Permissions

Only the Database Owner or a System Administrator can execute **sp_dropsegment.**

Tables Used

master.dbo.spt_values, *sysdatabases*, *sysdevices*, *sysindexes*, *sysobjects*, *syssegments*, *systhresholds*, *sysusages*

See Also

System procedures	**sp_addsegment**, **sp_addthreshold**, **sp_helpsegment**, **sp_helpthreshold**, **sp_placeobject**

sp_dropserver

Function

Drops a server from the list of known servers or drops remote logins and external logins in the same operation.

Syntax

```
sp_dropserver server [, droplogins]
```

Parameters

server – is the name of the server to be dropped.

droplogins – indicates that any remote logins for *server* should also be dropped.

Examples

1. `sp_dropserver GATEWAY`

   This command drops the remote server GATEWAY.

2. `sp_dropserver RDBAM_ALPHA, droplogins`

   Drops the entry for the remote server RDBAM_ALPHA and drops all remote logins and external logins for that server.

Comments

- Executing **sp_dropserver** drops a server from the list of known servers by deleting the entry from the *master.dbo.sysservers* table.
- Running **sp_dropserver** on a server that has associated entries in the *master.dbo.sysremotelogins* table results in an error message stating that you must drop the remote users before you can drop the server. To drop all remote logins for a server when dropping the server, use **droplogins**.
- Running **sp_dropserver** without **droplogins** against a server that has associated entries in the *sysattributes* table results in an error. You must drop the remote logins and external logins before you can drop the server.
- The checks against *sysattributes* for external logins and for default mapping to a server apply when Component Integration Services is configured.

Messages

- `Can't run sp_dropserver from within a transaction.`

  sp_dropserver modifies system tables, so it cannot be run within a transaction.

- `Remote logins for remote server '`*`server`*`' have been dropped.`

  The **sp_dropserver** command dropped the remote server and associated logins.

- `Server dropped.`

  The procedure was successful. The server you specified is no longer accessible through this server and it can no longer access this server.

- `There are still remote logins for the server '`*`server`*`'.`

  The server you tried to drop has associated entries in the *sysremotelogins* table. You must either drop the remote logins with **sp_dropremotelogin** or use **droplogins**.

- `There is not a server named 'server'.`

  The server you tried to drop is not a known server. All known servers for an Adaptive Server are listed in the *master.dbo.sysservers* table.

- `Unable to drop server `*`server`*` because it is referenced in master.dbo.sysdatabases.`

  The server you tried to drop cannot be dropped while it is named in the *master..sysdatabases* table.

- `Usage: sp_dropserver server [, droplogins]`

  The only parameter you can use with **sp_dropserver** is **droplogins**.

Permissions

Only a System Security Officer can execute **sp_dropserver**.

Tables Used

master.dbo.sysremotelogins, *master.dbo.sysservers*, *sysobjects*

See Also

System procedures	**sp_addserver**, **sp_dropremotelogin**, **sp_helpremotelogin**, **sp_helpserver**

sp_dropthreshold

Function

Removes a free-space threshold from a segment.

Syntax

```
sp_dropthreshold dbname, segname, free_space
```

Parameters

dbname – is the database from which you are dropping the threshold. This must be the name of the current database.

segname – is the segment whose free space is monitored by the threshold. Use quotes when specifying the "default" segment.

free_space – is the number of free pages at which the threshold is crossed.

Examples

```
1. sp_dropthreshold mydb, segment1, 200
```

Removes a threshold from *segment1* of *mydb*. You must specify the database, segment, and amount of free space to identify the threshold.

Comments

- You cannot drop the last-chance threshold from the log segment.
- You can use the **no free space acctg** option of **sp_dboption** as an alternative to **sp_dropthreshold**. This option disables free-space accounting on non-log segments. You cannot disable free-space accounting on log segments.

Messages

- `Dropping threshold for segment 's*egname*' at '*free_space*' pages.`

  The **sp_dropthreshold** command succeeded.

- `Segment '*segname*' does not have a threshold at '*free_space*' pages.`

  Run **sp_helpthreshold** to see the names of the thresholds in the current database.

- `Table 'systhresholds' does not exist in database '`*`dbname`*`' -- cannot drop thresholds.`

  The *systhresholds* table is missing. This table is created when the database is created (or an upgrade to release 10.0 or later is performed), and it must not be removed.

- `There is no segment named '`*`segname`*`'.`

  Run **sp_helpsegment** to see the names of the segments in the current database.

- `You may not drop the log's last-chance threshold.`

  The threshold name and size you specified identify the last-chance threshold. You cannot drop this threshold.

Permissions

Only the Database Owner or a System Administrator can execute **sp_dropthreshold**.

Tables Used

sysobjects, *syssegments*, *systhresholds*

See Also

System procedures	**sp_addthreshold**, **sp_dboption**, **sp_helpthreshold**, **sp_thresholdaction**

sp_drop_time_range

Function

Removes a user-defined time range from Adaptive Server.

Syntax

sp_drop_time_range ***name***

Parameters

name – is the name of the time range to be dropped.

Examples

1. sp_drop_time_range evenings

Removes the "evenings" time range.

Comments

- You cannot remove the "at all times" time range.
- You cannot drop a time range if a resource limit exists for that time range.
- Dropping a time range does not affect the active time ranges for sessions currently in progress.
- For more information on time ranges, see Chapter 12, "Limiting Access to Server Resources," in the *System Administration Guide.*

Messages

- `'at-all-times' range may not be dropped.`

  You cannot drop the "at all times" time range.

- `Can't run sp_drop_time_range from within a transaction.`

  Because **sp_drop_time_range** modifies system tables, it cannot be run from within a transaction.

- `Only the System Administrator (SA) may execute this procedure.`

  You must be a System Administrator to run **sp_drop_time_range**.

- `There are still limits using this range.`

  You cannot drop a time range that is referenced by a resource limit. Use **sp_help_resource_limit** to determine which resource limits

reference this time range; then use **sp_drop_resource_limi**t to drop these limits.

- `Time range` *`name`* `dropped.`

  The time range was successfully dropped from the Adaptive Server.

- `Timerange name must be non-NULL.`

  You must specify a time range.

- `Unknown time range name` *`name.`*

  The time range you specified does not exist in the *systimeranges* system table of the *master* database.

Permissions

Only a System Administrator can execute **sp_drop_time_range**.

Tables Used

master..systimeranges

See Also

System procedures	**sp_add_resource_limit**, **sp_add_time_range**, **sp_modify_time_range**

sp_droptype

Function

Drops a user-defined datatype.

Syntax

sp_droptype *typename*

Parameters

typename – is the name of a user-defined datatype that you own.

Examples

1. sp_droptype birthday

Drops the user-defined datatype named *birthday*.

Comments

- Executing **sp_droptype** deletes a user-defined datatype from *systypes*.
- A user-defined datatype cannot be dropped if it is referenced by tables or another database object.

Messages

- `The type doesn't exist or you don't own it.`

  You do not own a user-defined datatype with that name.
- `Type is being used. You cannot drop it.`

  You cannot drop a user-defined datatype that is referenced by a table or another database object. Drop the tables and database objects first.
- `Type has been dropped.`

  The user-defined datatype no longer exists in the current database.

Permissions

Only the Database Owner or datatype owner can execute **sp_droptype**.

Tables Used

syscolumns, sysobjects, systypes, sysusers

See Also

Datatypes	"System and User-Defined Datatypes"
System procedures	**sp_addtype**, **sp_rename**

sp_dropuser

Function

Drops a user from the current database.

Syntax

sp_dropuser *name_in_db*

Parameters

name_in_db – is the user's name in the current database's *sysusers* table.

Examples

1. sp_dropuser albert

Drops the user "albert" from the current database. The user "albert" can no longer use the database.

Comments

- **sp_dropuser** drops a user from the current database by deleting the user's row from *sysusers*.
- You cannot drop a user who owns objects in the database.
- You cannot drop a user who has granted permissions to other users.
- You cannot drop the Database Owner from a database.
- If other users are aliased to the user being dropped, their aliases are also dropped. They no longer have access to the database.
- You cannot drop a user from a database if the user owns a stored procedure that is bound to an execution class in that database. See **sp_bindexeclass**.

Messages

- `The dependent aliases were also dropped.`

  Other users were aliased to the user being dropped. Their aliases have been dropped, and they can no longer access the database.

- `No user with the specified name exists in the current database.`

  The specified user does not exist in the current database.

- `User has been dropped from current database.`

  The specified user is no longer known to the current database.

- `You cannot drop the 'database owner'.`

  The *name_in_db* is the Database Owner.

- `You cannot drop the 'guest' user from master or tempdb.`

  The "guest" user must exist in *master* and *tempdb* to allow the "guest" mechanism to work in other databases.

- `You cannot drop user because user 'name_in_db' owns objects in database.`

  A user who owns objects in the current database cannot be dropped. Drop the owned objects first. A list of datatypes and their owners appears after this message.

- `You cannot drop user because user 'name_in_db' owns thresholds in database.`

  A user who owns thresholds in the current database cannot be dropped. Drop the owned thresholds first.

- `You cannot drop user because user 'name_in_db' owns types in database.`

  A user who owns user-defined datatypes in the current database cannot be dropped. Drop the owned datatypes first. A list of datatypes and their owners appears after this message.

- `You cannot drop user because he or she owns grantable privileges and granted them to other users. Use sp_helprotect for more information.`

  You must remove the grantable permissions from the user before he or she can be dropped.

Permissions

Only the Database Owner, a System Administrator, or a System Security Officer can execute **sp_dropuser.**

Tables Used

master.dbo.spt_values, sysalternates, syscolumns, sysobjects, sysprotects, syssegments, systhresholds, systypes, sysusers

See Also

Commands	**grant**, **revoke**, **use**
System procedures	**sp_addalias**, **sp_adduser**, **sp_bindexeclass**, **sp_droplogin**

sp_estspace

Function

Estimates the amount of space required for a table and its indexes, and the time needed to create the index.

Syntax

```
sp_estspace table_name, no_of_rows [, fill_factor
   [, cols_to_max [, textbin_len [, iosec]]]]
```

Parameters

table_name – is the name of the table. It must already exist in the current database.

no_of_rows – is the estimated number of rows that the table will contain.

fill_factor – is the index fillfactor. The default is null, which means that Adaptive Server uses its default fillfactor.

cols_to_max – is a comma-separated list of the variable-length columns for which you want to use the maximum length instead of the average. The default is the average declared length of the variable-length columns.

textbin_len – is the length, per row, of all *text* and *image* columns. The default value is 0. You need to provide a value only if the table stores *text* or *image* data. *text* and *image* columns are stored in a separate set of data pages from the rest of the table's data. The actual table row stores a pointer to the *text* or *image* value. **sp_estspace** provides a separate line of information about the size of the *text* or *image* pages for a row.

iosec – is the number of disk I/Os per second on this machine. The default is 30 I/Os per second.

Examples

1. sp_estspace titles, 10000, 50, "title,notes", 0, 25

```
name             type         idx_level Pages        Kbytes
---------------- ------------ --------- ------------ ------------
titles           data                 0         3364         6728
titles           text/image           0            0            0
titleidind       clustered            0           21           43
titleidind       clustered            1            1            2
titleind         nonclustered         0         1001         2002
titleind         nonclustered         1           54          107
titleind         nonclustered         2            4            8
titleind         nonclustered         3            1            2

Total_Mbytes
-----------------
             8.68

name                 type         total_pages  time_mins
-------------------- ------------ ------------ ------------
titleidind           clustered            3386           13
titleind             nonclustered         1060            5
titles               data                    0            2
```

Calculates the space requirements for the *titles* table and its indexes, and the time required to create the indexes. The number of rows is 10,000, the fillfactor is 50 percent, two variable-length columns are computed using the maximum size for the column, and the disk I/O speed is 25 I/Os per second.

2. declare @i int
select @i = avg(datalength(pic)) from au_pix
exec sp_estspace au_pix, 1000, null, null, @i

```
au_pix has no indexes
 name             type         idx_level Pages     Kbytes
 ---------------- ------------ --------- --------- ---------
 au_pix           data                 0        31        63
 au_pix           text/image           0     21000     42000

 Total_Mbytes
 -----------------
             41.08
```

Uses the average length of existing *image* data in the *au_pix* table to calculate the size of the table with 1000 rows. You can also provide this size as a constant.

3. sp_estspace titles, 50000

```
name             type         idx_level Pages        Kbytes
---------------- ------------ --------- ------------ ------------
titles           data                 0         4912         9824
titleidind       clustered            0           31           61
titleidind       clustered            1            1            2
titleind         nonclustered         0         1390         2780
titleind         nonclustered         1           42           84
titleind         nonclustered         2            2            4
titleind         nonclustered         3            1            2

Total_Mbytes
-----------------
            12.46

name               type         total_pages  time_mins
------------------ ------------ ------------ ------------
titleidind         clustered            4943           19
titleind           nonclustered         1435            8
```

Calculates the size of the *titles* table with 50,000 rows, using defaults for all other values.

Comments

- To estimate the amount of space required by a table and its indexes:

  a. Create the table.

  b. Create all indexes on the table.

  c. Run **sp_estspace**, giving the table name, the estimated number of rows for the table, and the optional arguments, as needed.

  You do not need to insert data into the tables. **sp_estspace** uses information in the system tables—not the size of the data in the tables—to calculate the size of tables and indexes.

- If the **auto identity** option is set in a database, Adaptive Server automatically defines a 10-digit IDENTITY column in each new table that is created without specifying a **primary** key, a **unique** constraint, or an IDENTITY column. To estimate how much extra space is required by this column:

  a. In the master database, use **sp_dboption** to turn on the **auto identity** option for the database.

  b. Create the table.

  c. Run **sp_estspace** on the table and record the results.

d. Drop the table.

e. Turn the **auto identity** option off for the database.

f. Re-create the table.

g. Rerun **sp_estspace** on the table, and record the results.

- For information about tables or columns, use **sp_help** *tablename*.

Messages

- `Object does not exist in this database.`

  You can use **sp_estspace** only on tables that already exist in the current database.

- `Table contains text/image type columns. You must specify the total length per row for these columns in the argument list.`

  The table you specified contains *text* or *image* columns. Specify a length for these columns as the fifth argument. See example 2.

Permissions

Any user can execute **sp_estspace**.

Tables Used

syscolumns, *sysindexes*, *sysobjects*

See Also

Commands	**create index**, **create table**
System procedures	**sp_help**

sp_extendsegment

Function

Extends the range of a segment to another database device.

Syntax

sp_extendsegment *segname, dbname, devname*

Parameters

segname – is the name of the existing segment previously defined with **sp_addsegment**.

dbname – is the name of the database on which to extend the segment. *dbname* must be the name of the current database.

devname – is the name of the database device to be added to the current database device range already included in *segname*.

Examples

1. sp_extendsegment indexes, pubs2, dev2

This command extends the range of the segment *indexes* for the database *pubs2* on the database device *dev2*.

Comments

- After defining a segment, you can use it in the **create table** and **create index** commands to place the table or index on the segment. If you create a table or index on a particular segment, subsequent data for the table or index is located on that segment.
- To associate a segment with a database device, create or alter the database with a reference to that device. A database device can have more than one segment associated with it.
- A segment can be extended over several database devices.
- When you extend the *logsegment* segment, Adaptive Server recalculates its last-chance threshold.

Messages

- `Can't run sp_extendsegment from within a transaction.`

  sp_extendsegment updates system tables, so it cannot be run from within a transaction.

- `Device 'devname' is now exclusively used by 'segname'.`

  sp_extendsegment succeeded.

- `'devname' is reserved exclusively as a log device.`

  You cannot create a segment on a database device that is dedicated to the database log.

- `No such device exists -- run sp_helpdb to list the devices for the current database.`

  The named device does not exist in *master.dbo.sysdevices*.

- `Segment extended.`

  sp_extendsegment succeeded. The segment named *segname* now includes space on the database device *devname*.

- `'segname' is not a valid identifier.`

  Segment names must conform to the rules for identifiers. They must begin with a letter, an underscore character (_), or a pound sign (#). After the first character, identifiers can include letters, underscores, pound signs, or dollar signs ($).

- `The specified device is not used by the database.`

  Although the device named as the *devname* parameter exists in *master.dbo.sysdevices*, it is not used by the specified database. Segments can be extended only on database devices used by the database. Use **alter database** to extend a database on a device listed in the *master.dbo.sysdevices* table.

- `There is no such segment as 'segname'.`

  The segment you tried to extend does not exist. Run **sp_helpsegment** to list the segments for a database.

- `This command has been ignored. Extending the log segment on device 'devname' would leave no space for creating objects in database 'database_name'.`

  devname is the only or last database device with space available for the database *database_name*.

- `You must execute this procedure from the database in which you wish to add a segment. Please execute 'use database_name' and try again.`

  sp_extendsegment can extend segments only in the database from which it is run. Qualify **sp_extendsegment** with the database name or issue the **use** *database_name* command to open the database in

which you want to extend a segment. Then run **sp_extendsegment** again.

Permissions

Only the Database Owner or a System Administrator can execute **sp_extendsegment.**

Tables Used

master.dbo.sysdatabases, sysdevices, master.dbo.sysusages, sysobjects, syssegments

See Also

Commands	**alter database**, **create index**, **create table**
System procedures	**sp_addsegment**, **sp_dropsegment**, **sp_helpdb**, **sp_helpdevice**, **sp_helpsegment**, **sp_placeobject**

sp_familylock

Function

Reports information about all the locks held by a family (coordinating process and its worker processes) executing a statement in parallel.

Syntax

sp_familylock [*fpid1* [, *fpid2*]]

Parameters

fpid1 – is the family identifier for a family of worker processes from the *master.dbo.sysprocesses* table. Run **sp_who** or **sp_lock** to get the *spid* of the parent process.

pfid2 – is the Adaptive Server process ID number for another lock.

Examples

1. sp_familylock 5

```
fid  spid locktype     table_id   page dbname  class            context
---  ---- ----------   --------   ---- ------  ---------------  --------
5    5    Sh_intent     176003658  0    userdb  Non cursor lock Sync-
pt duration request
5    5    Sh_intent-blk 208003772  0    userdb  Non cursor lock Sync-
pt duration request
5    6    Sh_page       208003772  3972 userdb  Non cursor lock Sync-
pt duration request
5    7    Sh_page       208003772  3973 userdb  Non cursor lock Sync-
pt duration request
5    8    Sh_page       208003772  3973 userdb  Non cursor lock Sync-
pt duration request
```

Displays information about the locks held by all members of the family with an *fid* of 5.

Comments

- **sp_familylock** with no parameter reports information on all processes belonging to families that currently hold locks. The report is identical to the output from **sp_lock**; however, **sp_familylock** allows you to generate reports based on the family ID, rather than the process ID. It is useful for detecting family deadlocks.

- Use the **object_name** system function to derive a table's name from its ID number.
- The "locktype" column indicates whether the lock is a shared lock ("Sh" prefix), an exclusive lock ("Ex" prefix) or an update lock, and whether the lock is held on a table ("table" or "intent") or on a page ("page").

  The "blk" suffix in the "locktype" column indicates that this process is blocking another process that needs to acquire a lock. As soon as this process completes, the other process(es) moves forward. The "demand" suffix indicates that the process is attempting to acquire an exclusive lock.
- The "class" column indicates whether a lock is associated with a cursor. It displays one of the following:
  - "Non cursor lock" indicates that the lock is not associated with a cursor.
  - "Cursor Id *number*" indicates that the lock is associated with the cursor ID number for that Adaptive Server process ID.
  - A cursor name indicates that the lock is associated with the cursor *cursor_name* that is owned by the current user executing **sp_lock**.
- The "fid" column identifies the family (including the coordinating process and its worker processes) to which a lock belongs. Values for "fid" are as follows:
  - A zero value indicates that the task represented by the *spid* is executed in serial. It is not participating in parallel execution.
  - A nonzero value indicates that the task (*spid*) holding the lock is a member of a family of processes (identified by "fid") executing a statement in parallel. If the value is equal to the *spid*, it indicates that the task is the coordinating process in a family executing a query in parallel.
- The "context" column identifies the context of the lock. Worker processes in the same family have the same context value. Values for "context" are as follows:
  - "NULL" means that the task holding this lock is either executing a query in serial or is a query being executed in parallel in transaction isolation level 1.
  - "FAM_DUR" means that the task holding the lock will hold the lock until the query is complete.

A lock's context may be "FAM_DUR" if the lock is a table lock held as part of a parallel query, if the lock is held by a worker process at transaction isolation level 3, or if the lock is held by a worker process in a parallel query and must be held for the duration of the transaction.

Messages

- `The class column will display the cursor name for locks associated with a cursor for the current user and the cursor id for other users.`

  This message appears every time you run **sp_familylock**.

Permissions

Any user can execute **sp_familylock**.

Tables Used

master.dbo.spt_values, *master.dbo.syslocks*, *sysobjects*, *master.dbo.sysprocesses*.

See Also

Commands	**kill**, **select**
System procedures	**sp_lock**, **sp_who**

sp_forceonline_db

Function

Provides access to all the pages in a database that were previously marked suspect by recovery.

Syntax

```
sp_forceonline_db dbname,
   {"sa_on" | "sa_off" | "all_users"}
```

Parameters

dbname – is the name of the database to be brought online.

sa_on – allows only users with the **sa_role** access to the specified page.

sa_off – revokes access privileges created by a previous invocation of **sp_forceonline_page** with **sa_on**.

all users – allows all users access to the specified page.

Examples

1. `sp_forceonline_db pubs2, "sa_on"`

   Allows the System Administrator access to all suspect pages in the *pubs2* database.

2. `sp_forceonline_db pubs2, "sa_off"`

   Revokes access to all suspect pages in the *pubs2* database from the System Administrator. Now, no one can access the suspect pages in *pubs2*.

3. `sp_forceonline_db pubs2, "all_users"`

   Allows all users access to all pages in the *pubs2* database.

Comments

- A page that is forced online is not necessarily repaired. Corrupt pages can also be forced online. Adaptive Server does not perform any consistency checks on pages that are forced online.
- **sp_forceonline_page** with **all users** cannot be reversed. When pages have been brought online for all users, you cannot take them offline again.
- **sp_forceonline_db** cannot be used in a transaction.

- To bring only specific offline pages online, use **sp_forceonline_page**.

Messages

- `Can't run sp_forceonline_db from within a transaction.`

  sp_forceonline_db modifies system tables, so it cannot be run within a transaction.

- `Invalid option` *`option`*`. Use 'sa_on', 'sa_off', or 'all_users'.`

  Use one of the specified options.

- `No such database -- run sp_helpdb to list databases.`

  Run **sp_helpdb** and verify the database name.

- `No suspect pages in database` *`dbname`*`. Use sp_listsuspect_page to list suspect pages.`

  The command is ignored because the specified database has no offline pages.

- `Permission denied. This operation requires System Administrator (sa_role) role.`

  Only a System Administrator can use **sp_forceonline_db**.

Permissions

Only the System Administrator can execute **sp_forceonline_db**.

Tables Used

master.dbo.sysattributes

See Also

System procedures	**sp_forceonline_page, sp_listsuspect_db**, **sp_listsuspect_page**, **sp_setsuspect_granularity**, **sp_setsuspect_threshold**

sp_forceonline_page

Function

Provides access to pages previously marked suspect by recovery.

Syntax

```
sp_forceonline_page dbname, pgid,
   {"sa_on" | "sa_off" | "all_users"}
```

Parameters

dbname – is the name of the database containing the pages to be brought online.

pgid – is the page identifier of the page being brought online.

sa_on – allows only users with the **sa_role** access to the specified page.

sa_off – revokes access privileges created by a previous invocation of **sp_forceonline_page** with **sa_on**.

all_users – allows all users access to the specified page.

Examples

1. `sp_forceonline_page pubs2, 312, "sa_on"`

   Allows a System Administrator access to page 312 in the *pubs2* database.

2. `sp_forceonline_page pubs2, 312, "sa_off"`

   Revokes access to page 312 in the *pubs2* database from the System Administrator. Now, no one has access to this page.

3. `sp_forceonline_page pubs2, 312, "all_users"`

   Allows all users access to page 312 in the *pubs2* database.

Comments

- **sp_forceonline_page** with **all_users** cannot be reversed. When pages have been brought online for all users, you cannot take them offline again.
- A page that is forced online is not necessarily repaired. Corrupt pages can also be forced online. Adaptive Server does not perform any consistency checks on pages that are forced online.
- **sp_forceonline_page** cannot be used in a transaction.

- **sp_forceonline_page** works only for databases in which the recovery fault isolation mode is "page." Use **sp_setsuspect_granularity** to display the recovery fault isolation mode for a database.
- To bring all of a database's offline pages online in a single command, use **sp_forceonline_db**.

Messages

- `Can't run sp_forceonline_page from within a transaction.`

  sp_forceonline_page modifies system tables, so it cannot be run within a transaction.

- `No such database -- run sp_helpdb to list databases.`

  Run **sp_helpdb** and verify the database name.

- `All pages/objects in database` *`databasename`* `are now online.`

  This informational message appears if the *pgid* is the last offline page in the database.

- `Invalid option` *`option`*`. Use 'sa_on', 'sa_off', or 'all_users'.`

  Use one of the specified options.

- `Permission denied. This operation requires System Administrator (sa_role) role.`

  Only a System Administrator can use **sp_forceonline_page**.

- `The page` *`pageid`* `in` *`databasename`* `is either already online or is an invalid ID.`

  Use **sp_listsuspect_page** to list suspect pages.

- `The page` *`pageid`* `of database` *`datebasename`* `is already in state` *`state`*`.`

  The command is ignored because the specified page is already online.

- `The remaining suspect pages in database` *`databasenam`*`e are:`

  This informational message is displayed on a successful execution of **sp_forceonline_page**, if additional pages remain offline.

- `You must be in the 'master' database in order to change database options.`

  Change to the *master* database, and issue **sp_forceonline_page** again.

Permissions

Only the System Administrator can use **sp_forceonline_page**.

Tables Used

master.dbo.sysattributes

See Also

System procedures	**sp_forceonline_db, sp_listsuspect_db**, **sp_listsuspect_page**, **sp_setsuspect_granularity**, **sp_setsuspect_threshold**

sp_foreignkey

Function

Defines a foreign key on a table or view in the current database.

Syntax

```
sp_foreignkey tabname, pktabname, col1 [, col2] ...
   [, col8]
```

Parameters

tabname – is the name of the table or view that contains the foreign key to be defined.

pktabname – is the name of the table or view that has the primary key to which the foreign key applies. The primary key must already be defined.

col1 – is the name of the first column that makes up the foreign key. The foreign key must have at least one column and can have a maximum of eight columns.

Examples

1. `sp_foreignkey titles, publishers, pub_id`

   The primary key of the *publishers* table is the *pub_id* column. The *titles* table also contains a *pub_id* column, which is a foreign key of *publishers*.

2. `sp_foreignkey orders, parts, part, subpart`

   The primary key of the *parts* table has been defined with **sp_primarykey** as the *partnumber* and *subpartnumber* columns. The *orders* table contains the columns *part* and *subpart*, which make up a foreign key of *parts*.

Comments

- **sp_foreignkey** adds the key to the *syskeys* table. Keys make explicit a logical relationship that is implicit in your database design.
- **sp_foreignkey** does not enforce referential integrity constraints; use the **foreign key** clause of the **create table** or **alter table** command to enforce a foreign key relationship.
- The number and order of columns that make up the foreign key must be the same as the number and order of columns that make

up the primary key. The datatypes (and lengths) of the primary and foreign keys must agree, but the null types need not agree.

- The installation process runs **sp_foreignkey** on the appropriate columns of the system tables.
- To display a report on the keys that have been defined, execute **sp_helpkey.**

Messages

- `Datatypes of the column 'column_name' in the keys are different.`

  The datatypes of the columns of the foreign key *tabname* and the primary key *pktabname* must be the same.

- `Foreign key table doesn't exist.`

  The table or view you specified with the *tabname* parameter does not exist in the current database.

- `New foreign key added.`

  The foreign key has been defined and added to *syskeys*.

- `Only the owner of the table may define a foreign key.`

  You are not the owner of the table or view.

- `Primary key does not exist with the same number of columns as the foreign key.`

  The number of columns in the foreign key *tabname* and in the primary key *pktabname* must be the same.

- `Primary key table doesn't exist.`

  The table or view you specified with the *pktabname* parameter does not exist in the current database or does not have a primary key defined.

- `Table or view name must be in current database.`

  You cannot add foreign keys to a table or view in a different database.

- `The table does not have a column named 'column_name'.`

  tabname does not have a column of the specified name. Use **sp_help** *tablename* to list column names.

- `Primary key does not exist.`

  The primary key specified with the *col1–col8* parameters does not exist, or is not defined in the primary key table.

Permissions

You must be the owner of the table or view in order to define its foreign key.

Tables Used

syscolumns, *sysindexes*, *syskeys*, *sysobjects*, *sysreferences*

See Also

Commands	**alter table**, **create table**, **create trigger**
System procedures	**sp_commonkey**, **sp_dropkey**, **sp_helpkey**, **sp_helpjoins**, **sp_primarykey**

sp_freedll

Function

Unloads a dynamic link library (DLL) that was previously loaded into XP Server memory to support the execution of an extended stored procedure (ESP).

Syntax

```
sp_freedll dll_name
```

Parameters

dll_name – is the file name of the DLL being unloaded from XP Server memory.

Examples

```
1. sp_freedll "sqlsrvdll.dll"
```

Unloads the *sqlsrvdll.dll* DLL.

Comments

- **sp_freedll** cannot be executed from within a transaction.
- **sp_freedll** cannot free the DLL of a system ESP.
- An alternative to unloading a DLL explicitly, using **sp_freedll**, is to specify that DLLs always be unloaded after the ESP request that invoked them terminates. To do this, set the **esp unload dll** configuration parameter to 1 or start **xpserver** with the **-u** option.
- **sp_freedll** can be used to update an ESP function in a DLL without shutting down XP Server or Adaptive Server.
- If you use **sp_freedll** to unload a DLL that is in use, **sp_freedll** will succeed, causing the ESP currently using the DLL to fail.

Messages

- `Can't run sp_freedll from within a transaction.`

  sp_freedll modifies system tables, so it cannot be run within a transaction.
- `DLL for the extended stored procedure` *`procedure_name`* `is not found.`

This message appears when you attempt to free a DLL that is not recorded in the system tables. Use **sp_helpextendedproc** to see a list of the DLLs that support ESPs.

- *dll_name* `is not a valid name.`

  The *dll_name* must have the syntax of an identifier.

- `sp_freedll failed. Check the SQL Server error log file.`

  Make sure that the DLL exists and that you are using its correct name.

Permissions

Only the System Administrator can execute **sp_freedll**.

Tables Used

master.dbo.syscomments, sysobjects

See Also

System procedures	**sp_addextendedproc, sp_dropextendedproc, sp_helpextendedproc**

sp_getmessage

Function

Retrieves stored message strings from *sysmessages* and *sysusermessages* for **print** and **raiserror** statements.

Syntax

```
sp_getmessage message_num, result output [, language]
```

Parameters

message_num – is the number of the message to be retrieved.

result **output** – is the variable that receives the returned message text, followed by a space and the keyword **output**. The variable must have a datatype of *char*, *nchar*, *varchar*, or *nvarchar*.

language – is the language of the message to be retrieved. *language* must be a valid language name in *syslanguages* table. If you include *language*, the message with the indicated *message_num* and *language* is retrieved. If you do not include *language*, then the message for the default session language, as indicated by the variable *@@langid*, is retrieved.

Examples

1.
```
declare @myvar varchar(200)
exec sp_getmessage 20001, @myvar output
```

   Retrieves message number 20001 from *sysusermessages*.

2.
```
declare @myvar varchar(200)
exec sp_getmessage 20010, @myvar output, french
```

   Retrieves the French language version of message number 20010 from *sysusermessages*.

Comments

- Any application can use **sp_getmessage**, and any user can read the messages stored in *sysmessages* and *sysusermessages*.

Messages

- `Message number must be greater than or equal to 17000.`

  The message number you specified is invalid.

- '*language*' is not an official language name from syslanguages.

  The name you specified is not a valid name in the *syslanguages* table.

- Message number *message_num* does not exist in the language *language*.

  The message number you specified does not exist in the specified language.

Permissions

Any user can issue **sp_getmessage.**

Tables Used

master.dbo.syslanguages, *master.dbo.sysmessages*, *sysobjects*, *sysusermessages*

See Also

Commands	**print**, **raiserror**
System procedures	**sp_addmessage**, **sp_dropmessage**

sp_grantlogin

(Windows NT only)

Function

Assigns Adaptive Server roles or **default** permissions to Windows NT users and groups when Integrated Security mode or Mixed mode (with Named Pipes) is active.

Syntax

```
sp_grantlogin {login_name | group_name}
   ["role_list" | default]
```

Parameters

login_name – is the network login name of the Windows NT user.

group_name – is the Windows NT group name.

role_list – is a list of the Adaptive Server roles granted. The role list can include one or more of the following role names: **sa_role**, **sso_role**, **oper_role**. If you specify more than one role, separate the role names with spaces, not commas.

default – specifies that the *login_name* or *group_name* receive default permissions assigned with the **grant** statement or **sp_role** procedure.

Examples

1. `sp_grantlogin jeanluc, oper_role`

   Assigns the Adaptive Server **oper_role** to the Windows NT user "jeanluc".

2. `sp_grantlogin valle`

   Assigns the **default** value to the Windows NT user "valle". User "valle" receives any permissions that were assigned to her via the **grant** command or **sp_role** procedure.

3. `sp_grantlogin Administrators, "sa_role sso_role"`

   Assigns the Adaptive Server **sa_role** and **sso_role** to all members of the Windows NT administrators group.

Comments

- You must create the Windows NT login name or group before assigning roles with **sp_grantlogin**. See your Windows NT documentation for details.
- **sp_grantlogin** is active only when Adaptive Server is running in Integrated Security mode or Mixed mode when the connection is Named Pipes. If Adaptive Server is running under Standard mode or Mixed mode with a connection other than Named Pipes, use **grant** and **sp_role** instead.
- If you do not specify a *role_list* or **default**, the procedure automatically assigns the **default** value.
- The **default** value does not indicate an Adaptive Server role. It specifies that the user or group should receive any permissions that were assigned to it via the **grant** command or **sp_role** procedure.
- Using **sp_grantlogin** with an existing *login_name* or *group_name* overwrites the user's or group's existing roles.

Messages

- `Access granted.`

  sp_grantlogin successfully executed.
- `'`*`login_name`*`' is not a valid account name.`

  The specified Windows NT user name or group name does not exist.
- `Parameter '`*`role_list`*`' is invalid.`

  One or more role names in the specified *role_list* are invalid.
- `The account name provided is a domain. Unable to grant privileges to a domain.`

  The specified *login_name* or *group_name* matches a Windows NT domain name. Use only valid Windows NT user names or group names with **sp_grantlogin**.
- `The account name provided is a deleted account. Unable to grant privileges to a deleted account.`

  The specified login name was deleted.
- `Unable to get SQL Server security information.`

  A call to the Windows NT security API (Application Program Interface) failed. Contact your Windows NT administrator.

- `Unable to set SQL Server security information.`

  A call to the Windows NT security API failed. Contact your Windows NT administrator.

Permissions

Only users with System Administrator privileges can use **sp_grantlogin.**

Tables Used

sysobjects

See Also

Commands	**grant**, **setuser**
System procedures	**sp_addlogin**, **sp_addremotelogin**, **sp_adduser**, **sp_displaylogin**, **sp_droplogin**, **sp_dropuser**, **sp_locklogin**, **sp_logininfo**, **sp_modifylogin**, **sp_revokelogin**, **sp_who**

sp_help

Function

Reports information about a database object (any object listed in *sysobjects*) and about system or user-defined datatypes.

Syntax

sp_help [*objname*]

Parameters

objname – is the name of any object in *sysobjects* or any user-defined datatype or system datatype in *systypes*. You cannot specify database names. *objname* can include tables, views, stored procedures, logs, rules, defaults, triggers, referential constraints, and check constraints. Use owner names if the object owner is not the user running the command and is not the Database Owner.

Examples

1. sp_help

Displays a list of objects in *sysobjects* and displays each object's name, owner, and object type. Also displays a list of each user-defined datatype in *systypes*, indicating the datatype name, storage type, length, null type, default name, and rule name. Null type is 0 (null values not allowed) or 1 (null values allowed).

2. sp_help publishers

```
Name                        Owner                          Type
--------------------------- -----------------------------  -----------
publishers                  dbo                            user table

Data_located_on_segment        When_created
------------------------------ --------------------------
default                                 Apr 12 1994  3:31PM


Column_name Type Length Prec Scale Nulls Default_name Rule_name Identity
----------- ---- ------ ---- ----- ----- ------------ --------- --------
pub_id      char      4 NULL  NULL     0 NULL         pub_idrule       0
pub_name varchar     40 NULL  NULL     1 NULL         NULL             0
city     varchar     20 NULL  NULL     1 NULL         NULL             0
state       char      2 NULL  NULL     1 NULL         NULL             0
```

```
attribute_class attribute     int_value char_value            comments
--------------- ------------ --------- --------------------- --------
buffer manager  cache binding         1 publishers_cache          NULL


index_name         index_description                      index_keys
     index_max_rows_per_page
------------------ -------------------------------------- ----------
     ------------------------
pubind             clustered, unique located on default   pub_id
     0

name    attribute_class attribute  int_value char_value
         comments
------- --------------- ---------- --------- --------------------------
         --------
pubind  buffer manager  cache name      NULL cache for index pubind
         NULL

keytype     object                    related_object
         object_keys
         related_keys
----------- ------------------------- ---------------------------
        ------------
        ------------
primary     publishers                - none --
        pub_id, *, *, *, *, *, *, *
        *, *, *, *, *, *, *, *
foreign     titles                        publishers
        pub_id, *, *, *, *, *, *, *
        pub_id, *, *, *, *, *, *, *
Object is not partitioned.
```

Displays information about the *publishers* table. **sp_help** also lists any attributes assigned to the specified table and its indexes, giving the attribute's class, name, integer value, character value, and comments. The above example shows cache binding attributes for the *publishers* table.

3. sp_help partitioned_table

```
Name                         Owner        Type
-------------------------    ------------ -----------
partitioned_table            dbo          user table

Data_located_on_segment      When_created
----------------------       -------------------------
data1                        Mar 24 1995 10:48AM

Column_name  Type  Length Prec  Scale Nulls Default_name
-----------  ----- ------ ----  ----- ----- ------------
col1         char      5   NULL  NULL     0  NULL
```

```
Rule_name       Identity
----------      --------
NULL                   0

Object does not have any indexes.
No defined keys for this object.

partitionid      firstpage    controlpage
---------------- ----------- -----------
              1          145         146
              2          312         313
```

Displays information about a partitioned table.

4. sp_help "mary.marytrig"

```
Name          Owner              Type
------------ ------------------ ----------------
marytrig     mary               trigger

Data_located_on_segment When_created
----------------------- --------------------------
not applicable          Mar 20 1992  2:03PM
```

Displays information about the trigger *marytrig* owned by user "mary". The quotes are needed, because the period is a special character.

5. sp_help money

```
Type_name  Storage_type Length Prec  Scale
---------- ------------ ------ ----- -----

money      money             8  NULL  NULL

Nulls   Default_name  Rule_name   Identity
-----   ------------  ---------   --------
    1   NULL          NULL                0
```

Displays information about the system datatype *money*.

6. sp_help identype

```
Type_name  Storage_type Length Nulls Default_name
--------- ------------- ------ ----- ----------
identype   numeric           4     0 NULL

Rule_name       Identity
---------       --------
NULL                   1
```

Displays information about the user-defined datatype *identype*. The report indicates the base type from which the datatype was created, whether it allows nulls, the names of any rules and

defaults bound to the datatype, and whether it has the IDENTITY property.

Comments

- **sp_help** looks for an object in the current database only.
- **sp_help** follows the Adaptive Server rules for finding objects:
  - If you do not specify an owner name, and you own an object with the specified name, **sp_help** reports on that object.
  - If you do not specify an owner name, and do not own an object of that name, but the Database Owner does, **sp_help** reports on the Database Owner's object.
  - If neither you nor the Database Owner owns an object with the specified name, **sp_help** reports an error condition, even if an object with that name exists in the database for a different owner. Qualify objects that are owned by database users other than yourself and the Database Owner with the owner's name, as shown in example 4.
  - If both you and the Database Owner own objects with the specified name, and you want to access the Database Owner's object, specify the name in the format *dbo.objectname*.
- **sp_help** works on temporary tables if you issue it from *tempdb*.
- Columns with the IDENTITY property have an "Identity" value of 1; others have an "Identity" value of 0. In example 2, there are no IDENTITY columns.
- **sp_help** lists any indexes on a table, including indexes created by defining unique or primary key constraints in the **create table** or **alter table** statements. It also lists any attributes associated with those indexes. However, **sp_help** does not describe any information about the integrity constraints defined for a table. Use **sp_helpconstraint** for information about any integrity constraints.
- When Component Integration Services is enabled, **sp_help** displays information on the storage location of remote objects.

Messages

- `Object does not exist in this database.`

  The specified object does not exist in the current database.

- `Object must be in your current database.`

  sp_help only gives information about objects in the current database. Use **sp_helpdb** for information on the database itself.

Permissions

Any user can execute **sp_help**.

Tables Used

master.dbo.spt_values, *master.sysattributes*, *sysattributes*, *syscolumns*, *sysindexes*, *sysmessages*, *sysobjects*, *syspartitions*, *systypes*

See Also

System procedures	**sp_helpartition**, **sp_helpconstraint**, **sp_helpdb**, **sp_helpindex**, **sp_helpkey**, **sp_helprotect**, **sp_helpsegment**, **sp_helptext**, **sp_helpuser**

sp_helpartition

Function

Lists the partition number, first page, control page, and number of data pages and summary size information for each partition in a partitioned table.

Syntax

sp_helpartition [*table_name*]

Parameters

table_name – is the name of a partitioned table in the current database. If the table name is not supplied, the owner, tables name, and number of partitions is printed for all user tables in the database.

Examples

1. sp_helpartition sales

```
partitionid firstpage   controlpage ptn_data_pages
----------- ----------- ----------- --------------
          1         313         314           4227
          2       12802       12801           4285
          3       25602       25601           4404
          4       38402       38401           4523
          5       51202       51201           4347
          6       64002       64001           4285

(6 rows affected)
Partitions Average Pages Maximum Pages Minimum Pages Ratio (Max/Avg)
---------- ------------- ------------- ------------- ---------------
         6           4345          4523          4227        1.040967
```

Returns information about the partitions in *sales*.

Comments

- **sp_helpartition** lists the partition number, first page, control page, and number of data pages for each partition in a partitioned table. The number of pages per partition shows how evenly the data is distributed between partitions.

  The summary information display the number of partitions, the average number of pages per partition, the minimum and maximum number of pages, and the ratio between the average number of pages and the maximum number. This ratio is used

during query optimization. If the ratio is 2 or greater (meaning that the maximum size is twice as large as the average size), the optimizer chooses a serial query plan rather than a parallel plan.

- Partitioning a table creates additional page chains. Use the partition clause of the **alter table** command to partition a table. Each chain has its own last page, which is available for concurrent insert operations. This improves insert performance by reducing page contention. If the table is spread over multiple physical devices, partitioning improves insert performance by reducing I/O contention while Adaptive Server is flushing data from cache to disk.
- Partitioning a table does not affect its performance for update or delete commands.
- Use the **unpartition** clause of the **alter table** command to concatenate all existing page chains.
- Neither partitioning nor unpartitioning a table moves existing data.
- To change the number of partitions in a table, first use the **unpartition** clause of **alter table** to concatenate its page chains. Then use the **partition** clause of **alter table** to repartition the table.
- **sp_helpartition** looks only in the current database for the table.
- Use **sp_helpsegment** to display the number of used and free pages on the segment on where the partitioned table is stored.

Accuracy of Results

- The values reported in the "data_pages" column may be greater than the actual values. To determine whether the count is inaccurate, run **sp_statistics** and **sp_helpartition** to compare the data page count. The count provided by **sp_statistics** is always accurate.

  If the page count reported by **sp_statistics** differs from the sum of the partition pages reported by **sp_helpartition** by more then 5 percent, run one of the following commands to update the partition statistics:

  - **dbcc checkalloc**
  - **dbcc checkdb**
  - **dbcc checktable**
  - **update all statistics**
  - **update partition statistics**

Then, rerun **sp_helpartition** for an accurate report.

Messages

- `Object is not partitioned.`

  The table you specified is not partitioned.

- `Object does not exist in this database.`

  The table you specified does not exist in the current database.

- `Object must be in the current database.`

  You cannot run **sp_helpartition** on a table that is not in the current database. Either qualify **sp_helpartition** with the database name, or issue the **use** *database_name* command to open the database where the table resides, and then reissue **sp_helpartition**.

Permissions

Any user can execute **sp_helpartition**.

Tables Used

syspartitions

See Also

Commands	**alter table**, **insert**
System procedures	**sp_help**, **sp_helpsegment**

sp_helpcache

Function

Displays information about the objects that are bound to a data cache or the amount of overhead required for a specified cache size.

Syntax

```
sp_helpcache {cache_name | "cache_size[P|K|M|G]"}
```

Parameters

cache_name – is the name of an existing data cache.

cache_size – specifies the size of the cache, specified by **P** for pages, **K** for kilobytes, **M** for megabytes, or **G** for gigabytes. The default is **K**.

Examples

1. **`sp_helpcache pub_cache`**

   Displays information about items bound to *pub_cache*.

2. **`sp_helpcache "80M"`**

   Shows the amount of overhead required to create an 80MB data cache.

3. **`sp_helpcache`**

   Displays information about all caches and all items bound to them.

Comments

- To see the size, status, and I/O size of all data caches on the server, use **sp_cacheconfig**.
- When you configure data caches with **sp_cacheconfig**, all the memory that you specify is made available to the data cache. Overhead for managing the cache is taken from the default data cache. The **sp_helpcache** displays the amount of memory required for a cache of the specified size.
- To bind objects to a cache, use **sp_bindcache**. To unbind a specific object from a cache, use **sp_unbindcache**. To unbind all objects that are bound to a specific cache, use **sp_unbindcache_all**.

- The procedure **sp_cacheconfig** configures data caches. The procedure **sp_poolconfig** configures memory pools within data caches.
- **sp_helpcache** computes overhead accurately up to 74GB.

Messages

- `Can't run sp_helpcache from within a transaction.`

  sp_helpcache creates a temporary table, so you cannot run it in a transaction.

- `The database '`*`dbname`*`' is offline. To obtain cache-bindings for objects in this database, please online the database and rerun sp_helpcache.`

  The database is off line. Use the **online database** command to bring the database online when all loads are complete, and execute **sp_helpcache** again.

- `The specified named cache '`*`cache_name`*`' does not exist.`

  To see the caches available, run **sp_cacheconfig** with no parameters.

Permissions

All users can execute **sp_helpcache**.

Tables Used

master..sysattributes, *master..sysdatabases*, *sysattributes*, *sysindexes*, *sysobjects*

See Also

System procedures	**sp_bindcache**, **sp_cacheconfig**, **sp_poolconfig**, **sp_unbindcache**, **sp_unbindcache_all**

sp_helpconfig

Function

Reports help information on configuration parameters.

Syntax

sp_helpconfig "*configname*", ["*size*"]

Parameters

configname – is the configuration parameter being queried, or a non-unique parameter fragment.

size – is the size of memory, specified by **B** (bytes), **K** (kilobytes), **M** (megabytes), **G** (gigabytes), or **P** (pages). Used without the type of size specified, *size* specifies the number of the entity being configured using this parameter, for examples, locks, open indexes, and so on. *size* is ignored if *configname* is not a unique parameter name.

Examples

1. sp_helpconfig "allow"

```
Configuration option is not unique.
 option_name                    config_value run_value
 ------------------------------ ------------ -----------
 allow backward scans                      1           1
 allow nested triggers                     1           1
 allow procedure grouping                  1           1
 allow remote access                       1           1
 allow resource limits                     0           0
 allow sendmsg                             0           0
 allow sql server async i/o                1           1
 allow updates to system tables            0           0
```

2. sp_helpconfig "open objects", "421"

```
number of open objects sets the maximum number of database objects
that are open at one time on SQL Server. The default run value is 500.

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
-------------  -------------  -------------  -------------  -----------
          100     2147483647            500            500          243

Configuration parameter, 'number of open objects', will consume 207K
of memory if configured at 421.
```

Returns a report on how much memory is needed to create a metadata cache for 421 object descriptors.

3. sp_helpconfig "open databases", "1M"

```
number of open databases sets the maximum number of databases that can
be open at one time on SQL Server. The default run value is 12.

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
-------------  -------------  -------------  -------------  -----------
            5     2147483647             12             12          433

Configuration parameter, 'number of open databases', can be configured
to 28 to fit in 1M of memory.
```

Returns a report on how many database descriptors would fill a 1MB database cache.

4. sp_helpconfig "number of locks", "512K"

```
number of locks sets the number of available locks. The default run
value is 5000.

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
-------------  -------------  -------------  -------------  -----------
         1000    2147483647           5000           5000          528

Configuration parameter 'number of locks', can be configured to 4848
to fit in 512K of memory.
```

Returns a report on how many locks will use 512K of memory.

5. sp_helpconfig "allow updates to system tables"

```
allow updates to system tables allows system tables to be updated
directly. The default is 0 (off).

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
-------------  -------------  -------------  -------------  -----------
            0              1              0              0            0
```

Returns a report on the status of the **allow updates to system tables** configuration parameter.

Comments

- **sp_helpconfig** reports help information on configuration parameters, such as how much memory would be needed if the parameter were set to a certain value. **sp_helpconfig** also displays the current setting, the amount of memory used for that setting, the default value, and the minimum and maximum settings.

