Reference Manual: Procedures

SAP® Adaptive Server® Enterprise 16.0

DOCUMENT ID: DC36273-01-1600-01
LAST REVISED: May 2014

Contents

CHAPTER 1 System Procedures

SAP® Adaptive Server® Enterprise system procedures are similar to the stored procedures that you create using the Transact-SQL® language, but are supplied in SAP® ASE to use for updating and getting reports from system tables.

System procedures are created by **installmaster** at installation. They are located in the `sybsystemprocs` database, and owned by the system administrator. Use **sp_version** to determine which version of `installmaster` was most recently run.

Some system procedures can only run in a specific database, but many of them can run in any database. You can create your own system procedures to execute from any database.

You can declare up to 10,000 variables in a stored procedure.

All system procedures:

- Execute at isolation level 1.
- Report a *return status* that indicates whether or not they completed successfully, and if they did not, the reasons for failure.
  The following example means that the procedure executed successfully:

  ```
  return status = 0
  ```

  The examples in this book do not include the return status.

For more information:

- About creating your own system procedures, see the *System Administration Guide*.
- About the return values for system procedures, see *Return Values* in the *Transact-SQL User's Guide*.

Permissions on System Procedures

Set permissions for system procedures in the `sybsystemprocs` database.

Some system procedures can run only by a user with specific privileges or roles. Permission check for a system procedure may differ based on the granular permissions setting. Check the Permission section for each system procedure for details. See *Using Granular Permissions* in the *Security Administration Guide* for more information on granular permissions.

Other system procedures (for example, all the **sp_help** procedures) can be executed by any user, provided that the **execute** permission on the procedure was granted to public in `sybsystemprocs`.

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

Auditing System Procedures

In general, you can audit executed stored procedures by enabling the audit option "**exec_procedure**", which generates an audit record containing the name of the stored procedure and the parameters.

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`.

Run a system procedure in a specific database by either:

- Opening that database with the **use** command and execute the procedure, or
- Qualifying 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)
```

Entering Parameter Values

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 following form instead of supplying all the parameters:

```
@parametername = value
```

You can also use "null" as a placeholder for a parameter. Do not enclose "null" in quotes.

SQL has 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, the SAP ASE server attempts to execute the system procedure, then the command. For example, if you execute the following command, the SAP ASE server returns the output from **sp_help**, then runs the **checkpoint** command:

```
sp_help checkpoint
```

If you specify more parameters than the number of parameters expected by the system procedure, the extra parameters are ignored by the SAP ASE server.

Messages

System procedures return informational and error messages. System procedure error messages start at error number 17000.

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

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

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

- **Example 1 –** Displays all active roles.

  ```
  sp_activeroles
  ```

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

- **Example 2 –** Displays active roles and their hierarchy tree:

  ```
  sp_activeroles expand_down
  ```

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

Usage

sp_activeroles displays all your active roles and all roles contained by those roles.

See also:

- **alter role**, **create role**, **drop role**, **grant**, **revoke**, **set** in *Reference Manual: Commands*
- For information about creating, managing, and using roles, see the *System Administration Guide*.
- **mut_excl_roles**, **proc_role**, **role_contain**, **role_name** in *Reference Manual: Building Blocks*

Permissions

Any user can execute **sp_activeroles**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_displayroles* on page 240

sp_add_qpgroup

Adds an abstract plan group.

Syntax

```
sp_add_qpgroup new_name
```

Parameters

- ***new_name* –** is the name of the new abstract plan group. Group names must be valid identifiers.

Examples

- **Example 1 –** Creates a new abstract plan group named `dev_plans`:

  ```
  sp_add_qpgroup dev_plans
  ```

Usage

Use **sp_add_qpgroup** to add abstract plan groups for use in capturing or creating abstract plans. The abstract plan group must exist before you can create, save, or copy plans into a group.

You cannot run **sp_add_qpgroup** in a transaction.

See also **set** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_add_qpgroup** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_help_qpgroup* on page 370

sp_add_resource_limit

Creates a limit on the number of server resources that can be used by an SAP ASE login, or by an application, or both, 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 SAP ASE 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 SAP ASE 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 the SAP ASE 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:
  - **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.
  - **tempdb_space** – limits the number of pages a `tempdb` database can have during a single session.
- ***limitvalue*** **–** is the maximum amount of the server resource (I/O cost, elapsed time in seconds, row count, or tempdb space) that can be used by the login or application before the SAP ASE 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:
  - **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.
    - **tempdb_space** – the number of pages used in `tempdb` per session.
- ***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, the SAP ASE server performs a logical "or" of 1 and 2. For example, assume *enforced* is set to 3. If you run a query with **io_cost** that 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, the SAP ASE 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:

    - 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, the SAP ASE 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:

    - **1** – Query
    - **2** – Query batch (one or more SQL statements sent by the client to the server)
    - **4** – Transaction
    - **6** – Query batch and transaction

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

Examples

- **Example 1 –** 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, the SAP ASE server issues a warning:

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

- **Example 2 –** 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, the SAP ASE server aborts the transaction:

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

- **Example 3 –** 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, the SAP ASE server aborts the transaction:

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

Usage

Additional considerations for using **sp_add_resource_limit**.

- You must enable **sp_configure "allow resource limits"** for resource limits to take effect.
- 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 the SAP ASE 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, the SAP ASE 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**

Level	Login Name	Application Name
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, the SAP ASE server rebinds the limits for each session for that login and/or application at the beginning of the next query batch for that session.
- When you change the currently active time ranges, the SAP ASE server rebinds limits for the session. 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.

Note: Although the SAP ASE server terminates the current transaction when it reaches its time limit, you receive no 1105 error message until you issue another SQL command or batch; in other words, the message appears only when you attempt to use the connection again.

For more information on resource limits, see the *System Administration Guide*.

See also **isql** in the *Utility Guide*.

Permissions

The permission checks for **sp_add_resource_limit** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage resource limit` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_configure* on page 167
- *sp_drop_resource_limit* on page 258
- *sp_help_resource_limit* on page 368
- *sp_modify_resource_limit* on page 510

sp_add_time_range

Adds a named time range to an SAP ASE 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 255 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

- **Example 1 –** Creates the `business_hours` time range, which is active Monday through Friday, from 9:00 a.m. to 5:00 p.m.:

  ```
  sp_add_time_range business_hours, monday, Friday, "09:00",
  "17:00"
  ```
- **Example 2 –** 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:

  ```
  sp_add_time_range before_hours, Monday, Friday, "00:00", "09:00"
  ```

  ```
  sp_add_time_range after_hours, Monday, Friday, "18:00", "00:00"
  ```
- **Example 3 –** Creates the `weekends` time range, which is 12:00 midnight Saturday to 12:00 midnight Sunday:

  ```
  sp_add_time_range weekends, Saturday, Sunday, "00:00", "00:00"
  ```
- **Example 4 –** Creates the `Fri_thru_Mon` time range, which is 9:00 a.m. to 5:00 p.m., Friday, Saturday, Sunday, and Monday:

  ```
  sp_add_time_range Fri_thru_Mon, Friday, Monday, "09:00", "17:00"
  ```
- **Example 5 –** Creates the `Wednesday_night` time range, which is Wednesday from 5:00 p.m. to 12:00 midnight:

  ```
  sp_add_time_range Wednesday_night, Wednesday, Wednesday, "17:00",
  "00:00"
  ```

Usage

There are additional considerations when using **sp_add_time_range**:

- The SAP ASE 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.
- The SAP ASE 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, the SAP ASE 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).
- You can 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 the System Administration Guide.

Permissions

The permission checks for **sp_add_time_range** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage resource limit` privilege.
Disabled	With granular permissions disabled, you must be a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_add_resource_limit* on page 7
- *sp_drop_time_range* on page 261
- *sp_modify_time_range* on page 513

sp_addalias

Allows an SAP ASE 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

- **Example 1 –** 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”:

  ```
  sp_addalias victoria, albert
  ```

Usage

There are additional considerations when using **sp_addalias**:

- 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, the SAP ASE server checks `sysusers` to confirm that the user is listed there. If the user is not listed there, the SAP ASE server then checks `sysalternates`. If the user's `suid` is listed in `sysalternates`, mapped to a database user's `suid`, the SAP ASE 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, the SAP ASE server does not use the user's alias identity, because it checks `sysusers` and finds the `loginame` before checking `sysalternates`, where the alias is listed.

See also **use** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_addalias** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any user` privilege.
Disabled	With granular permissions disabled, you must be the database owner, a user with **sa_role**, or a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addlogin* on page 35
- *sp_adduser* on page 59
- *sp_dropalias* on page 252
- *sp_helpuser* on page 446

sp_addauditrecord

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

- **Example 1 –** 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:

```
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
```

- **Example 2 –** Adds this record to the audit trail. This example uses parameter names with the @ prefix, which allows you to leave some fields empty:

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

Usage

The SAP ASE 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**

Permissions

The permission checks for **sp_addauditrecord** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled: • Users with execute permission on the procedure can execute **sp_addauditrecord**. • By default, **sso_role** has execute permission. • The database owner of `sybsystemprocs` can grant execute permission.
Disabled	With granular permissions disabled: • Users with execute permission on the procedure can execute **sp_addauditrecord**. • By default **sso_role** has execute permission. • Users with **sa_role** can grant execute permission. • The database owner of `sybsystemprocs` can grant **execute** permission to other users.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	1
Audit option	**adhoc**
Command or access audited	User-defined audit record
Information in `extrainfo`	`extrainfo` is filled by the *text* parameter of **sp_addauditrecord**

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_audit* on page 63

sp_addaudittable

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”, the SAP ASE server creates the audit table on the same device as the `sybsecurity` database. Otherwise, the SAP ASE server creates the table on the device you specify.

Examples

- **Example 1 –** Creates a system audit table on `auditdev2`. If only one system audit table (`sysaudits_01`) exists when you execute the procedure, the SAP ASE server names the new audit table `sysaudits_02` and places it on its own segment, called `aud_seg_02`, on `auditdev2`:

  ```
  sp_addaudittable auditdev2
  ```

- **Example 2 –** 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, the SAP ASE 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:

  ```
  sp_addaudittable "default"
  ```

Usage

There are additional considerations when using **sp_addaudittable**:

- Auditing must already be installed when you run **sp_addaudittable**. To add a system audit table:

  1. 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",
     size = "5K"
     ```

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

   ```
   alter database sybsecurity on auditdev2
   ```
3. Execute **sp_addaudittable** to create the table.

- The SAP ASE 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, the SAP ASE server names the new audit table `sysaudits_06` and the new segment `aud_seg_06`. If you specify "default", the SAP ASE server places the segment on the same device as the `sybsecurity` database. Otherwise, the SAP ASE 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, the SAP ASE server displays an error message.
- For information about how to install auditing, see the installation documentation for your platform. See the *System Administration Guide* for information on how to use auditing.

See also **alter database**, **disk init** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_addaudittable** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage auditing` privilege.
Disabled	With granular permissions disabled, you must be a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_audit* on page 63

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.

Considerations for process mode

sp_addengine does not run in threaded mode.

Syntax

```
sp_addengine engine_number, engine_group [, instance_id]
```

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, the SAP ASE server creates it and adds the engine to it. Engine group names must conform to the rules for identifiers. For details, see *Expressions, Identifiers, and Wildcard Characters* in *Reference Manual: Building Blocks*.
- ***instance_id* –** (in cluster environments) ID of the instance to which you are adding an engine or engine group.

Examples

- **Example 1 –** 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:

  ```
  sp_addengine 2, DS_GROUP
  ```

- **Example 2 –** Adds engine number 5 to instance ID 8:

```
sp_addengine 5, 8
```

Usage

There are additional considerations when using **sp_addengine**:

- **sp_addengine** creates a new engine group if the value of `engine_group` does not already exist.
- If **sp_cluster set *system_view*** is set to **cluster**, you can add an engine or engine group to any instance in the cluster. If *system_view* is set to **instance**, you can add and engine or engine group only to a local instance.
- 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`.
- **sp_engine** can run in sessions using chained transactions after you use **sp_procxmode** to change the transaction mode to **anymode**.
- Prior to making engine affinity assignments, study the environment and consider the number of nonpreferred applications and the number of SAP ASE engines available. See the *Performance and Tuning Guide* for more information about non-preferred applications.

Permissions

The permission checks for **sp_addengine** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any execution class` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**

Information	Values
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addexeclass* on page 23
- *sp_bindexeclass* on page 83
- *sp_clearpsexe* on page 128
- *sp_dropengine* on page 264
- *sp_setpsexe* on page 634
- *sp_showcontrolinfo* on page 649
- *sp_showexeclass* on page 652
- *sp_showpsexe* on page 662
- *sp_unbindexeclass* on page 716

sp_addexeclass

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

Considerations for process mode

The predefined engine group parameter `ANYENGINE` and `LASTONLINE` are valid only in process mode.

Syntax

```
sp_addexeclass classname, priority, timeslice, engine_group [,
instance_id]
```

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. The SAP ASE server currently ignores this parameter.
- ***engine_group* –** identifies an existing group of engines on which processes associated with this class can run.
- ***instance_id* –** (in cluster environments) ID of the instance to which you are binding a user-defined execution class.

Examples

- **Example 1 –** Defines a new execution class called `DS` with a *priority* value of **LOW** and associates it with the engine group `DS_GROUP`:

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

- **Example 2 –** (Cluster Edition) Defines a new execution class called DS with a priority value of LOW and associates it with the engine group DS_GROUP on instance number 8, enter:

  ```
  sp_addexeclass "DS", "LOW", 0, "DS_GROUP", 8
  ```

Usage

There are additional considerations when using **sp_addexeclass**:

- **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.
- When you run **sp_addexeclass** in threaded mode, the SAP ASE server uses *engine_group* for the name of a thread pool.
- (In cluster environments) If **sp_cluster set *system_view*** is set to **cluster**, you can add an execution class on any instance in the cluster. If the *system_view* is set to **instance**, you can add an execution class only to a local instance.
- 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.

Permissions

The permission checks for **sp_addexeclass** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any execution class` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addengine* on page 21
- *sp_bindexeclass* on page 83
- *sp_clearpsexe* on page 128
- *sp_dropengine* on page 264
- *sp_dropexeclass* on page 266
- *sp_setpsexe* on page 634
- *sp_showcontrolinfo* on page 649
- *sp_showexeclass* on page 652
- *sp_unbindexeclass* on page 716

sp_addextendedproc

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 SAP ASE 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 or `.so` on Solaris. If an extension is specified, the *dll_name* must be enclosed in quotation marks.

Examples

- **Example 1 –** Registers an ESP for the function named **my_esp**, which is in the `sqlsrvdll.dll` file. The name of the resulting ESP database object is also `my_esp`:

  ```
  sp_addextendedproc my_esp, "sqlsrvdll.dll"
  ```

Usage

There are additional considerations when using **sp_addextendedproc**:

- Execute **sp_addextendedproc** from the `master` database.
- You can only use **sp_addextendedproc** to add extended stored procedures that take no parameters. If your extended stored procedure requires a formal parameter list, you must use the **create procedure** command with the **as external name** option, together with the complete parameter list.
- 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:
  - Windows – `$PATH`
  - Compaq Tru64 – `$LD_LIBRARY_PATH`
  - HP – `$SH_LIBRARY_PATH`

  If the file is not found, the search mechanism also searches `$SYBASE/dll` on Windows and `$SYBASE/lib` on other platforms.
- (On Windows) 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.

See also **create procedure** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_addextendedproc** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any ESP` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_dropextendedproc* on page 267
- *sp_helpextendedproc* on page 404

sp_addexternlogin

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

Syntax

```
sp_addexternlogin server, loginame, externname
    [, externpasswd] [rolename]
```

Parameters

- ***server* –** is the name of the remote server. The *remote_server* must be known to the local server by an entry in the `master.dbo.sysservers` table.

- ***loginame* –** is an account known to the local server. *loginame* must be represented by an entry in the `master.dbo.syslogins` table. The "sa" account, the "sso" account, and the *loginame* account are the only users authorized to modify remote access for a given local user.
- ***externnname* –** is an account known to the *server* and must be a valid account on the node where the *server* runs. This is the account used for logging into the *server*.
- ***externpasswd* –** is the password for *externnname*.
- ***rolename* –** is the SAP ASE user's assigned role. If *rolename* is specified, *login_name* is ignored.

Examples

- **Example 1 –** Tells the local server that when the login name "bobj" logs in, access to the remote server OMNI1012 is by the remote name "jordan" and the remote password "hitchpost". Only the "bobj" account, the "sa" account, and the "sso" account have the authority to add or modify a remote login for the login name "bobj":

  ```
  sp_addexternlogin OMNI1012, bobj, jordan, hitchpost
  ```
- **Example 2 –** Shows a many-to-one mapping so that all SAP ASE users that need a connection to DB2 can be assigned the same name and password:

  ```
  sp_addexternlogin DB2, NULL, login2, password2
  ```
- **Example 3 –** SAP ASE roles can also be assigned remote logins. With this capability, anyone with a particular role can be assigned a corresponding login name and password for a given remote server:

  ```
  sp_addexternlogin DB2, NULL, login3, password3, role
  ```

Usage

There are additional considerations when using **sp_addexternlogin**:

- **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.

  Note: You can use **sp_addexternlogin** only when Component Integration Services is configured.
- Mappings can be one-to-one (for specific users), role-to-one (role-based), many-to-one (server-based), or based on the client login and password from the TDS loginrec.
- The login and password have a many to one mapping. That is, you can assign all the users who need to log into a remote server the same name and password.
- When several external logins are set for a user, the following precedence is followed for user connections to a remote server. 1) one-to-one mapping, 2) if there is no one-to-one mapping, active role is used, 3) if neither one-to-one mapping nor active role is present,

then many-to-one mapping, 4) if none of the above is used then SAP ASE login and password.

- You can assign external logins to SAP ASE roles. You can assign anyone with a particular role a corresponding login name and password for any given remote server.
- When you establish a connection to a remote server for a user that has more than one role active, each role is searched for an external login mapping and uses the first mapping it finds to establish the login. This is the same order as displayed by the stored procedure **sp_activeroles**.
- If you perform role mapping, and a user's role is changed (using **set role**), any connections made to remote servers that used role mapping must be disconnected. You cannot do this if a transaction is pending. You cannot use **set role** if a transaction is active and remote connections are present that used role mapping.
- Before running **sp_addexternlogin**, add the remote server to the SAP ASE server with **sp_addserver**.
- *externname* and *externpasswd* 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 *loginame* account are the only users who can modify remote access for a given local user.

Permissions

The permission checks for **sp_addexternlogin** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any remote login` privilege. Any user can execute **sp_addexternlogin** for their own login.
Disabled	With granular permissions disabled, you must be a user with **sa_role** or **sso_role**. Any user can execute **sp_addexternlogin** for their own login.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**

Information	Values
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addserver* on page 46
- *sp_dropexternlogin* on page 268
- *sp_helpexternlogin* on page 406
- *sp_helpserver* on page 434

sp_addgroup

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

- **Example 1 –** Creates a group named `accounting` in the current database:

  ```
  sp_addgroup accounting
  ```

Usage

There are additional considerations when using **sp_addgroup**:

- **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.

See also **grant**, **revoke** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_addgroup** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any user` privilege.
Disabled	With granular permissions disabled, you must be the database owner, a user with **sso_role**, or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_adduser* on page 59
- *sp_changegroup* on page 103
- *sp_dropgroup* on page 273
- *sp_helpgroup* on page 408

sp_addlanguage

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`, `smalldatetime`, `date` or `time` 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

- **Example 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:

```
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
```

Usage

Usually, you add alternate languages from one of SAP ASE's Language Modules using the **langinstall** utility or the SAP ASE 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 **alter login** 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.

See also:

- **set** in *Reference Manual: Commands*
- **langinstall** in the *Utility Guide*

Permissions

The permission checks for **sp_addlanguage** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_droplanguage* on page 276
- *sp_helplanguage* on page 419
- *sp_modifylogin* on page 515
- *sp_setlangalias* on page 629

System Table Changes

The **sp_addlanguage** system procedure performs changes to system tables.

- **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 SAP ASE.
- **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_setlanalias**.
- **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, the SAP ASE 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, where *pubdate* is `datetime` data and *table* is any table:

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

sp_addlogin

Deprecated in SAP ASE versions 15.7 and later. To add a login account in SAP ASE, use the **create login** command.

sp_addmessage

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 1024 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, the SAP ASE server assumes that messages are in the default session language indicated by ***@@langid***.
- ***with_log*** **–** specifies whether the message is logged in the SAP ASE error log as well as in the Windows Event Log on Windows 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 specified, the existing message is overwritten; if **replace** is omitted, it is not. If you do not specify a value for **replace**, the parameter's default behavior specifies that the existing message is not overwritten.

Examples

- **Example 1 –** Adds a message with the number 20001 to `sysusermessages`:

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

- **Example 2 –** Adds a message with the number 20002 to `sysusermessages`. This message is logged in the SAP ASE error log, as well as in the Windows Event Log on Windows servers, if event logging is enabled. If a message numbered 20002 exists in the default session language, this message overwrites the old message:

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

Usage

sp_addmessage does not overwrite an existing message of the same number and *langid* unless you specify **@replace = “replace"**.

print and **raiserror** 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 the SAP ASE 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.

Only the user who created a message can execute **sp_addmessage** with the **replace** option to replace that original message.

See also **print**, **raiserror** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_addmessage** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, any user can execute **sp_addmessage**. To add a message with **with_log**, you must be the database owner or a user with `own database` privilege on the database.
Disabled	With granular permissions disabled, any user can execute **sp_addmessage**. To add a message with **with_log**, you must be the database owner or a user with sa_role. Only the user who created the message can execute **sp_addmessage** with the **replace** option to replace that original message.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	15
Audit option	**create**
Command or access audited	**sp_addmessage**
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – Message number • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_altermessage* on page 61
- *sp_bindmsg* on page 86
- *sp_dropmessage* on page 278
- *sp_getmessage* on page 353

sp_addobjectdef

(Component Integration Services only) 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.
- ***objecttype* –** is one of the values that specify the format of the object named by *objectdef*.. Valid values are:

  - **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. The SAP ASE server processes the result set from the RPC as a read-only table.

  Enclose the *objecttype* value in quotes.

  This table summarizes how each *objecttype* is used:

Table 1. 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

- **Example 1 –** Maps the local table `accounts` in the database `finance` to the remote object `pubs.dbo.accounts` in the remote server named MYSERVER. 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:

  ```
  sp_addobjectdef "finance.dbo.accounts",
  "MYSERVER.pubs.dbo.accounts", "table"
  ```

- **Example 2 –** 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`:

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

Usage

There are additional considerations when using **sp_addobjectdef**:

- **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_addobjectdef** 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. However, if a remote table exists, you need not use **sp_addobjectdef** before executing **create proxy_table**.

  create table is valid only for the *objecttype* values **table** and **file**. When either **create table** or **create existing table** is used, the SAP ASE 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.

See also:

- **create existing table**, **create table**, **drop table** in *Reference Manual: Commands*
- *Server Classes* in the *Component Integration Services User's Guide*

Permissions

The permission checks for **sp_addobjectdef** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the table owner or a user with `manage database` privilege.
Disabled	With granular permissions disabled, you must be the table owner, the database owner, or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addlogin* on page 35
- *sp_addserver* on page 46
- *sp_defaultloc* on page 212
- *sp_dropobjectdef* on page 279
- *sp_helpserver* on page 434

sp_addremotelogin

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 *remotenames* 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

- **Example 1 –** 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:

  ```
  sp_addremotelogin GATEWAY
  ```

  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`.
- **Example 2 –** Creates an entry that maps all logins from the remote server GATEWAY to the local user name "albert". The SAP ASE server adds a row to `sysremotelogins` with Albert's server user ID in the `suid` column and a null value for the `remoteusername`:

  ```
  sp_addremotelogin GATEWAY, albert
  ```

  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_remotelogin**.
- **Example 3 –** Maps a remote login from the remote user "pogo" on the remote server GATEWAY to the local user "ralph". The SAP ASE server adds a row to `sysremotelogins` with Ralph's server user ID in the `suid` column and "pogo" in the `remoteusername` column:

  ```
  sp_addremotelogin GATEWAY, ralph, pogo
  ```

Usage

There are additional considerations when using **sp_raddremotelogin**:

- 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_remotelogin** 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_remotelogin** 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.
- 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_remotelogin**.

See also:

- *System Administration Guide* for more information about setting up servers for remote procedure calls and for using "unified login."
- **isql** in the *Utility Guide*

Permissions

The permission checks for **sp_addremotelogin** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any remote login` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addlogin* on page 35
- *sp_addserver* on page 46
- *sp_dropremotelogin* on page 281
- *sp_helpremotelogin* on page 424
- *sp_helprotect* on page 425
- *sp_helpserver* on page 434
- *sp_remoteoption* on page 601

sp_addsegment

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

- **Example 1 –** Creates a segment named `indexes` for the database `pubs2` on the database device named `dev1`:

  ```
  sp_addsegment indexes, pubs2, dev1
  ```

- **Example 2 –** Creates a segment named `indexes` for the `pubs2` database on the database device named `pubs2_dev`:

  ```
  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
  ```

Usage

There are additional considerations when using **sp_addsegment**:

- You cannot create a segment on a device that already has an exclusive segment. If you attempt to do so, you see an error message similar to:

  ```
  A segment with a virtually hashed table exists on
  device orders_dat.
  ```

- **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 the *System Administration Guide* for more information.
- Although you can use **sp_addsegment** in a database that has both data and the log on the same device, such as when the database is created without the **log on** option, the SAP ASE server returns an error message if you create a database using:

```
create database dbname on devicename log on devicename with
override
```

See also**alter database**, **create index**, **create table**, **disk init** in *Reference Manual: Commands* .

Permissions

The permission checks for **sp_addsegment** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage database` permission
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_dropsegment* on page 287
- *sp_extendsegment* on page 330
- *sp_helpdb* on page 394
- *sp_helpdevice* on page 402
- *sp_placeobject* on page 577

sp_addserver

Defines a remote server, or the name of the local server; specifies the server for remote procedure calls (RPCs) when using the host and port parameters.

Syntax

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

Component Integration Services (CIS) only:

```
sp_addserver 'logical_server_name', ASEnterprise, 'host:port:filter'
```

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. A server *class* of "null" defaults to "ASEnterprise". Allowable values for the *class* parameter are:
  - **local** – local server (there can be only one) used only once after start-up, or after restarting the SAP ASE server, to identify the local server name so that it can appear in messages printed by the SAP ASE server
  - **null** – remote server with no category defined
  - **ASEnterprise** – all versions of SAP ASE; support for SQL Server 4.9 is not provided.
  - **ASAnywhere** – Adaptive Server Anywhere version 6.0 or later.
  - **ASIQ** – a server with server class ASIQ is any version of Adaptive Server IQ of 12.0 or later.
  - **direct_connect** (Component Integration Services only) – an Open Server-based application that conforms to the **direct_connect** interface specification.
  - **sds**– conforms to the interface requirements of a Specialty Data Store™ as described in the SAP ASE Specialty Data Store Developer's Kit manual.

  Note: The SAP ASE server does not support server class **db2**. To use **db2**, migrate your **db2** server class to **direct_connect** class.

- ***pname*** **–** is the name in the interfaces file for the server named *lname*. This enables you to establish local aliases for other SAP ASE servers or Backup Servers that you may need to communicate with. If you do not specify a *pname*, *lname* is used.

  (Component Integration Services only) You can use *pname* to specify the hostname or IP address and the port of the server you wish to connect to. This enables you to bypass the need for directory services (such as LDAP or an interfaces file) for the server when using the CT-Library. Use the following format:

  - `"hostname:port"`
  - `"ipaddr:port"`

  Note: You must enclose the hostname and port with single or double quotes to use this option.
- ***filter*** **–** in cluster environments – adds a remote server for remote procedure calls (RPCs).

  ```
  filter = ssl [= 'CN = common_name']
  ```

  Use this format to declare the *host:port* number:

  ```
  ip_address:port
  ```

Examples

- **Example 1 –** (In cluster environments) Adds a remote server named big_logical_server:

  ```
  sp_addserver 'big_logical_server', ASEntrprise,
      'maynard:23954:ssl= "CN=ase1.big server 1.com"'
  ```

  The rules for common names are the same as those used for dynamic listeners and the directory service entries.
- **Example 2 –** Adds an entry for a remote server named GATEWAY in `master.dbo.sysservers`. The *pname* is also GATEWAY:

  ```
  sp_addserver GATEWAY
  ```
- **Example 3 –** 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:

  ```
  sp_addserver GATEWAY, null, VIOLET
  ```
- **Example 4 –** Adds an entry for the local server named PRODUCTION:

  ```
  sp_addserver PRODUCTION, local
  ```
- **Example 5 –** (Component Integration Services only) Adds an entry for a remote SAP ASE server with the host name “myhost” with port number 10224:

  ```
  sp_addserver S1, ASEnterprise, "myhost:10224"
  ```

Note: If you use this syntax for *pname*, the SAP ASE site handler cannot successfully connect to this server; only CIS connections recognize this syntax for *pname*.

- **Example 6 –** (Component Integration Services only) Adds an entry for a remote SAP ASE server with the host IP 192.123.456.010 with port number 11222:

  ```
  sp_addserver S3, direct_connect, "192.123.456.010:11222"
  ```

Usage

There are additional considerations when using **sp_addserver**:

- 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.
- The SAP ASE 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 the SAP ASE 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.

See also *Remote Servers* in *Component Integration Services User's Guide*.

Permissions

The permission checks for **sp_addserver** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege. To execute **sp_addserver** for a server that is a shared disk cluster, you must be a user with `manage cluster` privilege and `manage server` privilege.
Disabled	With granular permissions disabled, you must be a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addremotelogin* on page 40
- *sp_dropremotelogin* on page 281
- *sp_dropserver* on page 289
- *sp_helpremotelogin* on page 424
- *sp_helpserver* on page 434
- *sp_serveroption* on page 623

sp_addthreshold

Creates a threshold to monitor space on a database segment. When free space on the segment falls below the specified level, the SAP ASE 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, the SAP ASE 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 SAP ASE server or on an Open Server. Thresholds cannot execute procedures on remote SAP ASE servers.

Examples

- **Example 1 –** Creates a threshold for `segment1`. When the free space on `segment1` drops below 200 pages, the SAP ASE server executes the procedure **pr_warning**:

  ```
  sp_addthreshold mydb, segment1, 200, pr_warning
  ```

- **Example 2 –** Creates a threshold for the `user_data` segment. When the free space on `user_data` falls below 100 pages, the SAP ASE server executes a remote procedure call to the Open Server **mail_me** procedure:

  ```
  sp_addthreshold userdb, user_data, 100, "o_server...mail_me"
  ```

- **Example 3 –** Creates a threshold on the `indexes` segment of the `pubs2` database. You can issue this command from any database:

  ```
  pubs2..sp_addthreshold pubs2, indexes, 100, pr_warning
  ```

Usage

When a threshold is crossed, the SAP ASE server executes the associated stored procedure. The SAP ASE server uses the following search path for the threshold procedure:

- If the procedure name does not specify a database, the SAP ASE 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_”, the SAP ASE server looks in the `sybsystemprocs` database.

If the procedure is not found in either database, the SAP ASE server sends an error message to the error log.

The SAP ASE 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.

See also:

- **create procedure**, **dump transaction** in *Reference Manual: Commands*

- *System Administration Guide* for more information about using thresholds.
- **lct_admin** in *Reference Manual: Building Blocks*

Permissions

The permission checks for **sp_addthreshold** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage database` privilege.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_dboption* on page 193
- *sp_dropthreshold* on page 291
- *sp_helpthreshold* on page 445
- *sp_modifythreshold* on page 520
- *sp_thresholdaction* on page 700

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 twice the size of the **@@thresh_hysteresis** pages from the closest threshold.

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."

The SAP ASE 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

To change or delete thresholds, use **sp_helpthreshold**, **sp_modifythreshold**, and **sp_dropthreshold**.

Procedure	Desription
sp_helpthreshold	For information about existing thresholds.
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, the SAP ASE 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.
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.

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

You cannot disable free-space accounting on log segments.

The Last-Chance Threshold

By default, the SAP ASE 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 the SAP ASE server and cannot be changed by users.

If the last-chance threshold is crossed before a transaction is logged, the SAP ASE 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 the SAP ASE server to roll back all transactions that have not yet been logged when the last-chance threshold is crossed.

All databases have a last-chance threshold, including `master`. The threshold is an estimate of the number of free log pages that are required to back up the transaction log. As you allocate more space to the log segment, the SAP ASE server automatically adjusts the last-chance threshold.

Creating Threshold Procedures

Any user with **create procedure** privilege 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 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.

The SAP ASE 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 the SAP ASE server.

Include **print** and **raiserror** statements in the threshold procedure to send output to the error log.

sp_addtype

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 SAP ASE server-supplied datatype on which to base the user-defined datatype. You can specify any SAP ASE datatype except `timestamp`.

  The `char`, `varchar`, `unichar`, `univarchar`, `nchar`, `nvarchar`, `binary`, and `varbinary` datatypes expect a *length* in parentheses. If you do not supply one, the SAP ASE server uses the default length of 1 character.

  The `float` datatype expects a binary *precision* in parentheses. If you do not supply one, the SAP ASE 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, the SAP ASE 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 the SAP ASE 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*, the SAP ASE 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

- **Example 1 –** Creates a user-defined datatype called `ssn` to be used for columns that hold social security numbers. Since the *nulltype* parameter is not specified, the SAP ASE 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):

```
sp_addtype ssn, "varchar(11)"
```

- **Example 2 –** Creates a user-defined datatype called `birthday` that allows null values:

```
sp_addtype birthday, "datetime", null
```

- **Example 3 –** 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:

```
sp_addtype temp52, "numeric(5,2)"
```

- **Example 4 –** 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:

```
sp_addtype "row_id", "numeric(10,0)", "identity"
```

- **Example 5 –** 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`:

```
sp_addtype systype, sysname
```

Usage

- **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 SAP ASE-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.
- If `unichar` or `univarchar` 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 2 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
  ```
- If a user-defined datatype is defined with the IDENTITY property, all columns created from it are IDENTITY columns. You can specify IDENTITY, NOT NULL, or neither 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.