- If you use a nonunique parameter fragment for *configname*, **sp_helpconfig** returns a list of matching parameters with their configured values and current values. See example 1.
- Use **sp_helpconfig** when you are planning a metadata cache configuration for a server.

  For example, suppose you were planning to move a database that contained 2000 user indexes to a different server. To find how much memory you would need to configure for that server so that it would accommodate the database's user indexes, enter the following command:

  ```
  sp_helpconfig "open indexes", "2000"
  ```

```
number of open indexes sets the maximum number of indexes that can be
open at one time on SQL Server. The default run value is 500.

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
-------------  -------------  -------------  -------------  -----------
          100     2147483647            500            500          208

Configuration parameter, 'number of open indexes', will consume 829k
of memory if configured at 2000.
```

Alternatively, suppose you had 1MB of memory available for the index cache, and you needed to know how many index descriptors it would support. Run the following command:

```
sp_helpconfig "open indexes", "1M"
```

```
number of open indexes sets the maximum number of indexes that can be
open at one time on SQL Server. The default run value is 500.

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
-------------  -------------  -------------  -------------  -----------
          100     2147483647            500            500          208

Configuration parameter 'number of open indexes', can be configured to
2461 to fit in 1M of memory.
```

Based on this output, if you have 1MB of memory, you can create an index descriptor cache that can contain a maximum of 2461 index descriptors. To create this cache, set the **number of open indexes** configuration parameter as follows:

```
sp_configure "number of open indexes", 2461
```

Messages

- `Can't run sp_helpconfig from within a transaction.`

  sp_helpconfig cannot be run within a transaction because it modifies system tables.

- Component Integration Services must be enabled and loaded in order to perform memory calculations.

  To calculate values for **max cis remote servers**, enable Component Integration Services.
- Configuration parameter, '*configname*', will consume *value*K of memory if configured at *size*.
- onfiguration parameter, '*configname*', can be configured to *value* to fit in *size* of memory.
- Changing the value of '*configname*' does not increase the amount of memory SQL Server uses.
- Setting 'total memory' to a size of *size* will result in a 'total data cache size' of size *value*K.
- Setting 'total memory' to a size of *size* will result in a 'procedure cache' of size *value*K.

Permissions

Any user can use **sp_helpconfig**.

Tables Used

sysindexes, sysobjects, sysdatabases

See Also

System procedures	**sp_configure**, **sp_countmetadata, sp_monitorconfig**

sp_helpconstraint

Function

Reports information about integrity constraints used in the specified tables.

Syntax

sp_helpconstraint [*objname*] [, detail]

Parameters

objname – is the name of a table that has one or more integrity constraints defined by a **create table** or **alter table** statement.

detail – returns information about the constraint's user or error messages.

Examples

1. sp_helpconstraint store_employees

```
name                          defn
----------------------------  --------------------------------
store_empl_stor_i_272004000   store_employees FOREIGN KEY
                              (stor_id) REFERENCES stores(stor_id)
store_empl_mgr_id_288004057   store_employees FOREIGN KEY
                              (mgr_id) SELF REFERENCES
                              store_employees(emp_id)
store_empl_2560039432         UNIQUE INDEX( emp_id) :
                              NONCLUSTERED, FOREIGN REFERENCE

(3 rows affected)

Total Number of Referential Constraints: 2

Details:
-- Number of references made by this table: 2
-- Number of references to this table: 1
-- Number of self references to this table: 1

Formula for Calculation:
Total Number of Referential Constraints
= Number of references made by this table
+ Number of references made to this table
- Number of self references within this table
```

Displays the constraint information for the *store_employees* table in the *pubs3* database. The *store_employees* table has a foreign key to the *stores* table (*stor_id*) and a self-reference (*mgr_id* references *emp_id*).

2. sp_helpconstraint titles, detail

```
name                           type
      defn
        msg
-----------------------------  ------------------------
     ------------------------------------------------------------
       ------------------------------------------
datedflt                       default value
      create default datedflt as getdate()

typedflt                       default value
      create default typedflt as "UNDECIDED"

titles_pub_id_96003373         referential constraint
      titles FOREIGN KEY (pub_id) REFERENCES publishers(pub_id)
        standard system error message number : 547

roysched_title__144003544      referential constraint
      roysched FOREIGN KEY (title_id) REFERENCES titles(title_id)
        standard system error message number : 547

salesdetai_title__368004342    referential constraint
      salesdetail FOREIGN KEY (title_id) REFERENCES titles(title_id)
        standard system error message number : 547

titleautho_title__432004570    referential constraint
      titleauthor FOREIGN KEY (title_id) REFERENCES titles(title_id)
        standard system error message number : 547

titles_800033162               unique constraint
      UNIQUE INDEX ( title_id) : NONCLUSTERED, FOREIGN REFERENCE
        standard system error message number : 2601

(7 rows affected)

Total Number of Referential Constraints: 4

Details:
-- Number of references made by this table: 1
-- Number of references to this table: 3
-- Number of self references to this table: 0
```

```
Formula for Calculation:
Total Number of Referential Constraints
= Number of references made by this table
+ Number of references made to this table
- Number of self references within this table.
```

Displays more detailed information about the *pubs3..salesdetail* constraints, including the constraint type and any constraint error messages.

3. sp_helpconstraint

```
id          name                      Num_referential_constraints
----------- ------------------------- ---------------------------
   80003316 titles                                              4
   16003088 authors                                             3
  176003658 stores                                              3
  256003943 salesdetail                                         3
  208003772 sales                                               2
  336004228 titleauthor                                         2
  896006223 store_employees                                     2
   48003202 publishers                                          1
  128003487 roysched                                            1
  400004456 discounts                                           1
  448004627 au_pix                                              1
  496004798 blurbs                                              1

(11 rows affected)
```

Displays a listing of all tables in the *pubs3* database.

Comments

- **sp_helpconstraint** prints the name and definition of the integrity constraint, and the number of references used by the table. The **detail** option returns information about the constraint's user or error messages.
- Running **sp_helpconstraint** with no parameters lists all the tables containing references in the current database, and displays the total number of references in each table. **sp_helpconstraint** lists the tables in descending order, based on the number of references in each table.
- **sp_helpconstraint** reports only the integrity constraint information about a table (defined by a **create table** or **alter table** statement). It does not report information about rules, triggers, or indexes created using the **create index** statement. Use **sp_help** to see information about rules, triggers, and indexes for a table.

- For constraints that do not have user-defined messages, Adaptive Server reports the system error message associated with the constraint. Query *sysmessages* to obtain the actual text of that error message.
- You can use **sp_helpconstraint** only for tables in the current database.
- If a query exceeds the configured number of auxiliary scan descriptors, Adaptive Server returns an error message. You can use **sp_helpconstraint** to determine the necessary number of scan descriptors. For details, see the description of the **number of aux scan descriptors** configuration parameter in Chapter 11, "Setting Configuration Parameters," in the *System Administration Guide.*

Messages

- `Object must be in current database.`

  sp_helpconstraint provides information about objects in the current database only. Use **sp_helpdb** for information about the database itself.
- `Object does not exist in this database.`

  The table you specified does not exist in the current database.
- `Object does not have any declarative constraints.`

  The object you specified does not have any declarative integrity constraints. **sp_helpconstraint** reports information only about the constraints defined by the **create table** or **alter table** statements.

Permissions

Any user can execute **sp_helpconstraint.**

A System Security Officer can restrict **select** permission on the *text* column of the *syscomments* table to the object owner and the System Administrator by resetting the **allow select on syscomments.text column** parameter with the system procedure **sp_configure.** This restriction prevents the display of constraint definitions when other users execute **sp_helpconstraint.**

This restriction is required in order to run Adaptive Server in the evaluated configuration. For more information about the evaluated configuration, see Chapter 1, "Overview of Security Features," in the *Security Administration Guide.*

Tables Used

syscolumns, *syscomments*, *sysconstraints*, *sysindexes*, *sysobjects*, *sysreferences*, *sysusermessages*

See Also

Commands	**alter table**, **create table**
System procedures	**sp_help**, **sp_helpdb**, **sp_monitorconfig**

sp_helpdb

Function

Reports information about a particular database or about all databases.

Syntax

sp_helpdb [*dbname*]

Parameters

dbname – is the name of the database on which to report information. Without this optional parameter, **sp_helpdb** reports on all databases.

Examples

1. sp_helpdb

```
name           db_size  owner dbid created        status
-------------- -------- ----- ---- -------------- ------------------
master         5.0 MB   sa       1 Jan 01, 1900   no options set
model          2.0 MB   sa       3 Jan 01, 1900   no options set
pubs2          2.0 MB   sa       6 Sep 20, 1995   no options set
sybsystemprocs 16.0 MB  sa       4 Sep 20, 1995   trunc log on chkp
tempdb         2.0 MB   sa       2 Sep 20, 1995   select into/bulkcopy
```

Displays information about all the databases in Adaptive Server.

2. sp_helpdb pubs2

(Not issued from *pubs2*.)

```
name   db_size  owner  dbid  created         status
-----  -------  -----  ----  ------------  ----------------------
pubs2  2.0 MB    sa     4    Mar 05, 1993  abort tran when log full

device_fragments      size    usage           free kbytes
-----------------     ------  ------------  -------------
master                2.0 MB  data and log            576

name    attribute_class attribute     int_value char_value  comments
------- --------------- ------------- --------- ----------- --------
pubs2   buffer manager  cache binding         1 pubs2_cache     NULL
```

Displays information about the *pubs2* database.

3. sp_helpdb pubs2

(Issued from *pubs2*.)

```
name   db_size  owner  dbid  created       status
-----  -------  -----  ----  ------------  ----------------------
pubs2  2.0 MB   sa     4     Mar 05, 1993  abort tran when log full

device_fragments   size    usage         free kbytes
-----------------  ------  ------------  -------------
master             2.0 MB  data and log            576

device                         segment
------------------------------ ------------------------------
master                         default
master                         logsegment
master                         system

name    attribute_class attribute     int_value char_value  comments
------- --------------- ------------- --------- ----------- --------
pubs2   buffer manager  cache binding         1 pubs2_cache     NULL
```

Displays information about the *pubs2* database, and includes segment information.

Comments

- **sp_helpdb** reports on the specified database when *dbname* is given or on all the databases listed in *master.dbo.sysdatabases* when no parameter is supplied.
- Executing **sp_helpdb** *dbname* from *dbname* includes free space and segment information in the report.
- **sp_helpdb** displays information about a database's attributes, giving the attribute's class, name, integer value, character value, and comments, if any attributes are defined. Example 3 shows cache binding attributes for the *pubs2* database.
- **sp_helpdb** reports if a database is offline.
- A database created with the **for load** option has a status of "don't recover" in the output from **sp_helpdb**.
- When Component Integration Services is enabled, **sp_helpdb** lists the default storage location for the specified database or all databases. If there is no default storage location, the display indicates "NULL".

Messages

- `The specified database does not exist.`

  The specified database does not exist. Run **sp_helpdb** without the *dbname* parameter to see a list of all the databases.

Permissions

Any user can execute **sp_helpdb**.

Tables Used

master.dbo.spt_values, master.dbo.sysattributes, sysdatabases, sysdevices, syslogins, sysmessages, syssegments, sysusages

See Also

Commands	**alter database**, **create database**
System procedures	**sp_configure**, **sp_dboption**, **sp_renamedb**

sp_helpdevice

Function

Reports information about a particular device or about all Adaptive Server database devices and dump devices.

Syntax

sp_helpdevice [*devname*]

Parameters

devname – is the name of the device about which to report information. If you omit this parameter, **sp_helpdevice** reports on all devices.

Examples

1. sp_helpdevice

```
device_name  physical_name   description
-----------  --------------  ----------------------------------------
diskdump     null            disk, dump device
master       d_master        special, default disk, physical disk, 10 MB

status       cntrltype          device_number          low       high
------       ----------         -------------          ----      -----
16           2                  0                      0         20000
3            0                  0                      0         5120
```

Displays information about all the devices on Adaptive Server.

2. sp_helpdevice diskdump

Reports information about the dump device named *diskdump*.

Comments

- **sp_helpdevice** displays information on the specified device, when *devname* is given, or on all devices in *master.dbo.sysdevices*, when no argument is given.
- The *sysdevices* table contains dump devices and database devices.

  Database devices can be designated as default devices, which means that they can be used for database storage. This can occur when a user issues **create database** or **alter database** and does not specify a database device name or gives the keyword **default**. To make a database device a default database device, execute the system procedure **sp_diskdefault**.

- Add database devices to the system with **disk init**. Add dump devices with **sp_addumpdevice**.
- The number in the "status" column corresponds to the status description in the "description" column.

  The "cntrltype" column specifies the controller number of the device. The "cntrltype" is 2 for disk or file dump devices and 3–8 for tape dump devices. For database devices, the "cntrltype" is usually 0 (unless your installation has a special type of disk controller).

  The "device_number" column is 0 for dump devices, 0 for the master database device, and between 1 and 255 for other database devices. **sp_helpdevice** may report erroneous negative numbers for device numbers greater than 126.

  The "low" and "high" columns represent virtual page numbers, each of which is unique among all the devices in Adaptive Server.

Messages

- `No such i/o device exists.`

  The device name supplied for the *devname* parameter does not exist on Adaptive Server. Run **sp_helpdevice** without the *devname* parameter to see a list of all devices.

Permissions

Any user can execute **sp_helpdevice**.

Tables Used

master.dbo.spt_values, *sysdevices*, *sysmessages*

See Also

Commands	**disk init**, **dump database**, **dump transaction**, **load database**, **load transaction**
System procedures	**sp_addumpdevice**, **sp_configure**, **sp_diskdefault**, **sp_dropdevice**, **sp_helpdb**, **sp_logdevice**, **sp_who**

sp_helpextendedproc

Function

Displays extended stored procedures (ESPs) in the current database, along with their associated DLL files.

Syntax

sp_helpextendedproc [*esp_name*]

Parameters

esp_name – is the name of the extended stored procedure. It must be a procedure in the current database.

Examples

1. **use sybsystemprocs**
   sp_helpextendedproc xp_cmdshell

   ```
   ESP Name    DLL Name
   ----------- ----------
   xp_cmdshell sybsyesp
   ```

   Lists the **xp_cmdshell** ESP and the name of the DLL file in which its function is stored.

2. **sp_helpextendedproc**

   ```
   ESP Name    DLL Name
   ----------- ----------
   xp_freedl    sybsyesp
   xp_cmdshell  sybsyesp
   ```

   Lists all the ESPs in the current database, along with the names of the DLL files in which their functions are stored.

Comments

- If the *esp_name* is omitted, **sp_helpextendedproc** lists all the extended stored procedures in the database.
- The *esp_name* is case sensitive. It must match the *esp_name* used to create the ESP.

Messages

- *esp_name* `is not a valid name.`

  The *esp_name* must have the syntax of an identifier.

- `Object must be in the current database.`

  The object is not in the current database.

- `The extended stored procedure` *`procedure_name`* `is not in sysobjects for this user.`

  There is no record of the specified ESP in the system tables in the current database owned by the current user.

Permissions

Only the System Administrator can execute **sp_helpextendedproc** to see all the ESPs in the database. All users can execute **sp_helpextendedproc** to see ESPs owned by themselves or by the Database Owner.

Tables Used

master.dbo.syscomments, *sysobjects*

See Also

Commands	**create procedure, drop procedure**
System procedures	**sp_addextendedproc**, **sp_dropextendedproc**

sp_helpexternlogin

(Component Integration Services only)

Function

Reports information about external login names.

Syntax

```
sp_helpexternlogin [remote_server [, login_name]]
```

Parameters

remote_server – is the name of the remote server that has been added to the local server with **sp_addserver**.

login_name – is a login account on the local server.

Examples

1. `sp_helpexternlogin`

   Displays all remote servers, local login names, and external logins.

2. `sp_helpexternlogin SSB`

   Displays local login names and external logins for the server named SSB.

3. `sp_helpexternlogin NULL, milo`

   Displays remote servers, local login names and external logins for the user named “milo”.

4. `sp_helpexternlogin SSB, trixi`

   Displays external logins for remote server SSB where the local user name is “trixi”.

Comments

- **sp_helpexternlogin** displays all remote servers, the user’s local login name, and the user’s external login name.
- Add remote servers with **sp_addserver**. Add local logins with **sp_addlogin**.

Messages

- `There is not a server named` *`remote_server`*`.`

  The *remote_server* you specified is not defined to the local server. Check the spelling of the remote server name or use **sp_helpserver** to list the known remote servers.

- *`remote_server`* `is the local server - external login not applicable.`

  The name of the local server was specified for the *remote_server* parameter.

- *`login_name`* `is not a local user -- request denied.`

  Check the spelling of *login_name*. Use **sp_helpuser** to list user names.

Permissions

Any user can use **sp_helpexternlogin**.

Tables Used

master.dbo.syslogins, *master.dbo.sysattributes*, *master.dbo.sysservers*

See Also

System procedures	**sp_addexternlogin**, **sp_addlogin**,**sp_addserver**, **sp_helpserver**

sp_helpgroup

Function

Reports information about a particular group or about all groups in the current database.

Syntax

sp_helpgroup [*grpname*]

Parameters

grpname – is the name of a group in the database created with **sp_addgroup**.

Examples

1. sp_helpgroup

```
Group_name      Group_id
--------------- --------
hackers         16384
public          0
```

Displays information about all groups in the current database.

2. sp_helpgroup hackers

```
Group_name  Group_id    Users_in_group  Userid
----------- ---------   --------------  ------
hackers     16384       ann             4
hackers     16384       judy            3
```

Displays information about the group “hackers”.

Comments

- To get a report on the default group, “public,” enclose the name “public” in single or double quotes (“public” is a reserved word).
- If there are no members in the specified group, **sp_helpgroup** displays the header, but lists no users, as follows:

```
Group_name  Group_id    Users_in_group  Userid
----------- ---------   --------------  ------
```

Messages

- `No group with the specified name exists.`

  The specified group does not exist in the current database. Execute the procedure without the *grpname* parameter to see a list of all the groups in the database.

Permissions

Any user can execute **sp_helpgroup**.

Tables Used

syssrvroles, *sysusers*

See Also

Commands	**grant**, **revoke**
System procedures	**sp_addgroup**, **sp_changegroup**, **sp_dropgroup**, **sp_helprotect**, **sp_helpuser**

sp_helpindex

Function

Reports information about the indexes created on a table.

Syntax

sp_helpindex *objname*

Parameters

objname – is the name of a table in the current database.

Examples

1. sp_helpindex sysobjects

```
index_name  index_description                    index_keys
      index_max_rows_per_page
----------- ------------------------------------ ----------
      ----------------------
titleidind  clustered, unique located on default   title_id
      0
titleind    nonclustered located on default        title
      0

name     attribute_class attribute     int_value char_value    comments
-------- --------------- ------------- --------- ------------  --------
titleind buffer manager  cache binding         1 titleind_cache    NULL
```

Displays the types of indexes on the *sysobjects* table.

Comments

- **sp_helpindex** lists any indexes on a table, including indexes created by defining unique or primary key constraints defined by a **create table** or **alter table** statement.
- **sp_helpindex** displays any attributes (for example, cache bindings) assigned to the indexes on a table.
- **sp_helpindex** displays the **max_rows_per_page** setting of the indexes.

Messages

- `Object does not exist in this database.`

  The name you specified for the *objname* parameter does not exist in the current database.
- `Object does not have any indexes.`

The table you named has no indexes.

- `Object must be in the current database.`

  The name you specified for the *objname* parameter includes a database reference. Name references must be local to the current database.

Permissions

Any user can execute **sp_helpindex**.

Tables Used

master.dbo.spt_values, *sysattributes*, *sysindexes*, *sysobjects*, *syssegments*

See Alsofs

Commands	**create index**, **drop index**, **update statistics**
System procedures	**sp_help**, **sp_helpkey**

sp_helpjoins

Function

Lists the columns in two tables or views that are likely join candidates.

Syntax

sp_helpjoins *lefttab*, *righttab*

Parameters

lefttab – is the first table or view.

righttab – is the second table or view. The order of the parameters does not matter.

Examples

1. sp_helpjoins sales, salesdetail

```
a1       a2       b1       b2       c1       c2
   d1       d2       e1       e2       f1       f2
      g1       g2       h1       h2
-------- -------- -------- -------- -------- --------
   -------- -------- -------- -------- -------- --------
      -------- -------- -------- --------
stor_id  stor_id  ord_num  ord_num  NULL     NULL
   NULL     NULL     NULL     NULL     NULL     NULL
      NULL     NULL     NULL     NULL
```

Displays a list of columns that are likely join candidates in the *sales* and *salesdetail* tables.

2. sp_helpjoins sysobjects, syscolumns

```
a1   a2   b1   b2   c1   c2   d1   d2   e1   e2
        f1   f2   g1   g2   h1   h2
---- ---- ---- ---- ---- ---- ---- ---- ---- ----
        ---- ---- ---- ---- ---- ----
id   id   NULL NULL NULL NULL NULL NULL NULL NULL
        NULL NULL NULL NULL NULL NULL
```

Displays a list of columns that are likely join candidates in the *sysobjects* and *syscolumns* system tables.

Comments

- The column pairs that **sp_helpjoins** displays come from either of two sources. First, **sp_helpjoins** checks the *syskeys* table in the current database to see if any foreign keys have been defined with

sp_foreignkey on the two tables and then checks to see if any common keys have been defined with **sp_commonkey** on the two tables. If **sp_helpjoins** does not find any foreign keys or common keys there, it looks for any keys that can reasonably be joined: it checks for keys with the same user-defined datatypes; if that fails, it checks for columns with the same name and datatype.

- **sp_helpjoins** does not create any joins.

Messages

- `First table doesn't exist.`

  The table specified as the *lefttab* parameter is not a table or view in the current database.

- `Object must be in the current database.`

  Both *lefttab* and *righttab* must be local to your current database.

- `Second table doesn't exist.`

  The table specified as the *righttab* parameter is not a table or view in the current database.

Permissions

Any user can issue **sp_helpjoins**.

Tables Used

syscolumns, syskeys, sysobjects

See Also

System procedures	**sp_commonkey**, **sp_foreignkey**, **sp_help**, **sp_helpkey**, **sp_primarykey**

sp_helpkey

Function

Reports information about a primary, foreign, or common key of a particular table or view, or about all keys in the current database.

Syntax

sp_helpkey [*tabname*]

Parameters

tabname – is the name of a table or view in the current database. If you do not specify a name, the procedure reports on all keys defined in the current database.

Examples

1. sp_helpkey

```
keytype object      related_object object_keys         related_keys
------- -------     -------------- ---------------     --------------
primary authors     -- none --     au_id,*,*,*,*,*,*,*  *,*,*,*,*,*,*,*
foreign titleauthor authors        au_id,*,*,*,*,*,*,* au_id,*,*,*,*,*,
                                                          *,*
```

Displays information about the keys defined in the current database. The "object_keys" and "related_keys" columns refer to the names of the columns that make up the key.

Comments

- **sp_helpkey** lists information about all primary, foreign, and common key definitions that reference the table *tabname* or, if *tabname* is omitted, about all the keys in the database. Define these keys with the **sp_primarykey**, **sp_foreignkey**, and **sp_commonkey** system procedures.
- **sp_helpkey** does not provide information about the **unique** or **primary key** integrity constraints defined by a **create table** statement. Use **sp_helpconstraint** to determine what constraints are defined for a table.
- Create keys to make explicit a logical relationship that is implicit in your database design so that applications can use the information.

- If you specify an object name, **sp_helpkey** follows the Adaptive Server rules for finding objects:
  - If you do not specify an owner name, and you own an object with the specified name, **sp_helpkey** reports on that object.
  - If you do not specify an owner name, and you do not own an object of that name, but the Database Owner does, **sp_helpkey** reports on the Database Owner's object.
  - If neither you nor the Database Owner owns an object with the specified name, **sp_helpkey** reports an error condition, even if an object with that name exists in the database for a different owner.
  - If both you and the Database Owner own objects with the specified name, and you want to access the Database Owner's object, specify the name in the form *dbo.objectname*.
- Qualify objects that are owned by database users other than yourself and the Database Owner with the owner's name, as in "mary.myproc".

Messages

- `No defined keys for this object.`

  No primary, foreign, or common keys are defined for the specified table or view.
- `The name supplied for the` *`tabname`* `parameter is not a table or view in the current database.`

  The table or view you specified does not exist in the current database.
- `Table or view name must be in current database.`

  The name you supplied for the *tabname* parameter included a database reference. Name references must be local to the current database.

Permissions

Any user can execute **sp_helpkey.**

Tables Used

master.dbo.spt_values, syskeys, sysobjects

See Also

Commands	**create trigger**
System procedures	**sp_commonkey**, **sp_foreignkey**, **sp_help**, **sp_primarykey**

sp_helplanguage

Function

Reports information about a particular alternate language or about all languages.

Syntax

sp_helplanguage [*language*]

Parameters

language – is the name of the alternate language you want information about.

Examples

1. sp_helplanguage french

```
langid dateformat datefirst upgrade     name
       alias
       months
       shortmonths
       days
------ ---------- --------- ----------- ----------------------
      ----------------------------
      ------------------------------------------------------------
      ------------------------------------------------------------
      ------------------------------------------------------------
1     dmy        1         0         french
      french
      janvier,février,mars,avril,mai,juin,juillet,août,septembre,
          octobre,novembre,décembre
      jan,fév,mar,avr,mai,jui,juil,aoû,sep,oct,nov,déc
      lundi,mardi,mercredi,jeudi,vendredi,samedi,dimanche
```

Displays information about the alternate language, “french”.

2. sp_helplanguage

Displays information about all installed alternate languages.

Comments

- **sp_helplanguage** reports on a specified language, when the language is given, or on all languages in *master.dbo.syslanguages*, when no language is supplied.

Messages

- *language* is not an official language name from syslanguages.

  Adaptive Server did not find the *language* you specified. Use **sp_helplanguage** with no parameters to see the list of language names available in Adaptive Server.

- No alternate languages are available.

  No alternate languages are installed in Adaptive Server.

- us_english is always available, even though it is not in master.dbo.syslanguages.

  This message appears at the end of each report from **sp_helplanguage.**

Permissions

Any user can execute **sp_helplanguage.**

Tables Used

master.dbo.syslanguages

See Also

System procedures	**sp_addlanguage**, **sp_droplanguage**, **sp_setlangalias**

sp_helplog

Function

Reports the name of the device that contains the first page of the transaction log.

Syntax

```
sp_helplog
```

Parameters

None.

Examples

```
1. sp_helplog
```

```
In database 'master', the log starts on device
'master'.
```

Comments

- **sp_helplog** displays the name of the device that contains the first page of the transaction log in the current database.

Messages

- `In database '`*`database_name`*`', the log starts on device '`*`device_name`*`'.`

  The named device contains the first page of the database's transaction log.

Permissions

Any user can execute **sp_helplog**.

Tables Used

master.dbo.sysdevices, master.dbo.sysusages, sysindexes, sysobjects

See Also

Commands	**alter database**, **create database**
System procedures	**sp_helpdevice**, **sp_logdevice**

sp_helpobjectdef

(Component Integration Services only)

Function

Reports owners, objects, and type information for remote object definitions.

Syntax

```
sp_helpobjectdef [object_name]
```

Parameters

object_name – is the name of the object as it is defined in the *sysattributes* table. The *object_name* can be in any of the following forms:

- *dbname.owner.object*
- *dbname..object*
- *owner.object*
- *object*

dbname and *owner* are optional. *object* is required. If *owner* is not supplied, the *owner* defaults to the current user name. If *dbname* is supplied, it must be the current database, and *owner* must be supplied or marked with the placeholder *dbname..object*. Enclose a multipart *object_name* in quotes.

Examples

1. **`sp_helpobjectdef`**

   Displays all remote object definitions in the current database.

2. **`sp_helpobjectdef "dbo.tb1"`**

   Displays remote object definitions for the *tb1* table owned by the Database Owner.

Comments

- If no *object_name* is supplied, **sp_helpobjectdef** displays all remote object definitions.
- A server name is not permitted in the *object_name* parameter.

Messages

- *object_name* is not a valid name.

  The syntax for *object_name* is not valid. One or more of the elements *dbname.owner.object* does not conform to the rules for identifiers.

- A server name is not permitted in the local *object_name*.

  The *object_name* contains a server name.

- Database name *dbname* is not your current database.

  dbname is optional in *object_name*. If supplied, it must be the current database.

- *User owner* is not a valid user in the *dbname* database.

  The value specified for *owner* in the *object_name* is not a defined user in the current database.

- *object* does not exist in this database

  The format of the name supplied for the *object* component in *object_name* is not valid.

Tables Used

sysattributes, sysobjects, sysservers, spt_values

See Also

Commands	**create table, create existing table, drop table**
System procedures	**sp_addlogin**, **sp_addserver**, **sp_addobjectdef**, **sp_defaultloc**, **sp_dropobjectdef**, **sp_help**, **sp_helpserver**

sp_helpremotelogin

Function

Reports information about a particular remote server's logins or about all remote server logins.

Syntax

```
sp_helpremotelogin [remoteserver [, remotename]]
```

Parameters

remoteserver – is the name of the server about which to report remote login information.

remotename – is the name of a particular remote user on the remote server.

Examples

1. **`sp_helpremotelogin GATEWAY`**

   Displays information about all the remote users of the remote server GATEWAY.

2. **`sp_helpremotelogin`**

   Displays information about all the remote users of all the remote servers known to the local server.

Comments

- **sp_helpremotelogin** reports on the remote logins for the specified server, when *remoteserver* is given, or on all servers, when no parameter is supplied.

Messages

- `There are no remote logins.`
- `There are no remote logins defined.`

  There are no remote logins for any remote server in *master.dbo.sysremotelogins*.
- `There are no remote logins for 'remotename'.`

  The remote server has no entries in the *master.dbo.sysremotelogins* table.

- `There are no remote logins for 'remotename' on remote server 'remoteserver'.`

  There is no remote login for the user *remoteuser* on the remote server *remoteserver*.

- `There are no remote logins for the remote server 'remoteserver'.`

  The specified server is not listed in *master.dbo.sysservers*. Run the procedure without the *remoteserver* parameter to see remote login information for all servers. To get a list of all servers, run **sp_helpserver**.

- `There are no remote servers defined.`

  The *master.dbo.sysservers* table has no entries for remote servers.

Permissions

Any user can execute **sp_helpremotelogin**.

Tables Used

master.dbo.spt_values, *master.dbo.sysmessages*, *master.dbo.sysremotelogins*, *master.dbo.sysservers*, *sysobjects*

See Also

System procedures	**sp_addremotelogin**, **sp_dropremotelogin**, **sp_helpserver**

sp_help_resource_limit

Function

Reports on resource limits.

Syntax

```
sp_help_resource_limit [name [, appname [, limittime
   [, limitday [, scope [, action]]]]]]
```

Parameters

name – is the Adaptive Server login to which the limits apply. For information about limits that govern a particular login, specify the login *name*. For information about limits without regard to login, specify **null**.

➤ ***Note***

If you are are not a System Administrator, specify your own login, or a login of NULL, to display information about the resource limits that apply to you.

appname – is the name of the application to which the limit applies. For information about limits that govern a particular application, specify the application name that the client program passes to the Adaptive Server in the login packet. For information about limits without regard to application, specify **null**.

limittime – is the time during which the limit is enforced. For information about limits in effect at a given time, specify the time, with a value between "00:00" and "23:59", using the following form:

"*HH*:*MM*"

For information about limits without regard to time, specify **null**.

limitday – is any day on which the limit is enforced. For information about resource limits in effect on a given day of the week, specify the full weekday name for the default server language, as stored in the *syslanguages* system table of the *master* database. For information about limits without regard to the days on which they are enforced, specify **null**.

scope – is the scope of the limit. Specify one of the following:

Scope Code	For Help on All Limits That Govern
1	Queries
2	Query batches (one or more SQL statements sent by the client to the server)
4	Transactions
6	Both query batches and transactions
NULL	The specified *name*, *appname*, *limittime*, *limitday*, and *action*, without regard to their *scope*

action – is the action to take when the limit is exceeded. Specify one of the following:

Action Code	For Help on All Limits That
1	Issue a warning
2	Abort the query batch
3	Abort the transaction
4	Kill the session
NULL	Govern the specified *name*, *appname*, *limittime*, *limitday*, and *scope*, without regard to the *action* they take

Examples

1. **`sp_help_resource_limit`**

   Lists all resource limits stored in the *sysresourcelimits* system table.

2. **`sp_help_resource_limit joe_user`**

   Lists all limits for the user “joe_user”.

3. **`sp_help_resource_limit NULL, my_app`**

   Lists all limits for the application *my_app*.

4. **`sp_help_resource_limit NULL, NULL, "09:00"`**

   Lists all limits enforced at 9:00 a.m.

5. **`sp_help_resource_limit @limittype = "09:00"`**

   An alternative way of listing the limits enforced at 9:00 a.m.

6. **sp_help_resource_limit NULL, NULL, NULL, Monday**

   Lists all limits enforced on Mondays.

7. **sp_help_resource_limit joe_user, NULL, "09:00", Monday**

   Lists any limit in effect for "joe_user" on Mondays at 9:00 a.m.

Comments

- **sp_help_resource_limit** reports on all resource limits, limits for a given login or application, limits in effect at a given time or day of the week, or limits with a given scope or action.
- For more information on resource limits, see Chapter 12, "Limiting Access to Server Resources," in the *System Administration Guide.*

Messages

- `Illegal action` *`action.`*

  Specify an action of **1**, **2**, **3**, **4**, or **null**.

- `Illegal limit time argument` *`limittime.`*

  Specify a valid time between "00:00" and "23:59".

- `No login with the specified name exists.`

  The login you specified does not exist within the *syslogins* system table of the *master* database.

- `Unknown limitday` *`limitday.`*

  You must specify the full name for the day of the week, as stored in the *syslanguages* system table of the *master* database.

- `Users other than the System Administrator can only view limits applicable to themselves.`

  You must be a System Administrator to view resource limits that apply to other users. For information about limits that apply to yourself, rerun **sp_help_resource_limit**, and specify your own login or a login of **null**.

Permissions

Any user can execute **sp_help_resource_limit** to list his or her own resource limits. Only System Administrators can execute **sp_help_resource_limit** to list limits that apply to other users.

Tables Used

master..sysresourcelimits, master..systimeranges, master..spt_limit_types

See Also

System procedures	**sp_add_resource_limit**, **sp_drop_resource_limit**, **sp_modify_resource_limit**

sp_helprotect

Function

Reports on permissions for database objects, users, groups, or roles.

Syntax

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

Parameters

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

username – is a user's name in the current database.

grant – displays the privileges granted to *name* **with grant option.**

none – ignores roles granted to the user when determining permissions granted.

granted – includes information on all roles granted to the user when determining permissions granted.

enabled – includes information on all roles activated by the user when determining permissions granted.

role_name – displays permission information for the specified role only, regardless of whether this role has been granted to the user.

Examples

1.
```
grant select on titles to judy
grant update on titles to judy
revoke update on titles(price) from judy
grant select on publishers to judy
with grant option
```

After this series of **grant** and **revoke** statements, executing **sp_helprotect titles** results in this display:

```
grantor grantee type    action   object      column       grantable
------- ------  -----   ------   ------      ------       ---------
dbo     judy    Grant   Select   titles      All          FALSE
dbo     judy    Grant   Update   titles      advance      FALSE
dbo     judy    Grant   Update   titles      notes        FALSE
dbo     judy    Grant   Update   titles      pub_id       FALSE
dbo     judy    Grant   Update   titles      pubdate      FALSE
dbo     judy    Grant   Update   titles      title        FALSE
dbo     judy    Grant   Update   titles      title_id     FALSE
dbo     judy    Grant   Update   titles      total_sales  FALSE
dbo     judy    Grant   Update   titles      type         FALSE
dbo     judy    Grant   Select   publishers  all          TRUE
```

2. **grant select, update on titles(price, advance)**
   to mary
   with grant option
   sp_helprotect titles

   After this **grant** statement, **sp_helprotect** displays the following:

```
grantor   grantee    type    action   object   column   grantable
-------   -------    ------  -------  ------   ------   ---------
dbo       mary       Grant   Select   titles   advance  TRUE
dbo       mary       Grant   Select   titles   price    TRUE
dbo       mary       Grant   Update   titles   advance  TRUE
dbo       mary       Grant   Update   titles   price    TRUE
```

3. **sp_helprotect judy**

   Displays all the permissions that "judy" has in the database.

4. **sp_helprotect titles, csmith, "grant"**

   Displays any permissions that "csmith" has on the *titles* table, as well as whether "csmith" has **with grant option** which allows "csmith" to grant permissions to other users.

5. **sp_helprotect @role_name = doctor_role**

   Displays information about the permissions that the doctor role has in the database.

Comments

- **sp_helprotect** reports permissions on a database object. If you supply the *username* parameter, only that user's permissions on the database object are reported. If *name* is not an object, **sp_helprotect** checks to see if it is a user, a group, or a role. If it is, **sp_helprotect** lists the permissions for the user, group, or role.
- **sp_helprotect** looks for objects and users in the current database only.

- If you do not specify an optional value such as **granted**, **enabled**, **none**, or *role_name*, Adaptive Server returns information on all roles activated by the current specified user.
- If the specified user is not the current user, Adaptive Server returns information on all roles granted to the specified user.
- Displayed information always includes permissions granted to the group in which the specified user is a member.
- In granting permissions, a System Administrator is treated as the object owner. If a System Administrator grants permission on another user's object, the owner's name appears as the grantor in **sp_helprotect** output.

Messages

- `@rolename = 'enabled' option is allowed only for the current user.`

  You cannot use **enabled** to get information on another user's roles.
- `Illegal rolename` *`role_name`* `specified.`

  Check the spelling of the name of the role.
- `Illegal string found where the keyword grant is expected.`

  Check the syntax and issue **sp_helprotect** with the parameters in the correct order.
- `Object must be in current database.`

  The name supplied for the *name* parameter included a reference to a database. The name must be local to the database.
- `No user with the specified name exists in the current database.`

  The name supplied for *username* is not a user or group in the current database.
- `No such object or user exists in the database.`

  The name supplied for the *name* parameter is not an object, user, or group in the current database.

Permissions

Any user can execute **sp_helprotect** to view his or her own permissions. Only a System Security Officer can use **sp_helprotect** to view permissions granted to other users.

Tables Used

master.dbo.spt_values, *syscolumns*, *sysobjects*, *sysprotects*, *sysusers*

See Also

Commands	**grant**, **revoke**, **create**, **drop**, **set**
System procedures	**sp_help**, **sp_activeroles**, **sp_configure**, **sp_displaylogin**, **sp_displayroles**, **sp_modifylogin**

sp_helpsegment

Function

Reports information about a particular segment or about all segments in the current database.

Syntax

sp_helpsegment [*segname*]

Parameters

segname – is the name of the segment about which you want information. If you omit this parameter, information about all segments in the current database appears.

Examples

1. sp_helpsegment

```
segment name                          status
------- ------------------------------ ------
      0 system                              0
      1 default                             1
      2 logsegment                          0
```

Reports information about all segments in the current database.

2. sp_helpsegment order_seg

```
segment name                          status
------- ------------------------------ ------
      3 order_seg                         0

device                 size           free_pages
---------------------- -------------- -----------
tpcd_data1             25.0MB               8176
tpcd_data2             25.0MB               8512
tpcd_data3             25.0MB               8392
tpcd_data4             25.0MB               8272
```

```
tpcd_data5              25.0MB                8448
tpcd_data6              25.0MB                8512

table_name              index_name             indid
----------------------- ---------------------- ------
orders                  orders                      0

total_size     total_pages free_pages  used_pages
-------------- ----------- ----------- -----------
150.0MB              76800       50312       26488
```

Reports information about the segment named *order_seg*, including which database tables and indexes use that segment and the total number of pages, free pages and used pages on the segment.

3. sp_helpsegment "default"

Reports information about the *default* segment. The keyword **default** must be enclosed in quotes.

4. sp_helpsegment logsegment

```
segment name                          status
------- ------------------------------ ------
      2 logsegment                          0

device                  size              free_pages
----------------------- ----------------- -----------
tpcd_log1               20.0MB                  10200

table_name              index_name               indid
----------------------- ------------------------ ------
syslogs                 syslogs                       0

total_size        total_pages free_pages  used_pages
----------------- ----------- ----------- -----------
20.0MB                  10240       10200          40
```

Reports information about the segment on which the transaction log is stored.

Comments

- **sp_helpsegment** displays information about the specified segment, when *segname* is given, or about all segments in the current database, when no argument is given.
- When you first create a database, Adaptive Server automatically creates the *system*, *default*, and *logsegment* segments. Use **sp_addsegment** to add segments to the current database.

- The *system*, *default*, and *logsegment* segments are numbered 0, 1, and 2, respectively.
- The "status" column indicates which segment is the default pool of space. Use **sp_placeobject** or the **on** *segment_name* clause of the **create table** or **create index** command to place objects on specific segments.
- The "indid" column is 0 if the table does not have a clustered index and is 1 if the table has a clustered index.

Messages

- `There is no such segment as` *`segname`*`.`

  The segment name supplied for the *segname* parameter does not exist in the *syssegments* table. Run **sp_helpsegment** without the *segname* parameter to see a list of all segments for the current database.

Permissions

Any user can execute **sp_helpsegment**.

Tables Used

master.dbo.sysdevices, *master.dbo.sysusages*, *sysindexes*, *sysobjects*, *syssegments*

See Also

Commands	**create index**, **create table**
System procedures	**sp_addsegment**, **sp_dropsegment**, **sp_extendsegment**, **sp_helpdb**, **sp_helpdevice**, **sp_placeobject**

sp_helpserver

Function

Reports information about a particular remote server or about all remote servers.

Syntax

sp_helpserver [*server*]

Parameters

server – is the name of the remote server about which you want information.

Examples

1. sp_helpserver GATEWAY

Displays information about the remote server GATEWAY.

2. sp_helpserver SYB_BACKUP

```
name       network_name  status                                id
---------- ------------- ----------------------------------- ---
SYB_BACKUP SYB_BACKUP    timeouts, no net password encryption  1
```

Displays information about the local Backup Server.

3. sp_helpserver

Displays information about all the remote servers known to the local server.

Comments

- **sp_helpserver** reports information about all servers in *master.dbo.sysservers* or about a particular remote server, when *server* is specified.
- When Component Integration Services is installed, **sp_helpserver** lists the server class for each server.

Messages

- `There are no remote servers defined.`

  This Adaptive Server has no remote servers defined.

- `There is not a server named` *`server`*`.`

  The specified server is not listed in *master.dbo.sysservers*. Run **sp_helpserver** without the *server* parameter to see a list of all the servers.

Permissions

Any user can execute **sp_helpserver.**

Tables Used

master.dbo.spt_values, *master.dbo.sysservers*, *sysobjects*

See Also

System procedures	**sp_addserver**, **sp_dropserver**, **sp_helpremotelogin**, **sp_serveroption**

sp_helpsort

Function

Displays Adaptive Server's default sort order and character set.

Syntax

sp_helpsort

Parameters

None.

Examples

1. sp_helpsort

For Class 1 (single-byte) character sets, **sp_helpsort** displays the name of the server's default sort order, its character set, and a table of its primary sort values. On a 7-bit terminal, it appears as follows:

```
Sort Order Description
------------------------------------------------------------------
 Character Set = 1, iso_1
     ISO 8859-1 (Latin-1) - Western European 8-bit character set.
 Sort Order = 50, bin_iso_1
     Binary sort order for the ISO 8859/1 character set (iso_1).
Characters, in Order

------------------------------------------------------------------
   ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
   @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _
   ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
   ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
   @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _
   ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
```

On an 8-bit terminal, it appears as follows:

```
Sort Order Description
------------------------------------------------------------------
 Character Set = 1, iso_1
     ISO 8859-1 (Latin-1) - Western European 8-bit character set.
 Sort Order = 50, bin_iso_1
     Binary sort order for the ISO 8859/1 character set (iso_1).
Characters, in Order

------------------------------------------------------------------
  ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
  @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _
  ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
  ¡ ¢ £ ¤ ¥ | § ¨ © ª « ¬ –
 ® ¯ ° 2 3 ´ µ ¶ · ¸ 1 º » 1/4 1/2 3/4 ¿ À
  Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï D Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Y P ß à
  á â ã ä å æ ç è é ê ë ì í î ï  ñ ò ó ô õ ö ÷ ø ù ú û ü y p ÿ
```

For a Class 2 (multibyte) character set, the characters are not listed, but a description of the character set is included. For example:

```
Sort Order Description
------------------------------------------------------------------
Character Set = 140, euc_jis
    Japanese. Extended Unix Code mapping for JIS-X0201
    (hankaku katakana) and JIS-X0208 (double byte) roman,
     kana, and kanji.
    Class 2 character set
Sort Order = 50, bin_eucjis
    Binary sort order for Japanese using the EUC JIS
      character set as a basis.
```

Comments

- Binary sort order is the default.

Messages

- `Unknown character set: character_set`

  sp_helpsort does not recognize *character_set*, but displays the characters in order.

Permissions

Any user can execute **sp_helpsort.**

Tables Used

master.dbo.syscharsets, master.dbo.syscurconfigs, sysobjects

sp_helptext

Function

Displays the **source text** of a **compiled object**.

Syntax

sp_helptext *objname*

Parameters

objname – is the name of the compiled object for which the source text is to be displayed. The compiled object must be in the current database.

Examples

1. sp_helptext pub_idrule

```
# Lines of Text
---------------
1

text
------------------------------------
create rule pub_idrule
as @pub_id in ("1389", "0736", "0877",
      "1622", "1756")
    or @pub_id like "99[0-9][0-9]"
```

Displays the source text of *pub_idrule*. Since this rule is in the *pubs2* database, execute this command from *pubs2*.

2. sp_helptext sp_helptext

Displays the source text of **sp_helptext**. Since system procedures are stored in *sybsystemprocs*, execute this command from *sybsystemprocs*.

Comments

- **sp_helptext** prints out the number of rows in *syscomments* (255 characters long each) that are occupied by the compiled object, followed by the source text of the compiled object.
- **sp_helptext** looks for the source text in the *syscomments* table in the current database.
- Encrypt the source text with **sp_hidetext**.

Messages

- `Object must be in the current database.`

  The *objname* parameter included a database name reference. The *objname* must be in the current database.

- `There is no text for object` *`objname`*`.`

  objname is an object in the current database that does not have text in *syscomments* (a table or an index, for example).

- `Procedure text source for procedure object` *`procname`* `(id=`*`proc_id`*`) is hidden.`

  The source text for the specified compiled object was hidden. If you need to see the source text, request a backup version from your System Administrator. If no backup version is available, contact Sybase Technical Support.

Permissions

Any user can execute **sp_helptext**.

A System Security Officer can restrict **select** permission on the *text* column of the *syscomments* table to the object owner and the System Administrator by resetting the **allow select on syscomments.text column** parameter with the system procedure **sp_configure**. This restriction prevents the display of the source text when other users execute **sp_helptext**. This restriction is required in order to run Adaptive Server in the **evaluated configuration**.

For more information about the evaluated configuration, see Chapter 1, "Overview of Security Features," in the *Security Administration Guide*.

Tables Used

syscomments, sysobjects

See Also

Commands	**create default**, **create procedure**, **create rule**, **create trigger**, **create view**
System procedures	**sp_checksource**, **sp_help**, **sp_hidetext**

sp_helpthreshold

Function

Reports the segment, free-space value, status, and stored procedure associated with all thresholds in the current database or all thresholds for a particular segment.

Syntax

```
sp_helpthreshold [segname]
```

Parameters

segname – is the name of a segment in the current database.

Examples

1. `sp_helpthreshold logsegment`

   Shows all thresholds on the log segment.

2. `sp_helpthreshold`

   Shows all thresholds on all segments in the current database.

3. `sp_helpthreshold "default"`

   Shows all thresholds on the default segment. Note the use of quotes around the reserved word “default”.

Comments

- **sp_helpthreshold** displays threshold information for all segments in the current database. If you provide the name of a segment, **sp_helpthreshold** lists all thresholds in that segment.
- The *status* column is 1 for the last-chance threshold and 0 for all other thresholds. Databases that do not store their transaction logs on a separate segment have no last-chance threshold.

Messages

- `Database 'dbname' has no thresholds--table 'systhresholds' does not exist.`

  The *systhresholds* table is missing. This table is created when the database is created upgraded, and it must not be removed.

- `Segment 'segname' does not exist.`

  Use **sp_helpsegment** to see the names of segments in a database.

Permissions

Any user can execute **sp_helpthreshold**.

Tables Used

sysobjects, *syssegments*, *systhresholds*

See Also

System procedures	**sp_addthreshold**, **sp_dropthreshold**, **sp_helpsegment**, **sp_modifythreshold**, **sp_thresholdaction**

sp_helpuser

Function

Reports information about a particular user, group, or alias, or about all users, in the current database.

Syntax

sp_helpuser [*name_in_db*]

Parameters

name_in_db – is the user's name in the current database.

Examples

1. sp_helpuser

```
Users_name ID_in_db    Group_name  Login_name
---------  --------    ----------  ----------
ann        4           hackers     ann
dbo        1           public      sa
guest      2           public      NULL
judy       3           hackers     judy
```

Displays information about all users in the current database.

2. sp_helpuser dbo

```
Users_name     ID_in_db    Group_name  Login_name
----------     --------    ----------  ----------
dbo            1           public      sa

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

Displays information about the Database Owner (user name "dbo").

Comments

- **sp_helpuser** reports information about all users of the current database. If you specify a *name_in_db*, **sp_helpuser** reports information on the specified user only.

- If the specified user is not listed in the current database's *sysusers* table, **sp_helpuser** checks to see if the user is aliased to another user or is a group name.

Messages

- `The name supplied is a group name.`

  The name specified for the *name_in_db* parameter is a group name.

- `The name supplied is aliased to another user.`

  The name supplied is not a user in the database, but is aliased to a user in the database.

- `The name supplied is not a user, group, or aliased.`

  The name supplied is unknown in the database as a login, user, or group.

- `Users aliased to user.`

  If the user has other users aliased to him or her, the names of the other users are listed. (See example 2.)

- `The supplied name is a role name. Please resubmit the command with the name of a user, group or aliased.`

  sp_helpuser reports on users, aliases, or groups, not on roles.

Permissions

Any user can execute **sp_helpuser.**

Tables Used

master.dbo.syslogins, *sysalternates*, *sysobjects*, *sysusers*

See Also

Commands	**grant**, **revoke**, **use**
System procedures	**sp_adduser**, **sp_dropuser**, **sp_help**, **sp_helpgroup**

sp_hidetext

Function

Hides the **source text** for the specified **compiled object**.

Syntax

```
sp_hidetext [objname [, tabname [, username]]]
```

Parameters

objname – specifies the compiled object for which to hide the source text.

tabname – specifies the name of the table or view for which to hide the source text.

username – specifies the name of the user who owns the compiled object for which to hide the source text.

Examples

1. **`sp_hidetext`**

   Hides the source text of all compiled objects in the current database.

2. **`sp_hidetext @objname = "sp_sort_table", @username = "Mary"`**

   Hides the source text of the custom stored procedure, **sp_sort_table**, that is owned by Mary.

3. **`sp_hidetext "pr_phone_list"`**

   Hides the source text of the stored procedure **pr_phone_list**.

4. **`sp_hidetext @tabname = "my_tab"`**

   Hides the source text of all check constraints, defaults, and triggers defined on the table *my_tab*.

5. **`sp_hidetext "my_vu", "my_tab"`**

   Hides the source text of the view *my_vu* and all check constraints, defaults, and triggers defined on the table *my_tab*.

6. **`sp_hidetext @username = "Tom"`**

   Hides the source text of all compiled objects that are owned by Tom.

Comments

- **sp_hidetext** hides the **source text** for the specified **compiled object**.
- If you do not provide any parameters, **sp_hidetext** hides the source text for all compiled objects in the current database. You must be the Database Owner or System Administrator to use **sp_hidetext** with no parameters.
- For more information about hiding source text, see "Compiled Objects" on page 1-3 in the *Transact-SQL User's Guide.*

Messages

- `Failed to hide source text.`

  sp_hidetext could not hide the source text.

- `No user with the specified name exists in the current database.`

  You entered an invalid user name. Run **sp_hidetext** again and specify the correct user name.

- `Object does not exist in this database.`

  You entered an invalid name for the compiled object or table. Run **sp_hidetext** again and specify the correct name.

- `You must have the following role(s) to execute this command/procedure: sa_role. Please contact a user with the appropriate role for help.`

  You attempted to hide the source text for compiled objects that are not owned by you. You must be the Database Owner or a System Administrator. Obtain the necessary permissions, and then run **sp_hidetext** again.

Permissions

Any user can use **sp_hidetext** to hide the source text of his or her own compiled objects. Only a Database Owner or System Administrator can hide the source text of compiled objects that are owned by another user.

Tables Used

syscolumns, *syscomments*, *sysconstraints*, *sysobjects*, *sysprocedures*

See Also

System procedures	**sp_checksource**

sp_indsuspect

Function

Checks user tables for indexes marked as suspect during recovery following a sort order change.

Syntax

sp_indsuspect [*tab_name*]

Parameters

tab_name – is the name of the user table to be checked.

Examples

1. sp_indsuspect newaccts

Checks the table *newaccts* for indexes marked as suspect.

Comments

- **sp_indsuspect** with no parameter creates a list of all tables in the current database that have indexes that need to be rebuilt as a result of a sort order change. With a *tab_name* parameter, **sp_indsuspect** checks the specified table for indexes marked as suspect during recovery following a sort order change.
- Use **sp_indsuspect** to list all suspect indexes. The table owner or a System Administrator can use **dbcc reindex** to check the integrity of the listed indexes and to rebuild them if necessary.

Messages

- `Suspect indexes in database` *`database_name`*`:`

  The listed indexes are suspect and should be reindexed using **dbcc reindex.**

- `There are no suspect indexes in database` *`database_name`*`.`

  No tables in the current database contain suspect indexes.

- `Table must be in the current database.`

  sp_indsuspect checks only the current database for suspect indexes. You cannot use a fully qualified table name to check tables in another database. To check for suspect indexes in

another database, use the **use** *database_name* command to access the database or qualify **sp_indsuspect** with the databsae name.

- `There is no table named tab_name in the current database.`

  The current database does not contain the table name you specified. Check the table name and rerun **sp_indsuspect**.

- `Suspect indexes on table tab_name:`

  The listed indexes are suspect and should be reindexed using **dbcc reindex**.

- `There are no suspect indexes on table tab_name.`

  The specified table does not contain suspect indexes.

Permissions

Any user can execute **sp_indsuspect**.

Tables Used

sysindexes, *sysobjects*, *sysusers*

See Also

Commands	dbcc

sp_listsuspect_db

Function

Lists all databases that currently have offline pages because of corruption detected on recovery.

Syntax

```
sp_listsuspect_db
```

Parameters

None.

Examples

1. sp_listsuspect_db

Lists the databases that have suspect pages.

Comments

- **sp_listsuspect_db** lists the database name, number of suspect pages, and number of objects containing suspect pages.
- Use **sp_listsuspect_page** to identify the suspect pages.

Messages

- `The database` *`databasename`* `has` *`number_of_pages`* `suspect pages belonging to` *`number_of_objects`* `objects.`

  This informational message is displayed following successful execution of **sp_listsuspect_db**.

Permissions

Any user can execute **sp_listsuspect_db**.

Tables Used

master.dbo.sysattributes

See Also

System procedures	**sp_forceonline_db, sp_forceonline_page**, **sp_listsuspect_page**, **sp_setsuspect_granularity**, **sp_setsuspect_threshold**

sp_listsuspect_page

Function

Lists all pages in a database that are currently offline because of corruption detected on recovery.

Syntax

```
sp_listsuspect_page [dbname]
```

Parameters

dbname – is the name of the database.

Examples

1. `sp_listsuspect_page`

   Lists the suspect pages in the current database.

2. `sp_listsuspect_page pubs2`

   Lists the suspect pages in the *pubs2* database.

Comments

- **sp_listsuspect_page** lists the database name, page ID, object, index ID, and access status for every suspect page in the specified database or, if *dbname* is omitted, in the current user database.
- A value of SA_ONLYin the "access" column indicates that the page has been forced online for System Administrator use only. A value of BLOCK_ALL indicates that the page is offline for everyone.

Messages

- `No such database -- run sp_helpdb to list databases.`

  Reenter the database name.

Permissions

Any user can execute **sp_listsuspect_page**.

Tables Used

master.dbo.sysattributes

See Also

System procedures	**sp_forceonline_db, sp_forceonline_page, sp_listsuspect_db, sp_setsuspect_granularity, sp_setsuspect_threshold**

sp_lock

Function

Reports information about processes that currently hold locks.

Syntax

sp_lock [*spid1* [, *spid2*]]

Parameters

spid1 – is the Adaptive Server process ID number from the *master.dbo.sysprocesses* table. Run **sp_who** to get the *spid* of the lock.

spid2 – is another Adaptive Server process ID number to check for locks.

Examples

1. sp_lock

```
The class column will display the cursor name for locks associated
with a cursor for the current user and the cursor id for other users.
fid spid locktype        table_id  page  dbname  class           context
--- ---- -------------- ---------- ----- ------- --------------  -------
  0   7   Sh_intent       480004741     0 master  Non Cursor Lock NULL
  0  18   Ex_intent        16003088     0 pubtune Non Cursor Lock NULL
  0  18   Ex_page          16003088   587 pubtune Non Cursor Lock NULL
  0  18   Ex_page          16003088   590 pubtune Non Cursor Lock NULL
  0  18   Ex_page          16003088  1114 pubtune Non Cursor Lock NULL
  0  18   Ex_page          16003088  1140 pubtune Non Cursor Lock NULL
  0  18   Ex_page          16003088  1283 pubtune Non Cursor Lock NULL
  0  18   Ex_page          16003088  1362 pubtune Non Cursor Lock NULL
  0  18   Ex_page          16003088  1398 pubtune Non Cursor Lock NULL
  0  18   Ex_page-blk      16003088   634 pubtune Non Cursor Lock NULL
  0  18   Update_page      16003088  1114 pubtune Non Cursor Lock NULL
  0  18   Update_page-blk  16003088   634 pubtune Non Cursor Lock NULL
  0  23   Sh_intent        16003088    0  pubtune Non Cursor Lock NULL
  0  23   Sh_intent       176003658    0  pubtune Non Cursor Lock NULL
  0  23   Ex_intent       208003772    0  pubtune Non Cursor Lock NULL
  1   1   Sh_intent       176003658    0  tpcd   Non Cursor Lock Sync-pt
duration request
  1   1   Sh_intent-blk   208003772    0  tpcd   Non Cursor Lock Sync-pt
```

```
duration request
  1   8   Sh_page         176003658 41571 tpcd    Non Cursor Lock NULL
  1   9   Sh_page         176003658 41571 tpcd    Non Cursor Lock NULL
  1  10   Sh_page         176003658 41571 tpcd    Non Cursor Lock NULL
 11  11   Sh_intent       176003658     0 tpcd   Non Cursor Lock Sync-pt
duration request
 11  12   Sh_page         176003658 41571 tpcd    Non Cursor Lock NULL
 11  13   Sh_page         176003658 41571 tpcd    Non Cursor Lock NULL
 11  14   Sh_page         176003658 41571 tpcd    Non Cursor Lock NULL
```

This example shows the lock status of serial processes with *spids* 7, 18, and 23 and two families of processes. The family with *fid* 1 has the coordinating processes with *spid* 1 and worker processes with *spids* 8, 9, and 10. The family with *fid* 11 has the coordinating processes with *spid* 11 and worker processes with *spids* 12, 13, and 14.

2. sp_lock 7

```
The class column will display the cursor name for locks associated
with a cursor for the current user and the cursor id for other users.
fid spid locktype   table_id  page dbname  class             context
--- ---- ---------  --------- ---- ------  ----------------  ----------
 0   7   Sh_intent  480004741    0 master  Non Cursor Lock    NULL
```

Displays information about the locks currently held by *spid 7*.

Comments

- **sp_lock** with no parameters reports information on all processes that currently hold locks.
- The only user control over locking is through the use of the **holdlock** keyword in the **select** statement.
- Use the **object_name** system function to derive a table's name from its ID number.
- **sp_lock** output is ordered by *fid* and then *spid*.
- The "locktype" column indicates whether the lock is a shared lock ("Sh" prefix), an exclusive lock ("Ex" prefix) or an update lock, and whether the lock is held on a table ("table" or "intent") or on a page ("page").

  A"blk" suffix in the "locktype" column indicates that this process is blocking another process that needs to acquire a lock. As soon as this process completes, the other process(es) moves forward. A "demand" suffix in the "locktype" column indicates that the process is attempting to acquire an exclusive lock. For

more information about lock types, see Chapter 5, "Locking in Adaptive Server," in the *Performance and Tuning Guide*.

- The "class" column indicates whether a lock is associated with a cursor. It displays one of the following:
  - "Non Cursor Lock" indicates that the lock is not associated with a cursor.
  - "Cursor Id *number*" indicates that the lock is associated with the cursor ID number for that Adaptive Server process ID.
  - A cursor name indicates that the lock is associated with the cursor *cursor_name* that is owned by the current user executing **sp_lock**.
- The "fid" column identifies the family (including the coordinating process and its worker processes) to which a lock belongs. Values for "fid" are as follows:
  - A zero value indicates that the task represented by the *spid* is executed serially. It is not participating in parallel execution.
  - A nonzero value indicates that the task (*spid*) holding the lock is a member of a family of processes (identified by *fid*) executing a statement in parallel. If the value is equal to the *spid*, it indicates that the task is the coordinating process in a family executing a query in parallel.
- The "context" column identifies the context of the lock. Worker processes in the same family have the same context value. Legal values for "context" are as follows:
  - "NULL" means that the task holding this lock is either a query executing serially, or is a query executing in parallel in transaction isolation level 1.
  - "Sync-pt duration request" means that the task holding the lock will hold the lock until the query is complete.

    A lock's context may be "Sync-pt duration request" if the lock is a table lock held as part of a parallel query, if the lock is held by a worker process at transaction isolation level 3, or if the lock is held by a worker process in a parallel query and must be held for the duration of the transaction.

Messages

- `The class column will display the cursor name for locks associated with a cursor for the current user and the cursor id for other users.`

Permissions

Any user can execute **sp_lock.**

Tables Used

master.dbo.spt_values, master.dbo.syslocks, master.dbo.sysobjects

See Also

Commands	**kill**, **select**
System procedures	**sp_familylock**, **sp_who**

sp_locklogin

Function

Locks an Adaptive Server account so that the user cannot log in or displays a list of all locked accounts.

Syntax

```
sp_locklogin [loginame, "{lock | unlock}"]
```

Parameters

loginame – is the name of the account to be locked or unlocked.

lock | **unlock** – specifies whether to lock or unlock the account.

Examples

1. `sp_locklogin charles, "lock"`

   Locks the login account for the user “charles.”

2. `sp_locklogin`

   Displays a list of all locked accounts.

Comments

- Locking an Adaptive Server login account prevents that user from logging in. Use **sp_locklogin** instead of **sp_droplogin** for the following reasons:
  - You cannot drop a login who is a user in any database, and you cannot drop a user from a database if the user owns any objects in that database or has granted any permissions on objects to other users.
  - Adaptive Server may reuse the dropped login account’s server user ID (*suid*) when the next login account is created. This occurs only when the dropped login holds the highest *suid* in *syslogins*; however, it could compromise accountability if execution of **sp_droplogin** is not being audited. In addition, it is possible that the user with the reused *suid* will actually be able to access database objects that were authorized for the old *suid*.
  - You cannot drop the last remaining System Security Officer’s or System Administrator’s login account.
- **sp_locklogin** with no parameters returns a list of all the locked accounts.

- You can lock an account that is currently logged in. The user receives a warning that his or her account has been locked, but is not locked out of the account until he or she logs out.
- A locked account can be specified as a Database Owner and can own objects in any database.
- Locking an account that is already locked or unlocking an unlocked account has no effect.
- When locking a System Security Officer's login account, **sp_locklogin** verifies that at least one other unlocked System Security Officer's account exists. Similarly, **sp_locklogin** verifies that there is always an unlocked System Administrator's account. An attempt to lock the last remaining unlocked System Administrator or System Security Officer account causes **sp_locklogin** to return an error message and fail.

Messages

- `Can't run sp_locklogin from within a transaction.`

  sp_locklogin modifies system tables, so it cannot be run from within a transaction.
- `No such account -- nothing changed.`

  You specified an invalid *loginame*.
- `Locked account(s):`

  Lists all locked accounts.
- `Account unlocked.`

  You have successfully unlocked the account.
- `Account locked.`

  You have successfully locked the account.
- `Warning: the specified account is currently active.`

  The account you specified is currently logged in; it will be locked the next time that the user next tries to log in.
- `Cannot lock the last remaining unlocked SA login.`

  An active System Administrator account must always exist.
- `Cannot lock the last remaining unlocked SSO login`

  An active System Security Officer account must always exist.

Permissions

Only a System Administrator or a System Security Officers can use **sp_locklogin**.

Tables Used

master.dbo.sysloginroles, master.dbo.syslogins, master.dbo.sysprocesses, sysobjects

See Also

System procedures	**sp_addlogin**, **sp_modifylogin**, **sp_password**

sp_logdevice

Function

Moves the transaction log of a database with log and data on the same device to a separate database device.

Syntax

sp_logdevice *dbname*, *devname*

Parameters

dbname – is the name of the database whose *syslogs* table, which contains the transaction log, to put on a specific logical device.

devname – is the logical name of the device on which to put the *syslogs* table. This device must be a database device associated with the database (named in **create database** or **alter database**). Run **sp_helpdb** for a report on the database's devices.

Examples

1.
```
create database products on default = 10, logs = 2
go
sp_logdevice products, logs
go
```

   Creates the database *products* and puts the table *products.syslogs* on the database device *logs*.

2.
```
alter database test log on logdev
go
sp_logdevice test, logdev
go
```

   For the database *test* with log and data on the same device, places the log for *test* on the log device *logdev*.

Comments

- The **sp_logdevice** procedure affects only future allocations of space for *syslogs*. This creates a window of vulnerability during which the first pages of your log remain on the same device as your data. Therefore, the preferred method of placing a transaction log on a separate device is the use of the **log on** option to **create database**, which immediately places the entire transaction log on a separate device.

- Place transaction logs on separate database devices, for both recovery and performance reasons.

  A very small, noncritical database could keep its log together with the rest of the database. Such databases use **dump database** to back up the database and log and **dump transaction with truncate_only** to truncate the log.

- **dbcc checkalloc** and **sp_helplog** show some pages for *syslogs* still allocated on the database device until after the next **dump transaction**. After that, the transaction log is completely transferred to the device named when you executed **sp_logdevice**.
- The size of the device required for the transaction log varies, depending on the amount of update activity and the frequency of transaction log dumps. As a rule, allocate to the log device 10 percent to 25 percent of the space you allocate to the database itself.
- Use **sp_logdevice** only for a database with log and data on the same device. Do not use **sp_logdevice** for a database with log and data on separate devices.
- To increase the amount of storage allocated to the transaction log use **alter database**. If you used the **log on** option to **create database** to place a transaction log on a separate device, use:

  sp_extendsegment *segname, devname*

  to increase the size of the log segment. If you did not use **log on**, execute **sp_logdevice**.

  The device or segment on which you put *syslogs* is used **only** for the *syslogs* table. To increase the amount of storage space allocated for the rest of the database, specify any device other than the log device when you issue the **alter database** command.
- Use the **disk init** command to format a new database device for databases or transaction logs.
- See "Placing the Transaction Log on a Separate Device" in Chapter 15, "Creating and Managing User Databases," in the *System Administration Guide* for more details.

Messages

- `No such database -- run sp_helpdb to list databases.`

  No database with the supplied name exists. Run **sp_helpdb** to get a list of databases.

- `No such device exists -- run sp_helpdevice to list the Adaptive Server devices.`

  The *devname* device does not exist in Adaptive Server.

- `syslogs moved.`

  The procedure was successful and the *syslogs* table is now located on the *devname* device.

- `The last-chance threshold for database dbname is now n pages.`

  Adaptive Server created a last-chance threshold for the log segment of the database. When the amount of free space on the log segment falls below *n* pages, Adaptive Server executes **sp_thresholdaction**. Use **sp_modifythreshold** to change the procedure associated with the last-chance threshold.

- `Could not update the last-chance threshold for database dbname.`

  Adaptive Server was unsuccessful in creating a last-chance threshold for the log segment. Your *systhresholds* table may have been corrupted.

- `The specified device is not used by the database.`

  The database *dbname* has no space allocated on the device *devname*.

- `This command has been ignored. The device specified is the only non-log device available for the database and cannot be made log-only.`

  The *devname* you specified is the only, or the last, database device with space available for *dbname*. Making it a log device would leave no space for creating any more objects in the database.

Permissions

Only the Database Owner or a System Administrator can execute **sp_logdevice**.

Tables Used

master.dbo.sysdatabases, master.dbo.sysdevices, master.dbo.sysusages, sysobjects

See Also

Commands	**alter database**, **create database**, **dbcc**, **disk init**, **dump database**, **dump transaction**, **select**
System procedures	**sp_extendsegment**, **sp_helpdevice**

sp_loginconfig

(Windows NT only)

Function

Displays the value of one or all integrated security parameters.

Syntax

```
sp_loginconfig ["parameter_name"]
```

Parameters

parameter_name – is the name of the integrated security parameter you want to examine. Values are: **login mode**, **default account**, **default domain**, **set host**, **key _**, **key $**, **key @**, and **key #**.

Examples

1. sp_loginconfig

```
name                 config_item
-------------------- --------------------
login mode           standard
default account      NULL
default domain       NULL
set host             false
key _                domain separator
key $                space
key @                space
key #                -
```

Displays the values of all integrated security parameters.

2. sp_loginconfig "login mode"

```
name                 config_item
-------------------- --------------------
login mode           standard
```

Displays the value of the **login mode** security parameter.

Comments

- The values of integrated security parameters are stored in the Windows NT Registry. See the chapter on login security in *Configuring Adaptive Server for Windows NT* for instructions on changing the parameters.

- **sp_loginconfig** displays the *config_item* values that were in effect when you started Adaptive Server. If you changed the Registry values after starting Adaptive Server, those values are not reflected in the **sp_loginconfig** output.

Messages

- `Parameter 'parameter_name' is invalid.`

  The *parameter_name* does not match one of the valid parameter names described in this section.

Permissions

Only a System Administrator can execute **sp_loginconfig**.

Tables Used

sysobjects

See Also

System procedures	**sp_revokelogin**

sp_logininfo

(Windows NT only)

Function

Displays all roles granted to Windows NT users and groups with **sp_grantlogin.**

Syntax

```
sp_logininfo ["login_name" | "group_name"]
```

Parameters

login_name – is the network login name of the Windows NT user.

group_name – is the Windows NT group name.

Examples

1. sp_logininfo

```
account name             mapped login name
     type
     privilege
---------------------   ------------------
     ----------------------------
     ------------------------------------------------------------
BUILTIN\Administrators   BUILTIN\Administrators
    group
    'sa_role sso_role oper_role sybase_ts_role navigator_role
              replication_role'
HAZE\regularjoe          HAZE_regularjoe
    user
    'oper_role'

PCSRE\randy              PCSRE_alexander
    user
    'default'
```

Displays all permissions that were granted to Windows NT users and groups with **sp_grantlogin**.

2. sp_logininfo regularjoe

```
account name    mapped login name  type            privilege
--------------- ------------------ --------------- --------------
HAZE\regularjoe HAZE_regularjoe    user            'oper_role'
```

Displays the permissions granted to the Windows NT user "regularjoe."

Comments

- **sp_logininfo** displays all roles granted to Windows NT users and groups with **sp_grantlogin**.
- You can omit the domain name and domain separator (\) when specifying the Windows NT user name or group name.

Messages

- `'login_name' is not a valid account name.`

  The specified Windows NT user name or group does not exist.
- `The account name provided is a domain. Unable to grant privileges to a domain.`

  The specified *login_name* or *group_name* matches a Windows NT domain name. Use only valid Windows NT user names or group names with **sp_logininfo**.
- `Unable to get SQL Server security information.`

  A call to the Windows NT security API failed. Contact your Windows NT administrator.
- `Unable to set SQL Server security information.`

  A call to the Windows NT security API failed. Contact your Windows NT administrator.

Permissions

Only the System Administrator can execute **sp_logininfo**.

Tables Used

sysobjects

See Also

Commands	**grant**, **setuser**
System procedures	**sp_displaylogin**, **sp_grantlogin**, **sp_revokelogin**, **sp_role**, **sp_who**

sp_logiosize

Function

Changes the log I/O size used by Adaptive Server to a different memory pool when doing I/O for the transaction log of the current database.

Syntax

```
sp_logiosize ["default" | "size" | "all"]
```

Parameters

default – sets the log I/O size for the current database to Adaptive Server's default value (4K), if a 4K memory pool is available in the cache. Otherwise, Adaptive Server sets the log I/O size to 2K. Since **default** is a keyword, the quotes are required when specifying this parameter.

size – is the size to set the log I/O for the current database. Values are **2**, **4**, **8**, and **16**. You must enclose the value in quotes.

all – displays the log I/O size configured for all databases grouped by the cache name.

Examples

1. sp_logiosize

```
The transaction log for database 'master' will use
I/O size of 2 Kbytes.
```

Displays the log I/O size configured for the current database.

2. sp_logiosize "8"

Changes the log I/O size of the current database to use the 8K memory pool. If the database's transaction log is bound to a cache that does not have an 8K memory pool, Adaptive Server returns an error message indicating that such a pool does not exist, and the current log I/O size does not change.

3. sp_logiosize "default"

Changes the log I/O size of the current database to Adaptive Server's default value (4K). If a 4K memory pool does not exist in the cache used by the transaction log, Adaptive Server uses the 2K memory pool.

4. sp_logiosize "all"

```
Cache name: default data cache
Data base                      Log I/O Size
-----------------------------  ------------
master                         2 Kb
tempdb                         2 Kb
model                          2 Kb
sybsystemprocs                 2 Kb
pubs3                          2 Kb
pubtune                        2 Kb
dbccdb                         2 Kb
sybsyntax                      2 Kb
```

Displays the log I/O size configured for all databases.

Comments

- **sp_logiosize** displays or changes the log I/O size for the current database. Any user can execute **sp_logiosize** to display the configured log I/O size. Only a System Administrator can change the log I/O size.
- If you specify **sp_logiosize** with no parameters, Adaptive Server displays the log I/O size of the current database.
- When you change the log I/O size, it takes effect immediately. Adaptive Server records the new I/O size for the database in the *sysattributes* table.
- Any value you specify for **sp_logiosize** must correspond to an existing memory pool configured for the cache used by the database's transaction log. Specify these pools using the **sp_poolconfig** system procedure.

  Adaptive Server defines the default log I/O size of a database as 4K, if a 4K memory pool is available in the cache. Otherwise, Adaptive Server sets the log I/O size to 2K (a 2K memory pool is always present in any cache). For most workloads, a log I/O size of 4K performs much better than one of 2K, so each cache used by a transaction log should have a 4K memory pool.

  For more information about configuring caches and memory pools, see the *System Administration Guide* and the *Performance and Tuning Guide*.
- If the transaction logs for one or more databases are bound to a cache of type **logonly**, any memory pools in that cache that have I/O sizes larger than the log I/O size defined for those databases will **not** be used.

For example, assume that only two databases have their transaction logs bound to a "log only" cache containing 2K, 4K, and 8K memory pools. By default, **sp_logiosize** sets the log I/O size for these parameters at 4K, and the 8K pool is not used. Therefore, to avoid wasting cache space, be cautious when configuring the log I/O size.

- During recovery, only the 2K memory pool of the default cache is active, regardless of the log I/O size configured for a database. Transactions logs are read into the 2K pool of the default cache, and all transactions that must be rolled back, or rolled forward, read data pages into the default data cache.

Messages

- `Log I/O Size must be a power of 2. For example: 2, 4, 8, and 16.`

  Sizes for the log I/O are 2K, 4K, 8K, and 16K, which correspond with the I/O sizes of possible memory pools configured for a cache.

- `You must have System Administrator (SA) role to execute this stored procedure.`

  Only users with the System Administrator role can configure caches, memory pools, and the log I/O size for a database.

- `Unable to change the log I/O size. The memory pool for the specified log I/O size does not exist.`

  To change the log I/O size for a database, the cache used by the log must contain a memory pool of the size you specify. The system procedure **sp_poolconfig** defines these memory pools.

- `Log I/O size is set to` *`N`* `Kbytes.`

  Displays the log I/O size set for the current database.

- `Log I/O Size value '`*`N`*`' is illegal.`

  Legal sizes for the log I/O are 2K, 4K, 8K, and 16K, which correspond with the I/O sizes of possible memory pools configured for a cache.

- `Can't run sp_logiosize from within a transaction.`

  sp_logiosize modifies system tables, so it cannot be run from within a transaction.

- `The transaction log for database '`*`database`*`' will use I/O size of` *`N`* `Kbytes.`

  You have changed the log I/O size to this new value.

Permissions

Only the System Administrator can execute **sp_logiosize** to change the log I/O size for the current database. Any user can use **sp_logiosize** to display the log I/O size values.

Tables Used

sysattributes

See Also

System procedures	**sp_cacheconfig**, **sp_poolconfig**

sp_modifylogin

Function

Modifies the default database, default language, default role activation, or full name for a Adaptive Server login account.

Syntax

```
sp_modifylogin account, column, value
```

Parameters

account – is the login account to be modified.

column – specifies the name of the option to be changed. The options are:

Option	Definition
defdb	The "home" database to which the user is connected when he or she logs in.
deflanguage	The official name of the user's default language.
fullname	The user's full name.
"add default role"	The role or roles to be activated by default at login.
"drop default role"	The role or roles to be dropped from the list of roles activated by default at login.

value – is the new value for the specified option.

Examples

1. ```
   sp_modifylogin sarah, defdb, "pubs2"
   ```

   Changes the default database for "sarah" to *pubs2*.

2. ```
   sp_modifylogin claire, deflanguage, "french"
   ```

   Sets the default language for "claire" to French.

3. ```
   sp_modifylogin clemens, fullname, "Samuel Clemens"
   ```

   Changes the full name of user "clemens" to "Samuel Clemens."

4. ```
   sp_modifylogin csmith, "add default role",
   specialist_role
   ```

   Adds the *specialist* role to the list of roles activated by default when user csmith logs in.

5. **`sp_modifylogin hpillai, "drop default role", intern_role`**

   Drops the *intern* role from the list of roles activated by default when user "hpillai" logs in.

Comments

- Set a default database, language, or full name either with **sp_modifylogin** or with **sp_addlogin** when first adding the user's login to Adaptive Server.
  - If you do not specify a default database, the user's default is *master*.
  - If you do not specify a language, the user's default language is set to the server's default language.
  - If you do not specify a full name, that column in *syslogins* remains blank.
- After **sp_modifylogin** is executed, the user is connected to the new *defdb* the next time he or she logs in. However, the user cannot access the database until the Database Owner gives the user access through **sp_adduser** or **sp_addalias**, or unless there is a "guest" user in the database's *sysusers* table. If the user does not have access to the database by any of these means, she or he is connected to *master* and an error message appears.
- If a user's default database is dropped, or if the user is dropped from the database, the user is connected to *master* on his or her next login, and an error message appears.
- If a user's default language is dropped from the server, the server-wide default language is used as the initial language setting, and a message appears.
- Use **sp_modifylogin** to set a role to be activated by default at login or to drop a role from those activated by default at login.

Messages

- `Can't run sp_modifylogin from within a transaction.`

  sp_modifylogin modifies system tables so that it cannot be run from within a transaction.
- `No such account -- nothing changed.`

  You specified a nonexistent account name.

- `Column changed.`

  sp_modifylogin executed successfully.

- `Column name invalid -- nothing changed.`

  You specified an invalid name for the *column* parameter.

Permissions

Any user can use **sp_modifylogin** to change his or her own login account.

Only a System Administrator can use **sp_modifylogin** to change the default database, default language, or full name of another user.

Only a System Security Officer can use **sp_modifylogin** to activate another user's roles by default at login.

Tables Used

master..syslogins, sysobjects, sysloginroles

See Also

System procedures	**sp_activeroles**, **sp_addlogin**, **sp_displaylogin**, **sp_displayroles**, **sp_helprotect**

sp_modify_resource_limit

Function

Changes a resource limit by specifying a new limit value or the action to take when the limit is exceeded, or both.

Syntax

```
sp_modify_resource_limit {name, appname } ,
   rangename, limittype [, limitvalue] [, enforced]
   [, action] [, scope]
```

Parameters

name – is the Adaptive Server login to which the limit applies. You must specify either a *name* or an *appname* or both. To modify a limit that applies to all users of a particular application, specify a *name* of **null**.

appname – is the name of the application to which the limit applies. You must specify either a *name* or an *appname* or both. If the limit applies to all applications used by *name*, specify an *appnam*e of **null**. If the limit governs a particular application, specify the application name that the client program passes to the Adaptive Server in the login packet.

rangename – is the time range during which the limit is enforced. You cannot modify this value, but you must specify a non-null value to uniquely identify the resource limit.

limittype – is the type of resource to which the limit applies. You cannot modify this value, but you must specify a non-null value to uniquely identify the resource limit. The value must be one of the following:

Limit Type	Description
row_count	Limits the number of rows a query can return
elapsed_time	Limits the number of seconds in wall-clock time that a query batch or transaction can run
io_cost	Limits either the actual cost, or the optimizer's cost estimate, for processing a query

limit_value – is the maximum amount of the server resource that the login or application can use before Adaptive Server enforces the limit. This must be a positive integer less than or equal to 2^{31} or **null** to retain the existing value. The following table indicates what value to specify for each limit type:

Limit Type	Limit Value
row_count	The maximum number of rows a query can return before the limit is enforced
elapsed_time	The maximum number of seconds in wall-clock time that a query batch or transaction can run before the limit is enforced
io_cost	A unitless measure derived from optimizer's costing formula

enforced – determines whether the limit is enforced prior to or during query execution. You cannot modify this value. Use **null** as a placeholder.

action – is the action to take when the limit is exceeded. The following codes apply to all limit types:

Action Code	Description
1	Issues a warning
2	Aborts the query batch
3	Aborts the transaction
4	Kills the session
null	Retains the existing value

scope – is the scope of the limit. You cannot modify this value. You can use **null** as a placeholder.

Examples

1.
```
sp_modify_resource_limit robin, NULL, weekends,
row_count, 3000, NULL, 1, NULL
```

Modifies a resource limit that applies to all applications used by "robin" during the *weekends* time range. The limit issues a warning when a query is expected to return more than 3000 rows.

2. sp_modify_resource_limit NULL, acctg, "at all times", elapsed_time, 45, 2, 2, 6

Modifies a resource limit that applies to the *acctg* application on all days of the week and at all times of the day. The limit aborts the query batch when estimated query processing time exceeds 45 seconds.

Comments

- You cannot change the login or application to which a limit applies or specify a new time range, limit type, enforcement time, or scope.
- The modification of a resource limit causes the limits for each session for that login and / or application to be rebound at the beginning of the next query batch for that session.
- For more information on resource limits, see Chapter 12, "Limiting Access to Server Resources," in the *System Administration Guide.*

Messages

- `At least one of the login or application name must be non-NULL`

  You must specify the login or application, or both, to which the limit applies.

- `Can't run sp_modify_resource_limit from within a transaction.`

  Because it modifies system tables, **sp_modify_resource_limit** cannot be run from within a transaction.

- `Illegal action` *`action.`*

  You must specify an action of **1**, **2**, **3**, **4**, or **null**.

- `Illegal enforcement-time value` *`enforced`* `for this limit type.`

  The enforcement time you specified is not valid for the limit type.

- `Illegal limit value` *`limitvalue.`*

  The limit value must be a positive integer.

- `illegal scope` *`scope`* `for this limit type`

  The scope you specified is not valid for the limit type.

- `Limit type must be non-NULL.`

  You must specify a limit type of **row_count**, **elapsed_time**, or **io_cost**.

- `No login with the specified name exists.`

  The login you specified does not exist within the *syslogins* system table in the *master* database.

- `Only the System Administrator (SA) may execute this procedure.`

  You must be a System Administrator to run **sp_modify_resource_limit**.

- `Timerange name must be non-NULL.`

  You must specify the time range to which the limit applies.

- `Unknown limit type` *`limittype.`*

  The limit type you specified does not exist within the *spt_limit_types* system procedure table of the *master* database. You must specify a limit type of **row_count**, **elapsed_time**, or **io_cost**.

- `Unknown time range name` *`rangename.`*

  The time range you specified does not exist within the *systimeranges* system table in the *master* database.

Permissions

Only a System Administrator can execute **sp_modify_resource_limit**.

Tables Used

master..sysresourcelimits, master..systimeranges, master..spt_limit_types

See Also

System procedures	**sp_add_resource_limit**, **sp_drop_resource_limit**, **sp_help_resource_limit**

sp_modify_time_range

Function

Changes the start day, start time, end day, and / or end time associated with a named time range.

Syntax

```
sp_modify_time_range name, startday, endday,
   starttime, endtime
```

Parameters

name – is the name of the time range. This must be the name of a time range stored in the *systimeranges* system table of the *master* database.

startday – is the day of the week on which the time range begins. This must be the full weekday name for the default server language, as stored in the *syslanguages* system table of the *master* database, or **null** to keep the existing *startday*.

endday – is the day of the week on which the time range ends. This must be the full weekday name for the default server language, as stored in the *syslanguages* system table of the *master* database, or **null** to keep the existing end day. The *endday* can fall either earlier or later in the week than the *startday*, or it can be the same day as the *startday*.

starttime – is time of day at which the time range begins. Specify the *starttime* in terms of a twenty-four hour clock, with a value between 00:00 and 23:59. Use the following form:

"*HH*:*MM*"

or **null** to keep the existing *starttime*.

endtime – is the time of day at which the time range ends. Specify the *endtime* in terms of a twenty-four hour clock, with a value between 00:00 (midnight) and 23:59. Use the following form:

"*HH*:*MM*"

or **null** to keep the existing *endtime*. The *endtime* must occur later in the day than the *starttime*, unless *endtime* is 00:00.

➤ ***Note***

For time ranges that span the entire day, specify a start time of "00:00" and an end time of "23:59".

Examples

1. **`sp_modify_time_range business_hours, NULL, Saturday, NULL, NULL`**

   Changes the end day of the *business_hours* time range from Friday to Saturday. Retains the existing start day, start time, and end time.

2. **`sp_modify_time_range before_hours, Monday, Saturday, NULL, "08:00"`**

   Specifies a new end day and end time for the *before_hours* time range.

Comments

- You cannot modify the "at all times" time range.
- It is possible to modify a time range so that it overlaps with one or more other time ranges.
- The modification of time ranges through the system stored procedures does not affect the active time ranges for sessions currently in progress.
- Changes to a resource limit that has a transaction as its scope does not affect any transactions currently in progress.
- For more information on time ranges, see Chapter 12, "Limiting Access to Server Resources," in the *System Administration Guide*.

Messages

- `'at all times' range may not be modified.`

  You cannot modify the "at all times" time range.

- `At least one of starting day, ending day, starting time, or ending time must be non-NULL.`

  You must specify a new *startday*, *endday*, *starttime*, or *endtime* for the time range.

- `Can't run sp_modify_time_range from within a transaction.`

  Because **sp_modify_time_range** modifies system tables, it cannot be run from within a transaction.

- `Ending time must be later in the day than starting time, or ending time must be non-NULL.`

  The *starttime* cannot be later than the *endtime*.

- `Modification would cause overlap with range` *`name.`*

  Overlapping time ranges are permitted. However, overlapping limits for a given user and application combination are not. For example, assume that you have limited joe_user's ability to return rows when he is running the **payroll** application during business hours. You have a second limit that allows joe_user to return more rows when he runs **payroll** in the evening.

  If you attempt to modify the evening hours time range so that it would cause the two resource limits on joe_user to overlap, you will get this message that the modification failed.

- `Only the System Administrator (SA) may execute this procedure.`

  You must be a System Administrator to run **sp_modify_time_range**.

- `Timerange name must be non-NULL.`

  You must specify which time range you want to modify.

- `Unknown endday` *`endday.`*

  You must specify the full name of the day of the week, as stored in the *syslanguages* system table of the *master* database.

- `Unknown ending time value` *`endtime`*`.`

  The *endtime* must be a valid time between "00:00" and "23:59".

- `Unknown range name` *`name`*

  The time range you specified does not exist in the *systimeranges* system table of the *master* database.

- `Unknown startday` *`startday.`*

  You must specify the full name of the day of the week, as stored in the *syslanguages* system table of the *master* database.

- `Unknown starting time value` *`starttime`*`.`

  The *starttime* must be a valid time between "00:00" and "23:59".

Permissions

Only a System Administrator can execute **sp_modify_time_range.**

Tables Used

master..systimeranges, master..syslanguages

See Also

System procedures	**sp_add_resource_limit**, **sp_add_time_range**, **sp_drop_time_range**

sp_modifythreshold

Function

Modifies a threshold by associating it with a different threshold procedure, free-space level, or segment name. You **cannot** use **sp_modifythreshold** to change the amount of free space or the segment name for the last-chance threshold.

Syntax

```
sp_modifythreshold dbname, segname, free_space
   [, new_proc_name] [, new_free_space]
   [, new_segname]
```

Parameters

dbname – is the database for which to change the threshold. This must be the name of the current database.

segname – is the segment for which to monitor free space. Use quotes when specifying the "default" segment.

free_space – is the number of free pages at which the threshold is crossed. When free space in the segment falls below this level, Adaptive Server executes the associated stored procedure.

new_proc_name – is the new stored procedure to execute when the threshold is crossed. The procedure can be located in any database on the current Adaptive Server or on an Open Server. Thresholds cannot execute procedures on remote Adaptive Servers.

new_free_space – is the new number of free pages to associate with the threshold. When free space in the segment falls below this level, Adaptive Server executes the associated stored procedure.

new_segname – is the new segment for which to monitor free space. Use quotes when specifying the "default" segment.

Examples

```
1. sp_modifythreshold mydb, "default", 200, NULL, 175
```

Modifies a threshold on the "default" segment of the *mydb* database to execute when free space on the segment falls below 175 pages instead of 200 pages. NULL is a placeholder indicating that the procedure name is not being changed.

```
2. sp_modifythreshold mydb, data_seg, 250, new_proc
```

Modifies a threshold on the *data_seg* segment of *mydb* so that it executes the *new_proc* procedure.

Comments

- See Chapter 23, "Managing Free Space with Thresholds," in the *System Administration Guide* for more information about using thresholds.

Crossing a Threshold

- When a threshold is crossed, Adaptive Server executes the associated stored procedure. Adaptive Server uses the following search path for the threshold procedure:
  - If the procedure name does not specify a database, Adaptive Server looks in the database in which the threshold was crossed.
  - If the procedure is not found in this database and the procedure name begins with "sp_", Adaptive Server looks in the *sybsystemprocs* database.

  If the procedure is not found in either database, Adaptive Server sends an error message to the error log.
- Adaptive Server uses a **hysteresis value**, the global variable *@@thresh_hysteresis*, to determine how sensitive thresholds are to variations in free space. Once a threshold executes its procedure, it is deactivated. The threshold remains inactive until the amount of free space in the segment rises to *@@thresh_hysteresis* pages above the threshold. This prevents thresholds from executing their procedures repeatedly in response to minor fluctuations in free space.

The Last-Chance Threshold

- By default, Adaptive Server monitors the free space on the segment where the log resides and executes **sp_thresholdaction** when the amount of free space is less than that required to permit a successful dump of the transaction log. This amount of free space, the **last-chance threshold**, is calculated by Adaptive Server and cannot be changed by users.
- If the last-chance threshold is crossed before a transaction is logged, Adaptive Server suspends the transaction until log space is freed. Use **sp_dboption** to change this behavior for a particular

database. Setting the **abort tran on log full** option to **true** causes Adaptive Server to roll back all transactions that have not yet been logged when the last-chance threshold is crossed.

- You cannot use **sp_modifythreshold** to change the free-space value or segment name associated with the last-chance threshold.

Other Thresholds

- Each database can have up to 256 thresholds, including the last-chance threshold.
- Each threshold must be at least 2 times *@@thresh_hysteresis* pages from the next closest threshold.
- Use **sp_helpthreshold** for information about existing thresholds.
- Use **sp_dropthreshold** to drop a threshold from a segment.

Creating Threshold Procedures

- Any user with **create procedure** permission can create a threshold procedure in a database. Usually, a System Administrator creates **sp_thresholdaction** in the *master* database, and Database Owners create threshold procedures in user databases.
- **sp_modifythreshold** does not verify that the specified procedure exists. It is possible to associate a threshold with a procedure that does not yet exist.
- **sp_modifythreshold** checks to ensure that the user modifying the threshold procedure has been directly granted the "sa_role". All system roles active when the threshold procedure is modified are entered in *systhresholds* as valid roles for the user writing the procedure. However, only directly granted system roles are activated when the threshold fires. Indirectly granted system roles and user-defined roles are not activated.
- Adaptive Server passes four parameters to a threshold procedure:
  - *@dbname*, *varchar*(30), which identifies the database
  - *@segment_name*, *varchar*(30), which identifies the segment
  - *@space_left*, *int*, which indicates the number of free pages associated with the threshold
  - *@status*, *int*, which has a value of 1 for last-chance thresholds and 0 for other thresholds

These parameters are passed by position rather than by name; your threshold procedure can use other names for them, but the procedure must declare them in the order shown and with the correct datatypes.

- It is not necessary to create a different procedure for each threshold. To minimize maintenance, create a single threshold procedure in the *sybsystemprocs* database that can be executed by all thresholds.
- Include **print** and **raiserror** statements in the threshold procedure to send output to the error log.

Executing Threshold Procedures

- Tasks that are initiated when a threshold is crossed execute as background tasks. These tasks do not have an associated terminal or user session. If you execute **sp_who** while these tasks are running, the *status* column shows "background".
- Adaptive Server executes the threshold procedure with the permissions of the user who modified the threshold, at the time he or she executed **sp_modifythreshold**, minus any permissions that have since been revoked.
- Each threshold procedure uses one user connection, for as long as it takes to execute the procedure.

Disabling Free-Space Accounting

- Use the **no free space acctg** option of **sp_dboption** to disable free-space accounting on non-log segments.
- You cannot disable free-space accounting on log segments.

◆ ***WARNING!***

System procedures cannot provide accurate information about space allocation when free-space accounting is disabled.

Creating Last-Chance Thresholds for Pre-Release 10.0 Databases

- When you upgrade a pre-release 10.0 database to the current release, it does not automatically acquire a last-chance threshold. Use the **lct_admin** system function to create a last-chance threshold in an existing database.

- Only databases that store their logs on a separate segment can have a last-chance threshold. Use **sp_logdevice** to move the transaction log to a separate device.

Messages

- `This procedure can only affect thresholds in the current database. Say "USE database_name", then run this procedure again.`

  sp_modifythreshold can modify thresholds only in the current database. Either qualify **sp_modifythreshold** with the database name or issue the **use** *database_name* command to open the database in which you want to modify a threshold. Then run **sp_modifythreshold** again.

- `You may not alter the free space or segment name of the log's last-chance threshold.`

  sp_modifythreshold cannot change the free-space value or segment name associated with the last-chance threshold.

Permissions

Only the Database Owner or a System Administrator can execute **sp_modifythreshold**.

Tables Used

master..sysusages, *sysobjects*, *syssegments*, *systhresholds*

See Also

Commands	**create procedure**, **dump transaction**
System procedures	**sp_addthreshold**, **sp_dboption**, **sp_dropthreshold**, **sp_helpthreshold**, **sp_thresholdaction**

sp_monitor

Function

Displays statistics about Adaptive Server.

Syntax

sp_monitor

Parameters

None.

Examples

1. sp_monitor

```
last_run               current_run          seconds
-------------------    -------------------  ---------
Jan 29 1987 10:11AM    Jan 29 1987 10:17AM  314

cpu_busy           io_busy     idle
---------------    ---------   --------------
4250(215)-68%      67(1)-0%    109(100)-31%

packets_received       packets_sent    packet_errors
----------------       ------------    ------------
781(15)                10110(9596)     0(0)

total_read     total_write total_errors    connections
-----------    ------------------------    -----------
394(67)        5392(53)    0(0)            15(1)
```

Reports information about how busy Adaptive Server has been.

Comments

- Adaptive Server keeps track of how much work it has done in a series of global variables. **sp_monitor** displays the current values of these global variables and how much they have changed since the last time the procedure executed.
- For each column, the statistic appears in the form *number(number)-number%* or *number(number)*.
  - The first number refers to the number of seconds (for "cpu_busy", "io_busy", and "idle") or the total number (for the other columns) since Adaptive Server restarted.
  - The number in parentheses refers to the number of seconds or the total number since the last time **sp_monitor** was run. The

percent sign indicates the percentage of time since **sp_monitor** was last run.

For example, if the report shows "cpu_busy" as "4250(215)-68%", it means that the CPU has been busy for 4250 seconds since Adaptive Server was last started, 215 seconds since **sp_monitor** last ran, and 68 percent of the total time since **sp_monitor** was last run.

For the "total_read" column, the value 394(67) means there have been 394 disk reads since Adaptive Server was last started, 67 of them since the last time **sp_monitor** was run.

- Table 3-18 describes the columns in the **sp_monitor** report, the equivalent global variables, if any, and their meanings. With the exception of "last_run", "current_run" and "seconds", these column headings are also the names of global variables—except that all global variables are preceded by @@. There is also a difference in the units of the numbers reported by the global variables—the numbers reported by the global variables are not milliseconds of CPU time, but machine ticks.

Table 3-18: Columns in the sp_monitor report

Column Heading	Equivalent Variable	Meaning
last_run		Clock time at which the **sp_monitor** procedure last ran.
current_run		Current clock time.
seconds		Number of seconds since **sp_monitor** last ran.
cpu_busy	*@@cpu_busy*	Number of seconds in CPU time that Adaptive Server's CPU was doing Adaptive Server work.
io_busy	*@@io_busy*	Number of seconds in CPU time that Adaptive Server has spent doing input and output operations.
idle	*@@idle*	Number of seconds in CPU time that Adaptive Server has been idle.
packets_received	*@@pack_received*	Number of input packets read by Adaptive Server.
packets_sent	*@@pack_sent*	Number of output packets written by Adaptive Server.
packet_errors	*@@packet_errors*	Number of errors detected by Adaptive Server while reading and writing packets.
total_read	*@@total_read*	Number of disk reads by Adaptive Server.
total_write	*@@total_write*	Number of disk writes by Adaptive Server.

Table 3-18: Columns in the sp_monitor report (continued)

Column Heading	Equivalent Variable	Meaning
total_errors	*@@total_errors*	Number of errors detected by Adaptive Server while reading and writing.
connections	*@@connections*	Number of logins or attempted logins to Adaptive Server.

- The first time **sp_monitor** runs after Adaptive Server start-up, the number in parentheses is meaningless.
- Adaptive Server's housekeeper task uses the server's idle cycles to write changed pages from cache to disk. This process affects the values of the "cpu_busy", "io_busy", and "idle" columns reported by **sp_monitor.** To disable the housekeeper task and eliminate these effects, set the **housekeeper free write percent** configuration parameter to 0:

  ```
  sp_configure "housekeeper free write percent", 0
  ```

Messages

- `Can't run sp_monitor from within a transaction.`

  sp_monitor modifies system tables, so it cannot be run within a transaction.

Permissions

Only a System Administrator can execute **sp_monitor.**

Tables Used

master.dbo.sysengines, master.dbo.spt_monitor, sysobjects

See Also

System procedures	**sp_who**

sp_monitorconfig

Function

Displays cache usage statistics regarding metadata descriptors for indexes, objects, and databases. **sp_monitorconfig** also reports statistics on auxiliary scan descriptors used for referential integrity queries.

Syntax

sp_monitorconfig "*configname*"

Parameters

configname – is the all or part of the configuration parameter name whose monitoring information is being queried. Valid configuration parameters are **open indexes**, **open objects**, **open databases**, **aux scan descriptors**, and **all**. Specifying **all** displays descriptor help information for all indexes, objects, databases, and auxiliary scan descriptors in the server.

Examples

1. sp_monitorconfig "open"

```
Configuration option is not unique.
option_name                    config_value run_value
------------------------------ ------------ -----------
curread change w/ open cursors            1           1
number of open databases                 12          12
number of open indexes                  500         500
number of open objects                  500         500
open index hash spinlock ratio          100         100
open index spinlock ratio               100         100
open object spinlock ratio              100         100
```

2. sp_monitorconfig "open objects"

```
Usage information at date and time: Jan 14 1997 8:54AM.
Name            # Free   # Active   % Active  # Max Ever Used   Re-used
--------------  --------  ------   --------   ---------------   -------
number of open  217       283      56.60      300               No
objects
```

In this example, there are 283 active object metadata descriptors, with 217 free. The maximum used at a peak period since Adaptive Server was last started is 300. You can then reset the

size to 330, for example, to accommodate the 300 maximum used metadata descriptors, plus space for 10 percent more:

```
sp_configure "number of open objects", 330
```

3. **sp_monitorconfig "open indexes"**

```
Usage information at date and time: Jan 14 1997 8:55AM.