See also:

- **create default**, **create rule**, **create table** in *Reference Manual: Commands*
- *User-Defined Datatypes* in *Reference Manual: Building Blocks*

Permissions

Any user can execute **sp_addtype**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**

Information	Values
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_bindefault* on page 80
- *sp_bindrule* on page 88
- *sp_dboption* on page 193
- *sp_droptype* on page 292
- *sp_rename* on page 605
- *sp_unbindefault* on page 714
- *sp_unbindrule* on page 720

sp_addumpdevice

Adds a dump device to the SAP ASE 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 the SAP ASE 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. 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). You should specify a capacity that is slightly below the rated capacity for your device.

Examples

- **Example 1 –** Adds a 40MB tape device. Dump and load commands can reference the device by its physical name, `/dev/nrmt8`, or its logical name, `mytapedump`:

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

- **Example 2 –** Adds a disk device named `mydiskdump`. Specify an absolute or relative path name and a file name:

  ```
  sp_addumpdevice "disk", mydiskdump, "/dev/rxy1d/dump.dat"
  ```

Usage

There are additional considerations when using **sp_addumpdevice**:

- **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 SAP ASE 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.

See also **disk init**, **dump database**, **dump transaction**, **load database**, **load transaction** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_addumpdevice** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage disk` privilege.
Disabled	With granular permissions disabled, you must be a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_dropdevice* on page 262
- *sp_helpdevice* on page 402

sp_adduser

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

- **Example 1 –** Adds "margaret" to the database. Her database user name is the same as her SAP ASE login name, and she belongs to the default group, "public":

  ```
  sp_adduser margaret
  ```

- **Example 2 –** 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":

  ```
  sp_adduser haroldq, harold, fort_mudge
  ```

Usage

There are additional considerations when using **sp_adduser**:

- 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 SAP ASE. 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`.

See also **grant**, **revoke**, **use** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_adduser** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any user` privilege.
Disabled	With granular permissions disabled, you must be the database owner, a user with **sa_role**, or a user with sso_role.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addalias* on page 14
- *sp_addgroup* on page 30
- *sp_changegroup* on page 103
- *sp_dropalias* on page 252
- *sp_dropgroup* on page 273
- *sp_helpuser* on page 446

sp_altermessage

Enables and disables the logging of a system-defined or user-defined message in the SAP ASE 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 30 bytes. The only valid parameter is **with_log**.
- ***parameter_value*** **–** is the new value for the parameter specified in *parameter*. The maximum length is 5 bytes. Values are **true** and **false**.

Examples

- **Example 1 –** Specifies that message number 2000 in `sysmessages` should be logged in the SAP ASE error log and also in the Windows Event Log (if logging is enabled):

  ```
  sp_altermessage 2000, 'with_log', 'TRUE'
  ```

Usage

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 servers, **sp_altermessage** also enables and disables logging in the Windows Event Log.

Permissions

The permission checks for **sp_altermessage** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner or a user with `own database` privilege.
Disabled	With granular permissions disabled, you must be the database owner or a user with sa_role.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addmessage* on page 35
- *sp_dropmessage* on page 278

sp_audit

Allows a system security officer to configure auditing options.

Syntax

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

Or:

```
sp_audit 'restart'
```

Parameters

- ***option*** **–** is the name of the auditing option to set. The valid auditing options are:

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. You can only use this option to specify system roles. **Note:** Auditing 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.
cluster	Audits cluster commands.
cmdtext	Audits all actions of a particular user. When auditing is configured and enabled, and **cmdtext** is set, system stored procedure and command password parameters are replaced with a fixed length string of asterisks in the audit records contained in the audit logs. This protects passwords from being seen by other with access to the audit log.
config_history	Enables or disables auditing for configuration history.
create	Audits the creation of database objects.

Option	Description
dbaccess	Audits access to the current database from another database.
dbcc	Audits the execution of any **dbcc** command.
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**.
encryption_key	Audits **create encryption key**, **sp_encryption**, **drop encryption key,** and **alter encryption key**
errors	Audits errors, whether fatal or not.
exec_procedure	Audits the execution of a stored procedure.
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**.
insert	Audits the insertion of rows into a table or view.
install	Audits the installation of Java classes.
load	Audits the execution of the **load database** or **load transaction**.
login	Audits all login attempts into the SAP ASE server.
login_locked	Audits the hostname and network IP address when a login account is locked due to exceeding the configured number of failed login attempts.
logout	Audits all logout attempts from the SAP ASE server.
mount	Audits **mount database** commands.

Option	Description
network	Audits specific network-related events, such as listener events. The valid settings are: • **on** • **off** • **pass** • **fail**
quiesce	Audits **quiesce database** commands.
reference	Audits references between tables.
remove	Audits the removal of Java classes.
revoke	Audits the execution of the **revoke**.
rpc	Audits the execution of remote procedure calls.
security	Audits security-relevant events. See
select	Audits the execution of the **select**.
setuser	Audits the execution of the **setuser**.
sproc_auth	Enables auditing for authorization checks that are performed inside system stored procedures
table_access	Audits access to any table by a specific user.
transfer table	Audits the execution of the transfer table command
truncate	Audits the execution of the **truncate table**.
unbind	Audits the execution of the **sp_unbindrule**, **sp_unbindmsg**, and **sp_unbin-default**.
unmount	Audits the execution of the **umount database** command.
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 parameter that lets you specify **all**, a system role, or the name of a specific login to be audited. However, system roles can only be specified if you use the **all** option. You cannot audit individual options for a system role.
- ***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** – audits 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.
    - **network** – audits specific network-related events, such as listener events.

  See the *System Administration Guide* for more information about the *object_name* values that are valid with each *option* value.
- ***setting* –** is the level of auditing. If you do not specify a value for *setting*, the SAP ASE server displays the current auditing setting for the option. Valid values for the *setting* parameter are:

    - **on** – activates auditing for the specified option. The SAP ASE 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** – activates 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**. The SAP ASE server generates audit records regardless of whether events pass or fail permission checks.

  Settings of:

    - **on** or **off** – apply to all auditing options
    - **pass** and **fail** – apply to all options except **cmdtext**, **errors**, and **adhoc**. For these options, only **on** or **off** applies. The initial, default value of all options is **off**. If you select the **cmdtext** option to either **pass** or **fail**, the SAP ASE server replaces the value with **on**.
- **restart –** If the audit process is forced to terminate due to an error, **sp_audit** can be manually restarted by entering:

  ```
  sp_audit restart
  ```

  The audit process can be restarted provided that no audit was currently running, but that the audit process has been configured to run by entering **sp_configure "auditing" 1**.

Examples

- **Example 1 –** Sets the **login_locked** audit option to initiate auditing of hostname and network IP addresses when a login account is locked due to exceeding the configured number of failed login attempts:

  ```
  sp_audit "login_locked","all","all","ON"
  ```

  If the audit tables are full and the event cannot be logged, a message with the information is sent to the errorlog.

  Monitoring the audit logs for the **Locked Login** event (112) helps to identify attacks on login accounts.

- **Example 2 –** Initiates auditing for SSL security-relevant events. Both successful and failed events are audited:

  ```
  sp_audit "security", "all", "all", "on"
  ```

  ```
  sample records added:
  ```

  To view the events from `sybsecurity`:

  ```
  select * from sybsecurity..sysaudits_01 where event=99
  ```

- **Example 3 –** Displays the setting of the **security** auditing option:

  ```
  sp_audit "security", "all", "all"
  ```

- **Example 4 –** Initiates auditing for the creation of objects in the `master` database, including **create database**.

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

- **Example 5 –** Audits commands in the `pubs2` database:

  ```
  sp_audit "encryption_key", "all", "pubs2", "on"
  ```

- **Example 6 –** Initiates auditing for the creation of all objects in the `db1`database:

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

- **Example 7 –** Initiates auditing for all failed executions by a system administrator.

  ```
  sp_audit "all", "sa_role", "all", "fail"
  ```

- **Example 8 –** 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` are audited for updates. The auditing for existing tables is not affected.

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

- **Example 9 –** Initiates auditing for all transfer table commands entered for the `titles` table:

  ```
  sp_audit "transfer table", "all", "titles", "on"
  ```

Usage

- **sp_audit** determines what is 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. For more information, see **sp_configure**.
- 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**
  - **exec_procedure**
  - **exec_trigger**
  - **insert**
  - **select**
  - **update**
- The configuration parameters that control auditing are:
  - **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. The SAP ASE 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.

  All auditing configuration parameters are dynamic and take effect immediately.

See also:

- For more information about configuring the SAP ASE server for auditing, see **sp_configure** in the *System Administration Guide*.
- **bcp** in the *Utility Guide*

Permissions

The permission checks for **sp_audit** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage auditing` privilege.
Disabled	With granular permissions disabled, you must be a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_configure* on page 167
- *sp_addauditrecord* on page 16

sp_audit and Security

The **sp_audit security** option audits these security-relevant events:

- Starting up or shutting down the server
- Activating or deactivating a role
- Issuing these commands:
  - **addcert**
  - **connect**
  - **create** and **drop login**
  - **dropcert**
  - **create**, **drop**, **alter**, **grant**, and **revoke role**
  - **kill**
  - **online database**
  - **set proxy**
  - **set session authorization**
  - **sp_configure**
- Issuing these built-in functions.
  - **config_admin**
  - **attr_notify**
  - **ha_check_alive**
  - **ha_retrestrictionclass**
  - **ha_hacluster_verify**

    - **ssl_admin**
    - **set_password**
    - **ha_add_companion**
    - **ha_getversion**
    - **ha_getrcs**
    - **js_wakeup**
    - **ws_admin**
    - **valid_user**
    - **ha_remove_companion**
    - **ha_failback**
    - **ha_setrcs**
    - **unlock_admin_account**
- Issuing **proc_role** from within a system procedure
- Regenerating the SSO passwords

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.

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

- **Example 1 –** The current user is automatically connected to the server MYSERVER the next time that user logs in. The user’s connection is placed in passthrough mode:

  ```
  sp_autoconnect MYSERVER, true
  ```

- **Example 2 –** Disables the autoconnect feature for the user “steve”:

```
sp_autoconnect MYSERVER, false, steve
```

Usage

- **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.
- 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, then reconnect before the change is made.
- Use **disconnect** to exit passthrough mode.

See also **connect to...disconnect**, **grant** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_autoconnect** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any login` privilege. Any user can execute **sp_autoconnect** for themselves.
Disabled	With granular permissions disabled, you must be a user with **sa_role**. Any user can execute **sp_autoconnect** for themselves.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addlogin* on page 35
- *sp_addserver* on page 46
- *sp_passthru* on page 557
- *sp_remotesql* on page 603

sp_autoformat

A utility stored procedure that produces readable result set data, **sp_autoformat** reformats the width of variable-length character data to display only non-blank characters. Trailing blanks are truncated in the output.

Syntax

```
sp_autoformat fulltabname[, selectlist, whereclause, orderby]
```

Parameters

- ***fulltabname* –** specifies the name of table from which data is being selected. Use owner names if the object owner is not the user running the command.
- ***selectlist* –** specifies the comma-separated list of columns to be selected in the result set. Columns in the table can be renamed using the **<*name*> = <*column*>** notation. See examples. If *selectlist* is not provided, all columns in the table specified are output in column ID order.
- ***whereclause* –** is a search predicate, specified as a **where** clause, that filters out rows from the table being selected.
- ***orderby* –** is an optional **order by** clause that specifies the order in which the output result set is presented.

Examples

- **Example 1 –** Returns a result set from a **select** statement similar to `select id, colid, name from syscolumns where id = 3`, where the character columns are autoformatted:

```
1> sp_autoformat "syscolumns", "id, colid, name", "where id = 3"
2> go
```

```
id           colid  name
------------ ------ -----------
           3      1 id
           3      2 number
           3      3 colid
           3      4 status
           3      5 type
           3      6 length
           3      7 offset
           3      8 usertype
           3      9 cdefault
           3     10 domain
           3     11 name
           3     12 printfmt
           3     13 prec
           3     14 scale
           3     15 remote_type
           3     16 remote_name
           3     17 xstatus
           3     18 xtype
           3     19 xdbid
           3     21 accessrule
           3     22 status2
```

- **Example 2 –** Renames the output columns using the following syntax:

```
[ < AS-Name label of Column> ][ ]*=[ ]*<column name>
```

<AS-Name label of Column> is optional, and you can use white spaces around the **=** separator:

```
1> sp_autoformat syscolumns, "'Object Id' = id, 'Column
Name'=name,
     'Column ID'=colid", "where id = 3"
2> go
```

```
 Object Id   Column Name Column ID
------------ ----------- ---------
           3 id                  1
           3 number              2
           3 colid               3
           3 status              4
           3 type                5
           3 length              6
           3 offset              7
           3 usertype            8
           3 cdefault            9
           3 domain             10
           3 name               11
           3 printfmt           12
           3 prec               13
           3 scale              14
           3 remote_type        15
           3 remote_name        16
           3 xstatus            17
           3 xtype              18
           3 xdbid              19
           3 accessrule         21
           3 status2            22

(1 row affected)
```

- **Example 3 –** Uses the *orderby* parameter to specify an ordering in the result output:

```
sp_autoformat @fulltabname = 'syscolumns',
              @selectlist = "id, name",
              @orderby = "ORDER BY name"
```

- **Example 4 –** Generates an autoformatted result when you select from multiple tables, or if you have a complex SQL **select** statement with expressions in the **select** list, you must:

1. Use temporary tables to generate the result set:

   The following generates the list of the columns with the highest column ID on all system catalogs:

```
select o.id, o.name, c.colid, c.name
from sysobjects o, syscolumns c
where o.id < 100 and o.id = c.id
  and c.colid = (select max(c2.colid) from syscolumns c2
                where c2.id = c.id)
order by o.name
```

   The following generates the same result set with auto-formatting of character data using a temporary table to produce readable output, and includes minor changes to provide column names in the temporary table:

```
select o.id, ObjectName = o.name, c.colid, ColumnName = c.name
into #result
from sysobjects o, syscolumns c
where o.id < 100 and o.id = c.id
  and c.colid = (select max(c2.colid) from syscolumns c2
                 where c2.id = c.id)
```