Name            # Free   # Active   % Active  # Max Ever Used   Re-used
--------------  --------  ------   --------   ---------------   -------
number of open  556       44       7.33       44                No
indexes
```

In this example, the maximum number of index metadata descriptors is 44. You can reset the size to 100, the minimum acceptable value:

```
sp_configure "number of open indexes", 100
```

4. **sp_monitorconfig "aux scan descriptors"**

```
Usage information at date and time: Jan 14 1997 8:56AM.

Name            # Free   # Active   % Active  # Max Ever Used   Re-used
--------------  --------  ------   --------   ---------------   -------
number of aux   170       30       15.00      32                NA
scan
descriptors
```

In this example, the number of active scan descriptors is 30, though Adaptive Server is configured to use 200. Use the **number of aux scan descriptors** configuration parameter to reset the value to at least 32. A safe setting is 36, to accommodate the 32 scan descriptors, plus space for 10 percent more.

5. **sp_monitorconfig "number of open databases"**

```
Usage information at date and time: Jan 14 1997 8:57AM.

Name            # Free   # Active   % Active  # Max Ever Used   Re-used
--------------  --------  ------   --------   ---------------   -------
number of open  0         5        100.00     5                 Yes
databases
```

In this example, Adaptive Server is configured for 5 open databases, all of which have been used in the current session. However, as indicated by the “Re-used” column, an additional database needs to be opened. If all 5 databases are in use, an error may result, unless the descriptor for a database that is not in use can be reused. To prevent an error, reset **number of open databases** to a higher value.

Comments

- **sp_monitorconfig** displays cache usage statistics regarding metadata descriptors for indexes, objects, and databases, such as the number of metadata descriptors currently in use by the server.
- **sp_monitorconfig** also reports the number of auxiliary scan descriptors in use. A scan descriptor manages a single scan of a table when queries are run on the table.
- The columns in the **sp_monitorconfig** output provide the following information:
  - "# Free" specifies the number of available metadata or auxiliary scan descriptors not currently used.
  - "# Active" specifies the number of metadata or auxiliary scan descriptors installed in cache (that is, active).
  - "% Active" specifies the percentage of cached or active metadata or auxiliary scan descriptors.
  - "# Max Ever Used" specifies the maximum number of metadata or auxiliary scan descriptors that have been in use since the server was started.
  - "Re-used" specifies whether a metadata descriptor was reused in order to accommodate an increase in indexes, objects, or databases in the server. The returned value is "Yes", "No" or "NA" (for configuration parameters that do not support the reuse mechanism, such as the number of **aux scan descriptors**).
- Use the value in the "# Max Ever Used" column as a basis for determining an appropriate number of descriptors; be sure to add about 10 percent for the final setting. For example, if the maximum number of index metadata descriptors used is 142, you might set the **number of open indexes** configuration parameter to 157.
- If the "Re-used" column states "Yes," reset the configuration parameter to a higher value. When descriptors need to be reused, there can be performance problems, particularly with open databases. An open database contains a substantial amount of metadata information, which means that to fill up an open database, Adaptive Server needs to access the metadata on the disk many times; the server can also have a spinlock contention problem. To check for spinlock contention, use the system procedure **sp_sysmon**. See the *Performance and Tuning Guide* for more information. To find the current number of indexes, objects, or databases, use **sp_countmetadata**.

- To get an accurate reading, run **sp_monitorconfig** during a normal Adaptive Server peak time period. You can run **sp_monitorconfig** several times during the peak period to ensure that you are actually finding the maximum number of descriptors used.

Permissions

Only a System Administrator can use **sp_monitorconfig**.

Tables Used

master..sysindexes, sysobjects, sysdatabases

See Also

System procedures	**sp_configure**, **sp_countmetadata**, **sp_helpconfig**, **sp_helpconstraint**

sp_passthru

(Component Integration Services only)

Function

Allows the user to pass a SQL command buffer to a remote server.

Syntax

```
sp_passthru server, command, errcode, errmsg, rowcount
    [, arg1, arg2, ... argn]
```

Parameters

server – is the name of a remote server to which the SQL command buffer will be passed. The class of this server must be a supported, non-local server class.

command – is the SQL command buffer. It can hold up to 255 characters.

errcode – is the error code returned by the remote server, if any. If no error occurred at the remote server, the value returned is 0.

errmsg – is the error message returned by the remote server. It can hold up to 255 characters. This parameter is set only if *errcode* is a nonzero number; otherwise NULL is returned.

rowcount – is the number of rows affected by the last command in the command buffer. If the command was an **insert**, **delete**, or **update**, this value represents the number of rows affected even though none were returned. If the last command was a query, this value represents the number of rows returned from the external server.

arg1 ... argn – receives the results from the last row returned by the last command in the command buffer. You can specify up to 250 *arg* parameters. All must be declared as output parameters.

Examples

1.
```
sp_passthru ORACLE, "select date from dual",
@errcode output, @errmsg output, @rowcount output,
@oradate output
```

   Returns the date from the Oracle server in the output parameter *@oradate*. If an Oracle error occurs, the error code is placed in *@errcode* and the corresponding message is placed in *@errmsg*. The *@rowcount* parameter will be set to 1.

Comments

- **sp_passthru** allows the user to pass a SQL command buffer to a remote server. The syntax of the SQL statement or statements being passed is assumed to be the syntax native to the class of server receiving the buffer. No translation or interpretation is performed. Results from the remote server are optionally placed in output parameters.

  Use **sp_passthru** only when Component Integration Services is installed and configured.

- You can include multiple commands in the command buffer. For some server classes, the commands must be separated by semicolons. Refer to the *Component Integration Services User's Guide* for a more complete discussion of query buffer handling in passthrough mode.

Return Parameters

- The output parameters *arg1* ... *argn* will be set to the values of corresponding columns from the last row returned by the last command in the command buffer. The position of the parameter determines which column's value the parameter will contain. *arg1* receives values from column 1, *arg2* receives values from column 2, and so on.
- If there are fewer optional parameters than there are returned columns, the excess columns are ignored. If there are more parameters than columns, the remaining parameters are set to NULL.
- An attempt is made to convert each column to the datatype of the output parameter. If the datatypes are similar enough to permit **implicit** conversion, the attempt will succeed. For information on implicit conversion, see "Datatype Conversion Functions" in Chapter 2, "Transact-SQL Functions". For information on which datatype represents the datatypes from each server class when in passthru mode, see the *Component Integration Services User's Guide.*

Messages

- `Column number 'col_num' returned from the last query does not match the argument supplied to receive its value.`

  Output argument number *col_num* has a datatype to which corresponding column data cannot be converted.

- `Arithmetic overflow error for type` *`datatype`*`, value` *`value`*`.`

  A numeric column value cannot fit into the parameter that is to receive it.

- `Site` *`server_name`* `not found in sysservers`

  The *server* parameter does not match the name of any non-local server. Add remote servers with **sp_addserver**.

- `Can't open a connection to site` *`server_name`* `because it does not have PASSTHRU or RPC capabilities.`

  The server argument does not specify the name of a remote server that can be contacted in passthru mode.

Permissions

Any user can execute **sp_passthru**.

Tables Used

sysservers, *sysremotelogins*

See Also

System procedures	**sp_autoconnect**, **sp_remotesql**

sp_password

Function

Adds or changes a password for an Adaptive Server login account.

Syntax

sp_password *caller_passwd*, *new_passwd* [, *loginame*]

Parameters

caller_passwd – is your password. When you are changing your own password, this is your old password. When a System Security Officer is using **sp_password** to change another user's password, *caller_passwd* is the System Security Officer's password.

new_passwd – is the new password for the user, or for *loginame*. It must be at least 6 bytes long. Enclose passwords that include characters besides A-Z, a-z, or 0-9 in quotation marks. Also enclose passwords that begin with 0-9 in quotes.

loginame – the login name of the user whose account password is being changed by the System Security Officer.

Examples

1. **sp_password "3blindmice, "2mediumhot"**

   Changes your password from password from "3blindmice" to "2mediumhot." (Enclose the passwords in quotes because they begin with numerals.)

2. **sp_password "2tomato", sesame1, victoria**

   A System Security Officer whose password is "2tomato" has changed Victoria's password to "sesame1."

3. **sp_password null, "16tons"**

   Changes your password from NULL to "16tons." Notice that NULL is not enclosed in quotes. (NULL is not a permissible new password.)

4. **PRODUCTION...sp_password figaro, lilacs**

   Changes your password on the PRODUCTION server from "figaro" to "lilacs."

Comments

- Any user can change his or her password with **sp_password.**
- New passwords must be at least 6 characters long. They cannot be NULL.
- The encrypted text of *caller_passwd* must match the existing encrypted password of the caller. If it does not, **sp_password** returns an error message and fails. *master.dbo.syslogins* lists passwords in encrypted form.
- If a client program requires users to have the same password on remote servers as on the local server, users must change their passwords on all the remote servers before changing their local passwords. Execute **sp_password** as a remote procedure call on each remote server. See example 4.
- You can set the **systemwide password expiration** configuration parameter to establish a password expiration interval that forces all Adaptive Server login accounts to change passwords on a regular basis. See Chapter 11, "Setting Configuration Parameters," in the *System Administration Guide* for more information.

Messages

- `Can't run sp_password from within a transaction.`

  sp_password modifies system tables, so it cannot be run within a transaction.
- `Error: Unable to set the Password.`

  Check your syntax carefully and try again to set the password.
- `No such login -- no password changed.`

  The name supplied for the *loginame* parameter does not exist in Adaptive Server.
- `Invalid caller's password specified, password left unchanged.`

  The *caller_passwd* parameter is not the current password of the caller.
- `New password specified is too short. Minimum length of acceptable passwords is 6 characters.`

  You specified a password that is too short.

- `New password supplied is the same as previous password. Please supply a different password.`

  If *new_passwd* is at least 6 bytes long, it is encrypted and compared with the encrypted value of the existing encrypted password for *loginame*. If they differ, the encrypted text of *new_passwd* is saved; otherwise, **sp_password** fails and returns this message.

- `Password correctly set.`

  The password was successfully changed. Use the new password the next time you log into Adaptive Server.

Permissions

Any user can execute **sp_password** to change his or her own password. Only a System Security Officer can use **sp_password** to change another user's password.

Tables Used

master.dbo.syslogins, *sysobjects*

See Also

System procedures	**sp_addlogin**, **sp_adduser**

sp_placeobject

Function

Puts future space allocations for a table or index on a particular segment.

Syntax

sp_placeobject *segname*, *objname*

Parameters

segname – is the name of the segment on which to locate the table or index.

objname – is the name of the table or index for which to place subsequent space allocation on the segment *segname*. Specify index names in the form "*tablename.indexname*".

Examples

1. **sp_placeobject segment3, authors**

   This command places all subsequent space allocation for the table *authors* on the segment named "segment3".

2. **sp_placeobject indexes, 'employee.employee_nc'**

   This command places all subsequent space allocation for the *employee* table's index named *employee_nc* on the segment named *indexes*.

Comments

- You cannot change the location of future space allocations for system tables.
- Placing a table or an index on a particular segment does not affect the location of any existing table or index data. It affects only future space allocation. Changing the segment used by a table or an index can spread the data among multiple segments.
- If you use **sp_placeobject** with a clustered index, the table moves with the index.
- You can specify a segment when you create a table or an index with **create table** or **create index**. If you do not specify a segment, the data goes on the *default* segment.

- When **sp_placeobject** splits a table or an index across more than one disk fragment, the diagnostic command **dbcc** displays messages about the data that resides on the fragments that were in use for storage before **sp_placeobject** executed. Ignore those messages.
- You cannot use **sp_placeobject** on a partitioned table.

Messages

- `'objname' is now on segment 'segname'.`

  The command was successful.

- `Partitioned objects cannot be moved.`

  The table you specified is partitioned. To move the table, first use the **unpartition** clause of the **alter table** command to unpartition it, and then issue the **sp_placeobject** system procedure.

- `There is no index named 'indexname' for table 'tablename'.`

  The index referenced in the *objname* parameter does not exist. Use the system procedure **sp_helpindex** to list the table's indexes.

- `There is no such segment as segname.`

  The *segname* you referenced is not a segment. All segments for a database are listed in the *syssegments* table. Use **sp_helpsegment** to get a report on all segments.

- `There is no table named 'tablename'.`

  The table referenced in the *objname* parameter does not exist. Use the system procedure **sp_help** for a list of tables.

- `You do not own table 'tablename'.`

  Only the table owner, the Database Owner, or a System Administrator can place a table or its index on a segment.

- `Use sp_logdevice to move syslogs table.`
- `You can't move system tables.`

  System tables must remain on the *system* segment.

Permissions

Only the table owner, Database Owner, or System Administrator can execute **sp_placeobject**.

Tables Used

sysindexes, *sysobjects*, *syspartitions*, *syssegments*

See Also

Commands	**alter table**, **dbcc**
System procedures	**sp_addsegment**, **sp_dropsegment**, **sp_extendsegment**, **sp_help**, **sp_helpindex**, **sp_helpsegment**

sp_plan_dbccdb

Function

Recommends suitable sizes for new *dbccdb* and *dbccalt* databases, lists suitable devices for *dbccdb* and *dbccalt*, and suggests a cache size and a suitable number of worker processes for the target database.

Syntax

sp_plan_dbccdb [*dbname*]

Parameters

dbname – specifies the name of the target database. If *dbname* is not specified, **sp_plan_dbccdb** makes recommendations for all databases in *master..sysdatabases*.

Example

1. sp_plan_dbccdb master

```
Recommended size for dbccdb is 4MB.

dbccdb database already exists with size 8MB.

Recommended values for workspace size, cache size and process
count are:

dbname          scan ws    text ws    cache    process count


master          64K        64K       640K    1
```

Returns configuration recommendations for creating a *dbccdb* database suitable for checking the *master* database. The *dbccdb* database already existed at the time this command was run, so the size of the existing database is provided for comparison.

2. sp_plan_dbccdb

```
Recommended minimum size for dbccdb is 4MB.

Recommended values for workspace size, cache size and process
count are:

dbname          scan ws    text ws    cache    process count

master          64K        64K        640K     1
tempdb          64K        64K        640K     1
model           64K        64K        640K     1
sybsystemprocs  272K       80K        640K     1
dbccdb         128K       64K       640K     1
```

Returns configuration recommendations for creating a *dbccdb* database suitable for checking all databases in the server. No *dbccdb* database existed at the time this command was run.

3. sp_plan_dbccdb pubs2

```
Recommended size for dbccdb is 4MB.

Recommended devices for dbccdb are:

Logical Device Name      Device Size
       Physical Device Name

sprocdev                 28672
      /remote/sybase/devices/srv_sprocs_dat
tun_dat                  8192
      /remote/sybase/devices/srv_tun_dat
tun_log                  4096
      /remote/sybase/devices/srv_tun_log

Recommended values for workspace size, cache size and process
count are:

dbname      scan ws    text ws    cache    process count

pubs2      64K        64K        640K     1
```

Returns configuration recommendations for creating a *dbccdb* database suitable for checking *pubs2*.

Comments

- **sp_plan_dbccdb** recommends suitable sizes for creating new *dbccdb* and *dbccalt* databases, lists suitable devices for the new database,

and suggests cache size and a suitable number of worker processes for the target database.

- If you specify *dbccdb*, **sp_plan_dbccdb** recommends values for *dbccalt*, the alternate database. If you specify *dbccalt*, **sp_plan_dbccdb** recommends values for *dbccdb*.
- **sp_plan_dbccdb** does not report values for existing *dbccdb* and *dbccalt* databases. To gather configuration parameters for an existing *dbccdb* or *dbccalt* database, use **sp_dbcc_evaluatedb**.
- For information on the **dbcc** stored procedures for maintaining *dbccdb* and for generating reports from *dbccdb*, see Chapter 6, "dbcc Stored Procedures."

Messages

- `Can't run sp_plan_dbccdb from within a transaction.`

  sp_plan_dbccdb modifies system tables, so it cannot be run within a transaction.

- *dbname* `database already exists with size` *nMB*`.`

  sp_plan_dbccdb notifies you that the database (either *dbccdb* or *dbccalt*) already exists. To gather configuration parameters for an existing *dbccdb* or *dbccalt* database, use **sp_dbcc_evaluatedb**.

- `No such database -- run sp_helpdb to list databases.`

  The database you specified does not exist. Run **sp_helpdb** to list the databases and check the spelling.

- `No suitable devices for` *dbname* `in master..sysdevices.`

  sp_plan_dbccdb did not find any suitable devices in *master..sysdevices* for the database you specified. Use **disk init** to create a new device for *dbccdb*.

- `Recommended devices for` *dbname* `are:`

  sp_plan_dbccdb recommends one or more devices to use for the database specified by *dbname*. If you specified "dbccdb" for *dbname*, **sp_plan_dbccdb** recommends devices for the *dbccalt* database. If you specified "dbccalt" for *dbname*, **sp_plan_dbccdb** recommends devices for the *dbccdb* database.

- `Recommended size for` *dbname* `is` *n*`MB.`

  sp_plan_dbccdb recommends the minimum size needed for the database specified by *dbname*. If you specified "dbccdb" for *dbname*, **sp_plan_dbccdb** recommends the minimum size needed

for the *dbccalt* database. If you specified "dbccalt" for *dbname*, **sp_plan_dbccdb** recommends the minimum size needed for the *dbccdb* database.

- `Recommended values for workspace size, cache size and process count are:`

  sp_plan_dbccdb recommends the values for workspace size, cache size, and the number of worker process to use for the target database you specified.

- `You must be the System Administrator (SA) or the Database Owner (dbo) to execute this procedure.`

Permissions

Only the System Administrator or Database Owner can execute **sp_plan_dbccdb**. Only the System Administrator can execute **sp_plan_dbccdb** without specifying a database name.

Tables Used

master..sysdatabases, *master..sysdevices*, *master..sysusages*

See Also

Commands	**dbcc**
System procedures	**sp_dbcc_evaluatedb**

sp_poolconfig

Function

Creates, drops, resizes, and provides information about memory pools within data caches.

Syntax

To create a memory pool in an existing cache, or to change pool size:

```
sp_poolconfig cache_name [, "mem_size[P|K|M|G]",
   "config_poolK" [, "affected_poolK"]]
```

To change a pool's wash size:

```
sp_poolconfig cache_name, "io_size",
   "wash=size[P|K|M|G]"
```

To change a pool's asynchronous prefetch percentage:

```
sp_poolconfig cache_name, "io_size",
   "local async prefetch limit=percent"
```

Parameters

cache_name – is the name of an existing data cache.

mem_size – is the size of the memory pool to be created or the new total size for an existing pool, if a pool already exists with the specified I/O size. The minimum size of a pool is 512K. Specify size units with **P** for pages, **K** for kilobytes, **M** for megabytes, or **G** for gigabytes. The default is kilobytes.

config_pool – is the I/O size performed in the memory pool where the memory is to be allocated or removed.

Valid I/O sizes are **2K**, **4K**, **8K**, and **16K**.

affected_pool – is the size of I/O performed in the memory pool where the memory is to be deallocated. If *affected_pool* is not specified, the memory is taken from the 2K pool.

io_size – is the size of I/O performed in the memory pool where the wash size is to be reconfigured. The combination of cache name and I/O size uniquely identifies a memory pool.

wash=*size* – Changes the wash size (the point in the cache at which Adaptive Server writes dirty pages to disk) for a memory pool.

local async prefetch limit=*percent* – sets the percentage of buffers in the pool that can be used to hold buffers that have been read into cache by asynchronous prefetch, but that have not yet been used.

Examples

1. **`sp_poolconfig pub_cache, "10M", "16K"`**

   Creates a 16K pool in the data cache *pub_cache* with 10MB of space. All space is taken from the default 2K memory pool.

2. **`sp_poolconfig pub_cache, "3M", "8K", "16K"`**

   Moves 3MB of space to the 8K pool from the 16K pool of *pub_cache*.

3. **`sp_poolconfig "pub_cache"`**

   Reports the current configuration of *pub_cache*.

4. **`sp_poolconfig pub_cache, "0K", "16K"`**

   Removes the 16K memory pool from *pub_cache*, placing all of the memory assigned to it in the 2K pool.

5. **`sp_poolconfig pub_cache, "2K", "wash=508K"`**

   Changes the wash size of the 2K pool in *pubs_cache* to 508K.

6. **`sp_poolconfig pub_cache, "2K", "local async prefetch limit=15"`**

   Changes the asynchronous prefetch limit for the 2K pool to 15 percent.

Comments

- When you create a data cache with **sp_cacheconfig**, all space is allocated to the 2K memory pool. **sp_poolconfig** divides the data cache into additional pools with larger I/O sizes.
- If no large I/O memory pools exist in a cache, Adaptive Server performs I/O in 2K units, the size of a data page, for all of the objects bound to the cache. You can often enhance performance by configuring pools that perform large I/O. A 16K memory pool reads and writes eight data pages in a single I/O operation.
- The combination of cache name and I/O size must be unique. In other words, you can have only one pool of a given I/O size in a particular data cache.
- Only one **sp_poolconfig** command can be active on a single cache at one time. If a second **sp_poolconfig** command is issued before the first one completes, it sleeps until the first command completes.

- Figure 3-3 shows a data cache with:
  - The default data cache with a 2K pool and a 16K pool
  - A user cache with a 2K pool and a 16K pool
  - A log cache with a 2K pool and a 4K pool

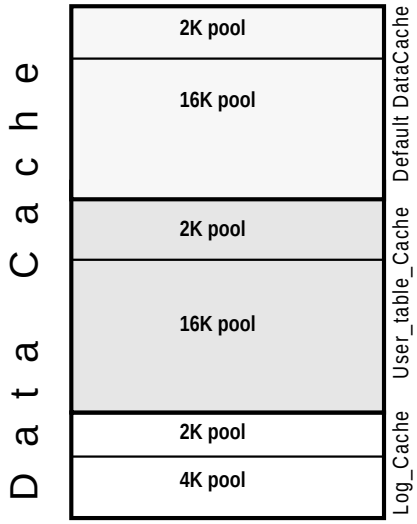

Figure 3-3: Data cache with default and user-defined caches

- You can create pools with I/O sizes up to 16K in the default data cache.
- The minimum size of a memory pool is 512K. You cannot reduce the size of any memory pool in any cache to less than 512K by transferring memory to another pool.
- Two circumstances can create pool less than 512K:
  - If you attempt to delete a pool by setting its size to zero, and some of the pages are in use, **sp_poolconfig** reduces the pool size as much as possible, and prints a warning message. The status for the pool is set to "Unavailable/deleted".
  - If you attempt to move buffers to create a new pool, and enough buffers cannot be moved to the new pool, **sp_poolconfig** moves as many buffers as it can, and the cache status is set to "Unavailable/too small."

In both of these cases, you can retry to command at a later time. The pool will also be deleted or be changed to the desired size when the server is restarted.

- You can create memory pools while Adaptive Server is active; no restart is needed for them to take effect. However, Adaptive Server can move only "free" buffers (buffers that are not in use or that do not contain changes that have not been written to disk). When you configure a pool or change its size, Adaptive Server moves as much memory as possible to the pool and prints an informational message showing the requested size and the actual size of the pool. After a restart of Adaptive Server, all pools are created at the configured size.
- The following commands perform only 2K I/O: **create database**, **alter database**, **create index**, **disk init**, **dbcc**, and **drop table.** Also, recovery uses only the 2K memory pool: all pages are read into and changed in the 2K pool of the default cache. Be sure that your default 2K pool is large enough for these activities.
- Most Adaptive Servers perform best with 4K I/O configured for transactions logs. Adaptive Server uses the default I/O size of 4K if the default cache or a cache with a transaction log bound to it is configured with a 4K memory pool. Otherwise, it uses the 2K memory pool.
- You can increase the default log I/O size for a database using the **sp_logiosize** system procedure. However, the I/O size you specify must have memory pools of the same size in the cache bound to the transaction log. If not, Adaptive Server uses the 4K or 2K memory pools.

Wash Percentage

- The default value for the wash size is computed as follows:
  - If the pool size is less than 300MB, the default wash size is set to 20 percent of the buffers in the pool
  - If the pool size is greater than 300MB, the default wash size is 20 percent of the number of buffers in 300MB
- The minimum setting for the wash size is 10 buffers, and the maximum setting is 80 percent of the size of the pool.
- Each memory pool contains a wash area at the least recently used (LRU) end of the chain of buffers in that pool. Once dirty pages (pages that have been changed while in cache) move into the wash area, Adaptive Server initiates asynchronous writes on

these pages. The wash area must be large enough so that pages can be written to disk before they reach the LRU end of the pool. Performance suffers when Adaptive Server needs to wait for clean buffers.

The default percentage, placing 20 percent of the buffers in the wash area, is sufficient for most applications. If you are using an extremely large memory pool, and your applications have a very high data modification rate, you may want to increase the size to 1 or 2 percent of the pool. Contact Sybase Technical Support for more information about choosing an effective wash size.

Local Asynchronous Prefetch Percentage

- The default value for a pool's asynchronous prefetch percentage is set by the configuration parameter **global async prefetch limit**. The pool limit always overrides the global limit.
- To disable prefetch in a pool (if the global limit is a nonzero number), set the pool's limit to 0.
- For information on the performance impact of changes to the asynchronous prefetch limit, see Chapter 18, "Tuning Asynchronous Prefetch," in the *Performance and Tuning Guide.*

Messages

- `A cache name must be supplied.`

  You did not specify a cache name. **sp_poolconfig** can report information about the pools in one cache, and requires a cache name. To see a report on all caches, use **sp_cacheconfig**.

- `Can't run sp_poolconfig from within a transaction.`

  You cannot run this command from within a transaction because it modifies system tables.

- `Invalid buffer size of 1K encountered. Valid buffer sizes are powers of two between 2k and 16k inclusive.`

  The argument for the configured pool or the affected pool was not specified correctly. Correct values are **2K**, **4K**, **8K** and **16K**.

- `I/O size of the memory pool is expected as the third argument.`

  If you are using **sp_poolconfig** to create or drop a pool in a cache, the third argument must be a valid I/O size.

- `2k pool must be a minimum of 512k.`

Check the pool size.

- `Less memory moved than requested in cache 'cache_name' Requested size = N Kb: from pool = io_size, to pool = io_size, actual memory moved = N Kb.`

  Only some of the memory was moved from one pool to another because some of the pages in the source pool were in use by active queries on the server. Try the command again later.

- `Source pool (io_size) and destination pool (io_size) are the same pool. The source and destination pools must be different.`

  You gave the same I/O size twice during a command to move memory between pools.

- `Syntax error encountered. Sizes must be of the form int[KMGP]. For example, a size of 5 megabytes may be specified as '5M'.`

  The third parameter to **sp_poolconfig** was similar to the wash size or local asynchronous prefetch configuration syntax but not exactly correct. Check the syntax, and retype the command.

- `The destination buffer pool size must be a minimum of 512 kilobytes.`

  The smallest pool that can be created is 512K.

- `The 'io_size' pool has been marked as being unavailable since it is  smaller than the minimum pool size of 512KB. You can size the pools correctly by retrying sp_poolconfig when the cache is not in use or by restarting the Server.`

  An attempt to move buffers to a pool did not succeed in moving enough buffers to create a pool of at least 512K because some of the buffers in the source pool were in use. You can try your command again later. Also, the pool will be created at the specified size when the server is restarted.

- `The source pool (io_size buffers, total size NKbytes) is not large enough to satisfy the request to move NKb of memory.`

  The pool configuration command you issued attempted to move more memory than allowed from a pool. If the source pool was the 2K pool, it must always have at least 512K.

- `The specified named cache 'cache_name' does not exist.`

To see the caches available, run **sp_cacheconfig** with no parameters.

- `The specified pool (io_size) does not exist in named cache 'cache_name'.`

  You specified a pool size that does not exist in the cache. Use **sp_cacheconfig** to see cache names and pool sizes.

- `The wash size for the 2k buffer pool in cache tuncache has been incorrectly configured. It must be a minimum of 10 buffers and a maximum of 80 percent of the number of buffers in the pool.`

  You tried to change the wash size for a pool to an illegal size, or you tried to change the pool's size to a size such that the wash size would not be within the valid range.

- `This pool has been marked unavailable. You can remove it completely by retrying sp_poolconfig when the cache is not in use or by restarting the Server.`

  A command to reduce the size of the pool or to drop it resulted in a cache that is smaller than the allowed minimum size.

- `Unable to delete the 'io_size' pool in cache 'cachename'`

  A command to delete a pool could not succeed because some of the pages were in use by active server processes. You can retry the command later, or it will be removed when the server is restarted.

Permissions

Only a System Administrator can execute **sp_poolconfig** to reconfigure memory pools within data caches. All users can use **sp_poolconfig** to get information about memory pools.

Tables Used

master..sysconfigures

See Also

System procedures	**sp_cacheconfig**, **sp_helpcache**, **sp_unbindcache**, **sp_unbindcache_all**

sp_primarykey

Function

Defines a primary key on a table or view.

Syntax

```
sp_primarykey tabname, col1 [, col2, col3, ..., col8]
```

Parameters

tabname – is the name of the table or view on which to define the primary key.

col1 – is the name of the first column that makes up the primary key. The primary key can consist of from one to eight columns.

Examples

1. `sp_primarykey authors, au_id`

   Defines the *au_id* field as the primary key of the table *authors*.

2. `sp_primarykey employees, lastname, firstname`

   Defines the combination of the fields *lastname* and *firstname* as the primary key of the table *employees*.

Comments

- Executing **sp_primarykey** adds the key to the *syskeys* table. Only the owner of a table or view can define its primary key. **sp_primarykey** does not enforce referential integrity constraints; use the **primary key** clause of the **create table** or **alter table** command to enforce a primary key relationship.
- Define keys with **sp_primarykey**, **sp_commonkey**, and **sp_foreignkey** to make explicit a logical relationship that is implicit in your database design. An application program can use the information.
- A table or view can have only one primary key. To display a report on the keys that have been defined, execute **sp_helpkey**.
- The installation process runs **sp_primarykey** on the appropriate columns of the system tables.

Messages

- `New primary key added.`

  You successfully defined a new primary key.

- `The table or view named doesn't exist in the current database.`

  The table or view supplied as the *tabname* parameter does not exist in the current database.

- `Only the owner of the table may define a primary key.`

  You are not the owner of the table or view and, therefore, cannot define its primary key.

- `Primary key already exists on table -- drop key first.`

  A table or view can have only have one primary key. A primary key already exists on the table or view supplied as the *tabname* parameter.

- `Table or view name must be in the current database.`

  You can define a primary key for a table in the current database only.

- `The table has no such` *`nth`* `column.`

  The column name supplied as one of the column names is not a column in *tabname*.

Permissions

Only the owner of the specified table or view can execute **sp_primarykey.**

Tables Used

syscolumns, *syskeys*, *sysobjects*

See Also

Commands	**alter table**, **create table**, **create trigger**
System procedures	**sp_commonkey**, **sp_dropkey**, **sp_foreignkey**, **sp_helpjoins**, **sp_helpkey**

sp_processmail

(Windows NT only)

Function

Reads, processes, sends, and deletes messages in the Adaptive Server message inbox, using the **xp_findnextmsg**, **xp_readmail**, **xp_sendmail**, and **xp_deletemail** system extended stored procedures (ESPs).

Syntax

```
sp_processmail [subject] [, originator [, dbuser
   [, dbname [, filetype [, separator]]]]]
```

Parameters

subject – is the subject header of the message. If you specify a *subject* but not an *originator*, **sp_processmail** processes all unread messages in the inbox that has the specified subject header. If you specify both *subject* and *originator*, **sp_processmail** processes all unread messages with the specified subject header sent by the specified originator. If you do not specify either *subject* or *originator*, **sp_processmail** processes all the unread messages in the Adaptive Server message inbox.

originator – is the sender of an incoming message. If you specify an *originator* and do not specify a *subject*, **sp_processmail** processes all unread messages in the inbox sent by the specified originator.

dbuser – specifies the Adaptive Server login name to use for the user context for executing the query in the message. The default is "guest."

dbname – specifies the database name to use for the database context for executing the query in the message. The default is "master."

filetype – specifies the file extension of the attached file that contains the results of the query. The default is ".txt".

separator – specifies the character to use as a column separator in the query results. It is the same as the **/s** option of **isql**. The default is the tab character.

Examples

1. **sp_processmail @subject="SQL REPORT", @originator="janet", @dbuser="sa", @dbname="salesdb", @filetype="res", @separator=";"**

   Processes all unread messages in the Adaptive Server inbox with the subject header "SQL Report" submitted by mail user "janet", processes the received queries in the *salesdb* database as user "sa", and returns the query results to "janet" in a *.res* file attached to the mail message. The columns in the returned results are separated by semicolons.

2. **sp_processmail @dbuser="sa"**

   Processes all unread messages in the Adaptive Server inbox as user "sa" in the *master* database and returns the query results in *.txt* files, which are attached to the mail messages. The columns in the returned results are separated by tab characters.

Comments

- **sp_processmail** reads, processes, sends, and deletes messages in the Adaptive Server message inbox, using the **xp_findnextmsg**, **xp_readmail**, **xp_sendmail**, and **xp_deletemail** system ESPs.
- **sp_processmail** sends outgoing mail to the originator of the incoming mail message being processed.
- **sp_processmail** uses the default parameters when invoking the ESPs, except for the *dbuser*, *dbname*, *attachname*, and *separator* parameters to **xp_sendmail**, which can be overridden by the parameters to **sp_processmail**.
- **sp_processmail** processes all messages as Adaptive Server queries. It reads messages from the Adaptive Server inbox and returns query results to the sender of the message and all its cc'd and bcc'd recipients in an attachment to an Adaptive Server message. **sp_processmail** generates a name for the attached file consisting of "syb" followed by five random digits, followed by the extension specified by the *filetype* parameter; for example, "syb84840.txt."
- **sp_processmail** deletes messages from the inbox after processing them.
- The *subject* and *originator* parameters specify which messages should be processed. If neither of these parameters is supplied, **sp_processmail** processes all the unread messages in the Adaptive Server message inbox.

- **sp_processmail** does not process attachments to incoming mail. The query must be in the body of the incoming message.

Messages

- `Failed to delete mail. Check the XP Server log file for more information.`
- `Failed to read mail from SQL Server's mailbox. Check the XP Server log file for more information.`
- `Failed to send mail to` *`recipient`*`. Check the XP Server log file for more information.`

Permissions

Only a System Administrator can execute **sp_processmail.**

See Also

System ESPs	**xp_deletemail, xp_findnextmsg, xp_readmail, xp_sendmail, xp_startmail**

sp_procqmode

Function

Displays the query processing mode of a stored procedure, view, or trigger.

Syntax

sp_procqmode [*object_name* [, detail]]

Parameters

object_name – is the name of the stored procedure, view, or trigger whose query processing mode you are examining. If you do not specify an *object_name*, **sp_procqmode** reports on all procedures, views, and triggers in the current database.

detail – returns information about whether the object contains a subquery, and whether there is information about the object in *syscomments*.

Examples

1. sp_procqmode

```
Object Owner.name  Object Type        Processing Mode
-----------------  ----------------   --------------
dbo.au_info        stored procedure   pre-System 11
dbo.titleview      view               System 11 or later
```

Displays the query processing mode for all stored procedures in the current database.

2. sp_procqmode old_sproc, detail

```
Object Owner.Name     Object Type       Processing Mode     Subq Text
--------------------- ----------------- ------------------- ---- ----
dbo.au_info           stored procedure  pre-System 11       no   yes
```

Displays the query processing mode of the stored procedure *old_sproc*, reports whether *old_sproc* contains any subqueries, and reports whether *syscomments* has information about *old_sproc*.

3. sp_procqmode null, detail

Displays detailed reports for all objects in the database.

Comments

- The processing mode identifies whether the object was created in SQL Server release 10.0 or earlier. Objects created on release 10.x

(or earlier) servers are "pre-System 11" objects. Objects created on release 11.0 or later servers are "System 11 or later" objects.

- Subqueries in "pre-System 11" objects use a different processing mode than subqueries in "System 11 or later" objects. Upgrading to release 11.0 or later does not automatically change the processing mode of the subquery.

  In general, the "System 11 or later" processing mode is faster than "pre-System 11" processing mode. To change the processing mode to "System 11 or later", drop and re-create the object. You cannot create an object with "pre-System 11" processing on the current release of Adaptive Server, so you may want to create the object with another name and test it before dropping the version that uses "pre-System 11" processing mode.
- The processing mode displayed for a given object is independent of whether that object actually includes a subquery, and pertains only to the specified object, not to any dependent objects. You must check each object separately.
- The detailed report shows if the object contains a subquery, and reports if text is available in *syscomments* (for **sp_helptext** to report, or for the **defncopy** utility to copy out). **sp_procqmode** does not check that the text in *syscomments* is valid or complete.

Messages

- `Object does not exist in this database.`

  The specified object does not exist in the current database.
- `Object must be a stored procedure, trigger, or view.`

  The specified object does not have a subquery processing mode. Only stored procedures, triggers, and views can have subqueries in their query trees.

Permissions

The database owner or object owner can execute **sp_procqmode** to display the query processing mode of a stored procedure, view, or trigger.

Tables Used

syscomments, *sysobjects*, *sysprocedures*

See Also

Stored Procedures	**sp_helptext**

sp_procxmode

Function

Displays or changes the transaction modes associated with stored procedures.

Syntax

sp_procxmode [*procname* [, *tranmode*]]

Parameters

procname – is the name of the stored procedure whose transaction mode you are examining or changing.

tranmode – is the new transaction mode for the stored procedure. Values are **"chained"**, **"unchained"**, and **"anymode"**.

Examples

1. sp_procxmode

```
procedure name      user name   transaction mode
------------------  ---------   ----------------
byroyalty           dbo         Unchained
discount_proc       dbo         Unchained
history_proc        dbo         Unchained
insert_sales_proc   dbo         Unchained
insert_detail_proc  dbo         Unchained
storeid_proc        dbo         Unchained
storename_proc      dbo         Unchained
title_proc          dbo         Unchained
titleid_proc        dbo         Unchained
```

Displays the transaction mode for all stored procedures in the current database.

2. sp_procxmode byroyalty

```
procedure name                transaction mode
----------------------------  ----------------
byroyalty                     Unchained
```

Displays the transaction mode of the stored procedure *byroyalty.*

3. sp_procxmode byroyalty, "chained"

Changes the transaction mode for the stored procedure *byroyalty* in the *pubs2* database from “unchained” to “chained”.

Comments

- To change the transaction mode of a stored procedure, you must be the owner of the stored procedure, the owner of the database containing the stored procedure, or the System Administrator. The Database Owner or System Administrator can change the mode of another user's stored procedure by qualifying it with the database and user name. For example:

  ```
  sp_procxmode "otherdb.otheruser.newproc", "chained"
  ```

- To use **sp_procxmode**, turn off chained transaction mode using the **chained** option of the **set** command. By default, this option is turned off.
- When you use **sp_procxmode** with no parameters, it reports the transaction modes of every stored procedure in the current database.
- To examine a stored procedure's transaction mode (without changing it), enter:

  ```
  sp_procxmode procname
  ```

- To change a stored procedure's transaction mode, enter:

  ```
  sp_procxmode procname, tranmode
  ```

- When you create a stored procedure, Adaptive Server tags it with the current session's transaction mode. This means:
  - You can execute "chained" stored procedures only in sessions using chained transaction mode.
  - You can execute "unchained" stored procedures only in sessions using unchained transaction mode.

  To execute a particular stored procedure in either chained or unchained sessions, set its transaction mode to "anymode".
- If you attempt to run a stored procedure under the wrong transaction mode, Adaptive Server returns a warning message, but the current transaction, if any, is not affected.

Messages

- `The new transaction-mode must be unchained, chained, or anymode.`

  You specified an invalid mode.
- `The specified object is not a stored procedure in the current database.`

  You specified an invalid object name.

- You must be either the system administrator (SA), the database administrator (dbo), or the owner of this stored procedure to change its transaction mode.

  You do not have the correct permissions.
- You cannot change the mode of a remote stored procedure.

Permissions

Any user can use **sp_procxmode** to display the transaction mode of a stored procedure.

Only a System Administrator, the Database Owner, or the owner of the procedure can change its transaction mode.

Tables Used

sysobjects

See Also

Commands	**begin transaction**, **commit**, **save transaction**, **set**

sp_recompile

Function

Causes each stored procedure and trigger that uses the named table to be recompiled the next time it runs.

Syntax

sp_recompile *objname*

Parameters

objname – is the name of a table in the current database.

Examples

1. sp_recompile titles

Recompiles each trigger and stored procedure that uses the table *titles* the next time the trigger or stored procedure is run.

Comments

- The queries used by stored procedures and triggers are optimized only once, when they are compiled. As you add indexes or make other changes to your database that affect its statistics, your compiled stored procedures and triggers may lose efficiency. By recompiling the stored procedures and triggers that act on a table, you can optimize the queries for maximum efficiency.
- **sp_recompile** looks for *objname* only in the current database and recompiles triggers and stored procedures only in the current database. **sp_recompile** does not affect objects in other databases that depend on the table.
- You cannot use **sp_recompile** on system tables.

Messages

- `Object 'objname' is not a table.`

  The specified object is not a table in the current database.
- `Table or view name must be in current database.`

  You can use **sp_recompile** on objects in the current database only.
- `'objname' is a system table. Cannot use sp_recompile on system tables.`

sp_recompile is allowed only on user tables.

- `You do not own table objname.`

  You can use **sp_recompile** only on tables that you own. If you are a System Administrator, you can run **sp_recompile** on any table.

- `Each stored procedure and trigger that uses table 'objname' will be recompiled the next time it is executed.`

  sp_recompile ran successfully. All stored procedures and triggers in this database that use the named table will be recompiled the next time they are run.

Permissions

Any user can execute **sp_recompile**.

Tables Used

sysobjects

See Also

Commands	**create index**, **update statistics**

sp_remap

Function

Remaps a stored procedure, trigger, rule, default, or view from releases later than 4.8 and prior to 10.0 to be compatible with releases 10.0 and later. Use **sp_remap** on pre-existing objects that the upgrade procedure failed to remap.

Syntax

```
sp_remap objname
```

Parameters

objname – is the name of a stored procedure, trigger, rule, default, or view in the current database.

Examples

1. `sp_remap myproc`

   Remaps a stored procedure called *myproc*.

2. `sp_remap "my_db..default_date"`

   Remaps a rule called *default_date*. Execute a **use my_db** statement to open the *my_db* database before running this procedure.

Comments

- If **sp_remap** fails to remap an object, drop the object from the database and re-create it. Before running **sp_remap** on an object, it is a good idea to copy its definition into an operating system file with the **defncopy** utility. For more information about **defncopy**, see the *Utility Programs* manual for your platform.
- **sp_remap** can cause your transaction log to fill rapidly. Before running **sp_remap**, use the **dump transaction** command to dump the transaction log, as needed.
- You can use **sp_remap** only on objects in the current database.
- **sp_remap** makes no changes to objects that were successfully upgraded to the current release.

Messages

- `Object does not exist in this database.`

  The object you tried to remap does not exist in the current database. Either qualify **sp_remap** with the database name or issue the **use** *database_name* command to open the correct database, and then reexecute **sp_remap**.

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

  sp_remap executed the **remap** option of the **dbcc** command.

- `You do not own object` *`objname`*`.`

  Only the owner of an object can remap it.

- `Remapping utility - procedure is corrupted in Sysprocedures. Recreate this procedure.`

  sp_remap cannot remap this object. Drop the object from the database and re-create it.

- `Remapping utility - a pointer exists in a tree when it should not.`

  sp_remap cannot remap this object. Drop the object from the database and re-create it.

- `Remapping utility - unable to locate the given procedure` *`procedure_name`* `in Sysprocedures.`

  sp_remap cannot remap this object. Drop the object from the database and re-create it.

- `Remapping utility -- Procedure needs to be recreated for this port.`

  sp_remap cannot remap this object. Drop the object from the database and re-create it.

Permissions

Only a System Administrator or the owner of an object can remap the object with **sp_remap**.

Tables Used

master.dbo.sysdatabases, sysobjects

See Also

Commands	**create default**, **create procedure**, **create rule**, **create trigger**, **create view**, **drop default**, **drop procedure**, **drop rule**, **drop trigger**, **drop view**, **dump transaction**
System procedures	**sp_helptext**
Utility programs	**defncopy**

sp_remoteoption

Function

Displays or changes remote login options.

Syntax

```
sp_remoteoption [remoteserver [, loginame
   [, remotename [, optname [, optvalue]]]]]
```

Parameters

remoteserver – is the name of the server that will be executing RPCs on this server.

➤ ***Note***

This manual page uses the term "local server" to refer to the server that is executing the remote procedures that are run from a "remote server".

loginame – is the login name that identifies the local login for the *remoteserver*, *loginame*, *remotename* combination.

remotename – is the remote user name that identifies the remote login for the *remoteserver*, *loginame*, *remotename* combination.

optname – is the name of the option to change. Currently, there is only one option, **trusted**, which means that the local server accepts remote logins from other servers without user-access verification for the particular remote login. The default is to use password verification. Adaptive Server understands any unique string that is part of the option name. Use quotes around the option name if it includes embedded blanks.

optvalue – is either **true** or **false. true** turns the option on, **false** turns it off.

Examples

1. sp_remoteoption

```
Settable remote login options.
remotelogin_option
------------------------
trusted
```

Displays a list of the remote login options.

2. **sp_remoteoption GATEWAY, churchy, pogo, trusted, true**

   Defines the remote login from the remote server GATEWAY to be **trusted** (that is, the password is not checked).

3. **sp_remoteoption GATEWAY, churchy, pogo, trusted, false**

   Defines the remote login "pogo" from the remote server GATEWAY as a login that is not trusted (that is, the password is checked).

4. **sp_remoteoption GATEWAY, albert, NULL, trusted, true**

   Defines all logins from GATEWAY that map to login "albert" on the local server to be trusted.

Comments

- To display a list of the remote login options, execute **sp_remoteoption** with no parameters.
- If you have used **sp_addremotelogin** to map all users from a remote server to the same local name, specify **trusted** for those users. For example, if all users from server GOODSRV that are mapped to "albert" are trusted, specify:

  ```
  sp_remoteoption GOODSRV, albert, NULL, trusted
    true
  ```

  If the logins are not specified as **trusted**, they cannot execute RPCs on the local server unless they specify local server passwords when they log into the remote server. When they use Open Client Client-Library, users can specify a password for server-to-server connections with the routine **ct_remote_pwd**. **isql** and **bcp** do not permit users to specify a password for RPC connections.

  If users are logged into the remote server using "unified login", the logins must also be trusted on the local server, or they must specify passwords for the server when they log into the remote server.

  For more information about setting up servers for remote procedure calls and for using "unified login", see the *Security Administration Guide*.

Messages

- `Option 'optname' turned off.`

  The procedure was successful.

- `Option 'optname' turned on.`

  The procedure was successful.

- `Remote login option doesn't exist or can't be set by user.`
  `Run sp_remoteoption with no parameters to see options.`

  Either the option does not exist or you do not have permission to turn it on or off.

- `Remote login option is not unique.`

  The name you supplied as the *optname* parameter is not unique. The remote login option value was not changed. The complete names that match the string you supplied are displayed after this message.

- `Settable remote login options.`

  When you execute **sp_remoteoption** with no parameters, this message appears, followed by a list of available options. (See example 1.)

- `There is no remote user 'remotename' mapped to local user 'loginame' on remote server 'remoteserver'.`

  You incorrectly identified the remote login or the remote server name. Use **sp_helpremotelogin** to list the remote logins. Use **sp_helpserver** to list the remote servers.

- `Usage: sp_remoteoption [remoteserver, loginame, remotename, optname, {true | false}]`

  Either the *optname* parameter was omitted or the *optvalue* parameter was not **true** or **false**.

Permissions

Only the System Security Officer can execute **sp_remoteoption**.

Tables Used

master.dbo.spt_values, *master.dbo.sysmessages*, *master.dbo.sysremotelogins*, *master.dbo.sysservers*, *sysobjects*

See Also

System procedures	**sp_addremotelogin**, **sp_dropremotelogin**, **sp_helpremotelogin**

sp_remotesql

(Component Integration Services only)

Function

Establishes a connection to a remote server, passes a query buffer to the remote server from the client, and relays the results back to the client.

Syntax

```
sp_remotesql server, query
   [, query2, ... , query254]
```

Parameters

server_name – is the name of a remote server defined with **sp_addserver**.

query – is a query buffer a with maximum length of 255 characters.

query2 through *query254* – is a query buffer with a maximum length of 255 characters. If supplied, these arguments are concatenated with the contents of *query1* into a single query buffer.

Examples

1.
```
sp_remotesql FREDS_SERVER, "select @@version"
```

   Passes the query buffer to FREDS_SERVER, which interprets **select @@version** and returns the result to the client. Adaptive Server does not interpret the result.

2.
```
create procedure freds_version
as
exec sp_remotesql FREDS_SERVER, "select @@version"
go
exec freds_version
go
```

   Illustrates the use of **sp_remotesql** in a stored procedure. This example and example 1 return the same information to the client.