2. Use **sp_autoformat** on that temporary table to produce formatted output:

   The **order by** clause in the original **select** statement is skipped when generating the temporary table, and is instead added to the call to **sp_autoformat** when generating the output result.

```
1> exec sp_autoformat @fulltabname = #result, @orderby = "order
by
   ObjectName"
2> go
```

```
id       ObjectName        colid ColumnName
-------- ----------------- ------ -------------
      11 sysalternates         2 altsuid
      21 sysattributes        13 comments
      55 syscertificates       6 suid
      45 syscharsets           8 sortfile
       3 syscolumns           22 status2
       6 syscomments           8 status
      37 sysconfigures          9 value4
      17 sysconstraints         7 spare2
      38 syscurconfigs         15 type
      30 sysdatabases          19 status4
      12 sysdepends            10 readobj
      35 sysdevices             7 mirrorname
      43 sysengines            12 starttime

      ...

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

You can further process the temporary table to report only on the required output for selected tables, as shown below:

```
1> exec sp_autoformat #result, "id, 'Object Name' =
ObjectName,
   'Column Name' = ColumnName", "where id < 5"
2> go
```

```
id       Object Name Column Name
------- ----------- -----------
      1 sysobjects  loginame
      2 sysindexes  crdate
      3 syscolumns  status2
      4 systypes    accessrule
```

Usage

- In SAP ASE version 15.0.3 and higher, **sp_autoformat** accepts columns of datatypes `int` (`smallint`, `bigint`, `tinyint`, `unsigned int`), `numeric`, `money`, `date/time`, and `float`, `real`, and `double` precision.
- **sp_autoformat** looks for an object only in the current database. To use **sp_autoformat** on temporary tables, issue the procedure from `tempdb`.
- **sp_autoformat** does not validate that the columns referenced in any of the parameters actually exist in the table specified by the *fulltabname* parameter. **sp_autoformat** fails if you reference any nonexistent columns.
- Provide only one instance of a column in the `select` list.

Return codes are:

- 0 – successful completion
- 1 – internal error, or usage error in invocation
- Other – any other errors raised by the SAP ASE server during the execution of the generated SQL statement are returned back to the caller.

Restrictions for **sp_autoformat** are:

- **sp_autoformat** uses internal SQL variables to generate SQL statements that are then executed using `execute immediate`. The length of the generated SQL statement is limited to 2K bytes. Auto-formatting result sets for a large column list, or columns with

long names can sometimes cause an error due to insufficient size of the buffer for the generated SQL statement.
- Quoted identifiers are not supported for either the table or column names. If you have result sets that use quoted idenfiers and that need autoformatting:
  1. Generate the required data in a temporary table, where the columns in the temporary table do not have any quoted identifiers.
  2. Use **sp_autoformat** to produce the required output using the temporary table.
  3. Rename the columns in the *selectlist* in the desired output format.

Permissions

No permission checks are performed for **sp_autoformat**. Permission checks do not differ based on the granular permissions settings. Users selecting from the tables must have appropriate **select** privileges.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_bindcache

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 with an index, `text` object, or `image` object that 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

- **Example 1 –** Binds the `titles` table to the cache named `pub_cache`:

  ```
  sp_bindcache pub_cache, pubs2, titles
  ```

- **Example 2 –** Binds the clustered index `titles.title_id_cix` to the `pub_ix_cache`:

  ```
  sp_bindcache pub_ix_cache, pubs2, titles, title_id_cix
  ```

- **Example 3 –** Binds `pubs2` to the `tempdb_cache`:

  ```
  sp_bindcache tempdb_cache, pubs2
  ```

- **Example 4 –** Binds the `pubs2` transaction log, `syslogs`, to the cache named `logcache`:

  ```
  sp_bindcache logcache, pubs2, syslogs
  ```

- **Example 5 –** Binds the `image` chain for the `au_pix` table to the cache named `pub_cache`:

  ```
  sp_bindcache pub_cache, pubs2, au_pix, "text only"
  ```

Usage

- 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, if the operation is not dynamic, restart the SAP ASE 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
  ```

  For more information, see **sp_dboption**.
- 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 the SAP ASE server is restarted. If a cache of the same name is created, the bindings become valid when the SAP ASE 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.
- Although you can still use **sp_bindcache** on a system `tempdb`, the binding of the system `tempdb` is now non-dynamic. Until you restart the server:
  - The changes do not take effect
  - **sp_helpcache** reports a status of “P” for pending, unless you have explicitly bound the system `tempdb` to the default data cache, in which case the status as “V” for valid, because by default the system `tempdb` is already bound to the default datacache.

Restrictions for **sp_bindcache** are:

- 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.

Permissions

The permission checks for **sp_bindcache** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage data cache` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_cacheconfig* on page 90
- *sp_configure* on page 167
- *sp_dboption* on page 193
- *sp_estspace* on page 324
- *sp_help* on page 358
- *sp_helpcache* on page 379
- *sp_helpdb* on page 394
- *sp_helpindex* on page 409
- *sp_poolconfig* on page 582
- *sp_spaceused* on page 676
- *sp_unbindcache* on page 710
- *sp_unbindcache_all* on page 713

sp_bindefault

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**.

You cannot bind defaults to computed columns.

- **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

- **Example 1 –** 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:

```
sp_bindefault today, "employees.startdate"
```

- **Example 2 –** 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):

```
sp_bindefault def_ssn, ssn
```

- **Example 3 –** 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:

```
sp_bindefault def_ssn, ssn, futureonly
```

Usage

There are additional considerations when using **sp_bindefault**:

- 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 SAP ASE 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, the SAP ASE 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.

See also **create default**, **create table**, **drop default** in *Reference Manual: Commands*.

Permissions

You must be the table owner or the user datatype owner to execute **sp_bindefault**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	6
Audit option	**bind**
Command or access audited	**sp_bindefault**
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – Name of default • *Proxy information* – Original login name, if **set proxy** in effect

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_unbindefault* on page 714

sp_bindexeclass

Associates an execution class with a client application, login, stored procedure, or default execution class.

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`. If *object_type* is **df**, it should be null.
- ***object_type*** **–** identifies the type of `object_name`. Use:
  - **ap** for application
  - **df** for user-defined default execution class
  - **lg** for login
  - **pr** for stored procedure
  - **sv** for a service task (valid only in threaded mode)
- ***scope*** **–** is the name of a client application or login, or it can be NULL for **ap**, **df**, **lg**, or **sv** objects. For objects, scope is the name of the stored procedure owner (user name). 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`

Examples

- **Example 1 –** This statement specifies that Transact-SQL applications 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:

  ```
  sp_bindexeclass 'isql', 'ap', NULL, 'EC3'
  ```

- **Example 2 –** 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** executes with `EC3` attributes:

```
sp_bindexeclass 'sa', 'lg', 'isql', 'EC1'
```

- **Example 3 –** This statement assigns `EC3` attributes to the stored procedure named `my_proc` owned by user `kundu`:

  ```
  sp_bindexeclass 'my_proc', 'PR', 'kundu', 'EC3'
  ```

- **Example 4 –** This statement assigns `CLASS1` attributes to all tasks that are running with default execution attributes:

  ```
  sp_bindexeclass NULL, 'DF', NULL, 'CLASS1'
  ```

- **Example 5 –** Binds the license heartbeat operation to the core execution task:

  ```
  sp_bindexeclass "License Heartbeat", sv, NULL, core
  ```

Usage

There are additional considerations when using **sp_bindexeclass**:

- When binding an execution class to a default execution class, all tasks running with default execution attributes run with attributes of the new class.
- You can bind service tasks to existing execution classes created to manage user tasks. That is, service tasks and user tasks can coexist in the same execution class.
- The `monServiceTask` monitoring table includes all services tasks, including their name and current binding.
- **sp_bindexeclass** associates an execution class with a client application, login, or stored procedure. It can also associate an execution class to the default execution class. Use **sp_addexeclass** to create execution classes.
- 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 invocations 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. If you do not specify a user-defined default execution class, then the object binds to the system-defined execution class `EC2`.
- You can use **sp_bindexeclass** to bind a RepAgent thread to an execution class using **rep agent** as the application without generating an error. However, because of restrictions in the SAP ASE server, the priority attribute is set to medium, and the binding has no effect.

- Binding fails when you attempt to bind an active process to an engine group with no online engines.
- The SAP ASE 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.

See also **isql** in the *Utility Guide*.

Permissions

The permission checks for **sp_bindexeclass** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any execution class` privilege. For ECO, you must be a user with `manage any execution class` and **sybase_ts_role**.
Disabled	With granular permissions disabled, you must be a user with **sa_role**. For ECO, you must be a user with **sa_role** and **sybase_ts_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addexeclass* on page 23
- *sp_showexeclass* on page 652
- *sp_unbindexeclass* on page 716

sp_bindmsg

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

- **Example 1 –** Binds user message number 20100 to the `positive_balance` constraint:

  ```
  sp_bindmsg positive_balance, 20100
  ```

Usage

There are additional considerations when using **sp_bindmsg**:

- **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*.

See also **alter table**, **create table** in *Reference Manual: Commands*.

Permissions

You must be the constraint owner to execute **sp_bindmsg**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	7
Audit option	**bind**
Command or access audited	**sp_bindmsg**
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – Message ID • *Proxy information* – Original login name, if **set proxy** in effect

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addmessage* on page 35
- *sp_getmessage* on page 353
- *sp_unbindmsg* on page 718

sp_bindrule

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

- **Example 1 –** 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`:

  ```
  sp_bindrule today, "employees.startdate"
  ```

- **Example 2 –** 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):

  ```
  sp_bindrule rule_ssn, ssn
  ```

- **Example 3 –** 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:

  ```
  sp_bindrule rule_ssn, ssn, futureonly
  ```

Usage

There are additional considerations when using **sp_bindrule**:

- 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 SAP ASE server-supplied datatype or to a `text` or an `image` column.
- You cannot bind a rule to a system table.
- You cannot bind a rule to a computed column.
- 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 datatypes. Binding a rule to a column replaces a rule bound to the datatype of that column; however, binding a rule to a datatype does not replace a rule bound to a column of that user-defined datatype.
- 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.

See also **create rule**, **drop rule** in *Reference Manual: Commands*.

Permissions

You must be the table owner or user datatype owner to execute **sp_bindmsg**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	8
Audit option	**bind**
Command or access audited	**sp_bindrule**
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – Name of the rule • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_unbindrule* on page 720

sp_cacheconfig

Creates, configures, reconfigures, and drops data caches, and provides information about them.

Syntax

```
sp_cacheconfig [cachename [, "cache_size[P | K | M | G]"]
    [, logonly | mixed | inmemory_storage][, strict | relaxed]]
    [, "cache_partition=[1 | 2 | 4 | 8 | 16 | 32 | 64]"]
    [, instance instance_name]
```

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 SAP ASE 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 256 times the logical page size of the server. 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. The cache size is in multiples of the logical page size.

- **logonly | mixed | inmemory_storage –** specifies the type of cache. **inmemory_storage** indicates you are creating a cache for an in-memory or relaxed-durability database.
- **strict | relaxed –** specifies the cache replacement policy.
- **cache_partition –** specifies the number of partitions to create in the cache.
- ***instance_name* –** (In cluster environments) Is the name of the instance with a cache that you are adjusting.

Examples

- **Example 1 –** Creates the data cache `pub_cache` with 10MB of space. All space is in the default logical page size memory pool:

  ```
  sp_cacheconfig pub_cache, "10M"
  ```

- **Example 2 –** Reports the current configuration of `pub_cache` and any memory pools in the cache:

  ```
  sp_cacheconfig pub_cache
  ```

- **Example 3 –** Drops `pub_cache` at the next start of the SAP ASE server:

  ```
  sp_cacheconfig pub_cache, "0"
  ```

- **Example 4 –** Creates `pub_log_cache` and sets its type to **logonly** in a single step:

  ```
  sp_cacheconfig pub_log_cache, "2000K", logonly
  ```

- **Example 5 –** 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:

  ```
  sp_cacheconfig pub_log_cache, "2000K"
  sp_cacheconfig pub_log_cache, logonly
  ```

- **Example 6 –** Creates a cache and sets the size, type, replacement policy and number of cache partitions:

  ```
  sp_cacheconfig 'newcache', '50M', mixed, strict,
  "cache_partition=2"
  ```

- **Example 7 –** Creates an in-memory storage named `pubs3_imdb`:

  ```
  sp_cacheconfig pubs_imdb, '500M', inmemory_storage
  ```

- **Example 8 –** (In cluster environments) Displays the cache for instance blade1:

  ```
  sp_cacheconfig 'instance blade1'
  ```

- **Example 9 –** (In cluster environments) Sets the size of the Sales Cache size on blade1 to 100 megabytes:

  ```
  sp_cacheconfig 'Sales Cache', '100M', 'instance blade1'
  ```

- **Example 10 –** (In cluster environments) Sets the size of the Sales Cache size on blade1 to 0 megabytes, effectively dropping the cache.

```
sp_cacheconfig 'Sales Cache', '0M', 'instance blade1'
```

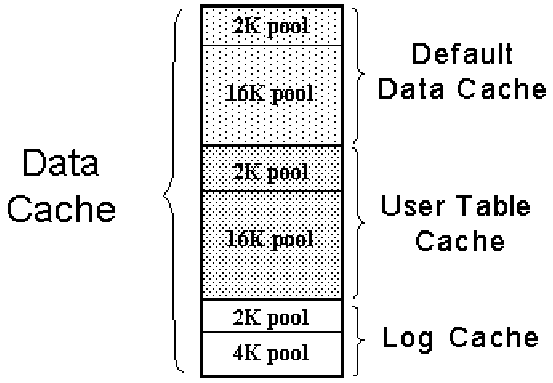
Usage

- The minimum cache size is 256 times the logical page size. For example, a 4K server would have a minimum cache size of 1024K.
- If the SAP ASE server is unable to allocate all the memory requested while you are creating a new cache or adding memory to an existing cache, it allocates all the available memory. However, this additional memory is allocated at the next restart of the SAP ASE server.
- If there are objects bound to cache (including the default cache), you cannot delete the cache until you unbind the objects.
- (In cluster environments) If you do not specify an instance_name, the cache for the cluster is displayed.
- Some of the actions you perform with **sp_cacheconfig** are dynamic (do not require a restart of the SAP ASE server) and some are static (require a reboot). The dynamic and static actions are:

Dynamic sp_cacheconfig Actions	Static sp_cacheconfig Actions
Adding a new cache	Changing the number of cache partitions
Adding memory to an existing cache	Reducing a cache size
Deleting a cache	Changing the replacement policy
Changing a cache type	

- When you first create a data cache:
  - All space is allocated to the logical page size memory pool.
  - The default type is **mixed**.
- This figure shows a data cache for a 2K server 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 1: Data Cache With Default and User-Defined Caches



- The default data cache must always have the type **default**, and no other cache can have the type **default**.
- The SAP ASE 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. Cache usage for Transact-SQL commands, depending on the binding of the database or object, are:

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**, **tableal-loc**	Bound cache	Bound cache	Default data cache
drop table	Bound cache	Bound cache	Default data cache

- Recovery uses only the logical page size 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 logical page size 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, from a server using 2K, shows the configuration for:

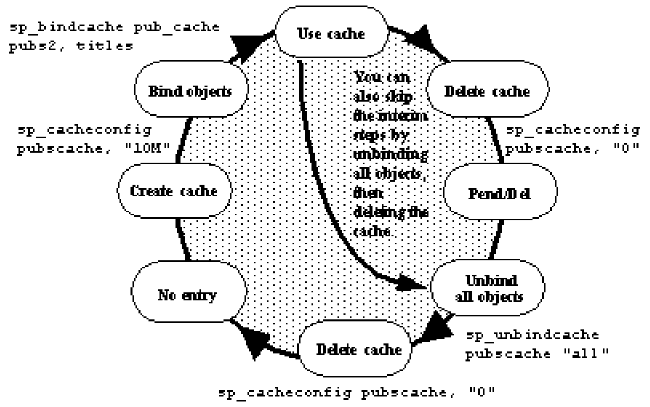
- The default data cache with two pools: a 2K pool and a 16K pool. The default data cache has 2 partitions.
- `pubs_cache` with two pools: 2K and 16K
- `pubs_log`, with the type set to **logonly** and cache replacement policy set to **relaxed**, with a 2K pool and a 4K pool

```
Cache Name              Status    Type     Config Value  Run Value
 ----------------------- --------- -------- ------------
--------
default data cache        Active   Default      0.00 Mb   26.09
Mb
pubs_cache               Active   Mixed       10.00 Mb  10.00 Mb
pubs_log                 Active    Log Only      2.40 Mb    2.40 M
                                             ------------- --------
                                   Total     12.40 Mb   38.49 Mb
==============================================================
=====
Cache: default data cache,   Status: Active,   Type: Default
      Config Size: 0.00 Mb,   Run Size: 26.09 Mb
     Config Replacement: strict LRU,   Run Replacement: strict
LRU
     Config Partition:          2,   Run Partition:          2
 IO Size  Wash Size Config Size  Run Size     APF Percent
 -------- --------- ------------ ------------ -----------
     2 Kb   3704 Kb      0.00 Mb     18.09 Mb     10
    16 Kb   1632 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
     Config Partition:          1,   Run Partition:          1
 IO Size  Wash Size Config Size  Run Size     APF Percent
 -------- --------- ------------ ------------ -----------
     2 Kb   1228 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
     Config Partition:          1,   Run Partition:          1
 IO Size  Wash Size Config Size  Run Size     APF Percent
 -------- --------- ------------ ------------ -----------
     2 Kb    206 Kb      0.00 Mb      1.01 Mb     10
    16 Kb    272 Kb      1.40 Mb      1.39 Mb     10
```

The meaning of the columns in the output are:

Column	Meaning
Cache Name	The name of the cache.
Status	One of the following: • "Active" • "Pend/Act" • "Pend/Del" 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.
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. If the value is 0, the size has not been explicitly configured, and a default value is used.
Run Value or Run Size	The size of the cache or pool now in use on the SAP ASE server.
Config/ Run Replacement	The cache policy (strict or relaxed) that is used for the cache after the next restart, and the current replacement policy. These differ only if the policy has been changed since the last reboot.
Config/Run Partition	The number of cache partitions that is used for the cache, and the current number of partitions. These differ if **sp_cacheconfig** has been used to change the number of partitions since the last reboot.
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.

Figure 2: Effects of Restarts and sp_cacheconfig on Cache Status



- You can also configure caches and pools by editing the configuration file. For more information, see the *System Administration Guide*.

Permissions

The permission checks for **sp_cacheconfig** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage data cache` privilege. Any user can execute **sp_cacheconfig** to view cache configurations
Disabled	With granular permissions disabled, you must be a user with **sa_role**. Any user can execute **sp_cacheconfig** to view cache configurations

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_bindcache* on page 76
- *sp_helpcache* on page 379
- *sp_poolconfig* on page 582
- *sp_unbindcache* on page 710
- *sp_unbindcache_all* on page 713

Data Cache Memory

When the SAP ASE server is first installed, all data cache memory is assigned to the logical page size 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 with databases that are not bound to a cache.

- When you create data caches, the memory allocation is validated against **max memory**. Memory for caches is allocated out of the memory allocated to the SAP ASE server with the **total logical_memory** configuration parameter. To increase the amount of space available for caches, increase **total logical memory**, or decrease other configuration settings that use memory. If the sum of **total logical memory** and additional memory requested is greater than **max memory**, then the SAP ASE server issues and error and does not perform the changes.

  The default cache is used for all objects, including system tables, that are not bound to another cache, and is the only cache used during recovery. For more information, see 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 free memory. To see the amount of overhead required for a specific size of cache, use **sp_helpcache**, giving the size:

  ```
  sp_helpcache "200M"
  ```

  ```
  10.38Mb of overhead memory will be needed to manage
  a cache of size 200M
  ```

  This is only an estimate of the overhead. The actual overhead may be larger because of runtime issues.

Creating Cache for In-Memory or Relaxed Durability Databases

Information on creating cache for in-memory or relaxed durability databases.

- The cache name cannot be longer than 127 bytes.
- The minimum size of in-memory storage cache is 256 logical pages (512K on a server using 2K logical pages).
- You cannot:
  - Include the **strict** or **relaxed** replacement strategies for in-memory storage. By default, **sp_cacheconfig** uses a replacement strategy of **none** for in-memory storage cache.
  - Create large I/O pools for in-memory storage cache (in-memory databases do not perform I/O). The SAP ASE server issues an error if you use **sp_poolconfig** to create buffer pools an in-memory storage cache.
  - Change the cache type from **mixed** to **logonly**, or vice-versa.

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 smallest pool.

To reduce the size of an existing cache, all of the space must be available in the logical page size pool. You may need to use **sp_poolconfig** to move space from other pools to this pool.

If you have a database or any nonlog objects bound to a cache, you cannot change its type to **logonly**.

Using Cache Partitions

Cache partitions can be used to reduce cache spinlock contention without needing to create separate caches and bind database objects to them.

For more information on monitoring cache spinlock contention, see the *Performance and Tuning Guide*.

You can set the default number of cache partitions for all caches with the configuration parameter **global cache partition number**. See the *System Administration Guide*.

Dropping Caches

To drop or delete a data cache, change its size to 0. When you set a cache’s size to 0, the cache is marked for deletion. 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 delete a data cache, and there are objects bound to the cache, the cache is left as-is in memory and the SAP ASE server issues the following message:

```
Cache (nmc3) not deleted dynamically. Objects are bound to the
cache. Use
sp_unbindcache_all to unbind all objects bound to the cache.
```

The entry corresponding to the cache in the configuration file is deleted, as well as the entries corresponding to the cache in `sysconfigures`, and the cache is deleted the next time the SAP ASE server is restarted.

- You cannot run **sp_cacheconfig** within a transaction.

sp_cachestrategy

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. You can use the **mru** strategy in all caches, regardless of available I/O size. Setting **prefetch "on"** has no effect on tables or indexes that are read into a cache that allows only 2K I/O.
- **on | off –** specifies the setting, **"on"** or **"off"**, enclosed in quotes.

Examples

- **Example 1 –** Displays information about cache strategies for the `titles` table:

  ```
  sp_cachestrategy pubs2, titles
  ```

  ```
  object name       index name    large IO  MRU
  ----------------- ------------- --------  ---------
  dbo.titles        titleidind    ON         ON
  ```

  When you use **sp_cachestrategy** without specifying the strategy and setting, it reports the current settings for the object.

- **Example 2 –** Displays information about cache strategies for the `titleind` index:

  ```
  sp_cachestrategy pubs2, titles, titleind
  ```

- **Example 3 –** Disables prefetch on the `titleind` index of the `titles` table:

  ```
  sp_cachestrategy pubs2, titles, titleind, prefetch, "off"
  ```

- **Example 4 –** Reenables MRU replacement strategy on the `authors` table:

  ```
  sp_cachestrategy pubs2, authors, "table only", mru, "on"
  ```

- **Example 5 –** Reenables prefetching on the text pages of the `blurbs` table:

  ```
  sp_cachestrategy pubs2, blurbs, "text only", prefetch, "on"
  ```

Usage

- 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. For more information on cache strategies, see the *Performance and Tuning Guide*.
- You can change the cache strategy only for objects in the current database.
- To see the size, status and I/O size of all data caches on the server, use **sp_cacheconfig**.

See also **delete**, **select**, **set**, **update** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_cachestrategy** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the object owner or a user with `manage data cache` privilege.
Disabled	With granular permissions disabled, you must be the object owner or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_cacheconfig* on page 90
- *sp_poolconfig* on page 582

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 I/O the size of the logical page size for **prefetch**, and by specifying **lru** strategy. For example, the following command forces LRU strategy, logical page size 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 the logical page size in a **select**, **delete**, or **update** command overrides an earlier **set prefetch "on"** statement. Specifying a larger I/O size in a **select**, **delete**, or **update** command does not override a **set prefetch "off"** command.

sp_changedbowner

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

- **Example 1 –** Makes the user "albert" the owner of the current database:

  ```
  sp_changedbowner albert
  ```

Usage

There are additional considerations when using **sp_changedbowner**:

- 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 the SAP ASE 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.

See also **create database** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_changedbowner** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `own any database` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addlogin* on page 35
- *sp_dropalias* on page 252
- *sp_dropuser* on page 293
- *sp_helpdb* on page 394

sp_changegroup

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

- **Example 1 –** The user "albert" is now a member of the "fort_mudge" group. It doesn't matter what group "albert" belonged to before:

  ```
  sp_changegroup fort_mudge, albert
  ```

- **Example 2 –** Removes "albert" from the group he belonged to without making him a member of a new group (all users are always members of "public"):

  ```
  sp_changegroup "public", albert
  ```

Usage

There are additional considerations when using **sp_changegroup**:

- 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.

See also **grant**, **revoke** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_changegroup** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any user` privilege.
Disabled	With granular permissions disabled, you must be the database owner, a user with **sa_role**, or a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addgroup* on page 30
- *sp_adduser* on page 59
- *sp_dropgroup* on page 273
- *sp_helpgroup* on page 408

sp_checknames

Checks the current database for names that contain characters not in the 7-bit ASCII set.

Syntax

```
sp_checknames [help | silent]
```

Parameters

- **help –** shows information about the system tables that are scanned.
- **silent –** checks the current database in a silent mode, returning either:
  - 0 – if there are no names with non-7 bit ASCII characters, or
  - 1 – if there is at least one name witha non-7 bit ASCII character

Examples

- **Example 1 –** Checks the `master` database for names that contain characters not in the 7-bit ASCII set:

```
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
```

- **Example 2 –** Displays information about the system tables scanned:

```
1> sp_checknames help
2> go
```

```
sp_checknames is used to search for non 7-bit ASCII characters
several important columns of system tables. The following
columns are searched:

In "master":
    sysdatabases.name
    sysdevices.name
    syslogins.name
    syslogins.dbname
    syslogins.password
    sysremotelogins.remoteusername
    sysservers.srvname
    sysservers.srvnetname

In all databases:
    syscolumns.name
    sysindexes.name
    sysobjects.name
    syssegments.name
    systypes.name
    sysusers.name

(return status = 0)
1>
```

- **Example 3 –** Suppresses the output of system table names, and displays just the return status:

```
1> sp_checknames silent
2> go
```

```
(return status = 1)
```

Usage

There are additional considerations when using **sp_checknames**:

- **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.

See also **update** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_checknames**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_password* on page 559
- *sp_rename* on page 605
- *sp_renamedb* on page 609

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.

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

- **Example 1 –** Shows the results if **sp_checkreswords** is executed in the `master` database:

```
1> /* executed in the master database */
2> sp_checkreswords
```

```
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
```

- **Example 2 –** Shows the results if **sp_checkreswords** is executed in the user database `user_db`:

```
1> /* executed in the user database, user_db */
2> sp_checkreswords
```

```
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.
```

Usage

- **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 Server 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 SAP ASE. For information on installing and running this procedure before performing the upgrade, see the installation documentation for your platform.

  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.

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. For more information, see **sp_renamedb**.

See also:

- **set** in *Reference Manual: Commands*
- **defncopy** in the *Utility Guide*

Permissions

Any user can execute **sp_checkreswords**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_configure* on page 167
- *sp_depends* on page 217
- *sp_rename* on page 605
- *sp_renamedb* on page 609

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 can 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 SAP ASE features. For example, you cannot use **bcp** on tables with names that are reserved words.

Before choosing the quoted identifier option, perform a test on various objects using all the tools you use to access the SAP ASE 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 with names that have been changed may work after the name change, but stop working when the query plan is recompiled. Recompilation takes place for many reasons, without notification to the user. To avoid unsuspected loss of functionality, 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 Guide*.

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 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, 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.

The types of identifiers that you can change with **sp_rename** and the changes you need to make on the server and in your application programs are:

- 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.
- 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.

To change the name of the view `isolation` to `isolated`, use:

```
sp_rename "isolation", isolated
```

To change the name of a column in the renamed view `isolated`, use:

```
sp_rename "isolated.key", keyname
```

Use **sp_depends** to get a list of all views, procedures, and triggers that references a view, procedure, or table that is renamed. To use **sp_depends** after renaming an object, give the new name. For example:

```
sp_depends new_name
```

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, then add or create it again. If you cannot do that, use the following command to allow direct updates to system tables:

```
sp_configure "allow updates to system tables", 1
```

Only a system security officer can set the **allow updates to system tables** configuration parameter.

Errors during direct updates to system tables can create severe problems in the SAP ASE server. Determine whether you can drop the objects or user, then re-create them:

Identifier Type	Suggested Actions to Avoid Updates to System Tables
User names and log-in names	To change the name of a user with no objects: **1.** Use **sp_helprotect *username*** in each database to record the user’s permissions. **2.** Drop the user from all of the databases (**sp_dropuser**). **3.** Drop the login (**drop login**). **4.** Add the new login name (**create login**). **5.** Add the new user name to the databases (**sp_adduser**). **6.** Restore the user’s permissions with **grant**.
Device names	If this device is completely allocated, you need not use its name in a **create database** command, so you can leave the name unchanged.

Identifier Type	Suggested Actions to Avoid Updates to System Tables
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 log-ins	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.
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.

This table shows possible dependencies on this set of identifiers. See this table for possible dependencies, whether you choose to upgrade by dropping and re-creating objects, by using delimited identifiers, or by performing direct updates to system tables.

Warning! Direct updates to system tables can be very dangerous. You can make mistakes that make it impossible for the SAP ASE 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 in the following table.

Table 2. 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.

Identifier	Remember To
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.

This 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:

```
sp_configure "allow updates to system tables", 1
```

Then you can execute:

```
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**.
- 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.

The following table 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. 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.syschar-sets`	`name`
Language name	`master.dbo.syslan-guages` `master.dbo.syslogins`	`name` `language`

Using Delimited Identifiers

Consideration for 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, the SAP ASE 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**.

sp_checksource

Checks for the existence of the source text of compiled objects such as views, defaults, rules, triggers, procedures, declarative defaults, check constraints, computed columns, function-based indexes and predicates. The predicate name may be a user-defined or internal name.

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

- **Example 1 –** Checks for the existence of the source text of all compiled objects in the current database:

```
sp_checksource
```

- **Example 2 –** Checks for the existence of the source text of the view named `titleview`:

```
sp_checksource titleview
```

- **Example 3 –** Checks for the existence of the source text of the view named `titls_vu` that is owned by Mary:

  ```
  sp_checksource title_vu, @username = Mary
  ```
- **Example 4 –** Checks for the existence of the source text of the custom stored procedure `list_phone_proc`:

  ```
  sp_checksource list_phone_proc
  ```
- **Example 5 –** Checks for the existence of the source text of all the check constraints, triggers, and declarative defaults defined on the table named `my_tab`:

  ```
  sp_checksource @tabname = "my_tab"
  ```
- **Example 6 –** 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`:

  ```
  sp_checksource @objname = "my_vu", @tabname = "my_tab"
  ```
- **Example 7 –** Checks for the existence of the source text of all compiled objects owned by Tom:

  ```
  sp_checksource @username = "Tom"
  ```
- **Example 8 –** Checks for the existence of the source text for the “pred1” predicate:

  ```
  sp_checksource pred1
  ```

Usage

There are additional considerations when using **sp_checksource**:

- **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.
- **sp_checksource** encrypts the text of user-defined functions.

Permissions

The permission checks for **sp_checksource** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage database` privilege to check for the existence of the source text of compiled objects that are owned by another user. Any user can execute **sp_checksource** to check for the existence of the source text for his or her own compiled objects.
Disabled	With granular permissions disabled, you must be the database owner or a user with sa_role to check for the existence of the source text of compiled objects that are owned by another user. Any user can execute **sp_checksource** to check for the existence of the source text for his or her own compiled objects.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_hidetext* on page 449

sp_chgattribute

Changes the **max_rows_per_page**, **fillfactor**, **reservepagegap**, or **exp_row_size** value for future space allocations of a table or an index; sets the **concurrency_opt_threshold** for a table. Provides the user interface for optimistic index locking.

Syntax