```
3. sp_remotesql DCO_SERVER,
   "insert into remote_table
   (numbercol,intcol, floatcol,datecol )",
   "values (109.26,75, 100E5,'10-AUG-85')"
   select @@error
```

The server concatenates two query buffers into a single buffer, and passes the complete **insert** statement to the server DCO_SERVER for processing. The syntax for the **insert** statement is a format that DCO_SERVER understands. The returned information is not interpreted by the server. This example also examines the value returned in *@@error*.

```
4. declare @servname varchar(30)
   declare @querybuf varchar(200)
   select  @servname = "DCO_SERV"
   select  @querybuf = "select table_name
        from all_tables
        where owner = 'SYS'"
   exec sp_remotesql @servname, @querybuf
```

Illustrates the use of local variables as parameters to **sp_remotesql**.

Comments

- **sp_remotesql** establishes a connection to a remote server, passes a query buffer to the remote server from the client, and relays the results back to the client. The local server does not intercept results.
- You can use **sp_remotesql** within another stored procedure.
- The query buffer parameters must be a character expression with a maximum length of 255 characters. If you use a query buffer that is not *char* or *varchar*, you will receive datatype conversion errors.
- **sp_remotesql** sets the global variable *@@error* to the value of the last error message returned from the remote server if the severity of the message is greater than 10.
- If **sp_remotesql** is issued from within a transaction, Adaptive Server verifies that a transaction has been started on the remote server before passing the query buffer for execution. When the transaction terminates, the remote server is directed to commit the transaction. The work performed by the contents of the query buffer is part of the unit of work defined by the transaction.

  If transaction control statements are part of the query buffer, it is the responsibility of the client to ensure that the transaction

commit and **rollback** occur as expected. Mixing Transact-SQL with transaction control commands in the query buffer can cause unpredictable results.

- The local server manages the connection to the remote server. Embedding **connect to** or **disconnect** commands in the query buffer causes results that require interpretation by the remote server. This is not required or recommended. Typically, the result is a syntax error.

Messages

- `Site server not found in sysservers.`

  The server *server* is not defined. Add the server with **sp_addserver** before naming the server in an **sp_remotesql** procedure.

- `Procedure sp_remotesql expects parameter @parm_name, which was not supplied.`

  Either the *query_buf1* parameter was not supplied or the *server_name* and *query_buf1* parameters were not supplied. These are required.

- `Parameter cannot be NULL.`

  Either the *server* or the *query* parameter was NULL. You must specify a server name and query buffer.

Permissions

Any user can execute **sp_remotesql**.

Tables Used

No tables are used.

See Also

Commands	**connect to...disconnect**
System procedures	**sp_autoconnect**, **sp_passthru**

sp_rename

Function

Changes the name of a user-created object or user-defined datatype in the current database.

Syntax

sp_rename *objname, newname*

Parameters

objname – is the original name of the user-created object (table, view, column, stored procedure, index, trigger, default, rule, check constraint, referential constraint, or user-defined datatype). If the object to be renamed is a column in a table, *objname* must be in the form "*table.column*". If the object is an index, *objname* must be in the form "*table.indexname*".

newname – is the new name of the object or datatype. The name must conform to the rules for identifiers and must be unique to the current database.

Examples

1. **sp_rename titles, books**

   Renames the *titles* table to *books*.

2. **sp_rename "books.title", bookname**

   Renames the *title* column in the *books* table to *bookname*.

3. **sp_rename "books.titleind", titleindex**

   Renames the *titleind* index in the *books* table to *titleindex*.

4. **sp_rename tid, bookid**

   Renames the user-defined datatype *tid* to *bookid*.

Comments

- **sp_rename** changes the name of a user-created object or datatype. You can change only the name of an object or datatype in the database in which you issue **sp_rename**.
- When you are renaming a column or index, do not specify the table namein the *newname*. See examples 2 and 3.

- You can change the name of a an object referenced by a view. For example, if a view references the *new_sales* table and you rename *new_sales* to *old_sales*, the view will reference *old_sales*.
- You cannot change the names of system objects and system datatypes.

◆ ***WARNING!***

Procedures, triggers, and views that depend on an object whose name has been changed work until they are dropped and re-created. Also, the old object name appears in query results until the user changes and re-creates the procedure, trigger, or view. Change the definitions of any dependent objects when you execute sp_rename. Find dependent objects with sp_depends.

Messages

- `Can't run sp_rename from within a transaction.`

  sp_rename modifies system tables, so it cannot be run within a transaction.
- `Column name has been changed.`

  The specified column name was renamed to *newname*.
- `Index name has been changed.`

  The specified index name was renamed to *newname*.
- `Name of user-defined type name changed.`

  The specified user-defined datatype was renamed to *newname*.
- *`Newname`* `already exists in sysobjects.`

  An object named *newname* already exists. Object names must be unique to the database.
- *`Newname`* `already exists in systypes.`

  A datatype named *newname* already exists. Datatype names must be unique to the database.
- *`newname`* `is not a valid name.`

  newname does not conform to the rules for identifiers.
- `Object must be in the current database.`

The name supplied for the *objname* parameter included a reference to a database. The object must be in the current database.

- `Object name cannot be changed either because it does not exist in this database, or you don't own it, or it is a system name.`

  No object of the specified name exists, or you do not own the object.

- `Object name has been changed.`

  The specified object was renamed to *newname*.

- `There is already a column named '`*`newname`*`' in table '`*`tablename`*`'.`

  Column names must be unique within a table. The table already contains a column with the name you chose.

- `Table or view names beginning with '#' are not allowed.`

  You cannot begin the name of a table or view with "#".

- `There is already an index named '`*`newname`*`' for table '`*`tablename`*`'.`

  Index names for a table must be unique. The table already has an index with the name you chose.

- `You do not own a table, column or index of that name in the current database.`

  No column of the specified name exists in the specified table, or you do not own the table.

Permissions

Users can execute **sp_rename** to rename their own objects only. Database Owners and System Administrators can also rename only the objects that they own. However, they can use the **setuser** command to assume another database user's identity.

Tables Used

syscolumns, *sysindexes*, *sysobjects*, *systypes*

See Also

Commands	**alter table**, **create default**, **create procedure**, **create rule**, **create table**, **create trigger**, **create view**
System procedures	**sp_addtype**, **sp_checkreswords**, **sp_depends**, **sp_renamedb**

sp_renamedb

Function

Changes the name of a user database.

Syntax

sp_renamedb *dbname, newname*

Parameters

dbname – is the original name of the database.

newname – is the new name of the database. Database names must conform to the rules for identifiers and must be unique.

Examples

1. **sp_renamedb accounting, financial**

   Renames the *accounting* database to *financial*.

2. ```
   sp_dboption work, single, true
   go
   use work
   go
   checkpoint
   go
   sp_renamedb work, workdb
   go
   use master
   go
   sp_dboption workdb, single, false
   go
   use workdb
   go
   checkpoint
   go
   ```

   Renames the database named *work*, which is a Transact-SQL reserved word, to *workdb*.

Comments

- **sp_renamedb** changes the name of a database. You **cannot** rename system databases or databases with external referential integrity constraints.
   ```

- The System Administrator must place a database in single-user mode with **sp_dboption** before renaming it and must restore it to multi-user mode afterward.
- **sp_renamedb** fails if any table in the database references, or is referenced by, a table in another database. Use the following query to determine which tables and external databases have foreign key constraints on primary key tables in the current database:

  ```
  select object_name(tableid), db_name(frgndbname)
  from sysreferences
  where frgndbname is not null
  ```

  Use the following query to determine which tables and external databases have primary key constraints for foreign key tables in the current database:

  ```
  select object_name(reftabid), db_name(pmrydbname)
  from sysreferences
  where pmrydbname is not null
  ```

  Use **alter table** to drop the cross-database constraints in these tables. Then, rerun **sp_renamedb**.
- When you change a database name:
  - Drop all stored procedures, triggers, and views that include the database name
  - Change the source text of the dropped objects to reflect the new database name
  - Re-create the dropped objects
  - Change all applications and SQL source scripts that reference the database, either in a **use** *database_name* command or as part of a fully qualified identifier (in the form *dbname*.[*owner*].*objectname*).
- If you use scripts to run **dbcc** commands or **dump database** and **dump transaction** commands on your databases, be sure to update those scripts.

◆ ***WARNING!***

Procedures, triggers, and views that depend on a database whose name has been changed work until they are re-created. Change the definitions of any dependent objects when you execute sp_renamedb. Find dependent objects with sp_depends.

Messages

- `A database with the new name already exists.`

  The database you specified for the *newname* parameter is already a database. Database names must be unique.

- `Can't run sp_renamedb from within a transaction.`

  sp_renamedb modifies system tables, so it cannot be run within a transaction.

- `Database '`*`database_name`*`' has references to other databases. Drop those references and try again.`

  You cannot rename a database if any of its tables reference, or are referenced by, a table in another database. Before renaming the database, you must use **alter table** to drop any external referential integrity constraints.

- `Database is renamed and in single-user mode. System Administrator (SA) must reset it to multi-user mode with sp_dboption.`

  sp_renamedb succeeded.

- *`newname`* `is not a valid name.`

  The value for *newname* does not conform to the rules for identifiers.

- `The databases 'master', 'model', and 'tempdb' cannot be renamed.`

  You cannot rename system databases.

- `The specified database does not exist.`

  The database you specified with the *dbname* parameter does not exist.

- `System Administrator (SA) must set database '`*`dbname`*`' to single-user mode with sp_dboption before using sp_renamedb.`

  You cannot rename a database while someone is using it, and you must make sure that no one tries to use the database while it is being renamed.

Permissions

Only a System Administrator can execute **sp_renamedb**.

Tables Used

master.dbo.spt_values, *master.dbo.sysdatabases*, *sysobjects*

See Also

Commands	**create database**
System procedures	**sp_changedbowner**, **sp_dboption**, **sp_depends**, **sp_helpdb**, **sp_rename**

sp_reportstats

Function

Reports statistics on system usage.

Syntax

sp_reportstats [*loginame*]

Parameters

loginame – is the login name of the user to show accounting totals for.

Examples

1. sp_reportstats

```
Name    Since         CPU     Percent CPU  I/O     Percent I/O
------  -----------   -----   ------------ -----   -------------
julie   jun 19 1993   10000   24.9962%     5000    24.325%
jason   jun 19 1993   10002   25.0013%     5321    25.8866%
ken     jun 19 1993   10001   24.9987%     5123    24.9234%
kathy   jun 19 1993   10003   25.0038%     5111    24.865%

      Total CPU   Total I/O
      ---------   ---------
      40006       20555
```

Displays a report of current accounting totals for all Adaptive Server users.

2. sp_reportstats kathy

```
Name    Since         CPU     Percent CPU   I/O     Percent I/O
------  -----------   -----   ------------  -----   -------------
kathy   Jul 24 1993   498     49.8998%      48392   9.1829%

      Total CPU   Total I/O
      ---------   ----------
      998         98392
```

Displays a report of current accounting totals for user “kathy.”

Comments

- **sp_reportstats** prints out the current accounting totals for all logins, as well as each login’s individual statistics and percentage of the overall statistics. **sp_reportstats** accepts one parameter, the login name of the account to report. With no parameters, **sp_reportstats** reports on all accounts.

- **sp_reportstats** does not report statistics for any process with a system user ID (*suid*) of 0 or 1. This includes deadlock detection, checkpoint, houskeeper, network, auditing, mirror handlers, and all users with **sa_role**.
- The units reported for "CPU" are **machine** clock ticks, not Adaptive Server clock ticks.
- The "probe" user exists for the two-phase commit probe process, which uses a challenge-and-response mechanism to access Adaptive Server.

Messages

- `No login with the specified name exists.`

  Check the spelling of the user's name.

Permissions

Only a System Administrator can execute **sp_reportstats**.

Tables Used

master.dbo.syslogins, *sysobjects*

See Also

System procedures	**sp_clearstats, sp_configure**

sp_revokelogin

(Windows NT only)

Function

Revokes Adaptive Server roles and default permissions from Windows NT users and groups when Integrated Security mode or Mixed mode (with Named Pipes) is active.

Syntax

```
sp_revokelogin {login_name | group_name}
```

Parameters

login_name – is the network login name of the Windows NT user.

group_name – is the Windows NT group name.

Examples

1. `sp_revokelogin jeanluc`

   Revokes all permissions from the Windows NT user named "jeanluc".

2. `sp_revokelogin Administrators`

   Revokes all roles from the Windows NT Administrators group.

Comments

- Use **sp_revokelogin** only when Adaptive Server is running in Integrated Security mode or Mixed mode, when the connection is Named Pipes. If Adaptive Server is running in Standard mode, or in Mixed mode using a connection other than Named Pipes, use the **revoke** command.
- If you revoke a user's roles and default privileges with **sp_revokelogin**, that user can no longer log into Adaptive Server over a trusted connection.

Messages

- `Access revoked.`

  sp_revokelogin successfully executed.

- `'login_name' is not a valid account name.`

  The specified Windows NT user name or group does not exist.

- `No privilege to revoke.`

  The specified *login_name* or *group_name* has no privileges.

- `Unable to get SQL Server security information.`

  A call to the Windows NT security API failed. Contact your Windows NT administrator.

- `Unable to set SQL Server security information.`

  A call to the Windows NT security API failed. Contact your Windows NT administrator.

- `There must be at least one account with 'sso_role' privilege.`

  `There must be at least one account with 'sso_role' privilege other than the 'LocalSystem'.`

  You cannot revoke the last login that has **sso_role**. Adaptive Server requires a System Security Officer to manage security-sensitive tasks.

- `SQL Server's account cannot be modified.`

  Adaptive Server itself requires account privileges in order to manage login security. You cannot revoke privileges from this account.

Permissions

Only a System Administrator can use **sp_revokelogin**.

Tables Used

sysobjects

See Also

Commands	**grant**, **revoke**, **setuser**
System procedures	**sp_droplogin**, **sp_dropuser**, **sp_logininfo**

sp_role

Function

Grants or revokes roles to an Adaptive Server login account.

Syntax

```
sp_role {"grant" | "revoke"}, rolename, loginame
```

Parameters

grant | **revoke** – specifies whether to grant the role to or revoke the role from *loginame*.

rolename – is the role to be granted or revoked.

loginame – is the login account to or from which the role is to be granted or revoked.

Examples

```
1. sp_role "grant", sa_role, alexander
```

Grants the System Administrator role to the login account named “alexander”.

Comments

- **sp_role** grants or revokes roles to an Adaptive Server login account.
- When you grant a role to a user, it takes effect the next time the user logs into Adaptive Server. Alternatively, the user can enable the role immediately by using the **set role** command. For example, the command:

  ```
  set role sa_role on
  ```

  enables the System Administrator role for the user.
- You cannot revoke a role from a user while the user is logged in.
- When users log in, all roles that have been granted to them are active (**on**). To turn a role off, use the **set** command. For example, to deactivate the System Administrator role, use the command:

  ```
  set role "sa_role" off
  ```

Messages

- `Can't run sp_role from within a transaction.`

  sp_role modifies system tables, so it cannot be run within a transaction.

- `Neither 'grant' nor 'revoke' is specified -- nothing changed.`

  Specify either **grant** or **revoke**.

- `Role updated.`

  sp_role successfully executed.

Permissions

Only a System Administrator can grant the System Administrator role to other users. Only a System Security Officer can grant any role other than SA to other users.

Tables Used

master.dbo.sysloginroles, master.dbo.syslogins, master.dbo.sysprocesses, master.dbo.syssrvroles, sysobjects

See Also

Commands	**grant**, **revoke**, **set**
Functions	**proc_role**
System procedures	**sp_displaylogin**

sp_serveroption

Function

Displays or changes remote server options.

Syntax

```
sp_serveroption [server, optname, optvalue]
```

Parameters

server – is the name of the remote server for which to set the option.

optname – is the name of the option to be set or unset. Table 3-19 lists the option names.

Table 3-19: sp_serveroption options

Option	Meaning
net password encryption	Specifies whether to initiate connections with a remote server with the client side password encryption handshake or with the normal (unencrypted password) handshake sequence. The default is **false**, no network encryption.
readonly	Specifies that access to the server named is read only. This option is available only with Component Integration Services.
rpc security model A	The default model for handling RPCs. This model does not support mutual authentication, message integrity, or message confidentiality between the local server and the remote server.
timeouts	When unset (**false**), disables the normal timeout code used by the local server, so the site connection handler does not automatically drop the physical connection after one minute with no logical connection. The default is **true**.

Adaptive Server accepts any unique string that is part of the option name. Use quotes around the option name if it includes embedded blanks.

optvalue – is **true** (on) or **false** (off).

Examples

1. **sp_serveroption**

   ```
   Settable server options.

    ------------------------
   net password encryption
   readonly
   rpc security model A
   timeouts
   timeouts
   net password encryption
   ```

   Displays a list of the server options.

2. **sp_serveroption GATEWAY, "timeouts", false**

   Tells the server not to time out inactive physical connections with the remote server GATEWAY.

3. **sp_serveroption GATEWAY,**
   "net password encryption", true

   Specifies that when connecting to the remote server GATEWAY, GATEWAY sends back an encryption key to encrypt the password to send to it.

Comments

- To display a list of server options that can be set by the user, use **sp_serveroption** with no parameters.
- Once **timeouts** is set to **false**, the site handlers will continue to run until one of the two servers is shut down.
- The **net password encryption** option allows clients to specify whether to send passwords in plain text or encrypted form over the network when initiating a remote procedure call. If **net password encryption** is **true**, the initial login packet is sent without passwords, and the client indicates to the remote server that encryption is desired. The remote server sends back an encryption key, which the client uses to encrypt its passwords. The client then encrypts its passwords, and the remote server uses the key to authenticate them when they arrive.
- To set network password encryption for a particular **isql** session, you can use a command line option for **isql**. See the *Utility Programs* manual for your platform for more information.
- You cannot use the **net password encryption** option when connecting to a pre-release 10.0 SQL Server.

- See Chapter 7, "Managing Remote Servers," in the *Security Administration Guide* for additional details on server options.

Messages

- `Can't run sp_serveroption from within a transaction.`

  sp_serveroption modifies system tables, so it cannot be run within a transaction.

- `No such server -- run sp_helpserver to list servers.`

  You specified an incorrect server name. Run **sp_helpserver** to get a list of servers.

- `Option can be set for remote servers only -- not the local server.`

  You tried to set an option on the local server.

- `Server option doesn't exist or can't be set by user.`
  `Run sp_serveroption with no parameters to see options.`

  Either the option does not exist or you do not have permission to set or unset it. Run **sp_serveroption** with no parameters to display a list of options that can be set by the user.

- `Server option is not unique.`

  The name supplied as the *optname* parameter is not unique. No server option value was changed.

- `You must have the following role(s) to execute this command/procedure: role_name. Please contact a user with the appropriate role for help.`

  A particular role is required to set the server option. For example, **sa_role** is required for the **timeouts** option.

Permissions

Any user can execute **sp_serveroption** with no parameters to display a list of options. Only a System Administrator can set the **timeouts** option.

Tables Used

master.dbo.sysservers, sysobjects, syssecmechs

See Also

System procedures	**sp_helpserver**, **sp_password**

sp_setlangalias

Function

Assigns or changes the alias for an alternate language.

Syntax

```
sp_setlangalias language, alias
```

Parameters

language – is the official language name of the alternate language.

alias – is the new local alias for the alternate language.

Examples

1. `sp_setlangalias french, français`

   This command assigns the alias name “français” for the official language name “french”.

Comments

- *alias* replaces the current value of *syslanguages.alias* for the official name.
- The **set language** command can use the new *alias* in place of the official language name.

Messages

- *language* `is not an official language name from syslanguages.`

  Use **sp_helplanguage** to see a list of official names of alternate languages on this Adaptive Server.

- *alias* `alias already exists in syslanguages.`

  The new *alias* must be unique. Use **sp_helplanguage** to see a list of official names and aliases available on this Adaptive Server.

- `Language alias not changed.`

  An error occurred in the updating of *master.dbo.syslanguages*, so the alias was not added. The Adaptive Server message that appeared before this message provides more information.

- `Language alias reset.`

  The alias for this alternate language name was changed.

Permissions

Only a System Administrator can execute **sp_setlangalias.**

Tables Used

master.dbo.syslanguages, sysobjects

See Also

Commands	**set**
System procedures	**sp_addlanguage**, **sp_droplanguage**, **sp_helplanguage**

sp_setpglockpromote

Function

Sets or changes the lock promotion thresholds for a database, for a table, or for Adaptive Server.

Syntax

```
sp_setpglockpromote {"database" | "table"}, objname,
   new_lwm, new_hwm, new_pct
sp_setpglockpromote server, NULL, new_lwm, new_hwm,
   new_pct
```

Parameters

server – sets server-wide values for the lock promotion thresholds.

"database" | **"table"** – specifies whether to set the lock promotion thresholds for a database or table. "database" and "table" are Transact-SQL keywords, so the quotes are required.

objname – is either the name of the table or database for which you are setting the lock promotion thresholds or **null**, if you are setting server-wide values.

new_lwm – specifies the value to set for the low watermark (LWM) threshold. The LWM must be less than or equal to the high watermark (HWM). The minimum value for LWM is 2. This parameter can be **null**.

new_hwm – specifies the value to set for the lock promotion HWM threshold. The HWM must be greater than or equal to the LWM. The maximum HWM is 2,147,483,647. This parameter can be **null**.

new_pct – specifies the value to set for the lock promotion percentage (PCT) threshold. PCT must be between 1 and 100. This parameter can be **null**.

Examples

1. `sp_setpglockpromote "server", NULL, 200, 300, 50`

Sets the server-wide lock promotion LWM to 200, the HWM to 300, and the PCT to 50.

2. **sp_setpglockpromote "database", master, 1000, 1100, 45**

   Sets lock promotion thresholds for the *master* database.

3. **sp_setpglockpromote "table", "pubs2..titles", 500, 700, 10**

   Sets lock promotion thresholds for the *titles* table in the *pubs2* database. This command must be issued from the *pubs2* database.

4. **sp_setpglockpromote "database", master, @new_hwm=1600**

   Changes the HWM threshold to 160 for the *master* database. The thresholds were previously set with **sp_setpglockpromote**. This command must be issued from the *master* database.

Comments

- **sp_setpglockpromote** configures the lock promotion values for a table, for a database, or for Adaptive Server.

  Adaptive Server acquires page locks on a table until the number of locks exceeds the lock promotion threshold. **sp_setpglockpromote** changes the lock promotion thresholds for an object, a database, or the server. If Adaptive Server is successful in acquiring a table lock, the page locks are released.

  When the number of locks on a table exceeds the HWM threshold, Adaptive Server attempts to escalate to a table lock. When the number of locks on a table is below the LWM, Adaptive Server does not attempt to escalate to a table lock. When the number of locks on a table is between the HWM and LWM and the number of locks exceeds the PCT threshold, Adaptive Server attempts to escalate to a table lock.

- Lock promotion thresholds for a table override the database or server-wide settings. Lock promotion thresholds for a database override the server-wide settings.

- Lock promotion thresholds for Adaptive Server do not need initialization, but you must initialize database and table lock promotion thresholds by specifying LWM, HWM, and PCT with **sp_setpglockpromote**, which creates a row for the object in *sysattributes* when it is first run for a database or table. Once the thresholds have been initialized, then they can be modified individually, as in example 4.

- For a table or a database, **sp_setpglockpromote** sets LWM, HWM, and PCT in a single transaction. If **sp_setpglockpromote** encounters an error while updating any of the values, then all changes are aborted and the transaction is rolled back. For server-wide changes, one or more thresholds may fail to be updated while others are successfully updated. Adaptive Server returns an error message if any values fail to be updated.
- To view the server-wide settings for the lock promotion thresholds, use **sp_configure "lock promotion"** to see all three threshold values. To view lock promotion settings for a database, use **sp_helpdb**. To view lock promotion settings for a table, use **sp_help**.

Messages

- `Can't run sp_setpglockpromote from within a transaction.`

  sp_setpglockpromote updates system tables, so it cannot be run from within a transaction.

- `No such database -- run sp_helpdb to list databases.`

  The database does not exist. Check the spelling.

- `Object must be in the current database.`

  sp_setpglockpromote can configure lock promotion values for tables in the current database only. Either qualify **sp_setpglockpromote** with the database name or issue the **use** *database_name* command to open the database in which the table resides, and issue **sp_setpglockpromote** again.

- `The target object does not exist.`

  The table specified with the *objname* parameter does not exist in the current database, or the database does not exist.

- `At least one of the parameters 'new_lwm', 'new_hwm' or 'new_pct' must be non-NULL to execute sp_setpglockpromote.`

  Specify a value for one of the thresholds.

- `You must be in the 'master' database to add, change or drop lock promotion attribute for a user database.`

  To set lock promotion values for a database, issue **sp_setpglockpromote** from the *master* database.

- `You need to specify a non-NULL value for` *`value`*`, since it has not been set previously with a non-NULL value.`

  Reissue **sp_setpglockpromote** with values for HWM, LWM, and PCT to initialize the values.

- `Object name parameter must be NULL for Server-wide lock promotion attributes. Using NULL instead of` *`objname`*`.`
- *`'objname'`* `is a system table. This stored procedure cannot be used on system tables.`

  You can create lock promotion thresholds for user tables only.

- `LWM =` *`value`* `cannot be greater than HWM` *`value`*`.`

  You specified a lock promotion HWM that is lower than the current LWM or a LWM that is greater than the current HWM. Check the values and reissue **sp_setpglockpromote**.

- `Invalid value specified for 'scope' parameter. Valid values are 'SERVER','DATABASE' or 'TABLE'.`

  sp_setpglockpromote sets lock promotion values for a database, for a table, or for Adaptive Server.

- `'lock promotion' attributes of` *`objname parameter`* `have been changed. The new values are` *`value`*`.`

  sp_setpglockpromote succeeded.

Permissions

Only a System Administrator can execute **sp_setpglockpromote**.

Tables Used

master.dbo.sysattributes, master.dbo.sysconfigures, sysattributes

See Also

System procedures	**sp_configure**, **sp_dropglockpromote**, **sp_help**, **sp_helpdb**

sp_setpsexe

Function

Sets custom execution attributes "on the fly" for a session.

Syntax

```
sp_setpsexe spid, exeattr, value
```

Parameters

spid – is the ID of the session for which to set execution variables. Use **sp_who** to see *spid*s.

exeattr – identifies the execution attribute to be set. Values are **priority** and **enginegroup**.

value – is the new value of *exeattr*. Values for each attribute are as follows:

- If *exeattr* is **priority**, *value* is **HIGH**, **MEDIUM**, or **LOW**.
- If *exeattr* is **enginegroup**, *value* is the name of an existing engine group.

Examples

```
1. sp_setpsexe 1, "priority", "HIGH"
```

This statement sets the priority of the process with an ID of 1 to HIGH.

Comments

- Execution attribute values specified with **sp_setpsexe** are valid for the current session only and do not apply after the session terminates.
- Use **sp_setpsexe** with caution or it can result in degraded performance. Changing attributes "on the fly", using **sp_setpsexe**, can help if the process is not getting CPU time; however, if the performance problem is due to something else, such as locks, changing execution attributes could make the problem worse.
- Because you can only set execution attributes for sessions, **sp_setpsexe** cannot be set for a worker process *spid*.
- Except for the housekeeper *spid*, you cannot set execution attributes for system *spids*.

- **sp_setpsexe** does not work if there are no online engines in the associated engine group.

Messages

- `A non-SA user cannot modify attributes of another task.`

  Only users with System Administrator permissions can set these attributes.

- `A non-SA user can only lower its own priority value.`

  Users without System Administrator permissions can only lower the priority value.

- `Can't run sp_setpsexe from within a transaction.`

  sp_setpsexe modifies system tables, so it cannot be run within a transaction.

- *`exeattr`* `is not a valid execution attribute.`

  The attribute name is not valid.

- *`exeattr`* `value` *`value`* `is not valid.`

  The value of the execution attribute you supplied is not valid.

- `Failed to set attribute` *`exeattr`* `to` *`value`* `for spid` *`spid`*`.Check server errorlog for any additional information.`

  The execution attribute value could not be set for the specified Adaptive Server process. Please check the error log.

- `No SQL Server process with the specified ID exists.`

  The process ID specified either does not exist or is invalid for the operation; for example, a *spid* for a worker process. To see a list of valid *spid*s, run **sp_who**.

Permissions

All users can execute **sp_setpsexe** to lower the priority of a process that user owns. Only a System Administrator can execute **sp_setpsexe** without restriction.

Tables Used

sysattributes, *sysprocesses*

See Also

System procedures	**sp_addengine, sp_addexeclass, sp_bindexeclass, sp_clearpsexe, sp_dropengine, sp_dropexeclass, sp_showcontrolinfo, sp_showexeclass, sp_showpsexe**

sp_setsuspect_granularity

Function

Displays or sets the recovery fault isolation mode for a user database, which governs how recovery behaves when it detects data corruption.

Syntax

```
sp_setsuspect_granularity [dbname
   [, "database" | "page" [, "read_only"]]]
```

Parameters

dbname – is the name of the database for which to display or set the recovery fault isolation mode. For displaying, the default is the current database. For setting, you must be in the *master* database and specify the target *dbname*.

database – marks the entire database suspect, which makes it inaccessible, if the recovery process detects that any of its data is suspect.

page – marks only the corrupt pages suspect, making them inaccessible, if recovery detects corrupt data in the database. The rest of the data is accessible.

read_only –if specified, marks the entire database **read only** if recovery marks any pages suspect.

Examples

1. sp_setsuspect_granularity

```
DB Name  Cur. Suspect Gran.  Cfg. Suspect Gran.  Online mode
-------  -----------------  -----------------  -----------
pubs2    database           database           read/write
```

Displays the recovery fault isolation mode for the current database.

2. sp_setsuspect_granularity pubs2

Displays the current and configured recovery fault isolation mode for the *pubs2* database.

3. **sp_setsuspect_granularity pubs2, "page"**

```
DB Name       Cur. Suspect Gran. Cfg. Suspect Gran.
------------- ----------------- -----------------
pubs2          database             database
sp_setsuspect_granularity: The new values will become effective
      during the next recovery of the database 'pubs2'.
```

The next time recovery runs in the *pubs2* database, if any corrupt pages are detected, only the suspect pages will be taken offline and the rest of the database will be brought online.

4. **sp_setsuspect_granularity pubs2, "page", "read_only"**

The next time recovery runs in the *pubs2* database, if any corrupt pages are detected, only the suspect pages will be taken offline and the rest of the database will be brought online in read only mode.

5. **sp_setsuspect_granularity pubs2, "database"**

The next time recovery runs in the *pubs2* database, if any corrupt data is detected, the entire database will be marked suspect and taken offline.

Comments

- **sp_setsuspect_granularity** displays and sets the recovery fault isolation mode. This mode governs whether recovery marks an entire database or only the corrupt pages suspect when it detects that any data that it requires has been corrupted. For more information, see "Fault Isolation During Recovery" in Chapter 20, "Developing a Backup and Recovery Plan" of the *System Administration Guide.*
- The default recovery fault isolation mode of a user database is "database". You can set the recovery fault isolation mode only for a user database, not for a system database.
- You must be in the *master* database to set the recovery fault isolation mode.
- Data marked suspect due to corruption persists across Adaptive Server start-ups. When certain pages have been marked suspect, they remain offline after you reboot the server.
- When part or all of a database is marked suspect, the suspect data is not accessible to users unless a System Administrator has made the suspect data accessible with the **sp_forceonline_db** and **sp_forceonline_page** procedures.

- General database corruption, such as a corrupt database log or the unavailability of another resource not specific to a page, causes the entire database to be marked suspect, even if the recovery fault isolation mode is "page".
- If you do not specify **page** or **database**, Adaptive Server displays the current and configured settings. The current setting is the one that was in effect the last time recovery was executed in the database. The configured setting is the one that will be in effect the next time recovery is executed in the database.
- If the database comes online in **read_only** mode, no user can modify any of its data, including data that is unaffected by the suspect pages and is thus online. However, the system administrator can make the database writeable using the **sp_dboption** system procedure to set **read only** to **false**. In this case, users could then modify the online data, but the suspect data would remain inaccessible.

Messages

- `Can't run sp_setsuspect_granularity from within a transaction.`

  sp_setsuspect_granularity modifies system tables, so it cannot be run within a transaction.
- `No such database -- run sp_helpdb to list databases.`

  Reenter the database name.
- `Not allowed for System databases.`

  You can set the recovery fault isolation mode for a user database only.
- `Object level suspect granularity is not supported in this release.`

  Specify **database** or **page** for the recovery fault isolation mode.
- `Permission denied. This operation requires System Administrator (sa_role) role.`

  You are not a System Administrator.
- `suspect granularity option` *`option_name`* `is not valid.`

  Specify **database** or **page** for the recovery fault isolation mode.

- `The new values will become effective during the next recovery of the database` *`databasename`*`.`

  This informational message is displayed following successful execution of **sp_setsuspect_granularity.**

- `You must be in the 'master' database in order to change database options.`

  To set the recovery fault isolation mode for a database, issue **sp_setsuspect_granularity** from the *master* database.

Permissions

Only a System Administrator can execute **sp_setsuspect_granularity** to set the recovery fault isolation mode, but any user can use **sp_setsuspect_granularity** to display the settings.

Tables Used

master.dbo.sysattributes, master.dbo.sysdatabases

See Also

Commands	**dump database, dump transaction, load database**
System procedures	**sp_forceonline_db, sp_forceonline_page, sp_listsuspect_db, sp_listsuspect_page, sp_setsuspect_threshold**

sp_setsuspect_threshold

Function

Displays or sets the maximum number of suspect pages that Adaptive Server allows in a database before marking the entire database suspect.

Syntax

```
sp_setsuspect_threshold [dbname [, threshold]]
```

Parameters

dbname – is the name of the database for which you want to display or set the suspect escalation threshold. The default is the current database.

threshold – indicates the maximum number of suspect data pages that recovery will allow before marking the entire database suspect. The default is 20 pages. The minimum is 0.

Examples

1. `sp_setsuspect_threshold pubs2, 5`

   Sets the maximum number of suspect pages to five. If there are more than five suspect pages, recovery will mark the entire database suspect.

2. `sp_setsuspect_threshold pubs2`

   Displays the current and configured settings for the suspect escalation threshold for the *pubs2* database.

3. `sp_setsuspect_threshold`

   Displays the current and configured settings for the recovery fault isolation threshold for the current user database.

Comments

- You must be in the *master* database to set the suspect escalation threshold with **sp_setsuspect_threshold**.
- If you do not specify the number of pages, Adaptive Server displays the current and configured settings. The current setting is the one that was in effect the last time recovery was executed in the database. The configured setting is the one that will be in effect the next time recovery is executed in the database.

Messages

- `Can't run sp_setsuspect_threshold from within a transaction.`

  sp_setsuspect_threshold modifies system tables, so it cannot be run within a transaction.

- `No such database -- run sp_helpdb to list databases.`

  Check the spelling of the database name.

- `Not allowed because` *`databasename`* `has database level suspect granularity.`

  The recovery fault isolation mode must be "page". See **sp_setsuspect_granularity** for information about setting and displaying the recovery fault isolation mode.

- `Not allowed for System databases.`

  You can set the suspect escalation threshold for a user database only.

- `Permission denied. This operation requires System Administrator (sa_role) role.`

  You must be a System Administrator to change the recovery fault isolation mode with **sp_setsuspect_threshold**.

- `Suspect threshold value` *`threshold`* `is not valid.`

  The threshold must be a nonnegative integer.

- `The new values will become effective during the next recovery of the database` *`databasename`*`.`

  This informational message is displayed following successful execution of **sp_setsuspect_threshold**.

- `You must be in the 'master' database in order to change database options.`

  To set the suspect escalation threshold for a database, issue **sp_setsuspect_threshold** from the *master* database.

Permissions

Only a System Administrator can use **sp_setsuspect_threshold** to set the escalation threshold, but any database user can use it to display the current settings.

Tables Used

master.dbo.sysattributes, master.dbo.sysdatabases

See Also

System procedures	**sp_forceonline_db, sp_forceonline_page, sp_listsuspect_db, sp_listsuspect_page, sp_setsuspect_granularity**

sp_showcontrolinfo

Function

Displays information about engine group assignments, bound client applications, logins, and stored procedures.

Syntax

sp_showcontrolinfo [*object_type, object_name, spid*]

Parameters

object_type – is **AP**, **LG**, **PR**, **EG**, or **PS** for application, login, stored procedure, engine group, or process, respectively. If you do not specify an *object_type* (or specify an *object_type* of **null**), **sp_showcontrolinfo** displays information about all types.

object_name – is the name of the application, login, stored procedure, or engine group. Do not specify an *object_name* if you specify **PS** as the *object_type*. If you do not specify an *object_name* (or specify an *object_name* of **null**), **sp_showcontrolinfo** displays information about all object names.

spid – is the Adaptive Server process ID. Specify an *spid* only if you specify **PS** as the *object_type*. If you do not specify an *spid* (or specify an *spid* of **null**), **sp_showcontrolinfo** displays information for all *spid*s. Use **sp_who** to see *spid*s.

Examples

1. sp_showcontrolinfo

Shows all user-assigned execution class-to-object bindings.

2. sp_showcontrolinfo 'AP', 'isql'

Displays the execution class of the **isql** application.

3. sp_showcontrolinfo 'PS'

Displays the execution class for all processes assigned to engine groups.

4. sp_showcontrolinfo 'PS', null, 7

Displays the execution class for *spid* 7.

Comments

- When used with no parameters, **sp_showcontrolinfo** displays information about all user-assigned engine group assignments, bound client applications, logins, and stored procedures. When used with the object_type parameter, **sp_showcontrolinfo** provides information on an individual basis about application, login, or stored procedure bindings to an execution class, engine group compositions, and session-level attribute bindings. For more information, see Chapter 22, "Distributing Engine Resources Between Tasks," in the *Performance and Tuning Guide*.
- Unless *object_type* is **PR**, execute **sp_showcontrolinfo** from the *master* database. If *object_type* is **PR**, execute **sp_showcontrolinfo** from the database in which the procedure resides.
- If *object_type* is **null**, **sp_showcontrolinfo** displays execution class information for objects that match the other parameters.
- If *object_name* is **null**, **sp_showcontrolinfo** displays the binding information for all applications, logins, and stored procedures.
- If *spid* is **null**, **sp_showcontrolinfo** displays execution class information for objects that match the other parameters.

Messages

- `Can't run sp_showcontrolinfo from within a transaction.`

  sp_showcontrolinfo creates temporary tables, so it cannot be run within a transaction.

- *`object_type`* `is not a valid type of object.`

  The name you specified for *object_type* is not valid. *object_type* must be **AP**, **LG**, **PR**, **EG**, **PS**, or **null**.

Permissions

Any user can execute **sp_showcontrolinfo.**

Tables Used

sysattributes, syslogins

See Also

System procedures	**sp_addengine, sp_addexeclass, sp_bindexeclass, sp_clearpsexe, sp_dropengine, sp_dropexeclass, sp_showexeclass, sp_showpsexe, sp_unbindexeclass**

sp_showexeclass

Function

Displays the execution class attributes and the engines in any engine group associated with the specified execution class.

Syntax

sp_showexeclass [*execlassname*]

Parameters

execlassname – is the name of an execution class.

Examples

1. sp_showexeclass

```
classname   priority   engine_group   engines
---------  ---------- ------------- -----------
EC1         HIGH       ANYENGINE          ALL
EC2         MEDIUM     ANYENGINE          ALL
EC3         LOW        LASTONLINE         0
```

Displays the priority and engine group attribute values for all execution classes.

2. sp_showexeclass 'EC1'

```
classname   priority   engine_group           engines
----------- ---------- ---------------------- -----------
EC1         HIGH       ANYENGINE              ALL
```

Displays the attribute values of execution class *EC1*.

Comments

- **sp_showexeclass** displays the execution class attributes and the engines in any engine group associated with *execlassname*. For more information, see Chapter 22, "Distributing Engine Resources Between Tasks," in the *Performance and Tuning Guide*.
- If *execlassname* is NULL or absent, **sp_showexeclass** displays the priority and engine group attribute values for all execution classes, including the attribute values of the system-defined classes *EC1*, *EC2*, and *EC3*.

Messages

- `Can't run sp_showexeclass from within a transaction.`

  sp_showexeclass creates temporary tables, so it cannot be run within a transaction.

- `Execution class` *`execlassname`* `is not a valid class.`

  To see a list of valid class names, run **sp_showexeclass** with no *execlassname*.

Permissions

Any user can execute **sp_showexeclass**.

Tables Used

sysattributes, *sysengines*

See Also

System procedures	**sp_addengine, sp_addexeclass, sp_bindexeclass, sp_clearpsexe, sp_dropengine, sp_dropexeclass, sp_showcontrolinfo, sp_showpsexe, sp_unbindexeclass**

sp_showplan

Function

Displays the **showplan** output for any user connection for the current SQL statement or for a previous statement in the same batch.

Syntax

sp_showplan *spid*, *batch_id* output, *context_id* output, *stmt_num* output

To display the showplan output for the current SQL statement without specifying the *batch_id*, *context_id*, or *stmt_num*:

sp_showplan *spid*, null, null, null

Parameters

spid – is the process ID for any user connection. Use **sp_who** to see *spid*s.

batch_id – is a unique, nonnegative number for a batch

context_id – is a unique number for every procedure (or trigger) executed in a batch.

stmt_num – is the number of the current statement within a batch. The *stmt_num* must be a positive number.

Examples

1.
```
declare @batch int
declare @context int
declare @statement int
exec sp_showplan 99, @batch output, @context
output, @statement output
```

Displays the query plan for the current statement running in the user session with a *spid* value of 99, as well as values for the *batch_id*, *context_id*, and *statement_id* parameters. These values can be used to retrieve query plans in subsequent iterations of **sp_showplan** for the user session with a *spid* of 99.

2.
```
sp_showplan 99, null, null, null
```

Displays the **showplan** output for the current statement running in the user session with a *spid* value of 99.

Comments

- **sp_showplan** displays the **showplan** output for a currently executing SQL statement or for a previous statement in the same batch.
- To see the query plan for the previous statement within the same batch, execute **sp_showplan** again with the same parameter values, but subtract 1 from the statement number. Using this method, you can view all the statements in the statement batch back to query number one.
- **sp_showplan** can be run independently of Adaptive Server Monitor™ Server.
- If the *context_id* is greater than 0 for a SQL batch, the current statement is embedded in a stored procedure (or trigger) called from the original SQL batch. Select the *sysprocesses* row with the same *spid* value to display the procedure ID and statement ID.

Messages

- `Batch id must be non-negative.`

  Execute **sp_showplan** with null values for *batch_id*, *context_id*, and *stmt_num* to get a valid batch ID.

- `Context id must be non-negative.`

  Execute **sp_showplan** with null values for *batch_id*, *context_id*, and *stmt_num* to get a valid context ID.

- `If the batch id, context id, or statement number is specified, all three must be specified.`

  You tried to execute the procedure with at least one of these parameters, but not all three. You must either include values or *nulls* for all three parameters. Reissue the command entering values for all the parameters.

- `Statement number must be positive.`

  Execute **sp_showplan** with null values for *batch_id*, *context_id*, and *stmt_num* to get a valid statement number.

- `The current batch id` *`value`* `does not match the specified batch id` *`value`*`.`

  The batch you specified has finished, and a user started a new batch with a new batch ID. Reissue the command with a null value for the batch ID to determine the batch ID for this user.

- `The current context id value does not match the specified context id value.`

  The user session has changed its context ID by calling a new procedure or by returning from a previously issued procedure.

- `The query plan for spid value was marked available but can't be found in the process's plan space.`

  This may be a serious error. Rerun the same command. If Adaptive Server displays the same error message, report this to Sybase Technical Support.

- `The query plan for spid value is unavailable. Possibly the query has not started or has finished executing.`

  The query for which you issued the command is not available in the current text batch. **sp_showplan** shows query plans only for queries that are in the current text batch.

- `The specified spid value value applies to a server internal process, which does not execute a query plan.`

  The system process ID (*spid)* you specified is for an internal Adaptive Server process and not for a user connection. Check to make sure you are using the correct *spid* number.

- `The specified spid value value is out of range. It must be positive and not exceed the maximum number of user connections.`

  You specified a *spid* value for a statement number that is out of range.

- `The specified statement number value is greater than the number of statements in the batch or procedure. Specify a null statement number to obtain the current statement number.`

  You specified a value that is out of range. If you do not know the appropriate values for the parameters, reissue **sp_showplan** with null values in place of the parameter names, as follows:

  sp_showplan *spid*, null, null, null

- `The spid value must be a positive integer.`

  Reissue the command with a valid *spid*. Use **sp_who** for a list of valid *spids*.

Permissions

Only a System Administrator can execute **sp_showplan**.

Tables Used

None.

See Also

System procedures	**sp_who**

sp_showpsexe

Function

Displays execution class, current priority, and affinity for all client sessions running on Adaptive Server.

Syntax

sp_showpsexe [*spid*]

Parameters

spid – is the Adaptive Server session ID for which you want a report. The *spid* must belong to the application or login executing **sp_showpsexe**. Use **sp_who** to list *spid*s.

Examples

1. sp_showpsexe

```
spid  appl_name   login_name  exec_class current_priority task_affinity
----- ----------- ----------- ---------- ---------------- -------------
    1 isql        sa          EC1        HIGH             NONE
    5             NULL        NULL       LOW              NULL
    7 ctisql      sa          EC2        MEDIUM           NONE
    8 ctisql      sa          EC2        MEDIUM           NONE
```

Displays execution class, current priority, and affinity for all current client sessions.

2. sp_showpsexe 5

Displays the application name, login name, current priority, and engine affinity of the process with *spid* 5.

Comments

- **sp_showpsexe** displays execution class, current priority, and affinity for all sessions (objects with an *spid*). For more information, see Chapter 22, "Distributing Engine Resources Between Tasks," in the *Performance and Tuning Guide*.
- If the *spid* is NULL or absent, **sp_showpsexe** reports on all sessions currently running on Adaptive Server.
- **sp_showpsexe** does not report information for the following system processes: deadlock, checkpoint, network, auditing, and mirror handlers. It does display information for the housekeeper *spid*.

Messages

- `Can't run sp_showpsexe from within a transaction.`

  sp_showpsexe creates temporary tables, so it cannot be run within a transaction.

- `No SQL Server process with specified id exists.`

  To see a list of valid *spid*s, run **sp_who**.

Permissions

Any user can execute **sp_showpsexe**.

Tables Used

syslogins, sysprocesses

See Also

System procedures	**sp_addengine, sp_addexeclass, sp_bindexeclass, sp_clearpsexe, sp_dropengine, sp_dropexeclass, sp_showcontrolinfo, sp_showexeclass, sp_unbindexeclass**

sp_spaceused

Function

Displays estimates of the number of rows, the number of data pages, the size of indexes, and the space used by a specified table or by all tables in the current database.

Syntax

sp_spaceused [*objname* [,1]]

Parameters

objname – is the name of the table on which to report. If omitted, a summary of space used in the current database appears.

1 – prints separate information on the table's indexes and *text* / *image* storage.

Examples

1. sp_spaceused titles

```
name       rowtotal   reserved   data    index_size   unused
---------- ---------  ---------  ------- ----------   ----------
titles     18         46 KB      6 KB    4 KB         36 KB
```

Reports on the amount of space allocated (reserved) for the *titles* table, the amount used for data, the amount used for index(es), and the available (unused) space.

2. sp_spaceused titles, 1

```
index_name           size       reserved   unused
 -------------------- ---------- ---------- ----------
 titleidind           2 KB       32 KB      24 KB
 titleind             2 KB       16 KB      14 KB

 name       rowtotal   reserved   data   index_size unused
---------- --------  ---------  ------- ---------- ----------
titles           18      46 KB     6 KB       4 KB      36 KB
```

In addition to information on the *titles* table, prints information for each index on the table.

3. sp_spaceused blurbs,1

```
index_name           size       reserved   unused
-------------------- ---------- ---------- ----------
blurbs               0 KB       14 KB      12 KB
tblurbs              14 KB      16 KB      2 KB

name       rowtotal reserved    data    index_size unused
---------- -------- ----------- ------- ---------- ----------
blurbs            6       30 KB    2 KB      14 KB      14 KB
```

Displays the space taken up by the *text/image* page storage separately from the space used by the table. The object name for *text/image* storage is "t" plus the table name.

4. sp_spaceused

```
database_name   database_size
--------------- --------------------------
master          5 MB

reserved    data        index_size  unused
---------   ---------   ----------- --------
2176 KB     1374 KB     72 KB       730 KB
```

Prints a summary of space used in the current database.

5. sp_spaceused syslogs

```
name       rowtotal   reserved   data    index_size   unused
---------- ---------  ---------  ------- ----------   ----------
syslogs    Not avail. 32 KB     32 KB   0 KB         0 KB
```

Reports on the amount of space reserved and the amount of space available for the transaction log.

Comments

- **sp_spaceused** displays estimates of the number of data pages, space used by a specified table or by all tables in the current database, and the number of rows in the tables. **sp_spaceused** computes the *rowtotal* value using the **rowcnt** built-in function. This function uses a value for the average number of rows per data page based on a value in the allocation pages for the object. This method is very fast, but the results are estimates, and update and insert activity change actual values. The **update statistics** command, **dbcc checktable**, and **dbcc checkdb** update the rows-per-page estimate, so *rowtotal* is most accurate after one of these commands executes. Always use **select count(*)** if you need exact row counts.

- **sp_spaceused** reports on the amount of space affected by tables, clustered indexes, and nonclustered indexes.
- The amount of space allocated (reserved) reported by **sp_spaceused** is a total of the data, index size, and available (unused) space.
- Space used by *text* and *image* columns, which are stored as separate database objects, is reported separately in the *index_size* column and is included in the summary line for a table. The object name for *text/image* storage in the *index_size* column is "t" plus the table name.
- When used on *syslogs*, **sp_spaceused** reports *rowtotal* as "Not available". See example 5.

Messages

- `Object does not exist in this database.`

  The object specified does not exist in the current database.
- `Object is stored in 'sysprocedures' and has no space allocated directly.`

  The object is a trigger, stored procedure, rule, or default.
- `Object must be in the current database.`

  The object specified is not in the current database.
- `Views don't have space allocated.`

  sp_spaceused reports on the amount of space taken up by tables, clustered indexes, and nonclustered indexes only.

Permissions

Any user can execute **sp_spaceused**.

Tables Used

master.dbo.spt_values, *master.dbo.sysusages*, *sysindexes*, *sysobjects*

See Also

Catalog stored procedures	**sp_statistics**
Commands	**create index**, **create table**, **drop index**, **drop table**
System procedures	**sp_help**, **sp_helpindex**

sp_syntax

Function

Displays the syntax of Transact-SQL statements, system procedures, utilities, and other routines for Adaptive Server, depending on which products and corresponding **sp_syntax** scripts exist on your server.

Syntax

sp_syntax *word* [, *mod*][, *language*]

Parameters

word – is the name or partial name of a command or routine; for example, "help", to list all system procedures providing help. To include spaces or Transact-SQL reserved words, enclose the word in quotes.

mod – is the name or partial name of one of the modules such as "Transact-SQL" or "Utility". Each **sp_syntax** installation script adds different modules. Use **sp_syntax** without any parameters to see which modules exist on your server.

language – is the language of the syntax description to be retrieved. *language* must be a valid language name in the *syslanguages* table.

Examples

1. sp_syntax

```
sp_syntax provides syntax help for Sybase products.
These modules are installed on this Server:

 Module
 --------------------
 OpenVMS
 Transact-SQL
 UNIX Utility
 System Procedure

Usage: sp_syntax command [, module [, language]]
```

Displays all **sp_syntax** modules available on your server.

2. sp_syntax "disk"

Displays the syntax and functional description of all routines containing the word or word fragment "disk". Since "disk" is a Transact-SQL reserved word, enclose it in quotes.

Comments

- The text for **sp_syntax** is in the database *sybsyntax*. Load **sp_syntax** and the *sybsyntax* database onto Adaptive Server with the installation script described in the configuration documentation for your platform. If you cannot access **sp_syntax**, see your System Administrator for information about installing it on your server.
- You can use wildcard characters within the command name you are searching for. However, if you are looking for a command or function that contains the literal "_", you may get unexpected results, since the underscore wildcard character represents any single character.

Messages

- `Can't run sp_syntax from within a transaction.`

  sp_syntax creates temporary tables, so it cannot be run within a transaction.
- `No command or routine has a name like 'word'`

  The command name you used is not in the *sybsyntax* database.
- `No module has a name like 'mod'`

  The module name you used is not in the *sybsyntax* database.
- `No command or routine has a name like 'word' and a module like 'mod'`

  The combination of command name and module name is not in the *sybsyntax* database.
- `sp_syntax provides syntax help for Sybase products.`

  ```
  These modules are installed on this Server:
   Module
   --------------------
   module_name

  Usage: sp_syntax command [, module [, language]]
  ```

  These help message appear when you use **sp_syntax** with no arguments.

Permissions

Any user can execute **sp_syntax**.

Tables Used

sybsyntax..sybsyntax

See Also

System procedures	**sp_help**, **sp_helpdb**

sp_sysmon

Function

Displays performance information.

Syntax

```
sp_sysmon begin_sample

sp_sysmon { end_sample | interval }
   [, section [, applmon] ]

sp_sysmon { end_sample | interval } [, applmon ]
```

Parameters

begin_sample – starts sampling. You cannot specify a section when you specify **begin_sample.**

end_sample – ends sampling and prints the report.

interval – specifies the time period for the sample. It must be in HH:MM:SS form, for example “00:20:00”.

section – is the abbreviation for one of the sections printed by **sp_sysmon**. Table 3-20 lists the values and corresponding names of the report sections.

Table 3-20: *sp_sysmon* report sections

Report Section	Parameter
Application Management	**appmgmt**
Data Cache Management	**dcache**
Disk I/O Management	**diskio**
ESP Management	**esp**
Index Management	**indexmgmt**
Kernel Utilization	**kernel**
Lock Management	**locks**
Memory Management	**memory**
Metadata Cache Management	**mdcache**
Monitor Access to Executing SQL	**monaccess**
Network I/O Management	**netio**

Table 3-20: *sp_sysmon* report sections (continued)

Report Section	Parameter
Parallel Query Management	**parallel**
Procedure Cache Management	**pcache**
Recovery Management	**recovery**
Task Management	**taskmgmt**
Transaction Management	**xactmgmt**
Transaction Profile	**xactsum**
Worker Process Management	**wpm**

applmon – specifies whether to print application detail, application and login detail, or no application detail. The default is to omit the application detail. Valid values are listed in Table 3-21.

Table 3-21: Values for *applmon* parameter to *sp_sysmon*

Parameter	Information Reported
appl_only	CPU, I/O, priority changes and resource limit violations by application name.
appl_and_login	CPU, I/O, priority changes and resource limit violations by application name and login name.
no_appl	Skips the by application or by login section of the report. This is the default.

This parameter is only valid when printing the full report and when you specify **appmgmt** for the *section*.

Examples

1. **`sp_sysmon "00:10:00"`**

   Prints monitor information after 10 minutes.

2. **`sp_sysmon "00:05:00", diskio`**

   Prints only the "Disk Management" section of the **sp_sysmon** report after 5 minutes.

3. **sp_sysmon begin_sample**
   go
   execute proc1
   go
   execute proc2
   go
   select sum(total_sales) from titles
   go
   sp_sysmon end_sample, dcache
   go

   Starts the sample, executes stored procedures and a select query, ends the sample, and prints only the "Data Cache" section of the report.

4. **sp_sysmon "00:05:00", @applmon = appl_and_login**

   Prints the full report and includes application and login detail for each login.

Comments

- **sp_sysmon** displays information about Adaptive Server performance. It sets internal counters to 0, and then waits for the specified interval while activity on the server causes the counters to be incremented. When the interval ends, **sp_sysmon** prints information from the values in the counters. See Chapter 24, "Monitoring Performance with sp_sysmon," in the *Performance and Tuning Guide* for more information.
- To print only a single section of the report, use the values listed in Table 3-20 for the second parameter.

  If you use sp_sysmon in batch mode, with **begin_sample** and **end_sample**, the time interval between executions must be at least one second. You can use **waitfor delay "00:00:01"** to lengthen the execution time of a batch.

Messages

- `Can't run sp_sysmon from within a transaction.`

  sp_sysmon modifies system tables, so it cannot be run within a transaction.

- `Invalid parameter '`*`parameter`*`'. Usage: sp_sysmon {'begin_sample'|{'end_sample'|'hh:mm:ss'}[,section]}} [,@applmon={'appl_only'|'appl_and_login'|'no_appl'}]`

  You specified an invalid parameter.

- `Invalid value for parameter 2: sp_sysmon Valid values are: 'kernel','wpm','parallel','taskmgmt','appmgmt','esp','monaccess','xactsum','xactmgmt','indexmgmt','mdcache','locks','dcache','pcache','memory','recovery','diskio','netio'`

  You specified an invalid value for *section*. Use one of the values in the error message.

- `sp_sysmon: No report was produced. Sampling interval was less than 1 second.`

  When using **sp_sysmon** in batch mode, the amount of time between the **begin_sample** and **end_sample** executions was less than 1 second.

- `sp_sysmon: Section parameter 'section' ignored for begin_sample option.`

  You can specify a report section only with an interval or with **end_sample**.

- `You must have System Administrator (SA) role to execute this stored procedure.`

  Only a System Administrator can execute **sp_sysmon**.

Permissions

Only a System Administrator can execute **sp_sysmon**.

Tables Used

master.dbo.sysconfigures, master.dbo.syscurconfigs, master.dbo.sysdevices, master.dbo.ssysmonitors

sp_thresholdaction

Function

Executes automatically when the number of free pages on the log segment falls below the last-chance threshold, unless the threshold is associated with a different procedure. Sybase does not provide this procedure.

Syntax

When a threshold is crossed, Adaptive Server passes the following parameters to the threshold procedure by position:

```
sp_thresholdaction @dbname,
        @segment_name,
        @space_left,
        @status
```

Parameters

@dbname – is the name of a database where the threshold was reached.

@segment_name – is the name of the segment where the threshold was reached.

@space_left – is the threshold size, in 2K pages.

@status – is 1 for the last-chance threshold; 0 for all other thresholds.

Examples

```
1. create procedure sp_thresholdaction
        @dbname varchar(30),
        @segmentname varchar(30),
        @space_left int,
        @status int
   as
        dump transaction @dbname to tapedump1
```

Creates a threshold procedure for the last-chance threshold that dumps the transaction log to a tape device.

Comments

- **sp_thresholdaction** must be created by the Database Owner (in a user database), or a System Administrator (in the *sybsystemprocs* database), or a user with **create procedure** permission.

- You can add thresholds and create threshold procedures for any segment in a database.
- When the last-chance threshold is crossed, Adaptive Server searches for the **sp_thresholdaction** procedure in the database where the threshold event occurs. If it does not exist in that database, Adaptive Server searches for it in *sybsystemprocs*. If it does not exist in *sybsystemprocs*, it searches *master*. If Adaptive Server does not find the procedure, it sends an error message to the error log.
- **sp_thresholdaction** should contain a **dump transaction** command to truncate the transaction log.
- By design, the last-chance threshold allows enough free space to record a **dump transaction** command. There may not be enough space to record additional user transactions against the database. Only commands that are not recorded in the transaction log (**select**, fast **bcp**, **readtext**, and **writetext**) and commands that might be necessary to free additional log space (**dump transaction**, **dump database**, and **alter database**) can be executed. By default, other commands are suspended and a message is sent to the error log. To abort these commands rather than suspend them, use the **abort tran on log full** option of **sp_dboption** followed by the **checkpoint** command.

Waking Suspended Processes

- Once the **dump transaction** command frees sufficient log space, suspended processes automatically awaken and complete.
- If fast **bcp**, **writetext**, or **select into** have resulted in unlogged changes to the database since the last backup, the last-chance threshold procedure cannot execute a **dump transaction** command. When this occurs, use **dump database** to make a copy of the database, and then truncate the transaction log with **dump transaction**.
- If this does not free enough space to awaken the suspended processes, it may be necessary to increase the size of the transaction log. Use the **log on** option of the **alter database** command to allocate additional log space.
- As a last resort, System Administrators can use **sp_who** to determine which processes are suspended, and then use the **kill** command to kill them.

See Also

Commands	**create procedure**, **dump transaction**
System procedures	**sp_addthreshold**, **sp_dboption**, **sp_dropthreshold**, **sp_helpsegment**, **sp_helpthreshold**, **sp_modifythreshold**

sp_unbindcache

Function

Unbinds a database, table, index, *text* object, or *image* object from a data cache.

Syntax

```
sp_unbindcache dbname [,[owner.]tablename
   [, indexname | "text only"]]
```

Parameters

dbname – is the name of database to be unbound or the name of the database containing the objects to be unbound.

owner – is the name of the table's owner. If the table is owned by the Database Owner, the owner name is optional.

tablename – is the name of the table to be unbound from a cache or the name of a table whose index, *text* object, or *image* object is to be unbound from a cache.

indexname – is the name of an index to be unbound from a cache.

text only – unbinds *text* or *image* objects from a cache.

Examples

1. `sp_unbindcache pubs2, titles`

   Unbinds the *titles* table from the cache to which it is bound.

2. `sp_unbindcache pubs2, titles, titleidind`

   Unbinds the *titleidind* index from the from the cache to which it is bound.

3. `sp_unbindcache pubs2, au_pix, text`

   Unbinds the *text* or *image* object for the *au_pix* table from the cache to which it is bound.

4. `sp_unbindcache pubs2, syslogs`

   Unbinds the transaction log, *syslogs*, from its cache.

Comments

- When you unbind a database or database object from a cache, all subsequent I/O for the cache is performed in the default data

cache. All dirty pages in the cache being unbound are written to disk, and all clean pages are cleared from the cache. For more information, see Chapter 22, "Distributing Engine Resources Between Tasks," in the *Performance and Tuning Guide*.

- Cache unbindings take effect immediately and do not require a restart of the server.
- When you drop a database, table, or index, its cache bindings are automatically dropped.
- To unbind a database, you must be using the *master* database. For tables, indexes, *text* objects, or *image* objects, you must be using the database where the objects are stored.
- To unbind any system tables in a database, you must be using the database, and the database must be in single-user mode. Use the command:

  ```
  sp_dboption db_name, "single user", true
  ```

  See **sp_dboption** for more information.
- The following procedures provide information about the bindings for their respective objects: **sp_helpdb** for databases, **sp_help** for tables, and **sp_helpindex** for indexes.
- **sp_helpcache** prints the names of objects bound to caches.
- **sp_unbindcache** needs to acquire an exclusive table lock when you are unbinding a table or its indexes to a cache. No pages can be read while the unbinding takes place. If a user holds locks on a table, and you issue **sp_unbindcache** on that object, the **sp_unbindcache** task sleeps until the locks are released.
- When you change the cache binding for an object with **sp_bindcache** or **sp_unbindcache**, the stored procedures that reference the object are recompiled the next time they are executed. When you change the binding for a database, the stored procedures that reference objects in the database are recompiled the next time they are executed.
- To unbind all objects from a cache, use the system procedure **sp_unbindcache_all**.

Messages

- `Can't run sp_unbindcache from within a transaction.`

  sp_unbindcache modifies system tables, so it cannot be run within a transaction.

- `You must be in Master to bind or unbind a database.`

  Database unbinding can take place only from the *master* database. Issue the **use master** command, and try again.

- `The target database does not exist.`

  The database name you specified does not exist. To see the names of all databases, execute **sp_helpdb**.

- `The target index does not exist.`

  The index name you specified does not exist. To see the names of indexes on a table, execute **sp_helpindex** *tablename*.

- `The target object does not exist.`

  The table name you specified does not exist. You must be using a database in order to bind any of the objects in a database. To see the names of tables in a database, execute **sp_help**.

- `You must be in Master to bind or unbind a database.`

  Database binding can take place only from the *master* database. Issue the **use master** command, and try again.

Permissions

Only a System Administrator can execute **sp_unbindcache**.

Tables Used

master..sysattributes, *master..sysdatabases*, *sysindexes*, *sysobjects*

See Also

System procedures	**sp_bindcache**, **sp_dboption**, **sp_helpcache**, **sp_unbindcache_all**

sp_unbindcache_all

Function

Unbinds all objects that are bound to a cache.

Syntax

sp_unbindcache_all *cache_name*

Parameters

cache_name – is the name of the data cache from which objects are to be unbound.

Examples

1. sp_unbindcache_all pub_cache

Unbinds all databases, tables, indexes, *text* objects and *image* objects that are bound to *pub_cache*.

Comments

- When you unbind entities from a cache, all subsequent I/O for the cache is performed in the default cache.
- To unbind individual objects from a cache, use the system procedure **sp_unbindcache**.
- See **sp_unbindcache** for more information about unbinding caches.

Messages

- `Can't run sp_unbindcache_all from within a transaction.`

  sp_unbindcache_all modifies system tables, so it cannot be run within a transaction.

- `The specified named cache 'cachename' does not exist.`

  There is no cache with the name you specified. Use **sp_cacheconfig** with no parameters to see the names of existing caches.

- `Unable to allocate a DBTABLE descriptor to open database 'database_name'. Another database must be closed or dropped before opening this one.`

  There are objects from more than eight databases bound to the cache you named. **sp_unbindcache_all** can only open eight

databases. Use **sp_helpcache** to see the names of objects bound to this cache and **sp_unbindcache** to unbind individual caches. When the number of databases is eight or less, you can execute **sp_unbindcache_all**.

- `You must be in Master to unbind a database.`

  Database unbinding can take place only from the *master* database. Issue the **use master** command, and try again.

Permissions

Only a System Administrator can execute **sp_unbindcache_all**.

Tables Used

master..sysattributes, master..sysdatabases, sysindexes, sysobjects

See Also

System procedures	**sp_bindcache**, **sp_helpcache**, **sp_unbindcache**

sp_unbindefault

Function

Unbinds a created default value from a column or from a user-defined datatype.

Syntax

```
sp_unbindefault objname [, futureonly]
```

Parameters

objname – is the name of either the table and column or the user-defined datatype from which to unbind the default. If the parameter is not of the form *"table.column"*, then *objname* is assumed to be a user-defined datatype. When unbinding a default from a user-defined datatype, any columns of that type that have the same default as the user-defined datatype are also unbound. Columns of that type, whose default has already been changed, are unaffected.

futureonly – prevents existing columns of the specified user-defined datatype from losing their defaults. It is ignored when unbinding a default from a column.

Examples

1. `sp_unbindefault "employees.startdate"`

   Unbinds the default from the *startdate* column of the *employees* table.

2. `sp_unbindefault ssn`

   Unbinds the default from the user-defined datatype named *ssn* and all columns of that type.

3. `sp_unbindefault ssn, futureonly`

   Unbinds defaults from the user-defined datatype *ssn*, but does not affect existing columns of that type.

Comments

- Use **sp_unbindefault** to remove defaults created with **sp_bindefault.** Use **alter table** to drop defaults declared using the **create table** or **alter table** statements.

- Columns of a user-defined datatype lose their current default unless the default has been changed or the value of the optional second parameter is **futureonly.**
- To display the text of a default, execute **sp_helptext** with the default name as the parameter.

Messages

- `Column or usertype must be in current database.`

  The *objname* parameter cannot include a database reference.
- `Columns of the user datatype specified had their defaults unbound.`

  Defaults on other columns of the user-defined datatype specified were unbound, unless their defaults were changed previously.
- `Default unbound from datatype.`

  The user-defined datatype supplied for the *objname* parameter no longer has any default.
- `Default unbound from table column.`

  The table column supplied for the *objname* parameter no longer has any default.
- `The specified column has no default.`

  No default is bound to the column name supplied for the *objname* parameter.
- `The specified user datatype has no default.`

  No default is bound to the datatype name supplied for the *objname* parameter.
- `You do not own a table with a column of that name.`

  The table name supplied for the *objname* parameter either does not exist in the database or you do not own it. You can bind or unbind defaults from columns only in the tables that you own.
- `You do not own a user datatype of that name.`

  The user-defined datatype supplied for the *objname* parameter either does not exist in the database or you do not own it. You can bind or unbind defaults from columns only in the tables that you own.

Permissions

Only the object owner can execute **sp_unbindefault.**

Tables Used

syscolumns, *sysobjects*, *sysprocedures*, *systypes*

See Also

Commands	**create default**, **drop default**
System procedures	**sp_bindefault**, **sp_helptext**

sp_unbindexeclass

Function

Removes the execution class attribute previously associated with an client application, login, or stored procedure for the specified scope.

Syntax

sp_unbindexeclass *object_name, object_type, scope*

Parameters

object_name – is the name of the application, login, or stored procedure for which to remove the association to the execution class.

object_type – identifies the type of *object_name* as **ap**, **lg**, or **pr** for application, login, or stored procedure.

scope – is the application name or the login name for which the unbinding applies for an application or login. It is the stored procedure owner name (user name) for stored procedures.

Examples

1. sp_unbindexeclass 'sa', 'lg', 'isql'

Removes the association between "sa" login scoped to application **isql** and an execution class. "sa" automatically binds itself to another execution class, depending on other binding specifications, precedence, and scoping rules. If no other binding is applicable, the object binds to the default execution class, *EC2*.

Comments

- The parameters must match an existing entry in the *sysattributes* system table.
- If you specify a null value for scope, Adaptive Server unbinds the object for which the scope is null, if there is one.
- A null value for scope does not indicate that unbinding should apply to all bound objects.
- Whenun binding a stored procedure from an execution class, you must use the name of the stored procedure owner (user name) for the *scope* parameter.
- Stored procedures can be dropped before or after unbinding.

- A user cannot be dropped from a database if the user owns a stored procedure that is bound to an execution class in that database.
- Unbind objects of type **PR** before dropping them from the database.
- Unbinding will fail if the associated engine group has no online engines and active processes are bound to the associated execution class.
- Due to precedence and scoping rules, the execution class being unbound may or may not have been in effect for the object called *object_name*. The object automatically binds itself to another execution class, depending on other binding specifications and precedence and scoping rules. If no other binding is applicable, the object binds to the default execution class, *EC2*.

Messages

- `Can't run sp_unbindexeclass from within a transaction.`

  sp_unbindexeclass modifies system tables, so it cannot be run within a transaction.

- `Failed to unbind` *`object_name`* `from execution class.`

  Unbinding the object from the execution class resulted in error. Check the error log.

- `No definition for classname` *`classname`* `exists`

  You must define the execution class before binding an object to it.

- `No login with the specified name` *`object_name`* `exists.`

  object_name is not a valid Adaptive Server login.

- *`object_name`* `cannot be NULL.`

  You must supply the name of the object to which the execution class will be bound.

- *`object_type`* `is not a valid object type for this stored procedure.`

  The name you specified for *object_type* is not valid. Use **ap**, **lg**, or **pr** for *object_type*.

- `The specified binding to be dropped does not exist.`

  The object was not bound to the execution class, so it cannot be unbound.

Permissions

Only a System Administrator can execute **sp_unbindexeclass.**

Tables Used

sysattributes, syslogins

See Also

System procedures	**sp_addengine, sp_addexeclass, sp_bindexeclass, sp_clearpsexe, sp_dropengine, sp_dropexeclass, sp_showcontrolinfo, sp_showexeclass, sp_showpsexe**

sp_unbindmsg

Function

Unbinds a user-defined message from a constraint.

Syntax

sp_unbindmsg *constrname*

Parameters

constrname – is the name of the constraint from which a message is to be unbound.

Examples

1. sp_unbindmsg positive_balance

Unbinds a user-defined message from the constraint *positive_balance*.

Comments

- You can bind only one message to a constraint. To change the message bound to a constraint, use **sp_bindmsg**; the new message number replaces any existing bound message. It is not necessary to use **sp_unbindmsg** first.
- To retrieve message text from the *sysusermessages* table, execute **sp_getmessage**.

Messages

- `Constraint name must be in 'current' database.`

  You can unbind messages only from constraints that are defined in the current database.

- `Constraint name must belong to the current user.`

  You cannot unbind a message from a constraint created by another user.

- `No such referential or check constraint exists. Please check whether the constraint name is correct.`

  Use **sp_help** *tablename* to see a list of constraints on a table.

- `Constraint is not bound to any message.`

  No message is bound to *constrname*.

- `Unbinding message failed unexpectedly. Please try again.`

  An error occurred. Reissue the command.

- `Message unbound from constraint.`

  You have successfully unbound the user-defined message from *constrname*.

Permissions

Only the object owner can execute **sp_unbindmsg**.

Tables Used

sysconstraints, *sysobjects*

See Also

System procedures	**sp_addmessage**, **sp_bindmsg**, **sp_getmessage**

sp_unbindrule

Function

Unbinds a rule from a column or from a user-defined datatype.

Syntax

```
sp_unbindrule objname [, futureonly]
```

Parameters

objname – is the name of the table and column or of the user-defined datatype from which the rule is to be unbound. If the parameter is not of the form *"table.column"*, then *objname* is assumed to be a user-defined datatype. Unbinding a rule from a user-defined datatype also unbinds it from columns of the same type. Columns that are already bound to a different rule are unaffected.

futureonly – prevents columns of the specified user-defined datatype from losing their rules. It is ignored when unbinding a rule from a column.

Examples

1. `sp_unbindrule "employees.startdate"`

   Unbinds the rule from the *startdate* column of the *employees* table.

2. `sp_unbindrule def_ssn`

   Unbinds the rule from the user-defined datatype named *def_ssn* and all columns of that type.

3. `sp_unbindrule ssn, futureonly`

   The user-defined datatype *ssn* no longer has a rule, but existing *ssn* columns are unaffected.

Comments

- Executing **sp_unbindrule** removes a rule from a column or from a user-defined datatype in the current database. If you do not want to unbind the rule from existing *objname* columns, use **futureonly** as the second parameter.
- You cannot use **sp_unbindrule** to unbind a check constraint. Use **alter table** to drop the constraint.
- To unbind a rule from a table column, specify the *objname* argument in the form *"table.column"*.

- The rule is unbound from all existing columns of the user-defined datatype unless the rule has been changed or the value of the optional second parameter is **futureonly.**
- To display the text of a rule, execute **sp_helptext** with the rule name as the parameter.

Messages

- `Column or usertype must be in current database.`

  The *objname* parameter may not include a database reference.
- `Columns of the user datatype specified had their rules unbound.`

  Rules on other columns of the user-defined datatype specified were unbound, unless their rules were previously changed.
- `Rule unbound from datatype.`

  The user-defined datatype supplied for the *objname* parameter no longer has any rule.
- `Rule unbound from table column.`

  The table column supplied for the *objname* parameter no longer has any rule.
- `The specified column has no rule.`

  There is no rule bound to the table column supplied for the *objname* parameter. Nothing changed.
- `The specified user datatype has no rule.`

  There is no rule bound to the user-defined datatype supplied for the *objname* parameter. Nothing changed.
- `You do not own a table with a column of that name.`

  The table name supplied for the *objname* parameter either does not exist in the database or you do not own it. You can bind or unbind rules only on tables that you own.
- `You do not own a user datatype of that name.`

  The user-defined datatype supplied for the *objname* parameter either does not exist in the database or you do not own it. You can bind or unbind rules only from datatypes that you own.

Permissions

Only the object owner can execute **sp_unbindrule.**

Tables Used

syscolumns, *sysconstraints*, *sysobjects*, *sysprocedures*, *systypes*

See Also

Commands	**create rule**, **drop rule**
System procedures	**sp_bindrule**, **sp_helptext**

sp_volchanged

Function

Notifies the Backup Server that the operator performed the requested volume handling during a dump or load.

Syntax

```
sp_volchanged session_id, devname, action
   [, fname [, vname]]
```

Parameters

session_id – identifies the Backup Server session that requested the volume change. Use the *@session_id* parameter specified in the Backup Server's volume change request.

devname – is the device on which a new volume was mounted. Use the *@devname* parameter specified in the Backup Server's volume change request. If the Backup Server is not located on the same machine as the Adaptive Server, use the form:

```
device at backup_server_name
```

action – indicates whether the Backup Server should **abort**, **proceed** with, or **retry** the dump or load.

fname – is the file to be loaded. If you do not specify a file name with **sp_volchanged**, the Backup Server loads the **file** = *filename* parameter of the load command. If neither **sp_volchanged** nor the load command specifies which file to load, the Backup Server loads the first file on the tape.

vname – is the volume name that appears in the ANSI tape label. The Backup Server writes the volume name in the ANSI tape label when overwriting an existing dump, dumping to a brand new tape, or dumping to a tape whose contents are not recognizable. If you do not specify a *vname* with **sp_volchanged**, the Backup Server uses the **dumpvolume** value specified in the dump command. If neither **sp_volchanged** nor the dump command specifies a volume name, the Backup Server leaves the name field of the ANSI tape label blank.

During loads, the Backup Server uses the *vname* to confirm that the correct tape has been mounted. If you do not specify a *vname* with **sp_volchanged**, the Backup Server uses the **dumpvolume** specified in the load command. If neither **sp_volchanged** nor the

load command specifies a volume name, the Backup Server does not check the name field of the ANSI tape label before loading the dump.

Examples

1. sp_volchanged 8, "/dev/nrmt4", RETRY

The following message from Backup Server indicates that a mounted tape's expiration date has not been reached:

```
Backup Server: 4.49.1.1: OPERATOR: Volume to be overwritten on
'/dev/rmt4' has not expired: creation date on this volume is
Sunday, Nov. 15, 1992, expiration date is Wednesday, Nov. 25,
1992.
Backup Server: 4.78.1.1: EXECUTE sp_volchanged
       @session_id = 8,
       @devname = '/auto/remote/pubs3/SERV/Masters/testdump',
       @action = { 'PROCEED' | 'RETRY' | 'ABORT' }
```

The operator changes the tape, and then issues the command in example 1.

Comments

Roles of Operator, Adaptive Server, and Backup Server in Volume Changes

- If the Backup Server detects a problem with the currently mounted volume, it requests a volume change:
  - On OpenVMS systems, the Backup Server sends volume change messages to the operator terminal on the machine on which it is running. Use the **with notify = client** option of the dump or load command to route other Backup Server messages to the terminal session on which the **dump** or **load** request initiated.
  - On UNIX systems, the Backup Server sends messages to the client that initiated the dump or load request. Use the **with notify = operator_console** option of the dump or load command to route messages to the terminal where the Backup Server was started.
  - After mounting another volume, the operator executes **sp_volchanged** from any Adaptive Server that can communicate with the Backup Server performing the dump or load. The operator does not have to log into the Adaptive Server on which the dump or load originated.
- On OpenVMS systems, the operating system—not the Backup Server—requests a volume change when it detects the end of a

volume or when the specified drive is offline. The operator uses the OpenVMS **REPLY** command to reply to these messages.

- On UNIX systems, the Backup Server requests a volume change when the tape capacity has been reached. The operator mounts another tape and executes **sp_volchanged**. Table 3-22 illustrates this process.

Table 3-22: Changing tape volumes on a UNIX system

Sequence	Operator, Using *isql*	Adaptive Server	Backup Server
1	Issues the **dump database** command		
2		Sends dump request to Backup Server	
3			Receives dump request message from Adaptive Server Sends message for tape mounting to operator Waits for operator's reply
4	Receives volume change request from Backup Server Mounts tapes Executes **sp_volchanged**		
5			Checks tapes If tapes are okay, begins dump When tape is full, sends volume change request to operator
6	Receives volume change request from Backup Server Mounts tapes Executes **sp_volchanged**		

Table 3-22: Changing tape volumes on a UNIX system (continued)

Sequence	Operator, Using *isql*	Adaptive Server	Backup Server
7			Continues dump When dump is complete, sends messages to operator and Adaptive Server
8	Receives message that dump is complete Removes and labels tapes	Receives message that dump is complete Releases locks Completes the **dump database** command	

Messages

Volume Change Prompts for Loads

- `Dump file '`*`fname`*`' section `*`vname`*` found instead of '`*`fname`*`' section `*`vname`*`.`

  Backup Server issues this message if it cannot find the specified file on a single-file medium.

The Operator Can	By Entering
Abort the load	**sp_volchanged** ***session_id, devname***, **abort**
Mount another volume and try to load it	**sp_volchanged** ***session_id, devname***, **retry** [, ***fname*** [, ***vname***]]
Load the file on the currently mounted volume, even though it is not the specified file (not recommended)	**sp_volchanged** ***session_id, devname***, **proceed** [, ***fname*** [, ***vname***]]

- `Mount the next volume to read.`

  Backup Server issues this message when it is ready to read the next section of the dump file from a multivolume dump.

The Operator Can	By Entering
Abort the load	**sp_volchanged *session_id, devname*, abort**
Mount the next volume and proceed with the load	**sp_volchanged *session_id, devname*, proceed** [, ***fname*** [, ***vname***]]

- `Mount the next volume to search.`

  Backup Server issues this message if it cannot find the specified file on multifile media.

The Operator Can	By Entering
Abort the load	**sp_volchanged *session_id, devname*, abort**
Mount another volume and proceed with the load	**sp_volchanged *session_id, devname*, proceed** [, ***fname*** [, ***vname***]]

Volume Change Prompts for Dumps

- `Mount the next volume to search.`

  When appending a dump to an existing volume, Backup Server issues this message if it cannot find the end-of-file mark.

The Operator Can	By Entering
Abort the dump	**sp_volchanged *session_id, devname*, abort**
Mount a new volume and proceed with the dump	**sp_volchanged *session_id, devname*, proceed** [, ***fname*** [, ***vname***]]

- `Mount the next volume to write.`

  Backup Server issues this message when it reaches the end of the tape. This occurs when it detects the end-of-tape mark or dumps

the number of kilobytes specified by the **capacity** parameter of the dump command or the device's *sysdevices.high* value.

The Operator Can	By Entering
Abort the dump	**sp_volchanged** ***session_id, devname***, **abort**
Mount the next volume and proceed with the dump	**sp_volchanged** ***session_id, devname***, **proceed** [, ***fname*** [, ***vname***]]

- `Volume on device` *`devname`* `has restricted access (code` *`access_code`*`).`

  Dumps that specify the **init** option overwrite any existing contents of the tape. Backup Server issues this message if you try to dump to a tape with ANSI access restrictions without specifying the **init** option.

The Operator Can	By Entering
Abort the dump	**sp_volchanged** ***session_id, devname***, **abort**
Mount another volume and retry the dump	**sp_volchanged** ***session_id, devname***, **retry** [, ***fname*** [, ***vname***]]
Proceed with the dump, overwriting any existing contents	**sp_volchanged** ***session_id, devname***, **proceed** [, ***fname*** [, ***vname***]]

- `Volume on device` *`devname`* `is expired and will be overwritten.`

  Dumps that specify the **init** option overwrite any existing contents of the tape. During dumps to single-file media, Backup Server issues this message if you have not specified the **init**

option and the tape contains a dump whose expiration date has passed.

The Operator Can	By Entering
Abort the dump	**sp_volchanged** ***session_id, devname***, **abort**
Mount another volume and retry the dump	**sp_volchanged** ***session_id, devname***, **retry** [, ***fname*** [, ***vname***]]
Proceed with the dump, overwriting any existing contents	**sp_volchanged** ***session_id, devname***, **proceed** [, ***fname*** [, ***vname***]]

- `Volume to be overwritten on '`*`devname`*`' has not expired: creation date on this volume is `*`creation_date`*`, expiration date is `*`expiration_date`*`.`

  On single-file media, Backup Server checks the expiration date of any existing dump unless you specify the **init** option. The Backup Server issues this message if the dump has not yet expired.

The Operator Can	By Entering
Abort the dump	**sp_volchanged** ***session_id, devname***, **abort**
Mount another volume and retry the dump	**sp_volchanged** ***session_id, devname***, **retry** [, ***fname*** [, ***vname***]]
Proceed with the dump, overwriting any existing contents	**sp_volchanged** ***session_id, devname***, **proceed** [, ***fname*** [, ***vname***]]

- `Volume to be overwritten on '`*`devname`*`' has unrecognized label data.`

  Dumps that specify the **init** option overwrite any existing contents of the tape. Backup Server issues this message if you try

to dump to a new tape or a tape with non-Sybase data without specifying the **init** option.

The Operator Can	By Entering
Abort the dump	**sp_volchanged *session_id, devname*, abort**
Mount another volume and retry the dump	**sp_volchanged *session_id, devname*, retry** [, ***fname*** [, ***vname***]]
Proceed with the dump, overwriting any existing contents of the volume	**sp_volchanged *session_id, devname*, proceed** [, ***fname*** [, ***vname***]]

Permissions

Any user can execute **sp_volchanged** to respond to a volume change request. This need not be the same user who started the dump or load.

Tables Used

master..sysdevices, sysobjects

See Also

Commands	**dump database**, **dump transaction**, **load database**, **load transaction**

sp_who

Function

Reports information about all current Adaptive Server users and processes or about a particular user or process.

Syntax

sp_who [*loginame* | "*spid*"]

Parameters

loginame – is the Adaptive Server login name of the user you are requesting a report on.

spid – is the number of the process you are requesting a report on. Enclose process numbers in quotes (Adaptive Server expects a *char* type).

Examples

1. sp_who

```
fid spid  status    loginame origname hostname blk  dbname
       cmd
--- ----- --------   -------- -------- -------- --- ------
       --------------
  0    1  recv sleep  bird     bird     jazzy    0   master
       AWAITING COMMAND
  0    2  sleeping    NULL     NULL             0   master
       NETWORK HANDLER
  0    3  sleeping    NULL     NULL             0   master
       MIRROR HANDLER
  0    4  sleeping    NULL     NULL             0   master
       AUDIT PROCESS
  0    5  sleeping    NULL     NULL             0   master
       CHECKPOINT SLEEP
  0    6  recv sleep  rose     rose     petal    0   master
       AWAITING COMMAND
  0    7  running     robert   sa       helos    0   master
       SELECT
  0    8  send sleep  daisy    daisy    chain    0   pubs2
       SELECT
  0    9  alarm sleep lily     lily     pond     0   master
       WAITFOR
  0   10  lock sleep  viola    viola    cello    7   pubs2
       SELECT
```

Reports on the processes running on Adaptive Server. Process 10 (a **select** on a table) is blocked by process 7 (a **begin transaction** followed by an **insert** on the same table).

For process 7, the current loginame is "robert", but the original loginame is "sa". Login "sa" executed a **set proxy** command to impersonate the user "robert".

2. sp_who victoria

Reports on the processes being run by the user "victoria".

3. sp_who "17"

Reports what Adaptive Server process number 17 is doing.

4. sp_who

```
fid spid  status    loginame origname hostname blk dbname
      cmd
---- ---- --------  -------- -------- -------- --- ------
      ----------------
  0    1  running   sa       sa       helos    0   master
      SELECT
  0    2  sleeping  NULL     NULL              0   master
      NETWORK HANDLER
  0    3  sleeping  NULL     NULL              0   master
      DEADLOCK TUNE
  0    4  sleeping  NULL     NULL              0   master
      MIRROR HANDLER
  0    5  sleeping  NULL     NULL              0   master
      HOUSEKEEPER
  0    6  sleeping  NULL     NULL              0   master
      CHECKPOINT SLEEP
```

Reports on the processes running on Adaptive Server. Although no user processes other than **sp_who** are running, the server still shows activity. During idle cycles, the housekeeper task moves dirty buffers into the buffer wash region.

Comments

- **sp_who** reports information about a specified user or Adaptive Server process. Without parameters, **sp_who** reports which users are running what processes in all databases.
- If you enable mirrored disks or remote procedure calls, the mirror handler and the site handler also appear in the report from **sp_who**.
- The "spid" column contains the process identification numbers that are used in the Transact-SQL **kill** command. The "blk" column

contains the process IDs of the blocking process, if there is one. A blocking process (which may be infected or have an exclusive lock) is one that is holding resources needed by another process. The "fid" column identifies the family (including the coordinating process and its worker processes) to which a lock belongs (see **sp_familylock** for more information).

- Running **sp_who** on a single-engine server shows the **sp_who** process "running" and all other processes "runnable" or in one of the sleep states. In multi-engine servers, there can be a "running" process for each engine.
- **sp_who** reports NULL in the "loginame" column for all system processes.
- Evaluation of a conditional statement such as an **if** or **while** loop appears as "COND" in the "cmd" column.
- System Administrators can remove many processes with the **kill** command.

Messages

- `No login with the specified name exists.`

  The name you supplied for the *loginame* parameter does not exist in Adaptive Server.

Permissions

Any user can execute **sp_who**.

Tables Used

master..sysprocesses

See Also

Commands	**kill**
System procedures	**sp_lock**

Catalog Stored Procedures

4 Catalog Stored Procedures

This chapter describes catalog stored procedures, which retrieve information from the system tables in tabular form.

Table 4-1 lists the catalog stored procedures that are covered in this chapter.

Table 4-1: Catalog stored procedures

Procedure	Description
sp_column_privileges	Returns permissions information for one or more columns in a table or view.
sp_columns	Returns information about the type of data that can be stored in one or more columns.
sp_databases	Returns a list of the databases in Adaptive Server.
sp_datatype_info	Returns information about a particular datatype or about all supported datatypes.
sp_fkeys	Returns information about foreign key constraints created in the current database with the **create table** or **alter table** command.
sp_pkeys	Returns information about primary key constraints created for a single table with the **create table** or **alter table** command.
sp_server_info	Returns a list of Adaptive Server attribute names and current values.
sp_special_columns	Returns the optimal set of columns that uniquely identify a row in a table or view; can also return a list of the columns that are automatically updated when any value in the row is updated by a transaction.
sp_sproc_columns	Returns information about a stored procedure's input and return parameters.
sp_statistics	Returns a list of indexes on a single table.
sp_stored_procedures	Returns information about one or more stored procedures.
sp_table_privileges	Returns privilege information for all columns in a table or view.
sp_tables	Returns a list of objects that can appear in a **from** clause.

Introduction to Catalog Stored Procedures

Catalog stored procedures retrieve information from the system tables in tabular form.

Like the system procedures, the catalog stored procedures, created by **installmaster** at installation, are located in the *sybsystemprocs* database and are owned by the System Administrator, but many of them can be run from any database.

If a catalog stored procedure is executed from a database other than *sybsystemprocs*, it retrieves information from the system tables in the database from which it was executed.

All catalog stored procedures execute at isolation level 1.

All catalog stored report a return status. For example:

```
return status = 0
```

means that the procedure executed successfully. The examples in this book do not include the return status.

Specifying Optional Parameters

If a parameter value for a catalog stored procedure contains punctuation or embedded blanks, or is a reserved word, you must enclose it in single or double quotes. If the parameter is an object name qualified by a database name or owner name, enclose the entire name in single or double quotes.

➤ ***Note***

Do not use delimited identifiers as catalog stored procedure parameters; they may produce unexpected results.

In many cases, it is more convenient to supply parameters to the catalog stored procedures in the form:

```
@parametername = value
```

than to supply all the parameters. The parameter names in the syntax statements match the parameter names defined by the procedures.

For example, the syntax for **sp_columns** is:

```
sp_columns table_name [, table_owner]
[, table_qualifier] [, column_name]
```

To use **sp_columns** to find information about a particular column, you can use:

```
sp_columns publishers, @column_name = "pub_id"
```

This provides the same information as the command with all of the parameters specified:

```
sp_columns publishers, "dbo", "pubs2", "pub_id"
```

You can also use "null" as a placeholder:

```
sp_columns publishers, null, null, "pub_id"
```

If you specify more parameters then the number of parameters expected by the system procedure, Adaptive Server ignores the extra parameters.

Pattern Matching

Adaptive Server offers a wide range of pattern matching through regular expressions. However, for maximum interoperability, assume only SQL standards pattern matching (the % and _ wildcard characters).

Procedure Messages

Catalog stored procedures return informational and error messages, which are listed with each procedure in this book. Catalog stored procedure error messages start with error number 17000.

Error messages from the functions and commands included in a procedure are documented in *Error Messages*.

System Procedure Tables

The catalog stored procedures **sp_columns**, **sp_datatype_info**, **sp_special_columns**, and **sp_sproc_columns** use the catalog stored procedure tables *spt_datatype_info*, *spt_datatype_info_ext*, and *spt_server_info* in the *sybsystemprocs* database to convert internal system values (for example, status bits) into human-readable format.

In addition, **sp_column_privileges** and **sp_table_privileges** create and then drop temporary tables.

ODBC Datatypes

Table 4-2 and Table 4-3 list the datatype code numbers and matching datatype names returned by **sp_columns** and **sp_sproc_columns** in the "data_type" column. The source for the description is the Open Database Connectivity (ODBC) Application Programming Interface (API).

Table 4-2: Code numbers for ODBC datatypes

Name	Type
char	1
decimal	3
double precision	8
float	6
integer	4
numeric	2
real	7
smallint	5
varchar	12

Table 4-3: Code numbers for extended datatypes

Name	Type
bigint	-5
binary (bit datatype)	-2
bit	-7
date	9
long varbinary	-4
long varchar	-1
time	10
timestamp	11
tinyint	-6
varbinary (bit-varying datatype)	-3

sp_column_privileges

Function

Returns permissions information for one or more columns in a table or view.

Syntax

```
sp_column_privileges table_name [, table_owner
   [, table_qualifier [, column_name]]]
```

Parameters

table_name – is the name of the table. The use of wildcard characters in pattern matching is not supported.

table_owner – is the name of the table owner. The use of wildcard characters in pattern matching is not supported. If you do not specify the table's owner, **sp_column_privileges** looks first for a table owned by the current user and then for a table owned by the Database Owner.

table_qualifier – is the name of the database. Values are the name of the current database and **null**.

column_name – is the name of the column whose permissions you want to display. Use wildcard characters to request information for more than one column. If you do not specify a column name, permissions information for all columns in the specified table is returned.

Examples

1. sp_column_privileges discounts, null, null, discounttype

```
table_qualifier     table_owner    table_name          column_name
     grantor   grantee     privilege     is_grantable
------------------- ------------- ------------------- ----------------
     --------- ----------- ------------ ------------
pubs2               dbo            discounts           discounttype
     dbo       dbo          SELECT       YES
pubs2               dbo            discounts           discounttype
     dbo       dbo          UPDATE       YES
pubs2               dbo            discounts           discounttype
     dbo       dbo           REFERENCE   YES
pubs2               dbo            discounts           discounttype
     dbo       guest       SELECT        NO
pubs2               dbo            discounts           discounttype
     dbo       guest       UPDATE        NO
pubs2               dbo            discounts           discounttype
     dbo       guest       REFERENCE     NO
```

Comments

- Table 4-4 describes the results set:

Table 4-4: Results set for sp_column_privileges

Column	Datatype	Description
table_qualifier	*varchar(32)*	The database name. This field can be NULL.
table_owner	*varchar(32)*	
table_name	*varchar(32)*	NOT NULL
column_name	*varchar(32)*	
grantor	*varchar(32)*	NOT NULL
grantee	*varchar(32)*	NOT NULL
privilege	*varchar(32)*	Identifies the column privilege. May be one of the following: SELECT - The grantee is permitted to retrieve data for the column. UPDATE - The grantee is permitted to update data in the column. REFERENCE - The grantee is permitted to refer to the column within a constraint (for example, a unique, referential, or table check constraint).

Table 4-4: Results set for sp_column_privileges (continued)

Column	Datatype	Description
is_grantable	*varchar(3)*	Indicates whether the grantee is permitted to grant the privilege to other users. The values are YES, NO, and NULL.

Messages

- `Catalog procedure sp_column_privileges can not be run in a transaction.`

  This procedure updates system tables, so it cannot be run from within a transaction.
- `Object name must be qualified with the owner name.`
- `Object name can only be qualified with owner name.`
- `This may be a temporary object. Please execute procedure from tempdb.`

  You invoked **sp_column_priviliges** for a table name beginning with "#". Execute the **use** *database_name* command to change to *tempdb*, and then rerun **sp_column_privileges**.
- `The table or view named doesn't exist in the current database.`

  The specified table or view does not exist. Check the spelling of the *table_name*.
- `The table does not have a column named` *`column_name`*`.`

  The specified column does not belong to the table.
- `Table qualifier must be name of current database.`

  sp_column_privileges cannot be used to return information about tables in another database. Execute the **use** *database_name* command to change to the correct database, and then rerun **sp_column_privileges**.

Permissions

Any user can execute **sp_column_privileges**.

Tables Used

syscolumns, sysobjects, sysusers

See Also

System procedures	**sp_help**, **sp_helprotect**

sp_columns

Function

Returns information about the type of data that can be stored in one or more columns.

Syntax

sp_columns *table_name* [, *table_owner*]
[, *table_qualifier*] [, *column_name*]

Parameters

table_name – is the name of the table or view. Use wildcard characters to request information about more than one table.

table_owner – is the owner of the table or view. Use wildcard characters to request information about tables owned by more than one user. If you do not specify a table owner, **sp_columns** looks first for tables owned by the current user and then for tables owned by the Database Owner.

table_qualifier – is the name of the database. This can be either the current database or NULL.

column_name – is the name of the column for which you want information. Use wildcard characters to request information about more than one column.

Examples

1. sp_columns "publishers", null, null, "p%"

```
table_qualifier               table_owner
     table_name                    column_name
     data_type type_name                    precision   length
     scale  radix  nullable
     remarks

     ss_data_type colid
----------------------------- -----------------------------
    ----------------------------- ----------------------------- --------
    ------------------------------ ----------- ----------- ------ ------
    --------
    ------------------------------------------------------------------------
------------------------------------------------------------------------
    ------------ -----
pubs2                         dbo
     publishers                    pub_id
```

```
            1 char                                    NULL           4
      NULL   NULL        0
     NULL

              47     1
pubs2                          dbo
     publishers                     pub_name
            12 varchar                                NULL          40
      NULL   NULL        1
     NULL