```
sp_chgattribute objname,
    {"max_rows_per_page" | "fillfactor" | "reservepagegap" |
    "exp_row_size" | "concurrency_opt_threshold" |
    "optimistic_index_lock" | "identity_burn_max" | "plldegree"
    "ptn_locking"}, value,  optvalue
    {"identity_gap", set_number |
    "dealloc_first_txtpg", 0 | 1 | 2}
```

Parameters

- ***objname*** **–** is the name of the table or index for which you want to change attributes.
- **max_rows_per_page –** specifies the row size. Use this for tables with variable-length columns.
- **fillfactor –** specifies how full the SAP ASE server makes each page when it is re-creating an index or copying table pages as a result of a **reorg rebuild** command or an **alter table** command to change the locking scheme. The **fillfactor** percentage is relevant only at the time the index is rebuilt. Valid values are 0–100.
- **reservepagegap –** specifies the ratio of filled pages to empty pages that are to be left during extent I/O allocation operations. For each specified *num_pages*, an empty page is left for future expansion of the table. Valid values are 0–255. The default value is 0.
- **exp_row_size –** reserves a specified amount of space for the rows in data-only locked tables. Use this option to reduce the number of rows being forwarded, which can be expensive during updates. Valid values are 0, 1, and any value between the minimum and maximum row length for the table. 0 means a server-wide setting is applied, and 1 means to fully pack the rows on the data pages.
- **concurrency_opt_threshold –** specifies the table size, in pages, at which access to a data-only-locked table should begin optimizing for reducing I/O, rather than for concurrency. If the table is smaller than the number of pages specified by **concurrency_opt_threshold**, the query is optimized for concurrency by always using available indexes; if the table is larger than the number of pages specified by **concurrency_opt_threshold**, the query is optimized for I/O instead. Valid values are -1 to 32767. Setting the value to 0 disables concurrency optimization. Use -1 to enforce concurrency optimization for tables larger than 32767 pages. The default is 15 pages.
- **optimistic_index_lock –** enables a performance optimization that eliminates contention on the root page of an index. If the root page must change because of index splits, an exclusive table is acquired. For this reason, **optimistic_index_lock** is appropriate for tables where the number of modifications is relatively small. Valid values are 1 to turn on optimistic index locking or 0 to turn off optimistic index locking which is the default.
- **identity_burn_max –** allows you to reset the internal counter for the identity column. The value set represents the highest value already generated; the next automatically generated value is one larger than the value you specify. The value is passed as a varchar datatype in the fourth parameter.
- **identity_gap –** indicates that you want to change the identity gap.
- ***value*** **–** is the numeric input value for the various options you specify in the **sp_chgattribute**.

- ***optvalue* –** is the new value. Valid values and default values depend on which parameter is specified. This parameter is only used by the **identity_burn_max** parameter. For other parameters, this value is NULL.
- ***set_number* –** is the new size of the identity gap.
- **dealloc_first_txtpg –** updates a text or image column to null. Sets the corresponding text pointer to null after deallocating the previously referenced `text` or `image` pages. This result in reduced space allocation for null `text/images` columns. Valid values are:

  - 0 – default, existing value, if either the table option setting is 1, or the database option **deallocate first text page** is TRUE, then deallocate the first text page after NULL update; otherwise, do not deallocate the first text page.
  - 1 – deallocate the first text page after NULL update (overriding the setting of the database option **deallocate first text page**).
  - 2 – do not deallocate the first text page after NULL update (overriding the setting of the database option **deallocate first text page**).

  Whether the first text page will be deallocated after NULL update depends on the combination of this table parameter and the database option **deallocate first text page**.

```
DB setting (deallocate first text page)   |   0   1   2
---------------------------------------------------------
dealloc_first_txtp - true                 |   Y   Y   N
dealloc_first_txtp - false                |   N   N   N
```

  - Y: deallocate first text page after null update
  - N: not deallocate first text page after null update

  The output from **sp_help** indicates whether first text page will be deallocated.
- **plldegree –** specifies the maximum number of threads the query optimizer can use.
- **ptn_locking –** specifies whether to enable (1) or disable (0) locking at the partition level. By default, partition locking is disabled.

Examples

- **Example 1 –** Sets the **max_rows_per_page** to 1 for the `authors` table for all future space allocations:

```
sp_chgattribute authors, "max_rows_per_page", 1
```

- **Example 2 –** Sets the **max_rows_per_page** to 4 for the `titleidind` index for all future space allocations:

```
sp_chgattribute "titles.titleidind", "max_rows_per_page", 4
```

- **Example 3 –** Specifies a **fillfactor** of 90 percent for pages in `title_ix`:

```
sp_chgattribute "titles.title_ix", "fillfactor", 90
```

- **Example 4 –** Sets the **exp_row_size** to 120 for the `authors` table for all future space allocations:

```
sp_chgattribute "authors", "exp_row_size", 120
```

- **Example 5 –** Sets the **reservepagegap** to 16 for the `titleidind` index for all future space allocations:

```
sp_chgattribute "titles.titleidind", "reservepagegap", 16
```

- **Example 6 –** Turns off concurrency optimization for the `titles` table:

```
sp_chgattribute "titles", "concurrency_opt_threshold", 0
```

- **Example 7 –** Sets the identity gap for `mytable` to 20:

```
sp_chgattribute "mytable", "identity_gap", 20
```

- **Example 8 –** Changes `mytable` to use the identity burning set factor setting instead of the **identity_gap** setting:

```
sp_chgattribute "mytable", "identity_gap", 0
```

- **Example 9 –** Sets the value of **sp_chgattribute** to 1, turning the optimistic index locking feature on.

```
sp_chgattribute "mytable", "optimistic_index_lock", 1
```

- **Example 10 –** Sets the value of **sp_chgattribute** to 0, turning the optimistic index locking feature off.

```
sp_chgattribute "mytable", "optimistic_index_lock", 0
```

- **Example 11 –** Switches the deallocation for text and image space on using **dealloc_first_txtpg**:

```
sp_chgattribute "mytable", "dealloc_first_txtpg", 1
```

To switch the feature off:

```
sp_chgattribute "mytable", "dealloc_first_txtpg", 0
```

- **Example 12 –** The output from **sp_help** indicates whether the first text page will be deallocated:

```
> sp_chgattribute  mytab, "dealloc_first_txtpg", 1
'dealloc_first_txtpg' attribute of object 'mytab' changed to 1.
(return status = 0)
1>
2> sp_help mytab
 Name Owner Object_type     Object_status            Create_date
----- ----- ----------- ------------------------- -------------------
mytab  dbo  user table  deallocate first text page  Jan 22 2013 9:45PM

> sp_chgattribute  mytab, "dealloc_first_txtpg", 2
'dealloc_first_txtpg' attribute of object 'mytab' changed to 2.
(return status = 0)
1>
2> sp_help mytab
  Name Owner Object_type     Object_status         Create_date
----- ----- ------------- -------------------- --------------------
mytab  dbo   user table   keep first text page  Jan 22 2013  9:45PM
```

- **Example 13 –** Changes the **identity_burn_max** value for the `authors` table to 5:

```
sp_chgattribute "authors", "identity_burn_max", 0, "5"
```

- **Example 14 –** Tells the query optimizer to use a maximum of four threads:

  ```
  sp_chgattribute my_table, "plldegree", 4
  ```

  The query optimizer may choose less than four threads if it does not find enough resources. The same mechanism can be applied to an index. For example, the following example uses an index called `auth_ind` exists on `authors` to use two threads to access it:

  ```
  sp_chgattribute "authors.auth_ind", "plldegree", 4
  ```

  You must run **sp_chgatttribute** from the current database.
- **Example 15 –** Enables partition-level locking for the `authors` table:

  ```
  sp_chgattribute authors, "ptn_locking", 1
  ```

  To disable partition-level locking:

  ```
  sp_chgattribute authors, "ptn_locking", 0
  ```

Usage

There are additional considerations when using **sp_chgattribute**:

- You cannot change attributes for virtually hashed tables. For example, if you attempt to change the attributes for table `order_line` (a virtually-hashed table) like this:

  ```
  sp_chgattribute 'order_line', 'exp_row_size', 1
  ```

  The SAP ASE server issues an error message similar to:

  ```
  sp_chgattribute is not allowed for order_line, as it is a
  virtually hashed table.
  ```
- (Cluster Edition only) You cannot use **sp_chgattribute** to change the value of *identity_gap* at runtime.
- **sp_chgattribute** changes the **max_rows_per_page**, **fillfactor**, **reservepagegap**, **exp_row_size**, or **dealloc_first_txtpg** value for future space allocations or data modifications of the table or index. It does not affect the space allocations of existing data pages. You can change these values for an object only in the current database.
- Use **sp_help** to see the stored space management values for a table. Use **sp_helpindex** to see the stored space management values for an index.
- Setting **max_rows_per_page** to 0 tells the SAP ASE server to fill the data or index pages and not to limit the number of rows—this is the default behavior of the SAP ASE server if you do not set **max_rows_per_page**.
- Both the **identity_burn_max** value stored in `sysobjects` and the current identity value are set to the new value.
- If the table is:
  - Not empty – the new value of **identity_burn_max** must be greater than or equal to the current maximum value of the `identity` column.
  - Empty – you can set the value to any positive value in the valid range.

- Low values of **max_rows_per** page 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 an incorrect value for **max_rows_per_page**, **fillfactor**, **reservepagegap**, or **exp_row_size**, **sp_chgattribute** returns an error message specifying the valid values.
- You cannot run this stored procedure from within a transaction.
- Only a user with **sa_role** privileges can execute this stored procedure.
- You cannot set the optimistic index locking option for tables with datapages or datarow locking schemes.
- You cannot set the optimistic index locking option for tables in system databases, such as `master` or `tempdb`. You can set it only on user-defined tables.
- `text` and `image` pages are allocated space even when you perform a NULL update. You can use **dealloc_first_txtpg** to remove these empty text pages from the table.

  A new update to the column results in reallocation of a `text` or `image` page.

See also:

- **alter table**, **create index**, **create table** in *Reference Manual: Commands*
- For more information on **max_rows_per_page**, **fillfactor**, **reservepagegap**, **exp_row_size**, and **concurrency_opt_threshold**, see the *Performance and Tuning Guide*.

  For more information about identity gaps, see the section *Managing Identity Gaps in Tables* in the *Transact-SQL User's Guide*.

Permissions

The permission checks for **sp_chgattribute** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the object owner.
Disabled	With granular permissions disabled, you must be the object owner.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_helpindex* on page 409

sp_cleanpwdchecks

sp_cleanpwdchecks is a custom stored procedure that allows you to define when and how to remove login and password-related attributes stored in user-defined tables.

Syntax

```
sp_cleanpwdchecks, login_name
```

Parameters

- ***login_name* –** specifies the login name of the cleanup to be performed.

Usage

sp_cleanpwdchecks is user-defined, and is dynamically called in the `master` database when you drop a login.

sp_clearpsexe

Clears the execution attributes of an SAP ASE 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

- **Example 1 –** Drops the engine group entry for process 12.

  ```
  sp_clearpsexe 12, 'enginegroup'
  ```

Usage

There are additional considerations when using **sp_clearpsexe**:

- 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 (`spids`).

See also *Performance and Tuning Guide*.

Permissions

The permission checks for **sp_clearpsexe** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any execution class` privilege. Any user can execute **sp_clearpsexe** to clear the priority attributes of tasks owned by that user.

Setting	Description
Disabled	With granular permissions disabled, you must be a user with sa_role. Any user can execute **sp_clearpsexe** to clear the priority attributes of tasks owned by that user.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addexeclass* on page 23
- *sp_bindexeclass* on page 83
- *sp_dropexeclass* on page 266
- *sp_showexeclass* on page 652
- *sp_unbindexeclass* on page 716

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**.

Syntax

```
sp_clearstats [loginame]
```

Parameters

- ***loginame* –** is the user’s login name.

Examples

- **Example 1 –** Initiates a new accounting period for all users.

```
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.
```

- **Example 2 –** Initiates a new accounting period for the user “kathy.”

```
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.
```

Usage

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`.

Permissions

The permission checks for **sp_clearstats** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_reportstats* on page 612

sp_client_addr

Displays the IP (Internet Protocol) address of every SAP ASE task with an attached client application, including the `spid` and the client host name.

Syntax

```
sp_client addr [spid]
```

Parameters

- ***spid* –** specifies one task for which you require an IP address.

Examples

- **Example 1 –** Lists IP addresses for all tasks:

```
sp_client_addr
```

```
---------
spid   hostname  ipaddr
-----------------------------
11     FRED      162.66.131.36
21     BARNEY    162.66.100.233
22     WILMA     162.66.100.206
23     BETTY     162.66.100.119
24     PEBBLES   162.66.100.125
25     BAMBAM    162.66.100.124
(6 rows affected)
(return status = 0)
```

- **Example 2 –** Shows IP addresses for `spid` 21:

```
sp_client_addr 21
```

```
----------
spid   hostname  ipaddr
-----------------------------
21     BARNEY    162.66.100.233
(1 row affected)
(return status = 0)
```

- **Example 3 –** Shows the result when a client application is not connected via IP:

```
sp_client_addr 11
```

```
----------
spid   hostname  ipaddr
-----------------------------
11     FRED      0.0.0.0
(1 row affected)
(return status = 0)
```

- **Example 4 –** Shows the result of a task with no attached client; for example, Housekeeper:

```
sp_client_addr 9
```

```
----------
spid   hostname  ipaddr
-------------------------------
9                NULL
(1 row affected)
(return status = 0)
```

- **Example 5 –** Shows the result when an incorrect `spid` is specified:

```
sp_client_addr 99
```

```
----------
Msg 18934, Level 16, State 1:
Procedure "sp_client_addr", Line 32:
spid not found
(return status = 1)
```

Usage

If the client application is not attached by IP, the address appears as 0.0.0.0. The SAP ASE server does not support display of addresses of protocols other than IP.

If a task has no attached client (Housekeeper, for instance), the IP address appears as "NULL". Tasks with no attached client are not listed when you use **sp_client_addr** with no parameter.

Permissions

Any user can execute **sp_client_addr**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_who* on page 736

sp_cluster

(Cluster environments only) Performs a number of procedures related to clusters.

Syntax

Migrates a connection to a different logical cluster or instance:

```
sp_cluster connection, migrate, lc_name, instance_name, "spid_list"
```

Determines if previous connection migrations to a new instance are pending, and terminates the migrations if they are:

```
sp_cluster connection, [‘migrate_status’ | 'migrate_cancel' ][,
‘spid_list’]
```

Modifies an outstanding action, such as canceling the action or changing the timing of the action:

```
sp_cluster logical, "action", lc_name, {
    cancel, action_handle |
    modify_time, action_handle, wait_option[, timeout ] |
    release, action_handle }
```

Adds a resource or one or more routes to the logical cluster:

```
sp_cluster logical, "add", lc_name, {
     route, route_type, key_list |
       instance, instance_list |      failover, instance_list }
```

Moves a route from one logical cluster to another:

```
sp_cluster logical, “alter”, lc_name, route, route_type, key_list
```

Creates a new logical cluster:

```
sp_cluster logical, "create", lc_name
```

Stops the logical cluster on one or more instances or the entire logical cluster, and places the instances or the cluster in the inactive state:

```
sp_cluster logical, "deactivate", lc_name, {
    "cluster" |     "instance", instance_list }
   [, wait_option[, timeout,[, @handle output ]]]
```

Drops a logical cluster, or one or more resources from the logical cluster:

```
sp_cluster logical, "drop",  lc_name,
    {cluster | instance, instance_list |
     failover, instance_list | route, route_type, key_list }
```

Reverses a manual failover, reinstating the original base instances:

```
sp_cluster logical, “failback”, lc_name, {
    cluster[, wait_option[, timeout[, @handle output ]]] |
    instance, from_instance_list, to_instance_list[, wait_option[,
        timeout[, @handle output ]]] }
```

Initiates a manual failover from base instances to failover instances.