              39     2
```

Displays information about all columns in the *publishers* table that begin with "p".

2. **`sp_columns "s%", null, null, "st%"`**

   Displays information about all columns beginning with "st" in tables that begin with "s".

Comments

- Table 4-5 shows the results set:

Table 4-5: Results set for sp_columns

Column	Datatype	Description
table_qualifier	*varchar(32)*	The database name. This field can be NULL.
table_owner	*varchar(32)*	
table_name	*varchar(32)*	NOT NULL.
column_name	*varchar(32)*	NOT NULL.
data_type	*smallint*	Integer code for ODBC datatype. If this is a datatype that cannot be mapped into an ODBC type, it is NULL.
type_name	*varchar(30)*	String representing a datatype. The underlying DBMS presents this datatype name.
precision	*int*	Number of significant digits.
length	*int*	Length in bytes of a datatype.
scale	*smallint*	Number of digits to the right of the decimal point.
radix	*smallint*	Base for numeric types.
nullable	*smallint*	The value 1 means NULL is possible; 0 means NOT NULL.

Table 4-5: Results set for sp_columns (continued)

Column	Datatype	Description
remarks	*varchar(254)*	
ss_data_type	*smallint*	An Adaptive Server datatype.
colid	*tinyint*	A column appended to the results set.

Messages

- `Table qualifier must be name of current database.`

  sp_columns cannot be used to return information about tables in another database. Execute the **use** *database_name* command to change to the correct database, and then rerun **sp_columns**.

Permissions

Any user can execute **sp_columns.**

Tables Used

syscolumns, sysobjects, systypes, sybsystemprocs..spt_datatype_info

See Also

System procedures	**sp_help**

sp_databases

Function

Returns a list of databases in Adaptive Server.

Syntax

sp_databases

Parameters

None.

Examples

1. sp_databases

```
database_name       database_size     remarks
----------------    -------------     ------------
master                   5120         NULL
model                    2048         NULL
mydb                     2048         NULL
pubs2                    2048         NULL
sybsecurity              5120         NULL
sybsystemprocs          16384         NULL
tempdb                   2048         NULL
```

Comments

- Table 4-6 describes the results set:

Table 4-6: Results set for sp_databases

Column	Datatype	Description
database_name	*char*(32)	NOT NULL database name.
database_size	*int*	Size of database, in kilobytes.
remarks	*varchar*(254)	Adaptive Server always returns NULL.

Messages

None.

Permissions

Any user can execute **sp_databases**.

Tables Used

master..sysdatabases, *master..sysusages*, *sysobjects*

See Also

System procedures	**sp_helpdb**

sp_datatype_info

Function

Returns information about a particular ODBC datatype or about all ODBC datatypes.

Syntax

```
sp_datatype_info [data_type]
```

Parameters

data_type – is the code number for the specified ODBC datatype about which information is returned. Datatype codes are listed in Table 4-2 and Table 4-3 on page 4-4.

Comments

- Table 4-7 describes the results set:

Table 4-7: Results set for sp_datatype_info

Column	Datatype	Description
type_name	*varchar(30)*	A DBMS-dependent datatype name (the same as the *type_name* column in the **sp_columns** results set).
data_type	*smallint*	A code for the ODBC type to which all columns of this type are mapped.
precision	*int*	The maximum precision for the datatype on the data source. Zero is returned for datatypes where precision is not applicable.
literal_prefix	*varchar(32)*	Character(s) used to prefix a literal. For example, a single quotation mark (') for character types and 0x for binary.
literal_suffix	*varchar(32)*	Character(s) used to terminate a literal. For example, a single quotation mark (') for character types and nothing for binary.
create_params	*varchar(32)*	A description of the creation parameters for this datatype.
nullable	*smallint*	The value 1 means this datatype can be created allowing null values; 0 means it cannot.

Table 4-7: Results set for sp_datatype_info (continued)

Column	Datatype	Description
case_sensitive	*smallint*	The value 1 means all columns of this type are case sensitive (for collations); 0 means they are not.
searchable	*smallint*	The value 1 means columns of this type can be used in a **where** clause.
unsigned_attribute	*smallint*	The value 1 means the datatype is unsigned; 0 means the datatype is signed.
money	*smallint*	The value 1 means it is a money datatype; 0 means it is not.
auto_increment	*smallint*	The value 1 means the datatype is automatically incremented; 0 means it is not.
local_type_name	*varchar(128)*	Localized version of the data source dependent name of the datatype.

Messages

None.

Permissions

Any user can execute **sp_datatype_info**.

Tables Used

sybsystemprocs..spt_datatype_info, *systypes*, *sysdatabases*, *sysmessages*, *sysprocesses*

See Also

System procedures	**sp_help**

sp_fkeys

Function

Returns information about foreign key constraints created with the **create table** or **alter table** command in the current database.

Syntax

```
sp_fkeys pktable_name [, pktable_owner]
   [, pktable_qualifier] [, fktable_name]
   [, fktable_owner] [, fktable_qualifier]
```

Parameters

pktable_name – is the name of the primary key table. The use of wildcard characters in pattern matching is not supported. You must specify either the *pktable_name* or the *fktable_name*, or both.

pktable_owner – is the name of the primary key table owner. The use of wildcard characters in pattern matching is not supported. If you do not specify the table owner, **sp_fkeys** looks first for a table owned by the current user and then for a table owned by the Database Owner.

pktable_qualifier – is the name of the database that contains the primary key table. This can be either the current database or NULL.

fktable_name – is the name of the foreign key table. The use of wildcard characters in pattern matching is not supported. Either the *fktable_name* or the *pktable_name*, or both, must be given.

fktable_owner – is the name of the foreign key table owner. The use of wildcard characters in pattern matching is not supported. If an *fktable_owner* is not specified, **sp_fkeys** looks first for a table owned by the current user and then for a table owned by the Database Owner.

fktable_qualifier – is the name of the database that contains the foreign key table. This can be either the current database or **null**.

Comments

- **sp_fkeys** returns information about foreign key constraints created with the **create table** or **alter table** command in the current database.

A foreign key is a key column in a table that logically depends on a **primary key** column in another table.

- Table 4-8 describes the results set:

Table 4-8: Results set for sp_fkeys

Column	Datatype	Description
pktable_qualifier	*varchar(32)*	The database that contains the primary key table.
pktable_owner	*varchar(32)*	The owner of the primary key table.
pktable_name	*varchar(32)*	NOT NULL.
pkcolumn_name	*varchar(32)*	NOT NULL.
fktable_qualifier	*varchar(32)*	The database that contains the foreign key table.
fktable_owner	*varchar(32)*	The owner of the foreign key table.
fktable_name	*varchar(32)*	NOT NULL.
fkcolumn_name	*varchar(32)*	NOT NULL.
key_seq	*smallint*	NOT NULL. The sequence number of the column in a multi-column primary key.
update_rule	*smallint*	Action to be applied to the foreign key when the SQL operation is UPDATE. Zero is returned for this column.
delete_rule	*smallint*	Action to be applied to the foreign key when the SQL operation is DELETE. Zero is returned for this column.

- Both the primary key and foreign key must have been declared in a **create table** or **alter table** statement.
- If the primary key table name is supplied, but the foreign key table name is NULL, **sp_fkeys** returns all tables that include a foreign key to the given table. If the foreign key table name is supplied, but the primary key table name is NULL, **sp_fkeys** returns all tables that are related by a primary key / foreign key relationship to foreign keys in the foreign key table.
- **sp_fkeys** does not return information about keys declared with the **sp_commonkey**, **sp_foreignkey** or **sp_primarykey** system procedures.

Messages

- `Catalog procedure sp_fkeys can not be run in a transaction.`

  sp_fkeys updates system tables, so it cannot be run from within a transaction.

- `Foreign key table qualifier must be name of current database.`

  sp_fkeys cannot return information about tables in another database. Execute the **use** *database_name* command to change to the correct database, and then rerun **sp_fkeys**.

- `Primary key table qualifier must be name of current database.`

  sp_fkeys cannot return information about tables in another database. Execute the **use** *database_name* command to change to the correct database, and then rerun **sp_fkeys**.

- `Object does not exist in this database.`

  The specified primary key table or foreign key table does not exist in the current database. Check the spelling of the table name.

- `Primary key table name or foreign key table name or both must be given.`

  You must specify the name of the primary key table, the foreign key table, or both.

Permissions

Any user can execute **sp_fkeys**.

Tables Used

sysobjects, *sysreferences*

See Also

Commands	**alter table**, **create table**
System procedures	**sp_helpkey**

sp_pkeys

Function

Returns information about primary key constraints created with the **create table** or **alter table** command for a single table.

Syntax

```
sp_pkeys table_name [, table_owner]
   [, table_qualifier]
```

Parameters

table_name – is the name of the table. The use of wildcard characters in pattern matching is not supported.

table_owner – is the name of the table owner. The use of wildcard characters in pattern matching is not supported. If *table_owner* is not specified, **sp_pkeys** looks first for a table owned by the current user and then for a table owned by the Database Owner.

table_qualifier – is the name of the database that contains the table. This can be either the current database or NULL.

Comments

- Table 4-9 describes the results set:

Table 4-9: Results set for sp_pkeys

Column	Datatype	Description
table_qualifier	*varchar*(32)	The database name. This field can be NULL.
table_owner	*varchar*(32)	
table_name	*varchar*(32)	NOT NULL.
column_name	*varchar*(32)	NOT NULL.
key_seq	*smallint*	NOT NULL. The sequence number of the column in a multicolumn primary key.

- Primary keys must have been declared with the **create table** or **alter table** statement, not with the **sp_primarykey** system procedure.
- The term **primary key** refers to a logical primary key for a table. Adaptive Server expects that every logical primary key has a

unique index defined on it and that this unique index is also returned in **sp_statistics**.

Messages

- `Object does not exist in this database.`

  The specified primary key table or foreign key table does not exist in the current database. Check the spelling of the table name.

- `Table qualifier must be name of current database.`

  sp_pkeys cannot return information about tables in another database. Execute the **use** *database_name* command to change to the correct database, and then rerun **sp_pkeys**.

- `Catalog procedure sp_pkeys can not be run in a transaction.`

  This procedure updates system tables, so it cannot be run from within a transaction.

Permissions

Any user can execute **sp_pkeys**.

Tables Used

sysindexes, *sysobjects*

See Also

Commands	**alter table**, **create table**
System procedures	**sp_helpkey**

sp_server_info

Function

Returns a list of Adaptive Server attribute names and current values.

Syntax

sp_server_info [*attribute_id*]

Parameters

attribute_id – is the integer ID of the server attribute.

Examples

1. sp_server_info 12

```
attribute_id attribute_name           attribute_value
------------ ------------------------ -------------------------
          12 MAX_OWNER_NAME_LENGTH  0
```

2. sp_server_info

Returns the list of server attributes, described by the mandatory rows, and their values.

Comments

- Table 4-10 describes the results set:

Table 4-10: Results set for sp_server_info

Column	Datatype	Description
attribute_id	*int*	NOT NULL.
attribute_name	*varchar*(60)	NOT NULL.
attribute_value	*varchar*(255)	

- Table 4-11 shows the mandatory rows in the results set:

Table 4-11: Mandatory results returned by sp_server_info

ID	Server Attribute Name	Description	Value
1	DBMS_NAME	Name of the DBMS.	SQL SERVER
2	DBMS_VER	Version of the DBMS.	*@@version*

Table 4-11: Mandatory results returned by sp_server_info (continued)

ID	Server Attribute Name	Description	Value
6	DBE_NAME	Unused	
10	OWNER_TERM	Adaptive Server's term for a table owner (the second part of a three-part name).	`owner`
11	TABLE_TERM	Adaptive Server's term for a table (the third part of a three-part name).	`table`
12	MAX_OWNER_NAME_LENGTH	Maximum length of the name for a table owner (the second part of a three-part name).	`30`
16	IDENTIFIER_CASE	The case sensitivity of user-defined names (table names, column names, and stored procedure names) in the database (the case in which these objects are presented in the system catalogs).	`MIXED`
15	COLUMN_LENGTH	The maximum number of characters for a column name.	`30`
13	TABLE_LENGTH	The maximum number of characters for a table name.	`30`
100	USERID_LENGTH	The maximum number of characters for a user name.	`30`
17	TX_ISOLATION	The initial transaction isolation level the server assumes, corresponding to an isolation level defined in SQL92.	`2`
18	COLLATION_SEQ	The assumed ordering of the character set for this server.	
14	MAX_QUAL_LENGTH	Maximum length of the name for a table qualifier (the first part of a three-part table name).	`30`
101	QUALIFIER_TERM	Adaptive Server's term for a table qualifier (the first part of a three-part name).	`database`
19	SAVEPOINT_SUPPORT	Does the underlying DBMS support named savepoints?	`Y`
20	MULTI_RESULT_SETS	Does the underlying DBMS or the gateway itself support multiple results sets (can multiple statements be sent through the gateway, with multiple results sets returned to the client)?	`Y`
102	NAMED_TRANSACTIONS	Does the underlying DBMS support named transactions?	`Y`

Table 4-11: Mandatory results returned by sp_server_info (continued)

ID	Server Attribute Name	Description	Value
103	SPROC_AS_LANGUAGE	Can stored procedures be executed as language events?	Y
103	REMOTE_SPROC	Can stored procedures be executed through the remote stored procedure APIs in DB-Library?	Y
22	ACCESSIBLE_TABLES	In the **sp_tables** stored procedure, does the gateway return only tables, views, and so on, that are accessible by the current user (that is, the user who has at least **select** privileges for the table)?	Y
104	ACCESSIBLE_SPROC	In the **sp_stored_procedures** stored procedure, does the gateway return only stored procedures that are executable by the current user?	Y
105	MAX_INDEX_COLS	Maximum number of columns in an index for the DBMS.	16
106	RENAME_TABLE	Can tables be renamed?	Y
107	RENAME_COLUMN	Can columns be renamed?	Y
108	DROP_COLUMN	Can columns be dropped?	Y
109	INCREASE_COLUMN_LENGTH	Can column size be increased?	N
110	DDL_IN_TRANSACTION	Can DDL statements appear in transactions?	Y
111	DESCENDING_INDEXES	Are descending indexes supported?	N
112	SP_RENAME	Can a stored procedure be renamed?	Y
500	SYS_SPROC_VERSION	The version of the catalog stored procedures currently implemented.	01.01.2822

Messages

- `Attribute id` *`attribute_id`* `is not supported.`

  Check the number of the server attribute.

Permissions

Any user can execute **sp_server_info**.

Tables Used

sybsystemprocs..spt_server_info, sysobjects

See Also

Catalog stored procedures	**sp_stored_procedures**, **sp_tables**

sp_special_columns

Function

Returns the optimal set of columns that uniquely identify a row in a table or view; can also return a list of *timestamp* columns, whose values are automatically generated when any value in the row is updated by a transaction.

Syntax

```
sp_special_columns table_name [, table_owner]
   [, table_qualifier] [, col_type]
```

Parameters

table_name – is the name of the table or view. The use of wildcard characters in pattern matching is not supported.

table_owner – is the name of the table or view owner. The use of wildcard characters in pattern matching is not supported. If you do not specify the table owner, **sp_special_columns** looks first for a table owned by the current user and then for a table owned by the Database Owner.

table_qualifier – is the name of the database. This can be either the current database or NULL.

col_type – is **R** to return information about columns whose values uniquely identify any row in the table, or **V** to return information about *timestamp* columns, whose values are generated by Adaptive Server each time a row is inserted or updated.

Examples

1. sp_special_columns systypes

```
scope  column_name       data_type type_name     precision
         length      scale
 ------ --------------- --------- ------------- --------
        ----------- ------
      0 name                  12 varchar              30
              30   NULL
```

```
            2. sp_special_columns @table_name=authors, @col_type=R
scope  column_name         data_type type_name     precision
         length      scale
------ -------------------- --------- ------------ -----------
----------- ------
      0 au_id                       12 varchar                11
                 11   NULL
```

Comments

- Table 4-12 describes the results set:

Table 4-12: Results set for sp_special_columns

Column	Datatype	Description
scope	*int*	NOT NULL. Actual scope of the row ID. Adaptive Server always returns 0.
column_name	*varchar*(30)	NOT NULL. Column identifier.
data_type	*smallint*	The integer code for an ODBC datatype. If this datatype cannot be mapped to an ANSI/ISO type, the value is NULL. The native datatype name is returned in the *type_name* column. (See the ODBC datatypes Table 4-2 on page 4-4.)
type_name	*varchar*(13)	The string representation of the datatype. This is the datatype name as presented by the underlying DBMS.
precision	*int*	The number of significant digits.
length	*int*	The length in bytes of the datatype.
scale	*smallint*	The number of digits to the right of the decimal point.

Messages

- `There is no table named table_name in the current database.`

  The table does not exist in the current database. Check the spelling of the table name.

- `Table qualifier must be name of current database.`

  sp_special_columns cannot return information about tables in another database. Execute the **use** *database_name* command to change to the correct database, and then rerun **sp_special_columns**.

- `Illegal value for 'col_type' argument. Legal values are 'V' or 'R'.`

  Specify **V** or **R**.

Permissions

Any user can execute **sp_special_columns**.

Tables Used

sybsystemprocs..spt_datatype_info, syscolumns, sysindexes, sysobjects, systypes, sysusers

See Also

Datatypes	timestamp Datatype
System procedures	**sp_help**

sp_sproc_columns

Function

Returns information about a stored procedure's input and return parameters.

Syntax

```
sp_sproc_columns procedure_name [, procedure_owner]
   [, procedure_qualifier] [, column_name]
```

Parameters

procedure_name – is the name of the stored procedure. The use of wildcard characters in pattern matching is not supported.

procedure_owner – is the owner of the stored procedure. The use of wildcard characters in pattern matching is not supported. If you do not specify the owner of the procedure, **sp_sproc_columns** looks first for a procedure owned by the current user and then for a procedure owned by the Database Owner.

procedure_qualifier – is the name of the database. This can be either the current database or NULL.

column_name – is the name of the parameter about which you want information. If you do not supply a parameter name, **sp_sproc_columns** returns information about all input and return parameters for the stored procedure.

Comments

- Table 4-13 describes the results set:

Table 4-13: Results set for sp_sproc_columns

Column	Datatype	Description
procedure_qualifier	*varchar*(30)	
procedure_owner	*varchar*(30)	
procedure_name	*varchar*(41)	NOT NULL.
column_name	*varchar*(30)	NOT NULL.
column_type	*smallint*	

Table 4-13: Results set for sp_sproc_columns (continued)

Column	Datatype	Description
data_type	*smallint*	The integer code for an ODBC datatype. If this datatype cannot be mapped to an ANSI/ISO type, the value is NULL. The native datatype name is returned in the *type_name* column.
type_name	*char*(30)	The string representation of the datatype. This is the datatype name as presented by the underlying DBMS.
precision	*int*	The number of significant digits.
length	*int*	The length in bytes of the datatype.
scale	*smallint*	The number of digits to the right of the decimal point.
radix	*smallint*	Base for numeric types.
nullable	*smallint*	The value 1 means this datatype can be created allowing null values; 0 means it cannot.
remarks	*varchar*(254)	NULL .
ss_data_type	*tinyint*	An Adaptive Server datatype.
colid	*tinyint*	An Adaptive Server specific column appended to the result set.

Messages

- `Table qualifier must be name of current database.`

  sp_sproc_columns cannot return information about tables in another database. Execute the **use** *database_name* command to change to the correct database, and then rerun **sp_sproc_columns**.

Permissions

Any user can execute **sp_sproc_columns**.

Tables Used

sybsystemprocs..spt_datatype_info, *syscolumns*, *sysobjects*, *sysprocedures*, *systypes*

See Also

System procedures	**sp_help**, **sp_helptext**

sp_statistics

Function

Returns a list of indexes on a single table.

Syntax

```
sp_statistics table_name [, table_owner]
   [, table_qualifier] [, index_name] [, is_unique]
```

Parameters

table_name – is the name of the table. The use of wildcard character pattern matching is not supported.

table_owner – is the owner of the table. The use of wildcard character pattern matching is not supported. If *table_owner* is not specified, **sp_statistics** looks first for a table owned by the current user and then for a table owned by the Database Owner.

table_qualifier – is the name of the database. This can be either the current database or NULL.

index_name – is the index name. The use of wildcard character pattern matching is not supported.

is_unique – is **Y** to return only unique indexes; otherwise, is **N** to return both unique and nonunique indexes.

Examples

1. sp_statistics publishers

```
table_qualifier                 table_owner
    table_name                      non_unique
    index_qualifier                 index_name
    type   seq_in_index column_name                     collation
    cardinality pages
------------------------------ ------------------------------
   ------------------------------ ----------
   ------------------------------ --------------------------
   ------ ------------ ------------------------------ --------
   ----------- -----------
```

```
pubs2                          dbo
    publishers                     NULL
    NULL                           NULL
        0         NULL NULL                          NULL
             3           1
pubs2                          dbo
    publishers                           0
    publishers                     pubind
         1            1 pub_id                        A
              3           1
```

Comments

- Table 4-14 describes the results set:

Table 4-14: Results set for sp_statistics

Column	Datatype	Description
table_qualifier	*varchar*(32)	The database name. This field can be NULL.
table_owner	*varchar*(32)	
table_name	*varchar*(32)	NOT NULL.
non_unique	*smallint*	NOT NULL. The value 0 means unique, and 1 means not unique.
index_qualifier	*varchar*(32)	
index_name	*varchar*(32)	
type	*smallint*	NOT NULL. The value 0 means clustered, 2 means hashed, and 3 means other.
seq_in_index	*smallint*	NOT NULL.
column_name	*varchar*(32)	NOT NULL.
collation	*char*(1)	The value A means ascending; D means descending; and NULL means not applicable.
cardinality	*int*	Number of rows in the table or unique values in the index.
pages	*int*	Number of pages to store the index or table.

- The indexes in the results set appear in ascending order, ordered by the *non-unique*, *type*, *index_name*, and *seq_in_index* columns.

- The index type *hashed* accepts exact match or range searches, but searches involving pattern matching do not use the index.

Messages

- `Table qualifier must be name of current database.`

  sp_statistics cannot return information about tables in another database. Execute the **use** *database_name* command to change to the correct database, and then rerun **sp_statistics**.

- `Catalog procedure sp_statistics can not be run in a transaction.`

  sp_statistics modifies system tables, so it cannot be run within a transaction.

Permissions

Any user can execute **sp_statistics**.

Tables Used

syscolumns, *sysindexes*, *sysobjects*

See Also

System procedures	**sp_help**, **sp_helpindex**

sp_stored_procedures

Function

Returns information about one or more stored procedures.

Syntax

```
sp_stored_procedures [sp_name [, sp_owner
   [, sp_qualifier]]]
```

Parameters

sp_name – is the name of the stored procedure. Use wildcard characters to request information about more than one stored procedure.

sp_owner – is the owner of the stored procedure. Use wildcard characters to request information about procedures that are owned by more than one user.

sp_qualifier – is the name of the database. This can be the current database or NULL.

Comments

- **sp_stored_procedures** returns information about stored procedures in the current database only.
- Table 4-15 shows the results set:

Table 4-15: Results set for sp_stored_procedures

Column	Datatype	Description
procedure_qualifier	*varchar*(30)	The name of the database.
procedure_owner	*varchar*(30)	
procedure_name	*varchar*(41)	NOT NULL.
num_input_params	*int*	NOT NULL. Always returns -1.
num_output_params	*int*	NOT NULL. The value >= 0 shows the number of parameters; -1 means the number of parameters is indeterminate.
num_result_sets	*int*	NOT NULL. Always returns -1.
remarks	*varchar*(254)	NULL.

- **sp_stored_procedures** can return the name of stored procedures for which the current user does not have execute permission.owever, if the server attribute *accessible_sproc* is "Y" in the results set for **sp_server_info**, only stored procedures that are executable by the current user are returned.

Messages

- `Stored procedure qualifier must be name of current database.`

  sp_stored_procedures cannot return information about stored procedures in another database. Execute the **use** *database_name* command to change to the correct database, and then rerun **sp_stored_procedures**.

Permissions

Any user can execute **sp_stored_procedures**.

Tables Used

sysobjects, *sysprocedures*, *sysprotects*, *sysusers*

See Also

System procedures	**sp_help**, **sp_helptext**

sp_table_privileges

Function

Returns privilege information for all columns in a table or view.

Syntax

```
sp_table_privileges table_name [, table_owner
   [, table_qualifier]]
```

Parameters

table_name – is the name of the table. The use of wildcard characters in pattern matching is not supported.

table_owner – is the name of the table owner. The use of wildcard characters in pattern matching is not supported. If you do not specify the table owner, **sp_table_privileges** looks first for a table owned by the current user and then for a table owned by the Database Owner.

table_qualifier – is the name of the database. This can be either the current database or NULL.

Comments

- Table 4-16 shows the results set:

Table 4-16: Results set for sp_table_privileges

Column	Datatype	Description
table_qualifier	*varchar*(32)	The name of the database. This field can be NULL.
table_owner	*varchar*(32)	
table_name	*varchar*(32)	NOT NULL.
grantor	*varchar*(32)	NOT NULL.
grantee	*varchar*(32)	NOT NULL.

Table 4-16: Results set for sp_table_privileges (continued)

Column	Datatype	Description
privilege	*varchar*(32)	Identifies the table privilege. May be one of the following: SELECT - The grantee is permitted to retrieve data for one or more columns of the table. INSERT - The grantee is permitted to insert new rows containing data for one or more columns into the table. UPDATE - The grantee is permitted to update the data in one or more columns of the table. DELETE - The grantee is permitted to delete rows of data from the table. REFERENCE - The grantee is permitted to refer to one or more columns of the table within a constraint.
is_grantable	*varchar*(3)	Indicates whether the grantee is permitted to grant the privilege to other users. The values are YES, NO, and NULL.

Messages

- `Catalog procedure sp_table_privileges can not be run in a transaction.`

  sp_table_privileges updates system tables, so it cannot be run from within a transaction.

- `Object name can only be qualified with owner name.`
- `Object name must be qualified with the owner name.`
- `This may be a temporary object. Please execute procedure from tempdb.`

  You invoked **sp_table_priviliges** for a table name beginning with "#". Execute the **use** *database_name* command to change to *tempdb*, and then rerun **sp_table_priviliges.**

- `The table or view named doesn't exist in the current database.`

  The specified table does not exist in the current database. Check the spelling of the table name.

- `Table qualifier must be name of current database.`

  sp_table_privileges cannot return information about tables in another database. Execute the **use** *database_name* command to change to the correct database, and then rerun **sp_table_privileges**.

Permissions

Any user can execute **sp_table_privileges**.

Tables Used

sysobjects, *sysusers*

See Also

System procedures	**sp_help**, **sp_helprotect**

sp_tables

Function

Returns a list of objects that can appear in a **from** clause.

Syntax

```
sp_tables [table_name] [, table_owner]
   [, table_qualifier][, table_type]
```

Parameters

table_name – is the name of the table. Use wildcard characters to request information about more than one table.

table_owner – is the table owner. Use wildcard characters to request information about more than one table.

table_qualifier – is the name of the database. Acceptable values are the name of the current database and NULL.

table_type – is a list of values, separated by commas, giving information about all tables of the table type(s) specified, including the following:

```
"'TABLE', 'SYSTEM TABLE', 'VIEW'"
```

➤ ***Note***

Enclose each table type with single quotation marks, and enclose the entire parameter with double quotation marks. Enter table types in uppercase.

Examples

```
1. sp_tables @table_type = "'TABLE', 'VIEW'"
```

This procedure returns information about all tables in the current database of the type TABLE and VIEW and excludes information about system tables.

Comments

- Adaptive Server does not necessarily check the read and write permissions on *table_name.* Access to the table is not guaranteed, even if you can display information about it.

- The results set includes tables, views, and synonyms and aliases for gateways to DBMS products.
- If the server attribute *accessible_tables* is "Y" in the results set for **sp_server_info**, only tables that are accessible by the current user are returned.
- Table 4-17 describes the results set:

Table 4-17: Results set for sp_tables

Column	Datatype	Description
table_qualifier	*varchar*(30)	The database name. This field can be NULL.
table_owner	*varchar*(30)	
table_name	*varchar*(30)	NOT NULL. The table name.
table_type	*varchar*(32)	NOT NULL. One of the following: 'TABLE', 'VIEW', 'SYSTEM TABLE'.
remarks	*varchar*(254)	NULL

Messages

- `Table qualifier must be name of current database.`

  sp_tables cannot return information about tables in another database. Execute the **use** *database_name* command to change to the correct database, and then rerun **sp_tables**.

Permissions

Any user can execute **sp_tables**.

Tables Used

sysdatabases, *sysobjects*, *sysprotects*, *sysusers*

See Also

System procedures	**sp_help**
Catalog stored procedures	**sp_server_info**

System Extended Stored Procedures

5 System Extended Stored Procedures

This chapter describes the system extended stored procedures (ESPs), which are extended stored procedures supplied by Sybase.

Table 5-1 lists the system extended stored procedures discussed in this chapter.

Table 5-1: System extended stored procedures

Procedure	Description	Platform
xp_cmdshell	Executes a native operating system command on the host system running Adaptive Server.	All Supporting DLLs
xp_deletemail	Deletes a message from the Adaptive Server message inbox.	NT Only
xp_enumgroups	Displays groups for a specific Windows NT domain.	NT Only
xp_findnextmsg	Retrieves the message identifier of the next message in the Adaptive Server message inbox.	NT Only
xp_logevent	Provides for logging a user-defined event in the Windows NT Event Log.	NT Only
xp_readmail	Reads a message from the Adaptive Server message inbox.	NT Only
xp_sendmail	Sends a message to the specified recipients using the MAPI interface.	NT Only
xp_startmail	Starts an Adaptive Server mail session.	NT Only
xp_stopmail	Stops an Adaptive Server mail session.	NT Only

Introduction

The system extended stored procedures, created by **installmaster** at installation, are located in the *sybsystemprocs* database and are owned by the System Administrator. They can be run from any database.

Permissions on System ESPs

Since system extended stored procedures are located in the *sybsystemprocs* database, their permissions are also set there.

Users with the **sa_role** have default execution permissions on the system ESPs. These System Administrators can grant execution permissions to other users.

DLLs associated with System ESPs

You can get the names of the DLLs associated with the system ESPs by running **sp_helpextendedproc** in the *sybsystemprocs* database.

Using System ESPs

The system ESPs follow the same calling conventions as the regular system procedures.

The only additional requirement for system ESPs is that the Open Server application, XP Server, must be running. Adaptive Server starts XP Server the first time an ESP is invoked. XP Server continues to run until you shut down Adaptive Server.

xp_cmdshell

Function

Executes a native operating system command on the host system running Adaptive Server.

Syntax

```
xp_cmdshell command [, no_output]
```

Parameters

command – is the operating system command string; maximum length is 255 bytes.

no_output – if specified, suppresses any output from the command.

Examples

1. `xp_cmdshell 'copy C:\log A:\log.0102', no_output`

   Silently copies the file named *log* on the C drive to a file named *log.0102* on the A drive.

2. `xp_cmdshell 'date'`

   Executes the operating system's **date** command and returns the current date as a row of data.

Comments

- **xp_cmdshell** returns any output, including operating system errors, as rows of text in a single column.
- **xp_cmdshell** is run from the current directory of the XP Server.
- The width of the column of returned output is 80 characters. The output is not formatted.
- **xp_cmdshell** cannot perform commands that require interaction with the user, such as "login".
- The user context in which an operating system command is executed via **xp_cmdshell** is controlled by the value of the **xp_cmdshell context** configuration parameter. If this parameter is set to 1 (the default), **xp_cmdshell** restricts permission to users with System Administration privileges at the operating system level. If this parameter is set to 0, **xp_cmdshell** uses the security context of the operating system account under which Adaptive Server is running. Therefore, using **xp_cmdshell** with the **xp_cmdshell context**

configuration parameter set to 0, any user can execute operating system commands using the permissions of the account running Adaptive Server. This account may have fewer restrictions than the user's own account.

See "xp_cmdshell context" in Chapter 11, "Setting Configuration Parameters," in the *System Administration Guide* for details about **xp_cmdshell context**.

- Regardless of the value of **xp_cmdshell context**, if the user who is executing **xp_cmdshell** is not a System Administrator (does not have the **sa_role**), a System Administrator must have granted that user explicit permission to execute **xp_cmdshell**. For example, the following statement grants "joe" permission to execute **xp_cmdshell**:

  ```
  grant execute on xp_cmdshell to joe
  ```

Messages

- `Cannot allocate memory. Check the XP Server log file.`

  XP Server cannot allocate enough memory to execute the command.

- `Invalid number of parameters specified. Check the documentation for minimum and maximum number of parameters.`

  You specified an incorrect number of parameters. The minimum is one parameter; the maximum is two parameters.

- `Invalid parameter value specified. Refer to the documentation for the correct value(s).`

  You specified an incorrect parameter value.

- `User access denied. Not a member of the NT administrators group.`

  You do not have permission to execute this procedure (Windows NT only).

Permissions

By default, only a System Administrator can execute **xp_cmdshell**. A System Administrator can grant execute permission to other users.

See Also

System procedures	**sp_configure**

xp_deletemail

(Windows NT only)

Function

Deletes a message from the Adaptive Server message inbox.

Syntax

```
xp_deletemail [msg_id]
```

Parameters

msg_id – is the message identifier of the mail message to be deleted.

Examples

1.
```
declare @cur_msg_id binary(255)
xp_deletemail @msg_id = @cur_msg_id
```

   Deletes from the Adaptive Server message inbox the message with the message identifier specified in the *cur_msg_id* variable.

2.
```
xp_deletemail
```

   Deletes the first message from the Adaptive Server message inbox.

Comments

- Obtain the *msg_id* using **xp_findnextmsg**.
- If the *msg_id* parameter is not used, the message to be deleted defaults to the first message in the message inbox.

Messages

- `Invalid parameter param received. Check user documentation, and reenter the command.`

  Make sure the parameter is a valid message identifier.

- `No message retrieved and/or deleted because none found.`

  Check to see that you are specifying a valid message identifier.

- `The function xp_deletemail received an invalid number of parameters. Check user documentation and reenter the command.`

  xp_deletemail takes one parameter or zero parameters.

- The mail session was not active. Start the mail session using xp_startmail before calling xp_deletemail.

Permissions

By default, only a System Administrator can execute **xp_deletemail**. A System Administrator can grant this permission to other users.

See Also

System ESPs	**xp_findnextmsg**, **xp_startmail**
System procedures	**sp_processmail**

xp_enumgroups

(Windows NT only)

Function

Displays groups for a specified Windows NT domain.

Syntax

xp_enumgroups [*domain_name*]

Parameters

domain_name – is the Windows NT domain for which you are listing user groups.

Examples

1. **xp_enumgroups**

   Lists all user groups on the Windows NT computer running XP Server.

2. **xp_enumgroups 'PCS'**

   Lists all user groups in the PCS domain.

Comments

- **xp_enumgroups** displays all local user groups if no parameter is passed.
- A **domain** is a named collection of computers that share a common user account database and security policy.
- A return status of 0 indicates success; 1 indicates failure.

Messages

- `Invalid number of parameters specified. Check the documentation for minimum and maximum number of parameters.`

  An incorrect number of parameters was passed to **xp_enumgroups**. This occurs if you specify less than one or more than two parameters.

- `Invalid parameter value specified. Refer to the documentation for the correct value(s).`

  An incorrect parameter type was passed to **xp_enumgroups**. This occurs if you specify a number.

Permissions

By default, only a System Administrator can execute **xp_enumgroups**. A System Administrator can grant this permission to other users.

xp_findnextmsg

(Windows NT only)

Function

Retrieves the next message identifier from the Adaptive Server message inbox.

Syntax

```
xp_findnextmsg @msg_id = @msg_id output [, type]
   [, unread_only = {true | false}]
```

Parameters

msg_id – on input, specifies the message identifier that immediately precedes the one you are trying to retrieve. Places the retrieved message identifier in the *msg_id* output parameter, which must be of type binary.

type – is the input message type based on the MAPI mail definition. The only supported message type is **CMC:IPM**. A NULL value or no value defaults to **CMC:IPM**.

unread_only – if this parameter is set to **true**, **xp_findnextmsg** considers only unread messages. If this parameter is set to **false**, **xp_findnextmsg** considers all messages, both read and unread, when retrieving the next message identifier. The default is **true**.

Examples

1. ```
   xp_findnextmsg @msg_id = @out_msg_id output
   ```

   Returns, in the *@out_msg_id* output variable, the message identifier of the next unread message after the message specified by the *@out_msg_id*.

2. ```
   xp_findnextmsg @msg_id = @out_msg_id output, NULL,
   @unread_only = false
   ```

   Returns, in the *@out_msg_id* output variable, the message identifier of the next message after the message specified by the *@out_msg_id*. The message may be read or unread.

Comments

- When **xp_findnextmsg** can find no more messages in the inbox, it returns a status of 1.

- **xp_deletemail** and **xp_readmail** use the message identifier returned by **xp_findnextmsg**.

Messages

- `Invalid parameter param received. Check user documentation and reenter the command.`

  Check to see that you are passing a valid message identifier for the first parameter. If you specify a type, it must be **CMC:IPM**. If you set **unread_only**, it must be **true** or **false**.

- `The function xp_findnextmsg received an invalid number of parameters. Check user documentation and reenter the command.`

  xp_findnextmsg takes a maximum of three parameters.

- `The mail session was not active. Start the mail session using xp_startmail before calling xp_findnextmsg.`

- `The parameter param is not an output parameter.`

  msg_id must be an output parameter.

- `The required parameter param is missing. Check user documentation and reenter the command.`

  You must specify a *msg_id*.

- `The value of parameter param is invalid. Check user documentation for correct value.`

  Check to see that you are passing a valid message identifier for the first parameter. If you specify a type, it must be **CMC:IPM**. If you set **unread_only**, it must be **true** or **false**.

Permissions

By default, only a System Administrator can execute **xp_findnextmsg**. A System Administrator can grant this permission to other users.

See Also

System ESPs	**xp_deletemail**, **xp_readmail**, **xp_startmail**
System procedures	**sp_processmail**

xp_logevent

(Windows NT only)

Function

Provides for logging a user-defined event in the Windows NT Event Log from within Adaptive Server.

Syntax

```
xp_logevent error_number, message [, type]
```

Parameters

error_number – is the user-assigned error number. It must be equal to or greater than 50000.

message – is the text of the message that is displayed in the description field of the event viewer. The maximum length of the message is 255 bytes. Enclose the message in quotes.

type – describes the urgency of the event. Values are **informational**, **warning**, and **error**. The default is **informational**. Enclose the value in quotes.

Examples

1. ```
   xp_logevent 55555, 'Email message deleted.'
   ```

   An informational event, number 55555, will be logged in the Windows NT Event Log. The text of the description in the event detail window is “Email message deleted”.

2. ```
   xp_logevent 66666, 'DLL not found.', 'error'
   ```

   An error event, number 66666, will be logged in the Windows NT Event Log. The text of the description in the event detail window is “DLL not found”.

Comments

- The following table describes the default event details for events generated with **xp_logevent**:

Detail	Value
User	N/A
Computer	Name of machine running XP Server
Event ID	12
Source	Name of Adaptive Server
Category	**User**

Messages

- `Invalid number of parameters. Error number and message must be provided.`
- `Invalid parameter` *`param`* `received. Check user documentation and reenter the command.`
- `Invalid data type. An integer value must be supplied for error num.`
- `Invalid data type. A VARCHAR value must be supplied for message.`
- `Invalid data type. A VARCHAR value must be supplied for type.`
- `Event logging is not enabled. Cannot log event in event log`

  Event logging is not enabled for this session. You can enable event logging by setting the **event logging** parameter to 1 using **sp_configure**.

Permissions

Only a System Administrator can execute **xp_logevent**.

xp_readmail

(Windows NT only)

Function

Reads a message from the Adaptive Server message inbox.

Syntax

```
xp_readmail [msg_id]
   [, recipients output]
   [, sender output]
   [, date_received output]
   [, subject output]
   [, cc output]
   [, message output]
   [, attachments output]
   [, suppress_attach = {true | false}]
   [, peek = {true | false}]
   [, unread = {true | false}]
   [, msg_length output]
   [, bytes_to_skip [output]]
   [, type [output]]
```

Parameters

msg _id – specifies the message identifier of the message to be read by **xp_readmail**. If the *msg_id* parameter is not used, the message defaults to the first unread message in the message box, if **unread** is **true**, or to the first message in the message box, if **unread** is **false**.

recipients – is a semicolon-separated list of the recipients of the message.

sender – is the originator of the message.

date_received – is the date the message was received.

subject – is the subject header of the message.

cc – is a list of the message's copied (cc'd) recipients (separated by semicolons).

message – is the text of the message body. If the length of the message body, obtained from the *msg_length* output parameter, is greater than 255, use the *byte_to_skip* and *msg_length* parameters to read the message in 255-byte increments.

attachments – is a list of the temporary paths of the attachments (separated by semicolons). *attachments* is ignored if *supress_attach* is **true**.

suppress_attach – if set to **true**, prevents the creation of temporary files for attachments. The default is **true**.

peek– if set to **false**, flags the message as unread after it has been read. If set to **true**, flags the message as an unread message, even after it has been read. The default is **false**.

unread_only – if set to **true**, **xp_readmail** considers only unread messages. If set to **false**, **xp_readmail** considers all messages, whether they are flagged as read or unread. The default is **true**.

msg_length – is the total length of the message, in bytes. Used with the *bytes_to_skip* parameter, allows **xp_readmail** to read messages in 255-byte increments.

bytes_to_skip – on input, if not 0, specifies the number of bytes to skip before reading the next 255 bytes of the message into the **message** output parameter. On output, contains the offset in the message (the previous value of *bytes_to_skip* plus the *msg_length* that is output with the call) from which to start reading the next 255-byte increment.

type – is the message type based on the MAPI mail definition. The only supported message type is **CMC:IPM**. A NULL value or no value defaults to **CMC:IPM**.

Examples

1.
```
declare @msgid binary(255)
declare @originator varchar(20)
declare @mess varchar(255)
exec xp_findnextmsg @msg_id = @msgid output
exec xp_readmail @msg_id = @msgid,
@sender = @originator output,
@message = @mess output
```

xp_readmail reads the first unread message in the message inbox. It gets the message identifier for this message from the *@msgid* variable, where it has been stored by the **xp_findnextmsg** ESP. **xp_readmail** stores the sender's name in the *@originator* variable and the message body in the *@mess* variable.

2.
```
declare @msgid binary(255)
declare @mess varchar(255)
declare @len int
declare @skip int = 0
exec xp_findnextmsg @msgid output
exec xp_readmail @msg_id = @msgid,
@message = @mess output
@msg_length = @len output,
@bytes_to_skip = @skip output
print @mess
if (@len > 255)
begin
    while (@skip < @len)
    begin
        xp_readmail @msg_id = @msgid,
        @message = @mess output,
        @bytes_to skip = @skip output
        print @mess
    end
end
```

   Reads the first 255 bytes of the message for which the message identifier is output by **xp_findnextmsg**. If the total length of the message exceeds 255 bytes, reads the next 255 bytes and continues until there are no more bytes to read.

Comments

- **xp_readmail** reads a message from the Adaptive Server message inbox.
- To get the message identifier of the next message in the message inbox, use **xp_findnextmsg**.

Messages

- `Invalid parameter` *`param`* `received. Check user documentation and reenter the command.`
- `Invalid number of parameters specified. Check the documentation for minimum and maximum number of parameters.`
- `No message retrieved and/or deleted because none found.`

  Check to see that you are passing a valid message identifier as an input parameter.

- The mail session was not active. Start the mail session using xp_startmail before calling xp_readmail.
- The value of parameter *param* is invalid. Check user documentation for correct value.

Permissions

By default, only a System Administrator can execute **xp_readmail**. A System Administrator can grant this permission to other users.

See Also

System ESPs	**xp_deletemail**, **xp_findnextmsg**, **xp_sendmail**, **xp_startmail**
System procedures	**sp_processmail**

xp_sendmail

(Windows NT only)

Function

Sends a message to the specified recipients. The message is either text or the results of a Transact-SQL query.

Syntax

```
xp_sendmail recipient [; recipient] . . .
   [, subject]
   [, cc_recipient] . . .
   [, bcc_recipient] . . .
   [, {query | message}]
   [, attachname]
   [, attach_result = {true | false}]
   [, echo_error = {true | false}]
   [, include_file [, include_file] . . .]
   [, no_column_header = {true | false}]
   [, width]
   [, separator]
   [, dbuser]
   [, dbname]
   [, type]
   [, include_query = {true | false}]
```

Parameters

recipient – is the email address of the user who will receive the message. At least one recipient is required. Separate multiple recipients with semicolons.

subject – is the optional message subject header. If not used, defaults to “Sybase SQL Server Message”.

cc_recipient – is a list of the message’s copied (cc’d) recipients (separated by semicolons).

bcc_recipient – is the list of the message’s blind- copied (bcc’d) recipients (separated by semicolons).

query – is one or more Transact-SQL statements. The results are sent to the recipients of the message. If *query* is used, *message* cannot be used.

message – is the text of the message being sent. If *message* is used, *query* cannot be used.

attachname – is the name of the file containing the results of a query, which is included as an attachment to the message, when the *query* parameter is used. If *attachname* is used, *attach_result* must be set to **true**. If *attach_result* is **true** and *attachname* is not specified, the the prefix of the attached file's generated file name is "syb" followed by 5 random digits followed by the ".txt" extension; for example, *syb84840.txt*. This parameter is ignored if the *message* parameter is used.

attach_result – if set to **true**, sends the results of a query as an attachment to the message. If set to **false**, sends the results directly in the message body. The default is **false**. This parameter is ignored if the *message* parameter is used.

echo_error – if set to **true**, sends Adaptive Server messages, including the count of rows affected message, along with the query results. If set to **false**, does not send Adaptive Server messages. The default is **true**. This parameter is ignored if the *message* parameter is used.

include_file – is a list of files to be included as attachments to the message, separated by semicolons. The files can be specified as file names, path names, or relative path names and can be either text or binary files.

no_column_header – if set to **true**, column headers are sent with query results. If set to **false**, column headers are not sent. The default is **false**. This parameter is ignored if the *message* parameter is used.

no_output – if set to **true**, no output is sent to the session that sent the mail. If set to **false**, the session sending the mail receives output. The default is **false**. This parameter is ignored if the *message* parameter is used.

width – specifies, in characters, the width of the results sets when query results are sent in a message. *width* is the same as the **/w** option in **isql**. Result rows are broken by the newline character when the specified *width* is reached. The default is 80 characters. This parameter is ignored if the *message* parameter is used.

separator – specifies the character to be used as a column separator when query results are sent in a message. *separator* is the same as the **/s** option in **isql**. The default is the tab character. This parameter is ignored if the *message* parameter is used.

dbuser – specifies the database user name to be assumed for the user context for executing queries when the *query* parameter is used. The default is "guest." This parameter is ignored if the *message* parameter is used.

dname – specifies the database name to be assumed for the database context for executing queries when the *query* parameter is used. The default is "master." This parameter is ignored if the *message* parameter is used.

type – is the input message type based on the MAPI mail definition. The only supported message type is **CMC:IPM**. A NULL value or no value defaults to **CMC:IPM**.

include_query – if set to **true**, the query or queries used in the *query* parameter are appended to the results set. If set to **false**, the query is not appended. The default is **false**. *include_query* is ignored if the *message* parameter is used.

Examples

1. ```
   xp_sendmail @recipient = "sally";"ramon",
   @subject = "Adaptive Server Backup Status",
   @message = "Adaptive Server Backup for SERVER2 is
   complete.",
   @copy_recipient="admin"
   ```

   xp_sendmail sends a text message on the backup status of an Adaptive Server to "sally" and "ramon" with a copy to the "admin" group.