```
sp_cluster logical, “failover”, lc_name, {cluster
   [, to_instance_list[, wait_option[, timeout[, @handle output ]]]
    | instance, from_instance_list, to_instance_list[, wait_option[,
        timeout[, @handle output ]]] }
```

Manually gathers and migrates a group of connections to a different logical cluster:

```
sp_cluster logical, 'gather', lc_name
```

Displays complete syntax for **sp_cluster logical**:

```
sp_cluster logical, “help”
```

Stops the logical cluster on one or more instances or the entire logical cluster:

```
sp_cluster logical, "offline", lc_name,
    {cluster | instance, instance_list }
    [, wait_option[, timeout,[, @handle output ]]]
```

Starts the default logical cluster on one or more instances:

```
sp_cluster logical, "online", { lc_name[, instance_list]}
```

Sets logical cluster rules: the open logical cluster, the failover mode, the system view, the start-up mode, and the load profile:

```
sp_cluster logical, "set", lc_name, { open
        | failover, failover_mode | system_view, view_mode
        | startup, { automatic | manual } | load_profile,
profile_name }
    login_distribution, { affinity | "round-robin" }
```

Displays information about a logical cluster:

```
sp_cluster logical, "show"
[, lc_name[, {action[, state] | route[, type[, key]]}]]
```

Lets you set up and manage the load profile for the logical cluster:

```
sp_cluster profile, [ "show" [, profile_name ]
    | "create", profile_name | "drop", profile_name
    | "set", profile_name [, weight [, wt_metric [, wt_value ]
    | threshold [, thr_metric [, thr_value ] ] ]
```

Lets you set up and manage the load profile for the logical cluster:

```
sp_cluster profile, [ "show" [, profile_name ] | "create",
profile_name | "drop", profile_name | "set", profile_name [, weight
[, wt_metric [, wt_value ] | threshold [, thr_metric [,
thr_value ] ] ]
```

Parameters

- **sp_cluster connection, migrate, *lc_name*, *instance_name*, "*spid_list*" –** where:
  - *lc_name* – is the name of the logical cluster.
  - *instance_name* – is the name of the instance.
  - *spid_list* – is the list of spids you are migrating. Separate multiple spids with semicolons.
- **sp_cluster connection, ['migrate_status' | 'migrate_cancel'][, '*spid_list*'] –** where:
  - *spid_list* – is the list of spids you are investigating.
  - **migrate_cancel** – indicates you are terminating the connection migrations.
  - **migrate_status** – indicates you are investigating the status of connection migrations.
- **sp_cluster logical, "action", *lc_name*, {cancel, *action_handle* | modify_time, *action_handle*, *wait_option*[, *timeout*] |release, *action_handle*} –** where:
  - **cancel** – specifies an action to be canceled.
  - *action_handle* – is the action identifier.

    - **modify_time** – specifies that the time of the action is to be modified.
    - *wait_option* – is how the time of the action is to be modified. Values are:
        - **wait** – indicates that existing connections are given a specified amount of time (or an infinite amount of time if no *timeout* is given) to migrate or disconnect.
        - **nowait** – indicates that existing connections are migrated or disconnected immediately.
        - **until** – indicates that existing connections are given until a specific time of day to migrate or disconnect.
    - *timeout* – is a specific amount of time (when used with **wait**) or a specific time (when used with **until**). The format is “hh:mm:ss” according to a 24-hour clock. For example, *timeout* records “11:30 p.m” (or “11:30pm”) as “23:30:00”.
    - **release** – specifies that all resources held by a completed action are to be released.
- **sp_cluster logical, "add", *lc_name*, { route, *route_type*, *key_list* | instance, *instance_list* | failover, *instance_list*} –** where:
    - *lc_name* – is the name of a logical cluster.
    - **route** – specifies that one or more routes are to be added to the logical cluster
    - *route_type* – is the type of route to be added. Values are:
        - **application** – specifies a route for an application name to the logical cluster.
        - **login** – specifies a route for a login name to the logical cluster.
        - **alias** – specifies a route for a server name alias to the logical cluster.
    - *key_list* – is a list of applications, logins, or aliases, depending on the route type. Elements in the key list are delimited by semicolons.
    - **instance** – specifies that one or more base instances are to be added to the logical cluster.
    - *instance_list* – is the list of instances to be added. Separate multiple instances with semicolons.
    - **failover** – specifies that one or more failover instances are to be added to the logical cluster.
- **sp_cluster logical, “alter”, *lc_name*, route, *route_type*, *key_list* –** where:
    - *lc_name* – is the name of a logical cluster.
    - **route** – specifies a route is to be altered.
    - *route_type* – is the type of route to be altered. Values are:
        - **application** – specifies a route for an application name to the logical cluster.
        - **login** – specifies a route for a login name to the logical cluster.
        - **alias** – specifies a route for a server name alias to the logical cluster.
    - *key_list* – is a list of applications, logins, or aliases, depending on the route type. Elements in a key list are delimited with semicolons.
- **sp_cluster logical, "create", *lc_name* –** where:
    - *lc_name* – is name of the logical cluster.

- **sp_cluster logical, "deactivate",** ***lc_name*****, { "cluster" |"instance",** ***instance_list*** **} [,** ***wait_option*****[,** ***timeout*****,[, @handle output]]] –**
  - *lc_name* – name of a logical cluster.
  - **cluster** – specifies the entire cluster.
  - **instance** – specifies that only certain instances in the logical cluster are to be placed in the inactive state.
  - *instance_list* – list of selected instances in the logical cluster.
  - *wait_option* – the valid options are:
    - **wait** – indicates that existing connections are given a specified amount of time (or an infinite amount of time if no *timeout* is given) to migrate or disconnect.
    - **nowait** – indicates that existing connections are migrated or disconnected immediately.
    - **until** – indicates that existing connections are given until a specific time of day to migrate or disconnect.
  - *timeout* – a specific amount of time (when used with **wait**) or a specific time (when used with **until**). The format is “hh:mm:ss” according to a 24-hour clock. For example, *timeout* records 11:30 p.m. as 23:30:00.
  - **@handle output** – specifies that an action handle is to be retrieved for the action.
- **sp_cluster logical, "drop",** ***lc_name*****, { cluster | instance,** ***instance_list*** **| failover,** ***instance_list*** **| route,** ***route_type*****,** ***key_list*** **} –** where:
  - *lc_name* – name of a logical cluster.
  - **cluster** – specifies the entire cluster.
  - **instance** – specifies that only certain instances in the logical cluster are to be placed in the inactive state.
  - *instance_list* – list of selected instances in the logical cluster.
  - *wait_option* – where the valid options are:
    - **wait** – indicates that existing connections are given a specified amount of time (or an infinite amount of time if no *timeout* is given) to migrate or disconnect.
    - **nowait** – indicates that existing connections are migrated or disconnected immediately.
    - **until** – indicates that existing connections are given until a specific time of day to migrate or disconnect.
  - *timeout* – a specific amount of time (when used with **wait**) or a specific time (when used with **until**). The format is “hh:mm:ss” according to a 24-hour clock. For example, *timeout* records 11:30 p.m. as 23:30:00.
  - **@handle output** – specifies that an action handle is to be retrieved for the action.
- **sp_cluster logical, “failback”,** ***lc_name*****, { cluster[,** ***wait_option*****[,** ***timeout*****[, @handle output]]] | instance,** ***from_instance_list*****,** ***to_instance_list*****[,** ***wait_option*****[,** ***timeout*****[, @handle output]]] } –** where:
  - *lc_name* – name of a logical cluster.

    - **cluster** – specifies the entire cluster.
    - *to_instance_list* – list of predefined failover instances. A value of NULL activates the first failover group.
    - *from_instance_list* – list of instances that are to be taken offline.
    - *wait_option* – where the valid options are:
        - **wait** – indicates that existing connections are given a specified amount of time (or an infinite amount of time if no *timeout* is given) to migrate or disconnect.
        - **nowait** – indicates that existing connections are migrated or disconnected immediately.
        - **until** – indicates that existing connections are given until a specific time of day to migrate or disconnect.
    - *timeout* – a specific amount of time (when used with **wait**) or a specific time (when used with **until**). The format is "hh:mm:ss" according to a 24-hour clock. For example, *timeout* records 11:30 p.m. as 23:30:00.
    - **@handle output** – specifies that an action handle is to be retrieved for the action.
- **sp_cluster logical, "failover", *lc_name*, {cluster[, *to_instance_list[, wait_option*[, *timeout*[, @handle output]]] | instance, *from_instance_list, to_instance_list*[, *wait_option*[,*timeout*[, @handle output]]] } –** where:
    - *lc_name* – name of a logical cluster.
    - **cluster** – specifies the failover of the entire logical cluster.
    - *to_instance_list* – list of predefined failover instances. A value of NULL activates the first failover group.
    - *wait_option* – how the time of the action is to be recorded. Values are:
        - **wait** – indicates that existing connections are given a specified amount of time (or an infinite amount of time if no *timeout* is given) to migrate or disconnect.
        - **nowait** – indicates that existing connections are migrated or disconnected immediately.
        - **until** – indicates that existing connections are given until a specific time of day to migrate or disconnect.
    - *timeout* – is a specific amount of time (when used with **wait**) or a specific time (when used with **until**). The format is "hh:mm:ss" according to a 24-hour clock. For example, *timeout* records 11:30 pm as 23:30:00.
    - **@handle output** – specifies that an action handle is to be retrieved for the failover.
    - **instance** – specifies that only selected instances in the logical cluster are to fail over.
    - *from_instance_list* – list of instances that are to be taken offline
- **sp_cluster logical, 'gather', *lc_name* –** where:
    - **gather** – indicates you are gathering a set of qualified connections to migrate them to another logical cluster.
    - *lc_name* – name of a logical cluster to which you are migrating the connections.
- **sp_cluster logical, "offline", *lc_name*, { cluster | instance, *instance_list* } [, *wait_option*[, *timeout*,[, @handle output]]] –** where:

    - *lc_name* – name of a logical cluster.
    - **cluster** – specifies the entire cluster.
    - **instance** – specifies that only selected instances in the logical cluster are to taken offline.
    - *instance_list* – list of selected instances in the logical cluster.
    - *wait_option* – how the time of the action is to be specified. Values are:
        - **wait** – indicates that existing connections are given a specified amount of time (or an infinite amount of time if no *timeout* is given) to migrate or disconnect.
        - **nowait** – indicates that existing connections are migrated or disconnected immediately.
        - **until** – indicates that existing connections are given until a specific time of day to migrate or disconnect.
    - *timeout* – is a specific amount of time (when used with **wait**) or a specific time (when used with **until**). The format is "hh:mm:ss" according to a 24-hour clock. For example, *timeout* records 11:30 pm as 23:30:00.
    - **@handle output** – specifies that an action handle is to be retrieved for the action.
    - *from_instance_list* – list of instances that are to be taken offline
- **sp_cluster logical, "online", { *lc_name*[, *instance_list*]} –** where:
    - *lc_name* – name of a logical cluster.
    - *instance_list* – list of selected instances in the logical cluster.
- **sp_cluster logical, "set", *lc_name*, {open | failover, *failover_mode* | system_view, *view_mode* | startup, { automatic | manual } | load_profile, *profile_name* | action_release, { automatic | manual } | gather, { automatic | manual } | login_distribution, { affinity | "round-robin" } –** where:
    - *lc_name* – name of a logical cluster.
    - **open** – sets the open logical cluster. Unrouted connections are sent to the open logical cluster.
    - **failover *failover_mode*** – sets the failover mode of the logical cluster. Values for *failover_mode* are:
        - **instance** – specifies a 1:1 failover strategy; every time a base instance fails, a failover resource is brought online.
        - **group** – specifies that failover resources are brought online only after all base instances in the cluster fail.
    - **system_view *view_mode*** – sets the default system view for tasks running in the logical cluster. Values for *view_mode* are:
        - **instance** – specifies that monitoring and informational tools such as **sp_who**, **sp_lock**, and monitoring tables describe an instance.
        - **cluster** – specifies that monitoring and informational tools such as **sp_who**, **sp_lock**, and monitoring tables describe the whole cluster.
    - **startup { automatic | manual}** – sets the start-up mode of the logical cluster.

  - **automatic** – specifies that the logical cluster is started automatically when the cluster starts.
  - **manual** – specifies that the logical cluster must be started manually.
- **login_distribution** – specifies how the Cluster Edition distributes connections when a logical cluster spans multiple instances.
- **action_release** – enables or disables the automatic releasing and clearing of these logical cluster actions—**online**, **offline**, **failover**, and **failback**—after they are completed or cancelled.
  - **automatic** – specifies that logical cluster actions are cleared automatically.
  - **manual** – specifies that logical cluster actions are not cleared after they are completed or cancelled. This is the default.
- **gather** – enables or disables the movement of groups of connections to a different logical cluster when one of these predefined actions occurs—**online**, **add route**, **alter route**, or **drop route**.
  - **automatic** – specifies that the connections are moved automatically.
  - **manual** – specifies that the connections are not moved automatically. This is the default.
- **@handle output** – specifies that an action handle is to be retrieved for the action.
- *from_instance_list* – list of instances that are to be taken offline

- **sp_cluster logical, "show"[, *lc_name*[, {*action*[, *state*] | route[, *type*[, *key*]]}]] –** where:
  - *lc_name* – name of the logical cluster. If NULL is entered, summary information for all logical clusters is displayed.
  - **action** – specifies information about administrative actions: **failover**, **failback**, **online**, **offline**, **deactivate**.
  - *state* – one of: **cancelled**, **complete**, or **active**.
  - **route** – specifies information about routes.
  - *type* – is one of: **application**, **alias**, or **login**.
  - *key* – a specific login, alias, or application name.
- **sp_cluster profile, ["show" [, *profile_name*] | "create", *profile_name* | "drop", *profile_name* | "set", *profile_name* [, weight [, *wt_metric* [, *wt_value*] | threshold [, *thr_metric* [, *thr_value*]]] –**
  - **show** – displays configured load profiles and their settings.
  - *profile_name* – name of a load profile.
  - **creates** – creates a new load profile.
  - **drop** – drops a load profile.
  - **set** – specifies attributes of a load profile. You must set each attribute individually.
  - **weight** – specifies a weight attribute.
  - *wt_metric* – specifies an individual weight metric. Values are:
    - **user connections** – the capacity of an instance to accept a new connection, based on resource availability.

  - **cpu utilization** – the capacity of an instance to accept a new connection, based on resource availability.
  - **run queue** – the capacity of an instance to accept a new connection, based on resource availability.
  - **io load** – outstanding asynchronous I/Os.
  - **engine deficit** – the difference in the number of online engines among instances in the cluster.

    Note: engine deficit is measurable only when instances in the cluster have unequal numbers of engines. **engine deficit** adds a metric for maximum relative capacity to the load score.
  - **user metric** – an optional, user-supplied metric.
- *wt_value* – specifies a weight value. Valid values are 0 to 255. A weight of zero (0) excludes the metric from calculation.
- **threshold** – specifies a threshold attribute.
- *thr_metric* – specifies a particular threshold attribute. Values are:
  - **dynamic** – specifies a threshold for dynamic load distribution.
  - **login** – specifies a threshold for login redirection
  - **hysteresis** – specifies a minimum load score for any connection redirection.
- *thr_value* – depends on value of *thr_metric*. When *thr_metric* is:
  - **dynamic** or **login** – *thr_value* is the percentage difference between the load scores of two instances. Valid values are 0 to 100. A weight of zero (0) disables this form of load distribution.
  - **hysteresis** – *thr_value* is the minimum load score for the target instance that must be met before dynamic load distribution or login redirection can occur.

Examples

- **Example 1 –** Moves the connection with a spid of 73 into the `SalesLC` cluster:

```
sp_cluster connection, migrate, SalesLC, NULL, '73'
```

- **Example 2 –** Moves the current connection to the “ase3” instance:

```
sp_cluster connection, migrate, NULL, ase3
```

- **Example 3 –** Moves connections with spid values of 73 and 75 into “ase3” instance and the `SalesLC` cluster:

```
sp_cluster connection, migrate, SalesLC, ase3, '73;75'
```

- **Example 4 –** Determines if there is a connection migration occurring on spid 73; if there is, the Cluster Edition cancels the migration:

```
sp_cluster connection, 'migrate_cancel', '73'
```

- **Example 5 –** Checks the status of migrated connections for connections with a spid value of 73:

```
sp_cluster connection, 'migrate_status', '73'
```

```
SPID LogicalCluster Instance MigrationLogicalCluster MigrationIns
tance Command
---- -------------- -------- ----------------------- ------------
------ -------
  73 SystemLC        ase1      SalesLC                 ase3
connection migrate
```

- **Example 6 –** Cancels a timed action on the “SalesLC” logical cluster. The action handle is 4390.

  ```
  sp_cluster logical, "action", SalesLC, cancel, "4390"
  ```

- **Example 7 –** Changes the wait option for existing action 5364 to **nowait**.

  ```
  sp_cluster logical, "action", SalesLC, modify_time, "5364", nowait
  ```

- **Example 8 –** Releases action 3456 for the “SalesLC” logical cluster.

  ```
  sp_cluster logical, "action", SalesLC, release, "3456"
  ```

- **Example 9 –** Releases all completed or cancelled actions for the “SalesLC” logical cluster.

  ```
  sp_cluster logical, "action", SalesLC, release, "all"
  ```

- **Example 10 –** Adds instances “ase1” and “ase2” to the “SalesLC” logical cluster.

  ```
  sp_cluster logical, "add", SalesLC, instance, "ase1;ase2"
  ```

- **Example 11 –** Creates one failover group with “ase3” for “SalesLC”.