2. ```
   xp_sendmail "peter",
   @query = "select * from authors",
   @attachname = "au_list.res",
   @attach_result= true
   ```

   Sends "peter" the results of a query on the *authors* table. The results are in an attachment to the message, which consists of a file named *au_lis.res*, which is in the directory from which the server is being executed.

Comments

- The following parameters are related to the results of queries sent in a message when the *query* parameter is used. They are ignored if the *message* parameter is used instead: *attachname*, *attach_result*, *echo_error*, *no_column_header*, *no_output*, *width*, *separator*, *dbuser*, *dname*, *include_query*.

Messages

- Invalid parameter *param* received. Check user documentation and reenter the command.
- The combination of input parameters is invalid. Check user documentation and reenter the command.

  The *attachname*, *attach_result*, *echo_error*, *no_column_header*, *no_output*, *width*, *separator*, *dbuser*, *dname*, and *include_query* parameters are ignored if the *message* parameter is used.
- The function xp_sendmail expected some recipients, but none were found. Reenter the command with a value for *@recipient*.
- The function xp_sendmail received an invalid number of parameters. Check user documentation and reenter the command.
- The mail session was not active. Start the mail session using xp_startmail before calling xp_sendmail.
- The value of parameter *param* is invalid. Check user documentation for correct value.

Permissions

By default, only a System Administrator can execute **xp_sendmail**. A System Administrator can grant this permission to other users.

See Also

System ESPs	**xp_deletemail**, **xp_findnextmsg**, **xp_readmail**, **xp_startmail**
System procedures	**sp_processmail**

xp_startmail

(Windows NT only)

Function

Starts an Adaptive Server mail session.

Syntax

```
xp_startmail [mail_user] [, mail_password]
```

Parameters

mail_user – is a mail profile name used by Adaptive Server to log into the Windows NT mail system. If *mail_user* is not used, **xp_startmail** uses the mail user name that was used to set up Sybmail's Adaptive Server account.

mail_password – is the mail password used by Adaptive Server to log into the Windows NT mail system. If *mail_password* is not used, **xp_startmail** uses the mail password that was used to set up Sybmail's Adaptive Server account.

Examples

1. `xp_startmail`

   Starts an Adaptive Server mail session using the mail user name and password for Sybmail's user account.

2. `xp_startmail "mailuser", "tre55uu"`

   Starts an Adaptive Server mail session with "mailuser" as the profile name and the password associated with that profile name.

Comments

- **xp_startmail** will not start an Adaptive Server mail session if one is already running.
- An Adaptive Server mail session must be started, either by an explicit call to **xp_startmail** or by configuring Adaptive Server to start an Adaptive Server mail session automatically at start-up, before any Sybmail-related system ESPs or the **sp_processmail** stored procedure can be executed. For information about initiating an Adaptive Server mail session automatically at start-up, see **start mail session** in Chapter 8, "System Tables," in the *System Administration Guide.*

- To see the default *mail_user* value from the *fullname* field for the "sybmail" user account, use the **sp_displaylogin** system procedure as follows:

```
sp_displaylogin sybmail
```

Messages

- `Invalid parameter param received. Check user documentation and reenter the command.`

  You provided an incorrect value for the *mail_user* or *mail_password* parameter.

- `Sybmail encountered CMC Error error.`

  You provided an incorrect value for the *mail_user* or *mail_password* parameter.

- `The function xp_startmail received an invalid number of parameters. Check user documentation and reenter the command.`

- `The mail session was already active when xp_startmail was invoked. Invocation of this ESP will have no effect.`

Permissions

By default, only a System Administrator can execute **xp_startmail**. A System Administrator can grant this permission to other users.

See Also

System ESPs	**xp_stopmail**
System procedures	**sp_configure**

xp_stopmail

(Windows NT only)

Function

Stops an Adaptive Server mail session.

Syntax

```
xp_stopmail
```

Parameters

None.

Examples

1. `xp_stopmail`

Stops an Adaptive Server mail session.

Comments

- Sybmail-related system ESPs and the **sp_processmail** stored procedure cannot be executed after an Adaptive Server mail session has been terminated with **xp_stopmail**.

Messages

- `Invalid parameter` *`param`* `received. Check user documentation and reenter the command.`
- `The mail session was not active. Start the mail session using xp_startmail before calling xp_stopmail.`
- `Sybmail encountered CMC Error` *`error`*`.`
- `The function xp_stopmail received an invalid number of parameters. Check user documentation and reenter the command.`

Permissions

By default, only a System Administrator can execute **xp_stopmail**. A System Administrator can grant this permission to other users.

See Also

System ESPs	**xp_startmail**
System procedures	**sp_processmail**

DBCC Stored Procedures

6 *dbcc* Stored Procedures

This chapter describes the **dbcc** stored procedures. These procedures access the tables only in the *dbccdb* database or in the alternate database, *dbccalt*. For details on setting up *dbccdb* or *dbccalt*, see Chapter 18, "Checking Database Consistency," in the *System Administration Guide*. For information on the tables in these databases, see Chapter 19, "dbccdb Tables" in the *System Administration Guide*.

Table 6-1 lists the **dbcc** stored procedures described in this chapter. For details on the **dbcc** system procedure **sp_plan_dbccdb**, see **sp_plan_dbccdb**. For more information on this system procedure and the **dbcc** stored procedures, see Chapter 18, "Checking Database Consistency," in the *System Administration Guide*.

Table 6-1: dbcc stored procedures

Procedure Name	Description
sp_dbcc_alterws	Changes the size of the specified workspace to a specified value, and initializes the workspace.
sp_dbcc_configreport	Generates a report that describes the configuration information used by the **dbcc checkstorage** operation for the specified database.
sp_dbcc_createws	Creates a workspace of the specified type and size on the specified segment and database.
sp_dbcc_deletedb	Deletes from *dbccdb* all the information related to the specified target database.
sp_dbcc_deletehistory	Deletes the results of **dbcc checkstorage** operations performed on the target database before the specified date and time.
sp_dbcc_differentialreport	Generates a report that highlights the changes in I/O statistics and faults that took place between two **dbcc** operations
sp_dbcc_evaluatedb	Recomputes configuration information for the target database and compares it to the current configuration information.

Table 6-1: dbcc stored procedures (continued)

Procedure Name	Description
sp_dbcc_faultreport	Generates a report covering fault statistics for the **dbcc checkstorage** operations performed for the specified object in the target database on the specified date.
sp_dbcc_fullreport	Runs **sp_dbcc_summaryreport**, **sp_dbcc_configreport**, **sp_dbcc_statisticsreport**, and **sp_dbcc_faultreport short** for *database..object_name* on or before the specified *date*.
sp_dbcc_runcheck	Runs **dbcc checkstorage** on the specified database, and then runs **sp_dbcc_summaryreport** or a report you specify
sp_dbcc_statisticsreport	Generates an allocation statistics report on the specified object in the target database.
sp_dbcc_summaryreport	Generates a summary report on the specified database. .
sp_dbcc_updateconfig	Updates the *dbcc_config* table in *dbccdb* with the configuration information of the target database.

Specifying the Object Name and Date

Several **dbcc** stored procedures use parameters for the object name and date. This section provides important information on specifying the object name and date.

Specifying the Object Name

The object name specifies only the name of the table or index for which to generate a report. When you specify an object name, you must also specify a database name (*dbname*). You cannot specify an owner for the object. If the specified object name is not unique in the target database, the system procedure generates a report on all objects with the specified name.

Specifying the Date

Use the following syntax to specify the date and time (optional):

```
mm/dd/yy[:hh:mm:ss]
```

A 24-hour clock is assumed.

When you specify the date, the system procedures interpret it as follows:

- If both the date and the time are specified, the **dbcc** operation that completed at the specified date and time is selected for the report.
- If the specified date is the current date, and no time is specified, the time is automatically set to the current time. The **dbcc** operation that completed within the previous 24 hours with a finish time closest to the current time is selected for the report.
- If the specified date is not the current date, and no time is specified, the time is automatically set to "23:59:59". The **dbcc checkstorage** operation that completed with a finish date and time closest to the specified date and system-supplied time is selected for the report.

For example, suppose the most recent **dbcc checkstorage** operation completed on March 4, 1997 at 10:20:45.

If you specify the date as "03/04/97", the system procedure interprets the date as 03/04/97:23:59:59. This date and time are compared to the actual finish date and time of 03/04/97:10:20:45.

If you specify the date as "03/04/97:10:00:00", the operation that completes at 10:20:45 is not selected for the report because only the operations that complete on or before the specified time meet the criteria.

If you specify the date as "03/06/97", no report is generated because the most recent operation completed more than 24 hours earlier.

sp_dbcc_alterws

Function

Changes the size of the specified workspace to a specified value, and initializes the workspace.

Syntax

```
sp_dbcc_alterws dbname, wsname, "wssize[K|M]"
```

Parameters

dbname – is the name of the database in which the workspace resides. Specify either **dbccdb** and **dbccalt.**

wsname – specifies the name of the workspace to alter.

wssize – is the new size of the workspace, specified by **K** (kilobytes) or **M** (megabytes). If you do not specify **K** or **M**, *wssize* specifies the number of pages. Page size is platform-dependent. The minimum size for a workspace is 24 pages.

Example

```
1. sp_dbcc_alterws dbccdb, scan_ws_000001, "30M"
```

```
Workspace scan_ws_000001 has been altered
successfully to size 30MB
```

Changes the size of the *scan_ws_000001* workspace on *dbccdb* to 30MB.

Comments

- **sp_dbcc_alterws** changes the size of the specified workspace to the specified value and initializes the workspace.
- To achieve maximum performance, make sure you have configured a buffer pool of at least 16K before you alter a workspace.
- Use **sp_plan_dbccdb** to determine size estimates before altering the workspace.
- The workspace must exist before it can be altered. For information on creating workspaces, see **sp_dbcc_createws**.
- For more information on the *scan* and *text* workspaces, see Chapter 18, “Checking Database Consistency,” and Chapter 19, “dbccdb Tables,” in the *System Administration Guide*. For

information on the *dbccalt* database, see Chapter 18, "Checking Database Consistency," in the *System Administration Guide.*

Messages

- *dbname* `is not a valid DBCC database. Database must be either dbccdb or dbccalt.`

  Specify either **dbccdb** or **dbccalt** for the database name.

- `DBCC database name cannot be null.`

  Specify either **dbccdb** or **dbccalt** for the database name.

- `No such database -- run sp_helpdb to list databases.`

  The name you specified for the target database does not exist. Use **sp_helpdb** to list the databases.

- `Workspace name cannot be null.`

  You must specify a workspace. Run **sp_help** to list all the workspaces.

- `Workspace size must be a valid number and must be at least` *n*`KB`

  Specify the correct workspace size. For information on how to determine the correct size, see "Planning Workspace Size" in Chapter 18, "Checking Database Consistency," in the *System Administration Guide.*

- `Workspace` *wsname* `has been altered successfully to size` *n*`KB`

  The workspace size has been changed, as specified.

- `Workspace` *wsname* `in` *dbname* `database could not be altered to size` *n*`KB`

  The size you requested for the workspace exceeds the capacity of the database. Increase the size of the database or delete some objects to increase the available space. Then run **sp_dbcc_alterws** again.

- *wsname* `is not a valid workspace in` *dbname* `database.`

  Workspaces must reside in either *dbccdb* or *dbccalt*. Use **sp_help** to list all the workspaces that currently exist in the database. If the workspace does not exist, use **sp_dbcc_createws** to create it.

- `You must be the System Administrator (SA) or the Database Owner (dbo) to execute this procedure.`

  Obtain the necessary permissions; then run **sp_dbcc_alterws** again.

- You must execute this procedure from the database in which you wish to create or alter workspaces. Execute 'use *dbname*' and try again.

Permissions

Only a System Administrator or the Database Owner can run **sp_dbcc_alterws**.

Tables Used

master..sysdatabases, *syssegments*, *sysobjects*

See Also

Commands	**dbcc**
dbcc stored procedures	**sp_dbcc_createws**, **sp_dbcc_evaluatedb**
System procedures	**sp_plan_dbccdb**, **sp_help**, **sp_helpdb**

sp_dbcc_configreport

Function

Generates a report that describes the configuration information used by the **dbcc checkstorage** operation for the specified database.

Syntax

sp_dbcc_configreport [*dbname*]

Parameters

dbname – specifies the name of the database. If *dbname* is not specified, the report contains information on all databases in *dbccdb..dbcc_operation_log*.

Examples

1. sp_dbcc_configreport

```
Reporting configuration information of database sybsystemprocs.

Parameter Name              Value                          Size

database name               sybsystemprocs                 51200K
dbcc named cache            default data cache             1024K
text workspace              textws_001 (id = 544004969)    128K
scan workspace              scanws_001 (id = 512004855)    1024K
max worker processes        1
operation sequence number    2
```

Generates a report on the configuration information related to **dbcc** for the *sybsystemprocs* database. The "Value" column lists the object name, where applicable, and the size.

Comments

- **sp_dbcc_configreport** generates a report that describes the configuration information used by **dbcc** operations for the specified database. This information is stored in the *dbcc_config* table.
- To evaluate the most current configuration parameters, run **sp_dbcc_updateconfig** before running **sp_dbcc_configreport**.
- To change the configuration values for a workspace, use **sp_dbcc_alterws**.

Messages

- `Can't run sp_dbcc_configreport from within a transaction.`

  sp_dbcc_configreport modifies tables, so it cannot be run within a transaction.

- `Reporting Configuration information of database` *`dbname`*`.`

  sp_dbcc_configreport provides a report of the configuration information for the specified database.

- `No configuration information available for` *`dbname`* `database.`

  sp_dbcc_configreport did not find any configuration information for the specified database in the *dbcc_operation_log* table. It could not provide a report of the configuration information for the specified database.

Permissions

Any user can run **sp_dbcc_configreport**.

Tables Used

master..sysdatabases, *dbccdb..dbcc_operation_log*, *dbccdb..dbcc_operation_results*, *dbccdb..dbcc_config*

See Also

Commands	dbcc
dbcc stored procedures	**sp_dbcc_alterws**, **sp_dbcc_fullreport**, **sp_dbcc_statisticsreport**, **sp_dbcc_summaryreport**, **sp_dbcc_updateconfig**

sp_dbcc_createws

Function

Creates a workspace of the specified type and size on the specified segment and database.

Syntax

```
sp_dbcc_createws dbname, segname, [wsname], wstype,
   "wssize[K|M]"
```

Parameters

dbname – is the name of the database in which the workspace is to be created. Values are *dbccdb* and *dbccalt*.

segname – is the name of the segment for the workspace.

wsname – is the name of the workspace. If the value is null, **sp_dbcc_createws** generates the name *scan_ws_nnnnnn* for the *scan* workspace and *text_ws_nnnnnn* for the *text* workspace, where *nnnnnn* is a unique 6-digit number.

wstype – specifies the type of workspace to be create. Values are **scan** and **text**.

wssize – is the workspace size, specified with **K** (kilobytes) or **M** (megabytes). If you do not specify **K** or **M**, *wssize* specifies the number of pages. The minimum size for a workspace is 24 pages.

Example

1. ```
   sp_dbcc_createws dbccdb, scanseg, scan_ws_pubs2,
   scan, "10M"
   ```

   Creates a 10MB *scan* workspace named *scan_ws_pubs2* on the *scanseg* segment in *dbccdb*.

2. ```
   sp_dbcc_createws dbccdb, textseg, text, "14M"
   ```

   Creates a 14MB *scan* workspace named *text_ws_000001* on the *textseg* segment in *dbccdb*.

Comments

- **sp_dbcc_createws** creates a workspace with the specified name and size and initializes it.

- Before you create a workspace, create the segment with **sp_addsegment.**
- Before you create a workspace, make sure you have configured a buffer pool of at least 16K, to achieve maximum performance.
- Use **sp_plan_dbccdb** to determine size estimates.
- After creating a workspace, run **sp_dbcc_updateconfig** to record the new configuration information in *dbcc_config*.
- Each workspace must have a unique name.
- For more information on the *scan* and *text* workspaces, see Chapter 18, "Checking Database Consistency," and Chapter 19, "dbccdb Tables," of the *System Administration Guide.*
- For information on the *dbccalt* database, see Chapter 18, "Checking Database Consistency," of the *System Administration Guide.*

Messages

- `DBCC database name cannot be null.`

  Specify either **dbccdb** or **dbccalt** for the database name.

- *dbname* `is not a valid DBCC database. Database must be either dbccdb or dbccalt.`

  Specify either **dbccdb** or **dbccalt** for the database name.

- `No such database -- run sp_helpdb to list databases.`

  The name you specified for the target database does not exist. Use **sp_helpdb** to list the databases.

- `Segment name cannot be null.`

  Specify a segment name.

- `There is no such segment as '`*segname*`'.`

  The segment where you want the workspace to reside must exist before you create the workspace. Use **sp_helpsegment** to list the segments that currently exist in the database. Use **sp_addsegment** to create segments.

- *wstype* `is not a valid workspace type. Valid types are 'scan' and 'text'.`

  Specify either **scan** or **text** for the workspace type.

- `Workspace size must be a valid number and must be at least` *n*`KB.`

  The workspace size must be specified as a whole integer, not a fraction or decimal number. If you do not specify **K** or **M** for the workspace size, Adaptive Server interprets the number specified to be the number of pages. The minimum size for a workspace is 24 pages.

- `Workspace` *wsname* `already exists in` *dbname* `database.`

  Each workspace must have a unique name. Use **sp_help** to list the workspaces that currently exist on the database.

- `Workspace` *wsname* `could not be created in` *dbname* `database.`

  sp_dbcc_createws could not create the workspace as specified. This error occurs if the workspace name is not unique or if the size you requested for the workspace exceeds the capacity of the database.

- `Workspace` *wsname* `of` *n*`KB size has been created successfully in` *dbname* `database.`

  sp_dbcc_createws created the workspace with the specified name and size in the specified database.

- `You must be the System Administrator (SA) or the Database Owner (dbo) to execute this procedure.`

  Obtain the necessary permissions; then run **sp_dbcc_createws** again.

- `You must execute this procedure from the database in which you wish to create or alter workspaces. Execute 'use` *dbname*`' and try again.`

Permissions

Only a System Administrator or the Database Owner can run **sp_dbcc_createws.**

Tables Used

master..sysdatabases, *syssegments*, *sysobjects*

See Also

Commands	**dbcc**
dbcc stored procedures	**sp_dbcc_alterws**, **sp_dbcc_evaluatedb**
System procedures	**sp_addsegment**, **sp_plan_dbccdb**, **sp_help**, **sp_helpsegment**

sp_dbcc_deletedb

Function

Deletes from *dbccdb* all the information related to the specified target database.

Syntax

```
sp_dbcc_deletedb [dbname]
```

Parameters

dbname – specifies the name of the target database for which you want the configuration information deleted. If you do not specify a value for *dbname*, Adaptive Server deletes data from all databases in *dbccdb..dbcc_config*. If the target database is *dbccdb*, and *dbccalt* exists, Adaptive Server deletes the data from *dbccalt*.

Example

```
1. sp_dbcc_deletedb "engdb"
```

```
All information for database engdb has been
deleted from dbccdb.
```

Deletes all information for the database named *engdb* from *dbccdb*.

Comments

- **sp_dbcc_deletedb** deletes from *dbccdb* all the information related to the specified target database, including configuration information and the results of previous **dbcc checkstorage** operations.
- If the deleted database is *dbccdb*, and the *dbccalt* database exists, **sp_dbcc_deletedb** deletes the configuration information and results of *dbccdb* from *dbccalt*.
- To remove the results of **dbcc checkstorage** operations created before a specific date, use **sp_dbcc_deletehistory**.
- For information about the *dbccalt* database, see Chapter 18, "Checking Database Consistency," in the *System Administration Guide*.

Messages

- `Can't run sp_dbcc_deletedb from within a transaction.`

  sp_dbcc_deletedb modifies tables, so it cannot be run within a transaction.

- `No configuration information to delete.`

  sp_dbcc_deletedb did not find any configuration information for the specified database in *dbccdb*.

- `Deleting Configuration information of database` *`dbname`*`.`

  sp_dbcc_deletedb is deleting from *dbccdb* all the configuration information for the target database.

- `Deleting results of dbcc operations completed on` *`date1`* `on the database` *`dbname`*`.`

  sp_dbcc_deletedb is deleting from *dbccdb* all the results of **dbcc** operations performed on the target database.

- `You must be the System Administrator (SA) or the Database Owner (dbo) to execute this procedure.`

  Obtain the necessary permissions, and then run **sp_dbcc_deletedb** again.

Permissions

Only a System Administrator or the Database Owner can run **sp_dbcc_deletedb**.

Tables Used

master..sysdatabases, *dbccdb..dbcc_config*, *dbccdb..dbcc_operation_log*, *dbccdb..dbcc_operation_results*, *dbccdb..dbcc_counters*, *dbccdb..dbcc_faults*, *dbccdb..dbcc_fault_params*

See Also

Commands	**dbcc**
dbcc stored procedures	**sp_dbcc_deletehistory**, **sp_dbcc_evaluatedb**
System procedures	**sp_plan_dbccdb**

sp_dbcc_deletehistory

Function

Deletes the results of **dbcc checkstorage** operations performed on the target database before the specified date and time.

Syntax

```
sp_dbcc_deletehistory [cutoffdate [, dbname]]
```

Parameters

cutoffdate – deletes all entries made on or before this date. This parameter is of type *datetime*. If a date is not specified, only the results of the last operation are retained. For more information, see "Specifying the Date" on page 6-2.

dbname – specifies the name of the database for which the data must be deleted. If not specified, **sp_dbcc_deletehistory** deletes the history information for all databases in *dbccdb..dbcc_config*.

Examples

```
1. sp_dbcc_deletehistory "03/04/1997", "pubs2"
```

Deletes results of all operations performed on the database *pubs2* on or before March 4, 1997.

Comments

- **sp_dbcc_deletehistory** deletes the results of **dbcc checkstorage** operations performed on the target database before the specified date and time.
- If the target database is *dbccdb*, and the *dbccalt* database exists, **sp_dbcc_deletehistory** deletes historical data for *dbccdb* from *dbccalt*.
- The value specified for *cutoffdate* is compared to the finish time of each **dbcc** operation.
- To see the dates when **dbcc checkstorage** was run so that you can choose the value for *cutoffdate*, run **sp_dbcc_summaryreport**.
- For information on the *dbccalt* database, see Chapter 18, "Checking Database Consistency," in the *System Administration Guide*.

Messages

- `Can't run sp_dbcc_deletehistory from within a transaction.`

  sp_dbcc_deletehistory modifies tables, so it cannot be run within a transaction.

- `Deleting results of dbcc operations completed on` *`date1`* `on the database` *`dbname`*`.`

  sp_dbcc_deletehistory is deleting from *dbccdb* all the results of **dbcc** operations performed on the target database.

- `No history information to delete.`

  sp_dbcc_deletehistory did not find any history information for the specified database in *dbccdb*.

- `You must be the System Administrator (SA) or the Database Owner (dbo) to execute this procedure.`

  Obtain the necessary permissions, and then run **sp_dbcc_deletehistory** again.

Permissions

Only a System Administrator or the Database Owner can run **sp_dbcc_deletehistory** on a specific database. Only a System Administrator can run **sp_dbcc_deletehistory** without specifying a database name.

Tables Used

master..sysdatabases, *master..sysdevices*, *master..sysusages*, *dbccdb..dbcc_operation_log*, *dbccdb..dbcc_operation_results*, *dbccdb..dbcc_counters*, *dbccdb..dbcc_faults*, *dbccdb..dbcc_fault_params*

See Also

Commands	**dbcc**
dbcc stored procedures	**sp_dbcc_deletedb**, **sp_dbcc_evaluatedb**
System procedures	**sp_plan_dbccdb**

sp_dbcc_differentialreport

Function

Generates a report that highlights the changes in I/O statistics and faults that took place between two **dbcc** operations.

Syntax

```
sp_dbcc_differentialreport [dbname [, objectname]],
   [db_op] [, "date1" [, "date2"]]
```

Parameters

dbname – specifies the name of the database. If you do not specify a *dbname*, the report contains information on all databases in *dbccdb..dbcc_operation_log*.

objectname – specifies the name of the table or index for which you want the report generated. If *object_name* is not specified, statistics on all objects in the target database are reported.

db_op – specifies the source of the data to be used for the report. The only value is **checkstorage**. The report is generated on the data specified by *db_op* on *date1* and *date2* for the specified object in the target database. If dates are not specified, the last two operations of the type *db_op* are compared.

date1 – specifies the first date of a **dbcc checkstorage** operation to be compared.

date2 – specifies the last date of a **dbcc checkstorage** operation to be compared.

Examples

1.
```
sp_dbcc_differentialreport master, sysprocedures,
checkstorage, "05/01/97", "05/04/97"
```

   Generates a report that shows the changes in I/O statistics and faults that occurred in the *sysprocedures* table between May 1, 1997 and May 4, 1997

Comments

- **sp_dbcc_differentialreport** generates a report that highlights the changes in I/O statistics and faults that occurred between two **dbcc** operations. It compares counter values reported from two

instances of **dbcc checkstorage**. Only the values that have been changed are reported.

- If only one date is specified, the results of the **dbcc checkstorage** operation selected by the specified date are compared to the results of the **dbcc checkstorage** operation immediately preceding the selected operation.
- If no dates are specified, the results of last two **dbcc checkstorage** operations are compared.
- If **sp_dbcc_differentialreport** returns a number for *object_name*, it means the object was dropped after the **dbcc checkstorage** operation completed.
- If no changes occurred between the specified operations, **sp_dbcc_differentialreport** does not generate a report.

Messages

- `Can't run sp_dbcc_differentialreport from within a transaction.`

  sp_dbcc_differentialreport modifies tables, so it cannot be run within a transaction.

- *`date1`* `and` *`date2`* `are the same. No differential report can be generated.`

  You must specify two different dates for **sp_dbcc_differentialreport** to make a comparison.

- *`db_op`* `is not a valid value for DBCC operation.`

  The only valid value for *db_op* is **checkstorage**.

- `Generating 'Differential Report' on object` *`object_name`* `in database` *`dbname`*`.`

  sp_dbcc_differentialreport created a report for the specified object in the specified database. This message appears at the beginning of each report for **sp_dbcc_differentialreport**.

- `Object name must be accompanied by database name.`

  You specified a value for *object_name*, but not for *dbname*. Run **sp_dbcc_differentialreport** again and specify values for both *dbname* and *object_name*.

- `No differential report on DBCC operations available.`

Permissions

Any user can run **sp_dbcc_differentialreport.**

Tables Used

master..sysdatabases, *dbccdb..dbcc_operation_log*, *dbccdb..dbcc_operation_results*, *dbccdb..dbcc_counters*

See Also

Commands	dbcc
dbcc stored procedures	**sp_dbcc_fullreport**, **sp_dbcc_statisticsreport**, **sp_dbcc_summaryreport**, **sp_dbcc_updateconfig**

sp_dbcc_evaluatedb

Function

Recomputes configuration information for the target database and compares it to the current configuration information.

Syntax

sp_dbcc_evaluatedb [*dbname*]

Parameters

dbname – specifies the name of the target database. If *dbname* is not specified, **sp_dbcc_evaluatedb** compares all databases listed in the *dbcc_config* table.

Example

1. sp_dbcc_evaluatedb

```
Recommended values for workspace size, cache size and worker
process count are:

Database name : sybsystemprocs
current scan workspace size : 400K      suggested scan workspace
size : 272K
current text workspace size : 208K      suggested text workspace
size : 208K
current cache size : 1024K      suggested cache size : 640K
current process count : 1        suggested process count : 1
```

Recomputes configuration information for the current database, *sybsystemprocs*, and suggests new values for some parameters.

Comments

- **sp_dbcc_evaluatedb** recomputes configuration information for the target database and compares the data to the current configuration information. It uses counter values recorded for the target database in the *dbcc_counters* table.
- The cache size is the size of the 16K buffer pool in the cache. For a 2K buffer pool, the minimum size of this cache must be the recommended value, plus 512.
- When the size and data distribution pattern of the target database changes, run **sp_dbcc_evaluatedb** to optimize the configuration information.

- To gather configuration information for the target database the first time, use **sp_plan_dbccdb**.
- To make sure you are evaluating the most current configuration parameters, run **sp_dbcc_updateconfig** before running **sp_dbcc_evaluatedb**.

Messages

- `Can't run sp_dbcc_evaluatedb from within a transaction.`

  sp_dbcc_evaluatedb modifies tables, so it cannot be run within a transaction.

- *dbname* `database is not configured for DBCC in the dbcc_config table.`

  You specified a database for which configuration information has not been added to *dbcc_config*. To gather configuration information for the target database the first time, use **sp_plan_dbccdb**.

- `Since dbccalt exists, it is the correct database for dbccdb. Use the dbccalt database.`

  *Y*ou ran **sp_dbcc_evaluatedb** and specified **dbccdb** for *dbname*. Change the current database to *dbccalt*, and then run **sp_dbcc_evaluatedb** and specify **dbccdb** for *dbname*.

- `You must be the System Administrator (SA) or the Database Owner (dbo) to execute this procedure.`

  Obtain the necessary permissions, and then run **sp_dbcc_evaluatedb** again.

Permissions

Only System Administrator or the Database Owner can run **sp_dbcc_evaluatedb**. Only a System Administrator can run **sp_dbcc_evaluatedb** without specifying a database name.

Tables Used

master..sysdatabases, *master..sysdevices*, *master..sysusages*, *dbccdb..dbcc_counters*, *dbccdb..dbcc_config*

See Also

Commands	**dbcc**
dbcc stored procedures	**sp_dbcc_updateconfig**
System procedures	**sp_plan_dbccdb**

sp_dbcc_faultreport

Function

Generates a report covering fault statistics for the **dbcc checkstorage** operations performed for the specified object in the target database on the specified date.

Syntax

```
sp_dbcc_faultreport [report_type [, dbname
   [, objectname [, date ]]]]
```

Parameters

report_type – specifies the type of fault report. Valid values are **short** and **long.** The default is **short.**

dbname – specifies the name of the target database; for example, *master..sysdatabases*. If *dbname* is not specified, the report contains information on all databases in *dbccdb..dbcc_operation_log*.

object_name – specifies the name of the table or index for which you want the report generated. If *object_name* is not specified, statistics on all objects in the target database are reported.

date – specifies the date on which the **dbcc checkstorage** operation was performed. If you do not specify *date*, Adaptive Server uses the date of the most recent operation.

Examples

1. sp_dbcc_faultreport "short"

```
Database Name : sybsystemprocs

 Table Name     Index  Type Code Description         Page Number
 -------------- ------ --------- ------------------- -----------
 sysprocedures        0    100031 page not allocated        5702
 sysprocedures        1    100031 page not allocated       14151
 syslogs              0    100022 chain start error        24315
 syslogs            0    100031 page not allocated     24315
```

Generates a short report of the faults found in tables in the *sybsystemprocs* database. The report includes the table name, the index number in which the fault occurred, the type code of the fault, a brief description of the fault, and the page number on which the fault occurred.

2. sp_dbcc_faultreport "long"

```
Generating 'Fault Report' for object sysprocedures in database
sybsystemprocs.

Type Code: 100031; Soft fault, possibly spurious
Page reached by the chain is not allocated.
page id:  14151
page header:
0x0000374700003788000037460000000500064 8B803EF0001000103FE0080000F
Header for 14151, next 14216, previous 14150, id = 5:1
 time stamp = 0x0001000648B8, next row = 1007, level = 0
 free offset = 1022, minlen = 15, status = 128(0x0080)
.
.
.
```

Generates a long report of the faults found in tables in the *sybsystemprocs* database. This example shows the first part of the output of a long report. The complete report repeats the information for each object in the **target database** in which **dbcc checkstorage** found a fault. The data following the long string of numbers shown under the "page header" field (“Header for 14151, next 14216, previous 14150 ...”) describes the components of the "page header" string.

Comments

- **sp_dbcc_faultreport** generates a report that shows all faults for the specified object in the target database.
- If **sp_dbcc_faultreport** returns a number for *object_name*, it means the object was dropped after the **dbcc checkstorage** operation completed.
- For information on the fault ID, see the *type_code* column described in “dbcc_faults” in Chapter 19, “dbccdb Tables,” in the *System Administration Guide*.
- For information on the fault status, see “Comparison of Soft and Hard Faults” in Chapter 18, “Checking Database Consistency,” in the *System Administration Guide*.

Messages

- `Can't run sp_dbcc_faultreport from within a transaction.`

  sp_dbcc_faultreport modifies tables, so it cannot be run within a transaction.

- `Generating 'Fault Report' on object object_name in database dbname.`

  sp_dbcc_faultreport created a report for the specified object in the specified database. This message appears at the beginning of each report for **sp_dbcc_faultreport**.

- `No fault report on DBCC operations available.`

  If no fault occurred during the specified operation, **sp_dbcc_faultreport** cannot generate a report.

- `Object name must be accompanied by database name.`

  You specified a value for *object_name*, but not for *dbname*.

Permissions

Any user can run **sp_dbcc_faultreport**.

Tables Used

master..sysdatabases, *dbccdb..dbcc_operation_log*, *dbccdb..dbcc_operation_results*, *dbccdb..dbcc_faults*, *dbccdb..dbcc_fault_params*

See Also

Commands	**dbcc**
dbcc stored procedures	**sp_dbcc_fullreport**, **sp_dbcc_statisticsreport**, **sp_dbcc_summaryreport**, **sp_dbcc_updateconfig**

sp_dbcc_fullreport

Function

Runs **sp_dbcc_summaryreport**, **sp_dbcc_configreport**, **sp_dbcc_statisticsreport**, and **sp_dbcc_faultreport short** for *database..object_name* on or before the specified *date*.

Syntax

```
sp_dbcc_fullreport [dbname [, objectname [, date]]]
```

Parameters

dbname – specifies the name of the database. If you do not specify *dbname*, the report contains information on all databases in *dbccdb..dbcc_operation_log*.

object_name – specifies the name of the table or index for which you want the report generated. If you do not specify *object_name*, statistics on all objects in the target database are reported.

date – specifies the date on which the **dbcc checkstorage** operation was performed. If you do not specify a *date*, the date of the last operation is used.

Examples

1. `sp_dbcc_fullreport master, sysprocedures`

Runs **sp_dbcc_summaryreport**, **sp_dbcc_configreport**, **sp_dbcc_statisticsreport**, and **sp_dbcc_faultreport short** for the most recent **dbcc checkstorage** operation run on the *sysprocedures* table in the *master* database.

Comments

- **sp_dbcc_fullreport** runs **sp_dbcc_summaryreport**, **sp_dbcc_configreport**, **sp_dbcc_statisticsreport**, and **sp_dbcc_faultreport short** for the specified database object on or before the specified date.

Messages

- `Can't run sp_dbcc_fullreport from within a transaction.`

  sp_dbcc_fullreport modifies tables, so it cannot be run within a transaction.

- `No full report on DBCC operations available.`

  If no information was recorded during the specified operation, **sp_dbcc_fullreport** cannot generate a report.
- `Object name must be accompanied by database name.`

  You specified a value for *object_name*, but not for *dbname*.

Permissions

Any user can run **sp_dbcc_fullreport**.

Tables Used

master..sysdatabases, *dbccdb..dbcc_operation_log*, *dbccdb..dbcc_operation_results*, *dbccdb..dbcc_faults*, *dbccdb..dbcc_fault_params*, *dbccdb..dbcc_counters*

See Also

Commands	dbcc
dbcc stored procedures	sp_dbcc_statisticsreport, sp_dbcc_summaryreport, sp_dbcc_updateconfig

sp_dbcc_runcheck

Function

Runs **dbcc checkstorage** on the specified database, and then runs **sp_dbcc_summaryreport** or a report you specify.

Syntax

```
sp_dbcc_runcheck dbname [, user_proc]
```

Parameters

dbname – specifies the name of the database on which the check is to be performed.

user_proc – specifies the name of the **dbcc** stored procedure or a user-created stored procedure that is to be run instead of **sp_dbcc_summaryreport**.

Example

1. `sp_dbcc_runcheck "engdb"`

   Checks the database *engdb* and generates a summary report on the information found.

2. `sp_dbcc_runcheck "pubs2", sp_dbcc_fullreport`

   Checks the database *pubs2* and generates a full report.

Comments

- **sp_dbcc_runcheck** runs **dbcc checkstorage** on the specified database.
- After the **dbcc checkstorage** operation is complete, **sp_dbcc_runcheck** runs **sp_dbcc_summaryreport** to generate a summary report. If you specify one of the other report-generating **dbcc** stored procedures for *dbcc_report*, **sp_dbcc_runcheck** runs that procedure instead of **sp_dbcc_summaryreport**. For a brief description and examples of all the report-generating stored procedures provided with *dbccdb*, see "Generating Reports from dbccdb" in Chapter 18, "Checking Database Consistency," in the *System Administration Guide*.
- You can write your own report-generating stored procedure and specify its name for *user_proc*. The stored procedure must be self-contained. **sp_dbcc_runcheck** cannot pass any parameters to Adaptive Server.

Messages

- `No such database -- run sp_helpdb to list databases.`

  The name you specified for the **target database** does not exist.

- `You must be the System Administrator (SA) or the Database Owner (dbo) to execute this procedure.`

  Obtain the necessary permissions, and then run **sp_dbcc_evaluatedb** again.

Permissions

Only a System Administrator or the Database Owner can run **sp_dbcc_runcheck**.

Tables Used

master..sysdatabases, dbccdb..dbcc_config, dbccdb..dbcc_counters, dbccdb..dbcc_fault_params, dbccdb..dbcc_faults, dbccdb..dbcc_operation_log, dbccdb..dbcc_operation_results

See Also

Commands	**dbcc**
dbcc stored procedures	**sp_dbcc_summaryreport**

sp_dbcc_statisticsreport

Function

Generates an allocation statistics report on the specified object in the target database.

Syntax

```
sp_dbcc_statisticsreport [dbname [, objectname
   [, date]]]
```

Parameters

dbname – specifies the **target database**. If *dbname* is not specified, the report contains information on all databases in *dbccdb..dbcc_operation_log*.

objectname – specifies the name of the table or index for which you want the report generated. If you do not specify *objectname*, Adaptive Server reports statistics on all objects in the target database.

date – specifies the date on which the **dbcc checkstorage** operation was performed. If you do not specify *date*, Adaptive Server uses the date of the most recent operation.

Examples

```
1. sp_dbcc_statisticsreport 'sybsystemprocs',
   'sysobjects'
Statistics Report on object sysobjects in database sybsystemprocs

 Parameter Name            Index Id Value
 ------------------------- -------- ------------
 count                     0        241.0
 max size                  0        99.0
 max count                 0        22.0
 bytes data                0        19180.0
 bytes used                0        22113.0
 count                     1        14.0
 max size                  1        9.0
 max level                 1        0.0
 max count                 1        14.0
 bytes data                1        56.0
 bytes used                1        158.0
 count                     2        245.0
```

```
max level                2        1.0
max size                 2        39.0
max count                2        71.0
bytes data               2        4377.0
bytes used               2        6995.0

 Parameter Name  Index Id  Partition  Value   Dev_name
---------------  --------  ---------  ------  -------------
page gaps               0          1    13.0  master
pages used              0          1    15.0  master
extents used            0          1     3.0  master
overflow pages          0          1     0.0  master
pages overhead          0          1     1.0  master
pages reserved          0          1     7.0  master
page extent gaps        0          1    11.0  master
ws buffer crosses       0          1     2.0  master
page extent crosses     0          1    11.0  master
pages used              1          1     2.0  master
extents used            1          1     1.0  master
overflow pages          1          1     0.0  master
pages overhead          1          1     1.0  master
pages reserved          1          1     6.0  master
page extent gaps        1          1     0.0  master
ws buffer crosses       1          1     0.0  master
page extent crosses     1          1     0.0  master
page gaps               2          1     4.0  master
pages used              2          1     6.0  master
extents used            2          1     1.0  master
overflow pages          2          1     0.0  master
pages overhead          2          1     1.0  master
pages reserved          2          1     2.0  master
page extent gaps        2          1     0.0  master
ws buffer crosses       2          1     0.0  master
page extent crosses      2         1     0.0  master
```

Generates a statistics report on the *sysobjects* table in the *master* database.

Comments

- **sp_dbcc_statisticsreport** generates an allocation statistics report on the specified object in the **target database**. It uses data from the *dbcc_counters* table, which stores information about page utilization and error statistics for every object in the target database.
- If **sp_dbcc_statisticsreport** returns a number for *object_name*, it means the object was dropped after the **dbcc checkstorage** operation completed.

- **sp_dbcc_statisticsreport** reports values recorded in the *dbcc_counters* table for the datatypes 5000–5019. See *dbcc_types* in Chapter 19, "dbccdb Tables" of the *System Administration Guide*.

  For *bytes data*, *bytes used*, and *overflow pages*, **sp_dbcc_statisticsreport** reports the sum of the values reported for all partitions and devices.

  For *count*, *max count*, *max size* and *max level*, **sp_dbcc_statisticsreport** reports the largest of the values reported for all partitions and devices.

  sp_dbcc_statisticsreport reports information for each device and partition used by objects in the **target database** for the following rows:

  - *extents used*
  - *io errors*
  - *page gaps*
  - *page extent crosses*
  - *page extent gaps*
  - *page format errors*
  - *pages reserved*
  - *pages overhead*
  - *pages misallocated*
  - *pages not allocated*
  - *pages not referenced*
  - *pages used*

  The *page gaps*, *page extent crosses*, and *page extent gaps* indicate how the data pages for the objects are distributed on the database devices. Large values indicate less effectiveness in using larger buffer sizes and in data prefetch.

- If multiple **dbcc checkstorage** operations were run on a target database on the same day, **sp_dbcc_statisticsreport** generates a report based on the results of the last **dbcc checkstorage** operation that finished before the specified time.

Messages

- `Can't run sp_dbcc_statisticsreport from within a transaction.`

  sp_dbcc_statisticsreport modifies tables, so it cannot be run within a transaction.

- `Generating 'Statistics Report' on object` *`object_name`* `in database` *`dbname`*`.`

  sp_dbcc_statisticsreport created a report for the specified object in the specified database. This message appears at the beginning of each report for **sp_dbcc_statisticsreport**.

- `No statistics report on DBCC operations available.`

  If no allocation statistics are available for the specified operation, **sp_dbcc_statisticsreport** cannot generate a report.

- `Object name must be accompanied by database name.`

  You specified a value for *object_name*, but not for *dbname*.

Permissions

Any user can run **sp_dbcc_statisticsreport**.

Tables Used

master..sysdatabases, *dbccdb..dbcc_operation_log*, *dbccdb..dbcc_operation_results*, *dbccdb..dbcc_counters*

See Also

Commands	**dbcc**
dbcc stored procedures	**sp_dbcc_fullreport**, **sp_dbcc_summaryreport**, **sp_dbcc_updateconfig**

sp_dbcc_summaryreport

Function

Generates a summary report on the specified database.

Syntax

sp_dbcc_summaryreport [*dbname* [, *date*]]

Parameters

dbname – specifies the name of the database for which you want the report generated. If you do not specify *dbname*, **sp_dbcc_summaryreport** generates reports on all databases in *dbccdb..dbcc_operation_log* for which the date is on or before the date and time specified by the *date* option.

date – specifies the date on which **dbcc checkstorage** was performed. If you do not specify a date, **sp_dbcc_summaryreport** uses the date of last **dbcc checkstorage** operation performed on the **target database**. This parameter is of the datatype *datetime*. If both the date and the time are specified for *date*, summary results of all the operations performed on or before the specified time are reported. If no date is specified, all operations are reported.

Examples

1. sp_dbcc_summaryreport

```
DBCC Operation : checkstorage

Database Name      Start time           End Time   Operation ID
    Hard Faults Soft Faults Text Columns Abort Count
       User Name
------------------ ------------------_ ---------- ------------
      ----------- ----------- ------------ -----------
      -----------------
sybsystemprocs     09/12/1997 10:54:45  10:54:53             1
                0           0            0           0
       sa
sybsystemprocs     09/12/1997 11:14:10  11:14:19             2
                 0           0            0           0
       sa
```

Generates a summary report on the *master* database, providing information on all **dbcc checkstorage** operations performed.

Comments

- **sp_dbcc_summaryreport** generates a summary report on the specified database.
- The report indicates the name of the database that was checked, the start and end time of the **dbcc checkstorage** run and the number of soft and hard faults found.
- The "Operation ID" column contains a number that identifies the results of each **dbcc checkstorage** operation on a given database at a specific time. The number provided in the report comes from the *opid* column of the *dbcc_operation_log* table. For more information, see "dbcc_operation_log" in Chapter 19, "dbccdb Tables," in the *System Administration Guide*.
- The "Text Colums" column shows the number of non-null text columns found by **dbcc checkstorage** during the run.
- The "Abort Count" column shows the number of tables that contained errors, which caused **dbcc checkstorage** to abort the check on the table. For details on the errors, run **sp_dbcc_faultreport**.

Messages

- `Can't run sp_dbcc_summaryreport from within a transaction.`

  sp_dbcc_summaryreport modifies tables, so it cannot be run within a transaction.

- `No summary report on DBCC operations available.`

  If no information is available for the specified operation, **sp_dbcc_statisticsreport** cannot generate a report.

- `Summary report on dbcc operations performed on database` *`dbname`* `on` *`date`*`.`

  sp_dbcc_summaryreport created a report on the specified database on the specified date. This message appears at the beginning of each report for **sp_dbcc_faultreport**.

Permissions

Any user can run **sp_dbcc_summaryreport**.

Tables Used

master..sysdatabases, *dbccdb..dbcc_operation_log*, *dbccdb..dbcc_operation_results*

See Also

Commands	dbcc
dbcc stored procedures	sp_dbcc_fullreport, sp_dbcc_statisticsreport, sp_dbcc_updateconfig

sp_dbcc_updateconfig

Function

Updates the *dbcc_config* table in *dbccdb* with the configuration information of the target database.

Syntax

```
sp_dbcc_updateconfig dbname, type, "str1" [, "str2"]
```

Parameters

dbname – is the name of the target database for which configuration information is being updated.

type – specifies the type name from the *dbcc_types* table. Table 6-2 on page 6-39 shows the valid values for *type*.

str1 – specifies the first configuration value for the specified *type* to be updated in the *dbcc_config* table. Table 6-2 on page 6-39 describes the expected value of *str1* for the specified *type*.

str2 – specifies the second configuration value for the specified *type* that you want to update in the *dbcc_config* table. Table 6-2 on page 6-39 describes the expected value of *str2* for the specified *type*.

Examples

1. **`sp_dbcc_updateconfig pubs2, "max worker processes", "4"`**

   Updates *dbcc_config* with the maximum number of worker processes for **dbcc checkstorage** to use when checking the *pubs2* database. The new maximum number of worker processes is 4.

2. **`sp_dbcc_updateconfig pubs2, "dbcc named cache", pubs2_cache, "10K"`**

   Updates *dbcc_config* with the size of the **dbcc** named cache "pubs2_cache". The new size is 10K.

3. **`sp_dbcc_updateconfig pubs2, "scan workspace", scan_pubs2`**

   Updates *dbcc_config* with the new name of the *scan* workspace for the *pubs2* database. The new name is *scan_pubs2*. This update is made after using **sp_dbcc_alterws** to change the name of the *scan* workspace.

4. **sp_dbcc_updateconfig pubs2, "text workspace", text_pubs2**

   Updates *dbcc_config* with the new name of the *text* workspace for the *pubs2* database. The new name is *text_pubs2*. This update is made after using **sp_dbcc_alterws** to change the name of the *text* workspace.

5. **sp_dbcc_updateconfig pubs2, "OAM count threshold", 5**

   Updates *dbcc_config* with the OAM count threshold value for the *pubs2* database. The new value is 5.

6. **sp_dbcc_updateconfig pubs2, "IO error abort", 3**

   Updates *dbcc_config* with the I/O error abort value for the *pubs2* database. The new value is 3.

7. **sp_dbcc_updateconfig pubs2, "linkage error abort", 8**

   Updates *dbcc_config* with the linkage error abort value for the *pubs2* database. The new value is 8.

Comments

- **sp_dbcc_updateconfig** updates the *dbcc_config* table for the **target database**.
- If the name of the target database is *dbccdb*, and the database *dbccalt* exists, **sp_dbcc_updateconfig** updates the *dbcc_config* table in *dbccalt*.
- If the target database name is not found in *dbcc_config*, **sp_dbcc_updateconfig** adds it and sets the operation sequence number to 0 before updating other configuration information.
- If the expected value for the specified *type* is a number, **sp_dbcc_updateconfig** converts the values you provide for *str1* and *str2* to numbers.

- Table 6-2 shows the valid type names to use for *type* and the expected value for *str1* or *str2*.

Table 6-2: Type names and expected values

type Name	Value expected for *str1* or *str2*
dbcc named cache	The name of the cache, specified by *str1*, and the new size (in kilobytes or megabytes) or the number of 2K pages, specified by *str2*.
IO error abort	The new error count, specified by *str1*. The value must be a number greater than 0. *str2* is not used with this type.
linkage error abort	The new linkage error count value specified in *str1*. The value must be a number greater than 0. *str2* is not used with this type.
max worker processes	The new number of worker processes, specified by *str1*. The value must be a number greater than 0. *str2* is not used with this type.
OAM count threshold	The new threshold count, specified by *str1*. The value must be a number greater than 0. *str2* is not used with this type.
scan workspace	The new name for the *scan* workspace, specified by *str1*. *str2* is not used with this type.
text workspace	The new name of the *text* workspace, specified by *str1*. *str2* is not used with this type.

- For more information on the *type* names and values, see "dbcc_types" in Chapter 19, "dbccdb Tables," of the *System Administration Guide*.

Messages

- *cache_name* `is not a valid cache name.`

  You specified a cache name for *str1* that does not match any cache name designated for the device on which *dbccdb* resides. Use **sp_helpcache** to list cache names.

- `Database name is null. Specify a valid database name.`

  Use **sp_helpdb** to list the databases.

- `No such database -- run sp_helpdb to list databases.`

  The name you specified for the **target database** does not exist.

- *`str1`* `is not a valid value for type.`

  The value you entered for *str1* is not valid for the *type*. See Table 6-2 on page 6-39 for a description of the expected values for the specified *type*.

- *`str1`* `is an invalid configuration parameter.`

  See Table 6-2 on page 6-39 for a description of the expected values for the specified *type*.

- *`str2`* `is not a valid value for` *`type`*`.`

  See Table 6-2 on page 6-39 for a description of the expected values for the specified *type*.

- *`str2`* `is an invalid configuration parameter.`

  You specified a value for *str2* that is not a valid configuration parameter. See Table 6-2 on page 6-39 for a description of the expected values for the specified *type*.

- `The specified size for cache` *`cache_name`* `is not valid.`

  You included a size for the cache name for *str1* that is not a valid size for that cache. Use **sp_dbcc_evaluatedb** to obtain a suggested value for the cache size. Use the suggested value or a larger value for the cache size.

- *`type`* `for database` *`dbname`* `updated in dbcc_config.`

  The *type* for the specified database was updated in the *dbcc_config* table.

- `Workspace` *`wsname`* `does not exist in database` *`dbname`*`.`

  You specified a workspace name for *str1* that does not match any workspace name found in the specified database. Use **sp_help** to list the workspaces.

- `You must be the System Administrator (SA) or the Database Owner (dbo) to execute this procedure.`

  Obtain the necessary permissions, and then run **sp_dbcc_updateconfig** again.

Permissions

Only a System Administrator or the Database Owner can run **sp_dbcc_updateconfig**.

Tables Used

master..sysdatabases, *dbccdb..dbcc_types*, *dbccdb..dbcc_config*

See Also

Commands	**dbcc**
dbcc stored procedures	**sp_dbcc_evaluatedb**
System procedures	**sp_plan_dbccdb**, **sp_help**