  ```
  sp_cluster logical, "add", SalesLC, failover, ase3
  ```

- **Example 12 –** Routes the logins “tom”, “dick”, and “harry” to the “SalesLC” logical cluster

  ```
  sp_cluster logical, "add", SalesLC, route, login, "tom;dick;harry"
  ```

- **Example 13 –** Routes the field_sales application to the “SalesLC” logical cluster.

  ```
  sp_cluster logical, "add", SalesLC, route, application,
  field_sales
  ```

- **Example 14 –** Creates a route of type alias to logical cluster “lc1” with the alias “SalesLC”. Then, changes the logical cluster association of the route from “lc1” to “lc2”. The route is identified by its route type (alias) and its key (SalesLC).

  ```
  sp_cluster logical, "add", "lc1", "route", "alias", "SalesLC"
  sp_cluster logical, "alter", "lc2", "route", "alias", "SalesLC"
  ```

- **Example 15 –** Creates a logical cluster named “SalesLC”:

  ```
  sp_cluster logical, "create", SalesLC
  ```

- **Example 16 –** Immediately stops all instances in the “SalesLC” logical cluster, and places “SalesLC” in the inactive state:

  ```
  sp_cluster logical, "deactivate", SalesLC, cluster, nowait
  ```

- **Example 17 –** Stops the “ase1” and “ase2” instances, and places “SalesLC” in the inactive state:

  ```
  sp_cluster logical, "deactivate", SalesLC, instance, "ase1;ase2"
  ```

- **Example 18 –** Drops the “SalesLC” logical cluster:

  ```
  sp_cluster logical, "drop", SalesLC, cluster
  ```

- **Example 19 –** Drops the base instances “ase1” and “ase2” from the “SalesLC” logical cluster.

  ```
  sp_cluster logical, "drop", SalesLC, instance, "ase1;ase2"
  ```

- **Example 20 –** Drops the routes from the applications field_sales and web_sales from the “SalesLC” logical cluster.

  ```
  sp_cluster logical "drop", SalesLC, route, application,
      "field_sales;web_sales"
  ```

- **Example 21 –** Fails back the “SalesLC” logical cluster:

  ```
  sp_cluster logical, "failback", SalesLC, cluster
  ```

- **Example 22 –** “SalesLC” is running on “ase3” and “ase1”. In this example, “ase3” fails back to “ase1”, and “SalesLC” continues to run on “ase2”. The action takes place in two minutes:

  ```
  declare @out_handle varchar(15)

  execute
  sp_cluster logical, "failback", SalesLC, instance,
  ase3, ase1, wait, "00:02:00", @handle = @out_handle
  output
  ```

- **Example 23 –** Fails over the “SalesLC” logical cluster to the first group of predefined failover resources. The failover waits 2 minutes before terminating connections.

  ```
  declare @out_handle varchar(15)

  execute
  sp_cluster logical, "failover", SalesLC, cluster, NULL, wait,
  "00:02:00",
  @handle = @out_handle output
  ```

  ```
  Action '2' has been issued for the 'failover cluster'
  command.Logical Cluster
  Handle      Action                           From     To
           State           InstancesWaiting ConnectionsRemaining
  WaitType
           StartTime                 Deadline         CompleteTime
   --------------- ----------- -------------------------------
  -------- --
           ---------------- ---------------- --------------------
  --------
           -------------------- --------------------
  -----------------------SalesLC                2 failover
  cluster    2, 4     NULL
  complete                    0                         0 wait
          Nov 15 2007  3:23PM    Nov 15 2007  3:25PM   Nov 15 2007
  3:23PM

  Remember to issue the 'sp_cluster logical, action, <logical
  ```

```
cluster name>,
release, <handle>' command for any cancelled or completed actions.
```

- **Example 24 –** "SalesLC" is running on "ase1" and "ase2". In this example, "ase1" fails over to "ase3", and "SalesLC" continues to run on "ase2". No wait option is specified, so it defaults to an indefinite wait.

```
sp_cluster logical, "failover", SalesLC, instance, ase1, ase3
```

```
Action '1' has been issued for the 'failover instance' command.
Logical Cluster Handle   Action               From
To    State     InstancesWaiting
   ConnectionsRemaining  WaitType    StartTime            Deadlin
e  CompleteTime
------- ---------------
 -------              ---- ----  --------- ----------------
  --------------------- ----------- ------------------  ---------
----------------
SalesLC             1   failover
instance       1  4    complete               0
                  0  infinite   Nov 15 2007  3:06PM  NULL  Nov 15
2007 3:06PM

Remember to issue the `sp_cluster logical, action, <logical
cluster name>,
release, <handle>' command for any cancelled or completed actions.
```

- **Example 25 –** Gathers and migrates a group of connections to the "new_stores" logical cluster:

```
sp_cluster logical, 'gather', new_stores
```

- **Example 26 –** Displays syntax for the **sp_cluster logical** stored procedures.

```
sp_cluster logical, "help"
```

```
Usage for sp_cluster 'logical':
sp_cluster 'logical', 'help' [, <module>]

To show the logical cluster configuration:
sp_cluster 'logical', 'show'
sp_cluster 'logical', 'show', <lcname>
sp_cluster 'logical', 'show', <lcname> | NULL, 'action' [,
<state>]
sp_cluster 'logical', 'show', <lcname> | NULL, 'route' [, <type [,
<key>]]

To create a logical cluster:
sp_cluster 'logical', 'create', <lcname>

To add resources to a logical cluster:
sp_cluster 'logical', 'add', <lcname>, 'failover',
<instance_list> [,<group>]
sp_cluster 'logical', 'add', <lcname>, 'instance', <instance_list
sp_cluster 'logical', 'add', <lcname>, 'route', <route_type>,
<key_list>

To drop resources from a logical cluster:
```

```
sp_cluster 'logical', 'drop', <lcname>, 'cluster'
sp_cluster 'logical', 'drop', <lcname>, 'failover',
<instance_list>
sp_cluster 'logical', 'drop', <lcname>, 'instance',
<instance_list>
sp_cluster 'logical', 'drop', <lcname>, 'route', <route_type>,
<key_list>

Argument details:
<lcname> is a logical cluster nam
 <instance_list> is a ';' separated list of instance
<route_type> is one of {'user', 'application', 'alias
 <key_list> is a ';' separated list of keys

To set attributes of a logical cluster:
sp_cluster 'logical', 'set', <lcname>, 'open'
sp_cluster 'logical', 'set', <lcname>, 'down_routing',
'disconnect' | 'system' |
    'open'
sp_cluster 'logical', 'set', <lcname>, 'failover', 'instance' |
'group'
sp_cluster 'logical', 'set', <lcname>, 'load_profile',
<profile_name>
sp_cluster 'logical', 'set', <lcname>, 'startup', 'automatic' |
'manual'
sp_cluster 'logical', 'set', <lcname>, 'system_view', 'instance' |
'cluster'

To start and stop a logical cluster:
sp_cluster 'logical', 'online', <lcname>[, <instance_list>]
sp_cluster 'logical', 'offline', <lcname>, 'cluster'[,
<wait_option>[,<time>[,
    @handle output]]]
sp_cluster 'logical', 'offline', <lcname>, 'instance',
    <instance_list>[,<wait_option>[, <time>[, @handle output]]]

To failover and failback a logical cluster:
sp_cluster 'logical', 'failover', <lcname>, 'cluster'[,
<instance_list>[,
    <wait_option>[, <time>[, @handle output]]]]
sp_cluster 'logical', 'failover', <lcname>, 'instance',
<from_instance_list>,
    <instance_list>[, <wait_option>[,<time>[, @handle output]]]
sp_cluster 'logical', 'failback', <lcname>,
'cluster'[,<instance_list>[,
    <wait_option>[, <time>[, @handle output]]]]
sp_cluster 'logical', 'failback', <lcname>, 'instance',
<from_instance_list>,
    <instance_list>[, <wait_option>[,<time>[, @handle output]]]

To work with action handles:
sp_cluster 'logical', 'action', <lcname>, 'cancel', <handle>
sp_cluster 'logical', 'action', <lcname>, 'modify_time',
<handle>, <wait_option>[,
    <time>]
sp_cluster 'logical', 'action', <lcname>, 'release', <handle>
```

```
Argument details:
<wait_option> is one of {'nowait', 'wait', 'until'}
<time> is a time in hh:mm:ss format
<handle> is an action handle
```

- **Example 27 –** Immediately stops all instances in the “SalesLC”, and places “SalesLC” in the offline state.

```
sp_cluster logical, "offline", SalesLC, cluster, nowait
```

- **Example 28 –** Stops the “ase1” and “ase2” instances in “SalesLC”, and places “SalesLC” in the offline state.

```
sp_cluster logical, "offline", SalesLC, instance, "ase1;ase2"
```

- **Example 29 –** Starts all base instances in the “SalesLC” logical cluster, and brings the cluster online.

```
sp_cluster logical, "online", SalesLC
```

- **Example 30 –** Starts the “ase1” instance in “SalesLC”, and brings the cluster online.

```
sp_cluster logical, "online", SalesLC, ase1
```

- **Example 31 –** Sets the load profile for the “SalesLC” logical cluster to the Sybase profile **sybase_profile_oltp**:

```
sp_cluster logical, "set", SalesLC, load_profile,
sybase_profile_oltp
```

- **Example 32 –** Sets the default system view to **cluster**:

```
sp_cluster logical, "set", SalesLC, system_view, cluster
```

- **Example 33 –** Displays summary information for all configured logical clusters.

```
sp_cluster logical, "show", NULL
```

```
 ID  Name           State   Online Instances
Connections--- -------        ------- ------------------
----------------1   mycluster     online       4              1
2   SalesLC        online        2                 0
3   HRLC           online        1                 0
4   CatchallLC     offline       0                 0

Logical cluster 'mycluster' is the system logical cluster.
Logical cluster 'CatchallLC' is the open logical cluster.

Logical Cluster  Instance    State   Type     Connections    Load
Score
---------------- --------   ------    ------   --------------
-----------
HRLC               silk      online     base             0
     0.01
SalesLC            cotton    offline    failover         0
     0.00
SalesLC            linen     online     base             0
     0.00
SalesLC            silk      offline    failover         0
     0.01
```

```
SalesLC            wool      online     base              0
     0.01
mycluster          cotton    online     base              0
     0.00
mycluster          linen     online     base              0
     0.00
mycluster          silk      online     base              0
     0.01
mycluster          wool      online     base              1
     0.01
```

- **Example 34 –** Displays a list of all outstanding actions.

```
sp_cluster logical, "show", NULL, action
```

- **Example 35 –** Displays information for the `SalesLC` logical cluster.

```
sp_cluster logical, "show", SalesLC
```

```
ID          Name           State         Online
Instances      Connections
----------- -------------- ------------  ----------------      --
--------
2           OrderLC        online       1                0

Instance         State       Type     Connections Load
Score  Failover Gro
---------------  ----------  -------  ----------- ----------  ---
--------
asedemo1         online      base              0        0.78  NU

Attribute                             Setting
------------------                    ---------------------------
---------
Down Routing Mode                     system
Failover Mode                         instance with fail_to_any
LC Roles                              none
Load Profile                          sybase_profile_oltp
Login Distribution                    affinity
Startup Mode                          automatic
System View                           cluster

Route Type             Route Key
---------------------- ------------------
application            order_app

Logical cluster 'OrderLC' has no associated actions.
(return status = 0)
```

- **Example 36 –** Creates the load profile “my_profile”:

```
sp_cluster profile, "create", my_profile
```

- **Example 37 –** Specifies the metric weights for “my_profile.” “user connections” is set to zero, which excludes that metric from the profile:

```
sp_cluster profile, "set", my_profile, weight, "user connections",
'0'
sp_cluster profile, "set", my_profile, weight, cpu utilization,
'20'
sp_cluster profile, "set", my_profile, weight, runqueue, '30'
sp_cluster profile, "set", my_profile, weight, io load, '10'
sp_cluster profile, "set", my_profile, weight, engine deficit,
'10'
sp_cluster profile, "set", my_profile, weight, user metric, '30'
```

- **Example 38 –** Sets the login redirection threshold to 80 and the hysteresis value to 10 for "my_profile:"

```
sp_cluster profile, "set", my_profile, threshold, login, '80'
sp_cluster profile, "set", my_profile, threshold, hysteresis, '10'
```

- **Example 39 –** Displays information about a configured profile:

```
sp_cluster profile, "show", my_profile
```

```
ID    Profile      Type    Connections   CPU Run    Queue
----  ------------ ------- ------------- --- ---    ---- ---- ---
--- --- ---
100   my_profile   user            0   20  30  10  10  30  30  0  20

Profile                    Logical Cluster
-------------------------  --------------
my_profile                 SalesLC

Profile                    Logical Cluster Instance

         Load Score        Connections Score
         CPU Score         Run Queue Score
         IO Load Score     User Score
---------------------------------- --------------------
-----------
    ------------------------------------
------------------------------
    ------------------------------------
------------------------------
    ------------------------------------
------------------------------
----------------------------------
my_profile                          SalesLC          ase1
                                 0.028871                0.000000
                                 0.028871                0.000000
                                 0.000000                0.000000
                                    0.000000
my_profile                                           ase2
                                 0.029474                0.000000
                                 0.029474                0.000000
                                 0.000000                0.000000
                                    0.000000
my_profile                                           ase3
                                 0.019503                0.000000
                                 0.019503                0.000000
                                 0.000000                0.000000
                                    0.000000
my_profile                                           ase4
                                 0.582675                0.000000
                                 0.290930                0.291745
```

```
                                    0.000000                 0.000000
                                      0.000000
```

Usage

The parameter usage for sp_cluster is:

Parameter	Usage Consideration
sp_cluster connection	• To migrate the current spid, omit *spid_list* from **sp_cluster connection, migrate**.
sp_cluster logical, action	• Retrieve an action handle by querying the `monLogicalClusterAction` table or executing: `sp_cluster logical, "show", NULL, action` • Any client that does not support migration is disconnected when it completes a SQL batch and has no open transactions, or when the *timeout* period expires, which ever comes first. • Any client remaining at the end of the *timeout* period is disconnected. • Cancelling an action does not roll back the action. Additional tasks may be necessary to restore the configuration to the original state. • Only completed actions can be released. Releasing an action removes the completed action from the system and from the `monLogicalClusterAction` table.

Parameter	Usage Consideration

sp_cluster logical, 'add'	• You cannot add a base instance or a failover resource to the system logical cluster.
	• Separate multiple instance, failover resources, or applications with semicolons.
	• Create multiple failover groups by enclosing the group in parenthesis, and separating groups with a comma. If you do not specify group, a new group is created and the instances are added to that group. You can specify a group into which the instances are placed (the group number must be quoted).
	For example:
	`1> sp_cluster logical, 'add', tempLC, failover, "asedemo3;asedemo2"`
	`2> go`
	`Added failover instance 'asedemo3' to group 1 for logical cluster 'tempLC'.`
	`Added failover instance 'asedemo2' to group 1 for logical cluster 'tempLC'.`
	And then add the instances to the group:
	`1> sp_cluster logical, 'add', tempLC, failover, asedemo4, "4"`
	`2> go`
	`Added failover instance 'asedemo4' to group 4 for logical cluster 'tempLC'.`

sp_cluster logical, "deactivate"	• You cannot use the **deactivate** command for the system logical cluster. • **offline** is identical to the **deactivate**, except **deactivate** places stopped instances or clusters in the inactive state and **offline** places them in the offline state.
sp_cluster logical "drop"	• You must place an instance or failover resource in the offline state before dropping it. • Dropping a cluster also drops all routes, resources, and settings associated with the cluster.
sp_cluster logical "failback"	• To initiate a failback, the logical cluster must first be failed over.
sp_cluster logical "gather"	• The logical cluster must be online to gather connections manually • The logical cluster must have defined routes to gather connections
sp_cluster logical, "offline"	• You cannot use the **offline** command for the system logical cluster. • **offline** is identical to **deactivate**, except **deactivate** places stopped instances or clusters in the inactive state.

Parameter	Usage Consideration
sp_cluster logical "online"	• You cannot use the **online** command for the system logical cluster.
sp_cluster logical "set"	• Only one logical cluster can have the open property. When you set the open property to a new logical cluster, the open property is removed from the previous open logical cluster.
sp_cluster profile	• The user metric value must be normalized so that it is compatible with values for metrics provided by SAP. Consider a user metric that measures response times. If the maximum acceptable response time is 10 seconds and the measured value is 5, the metric value is 50 (5/10 x 100 = 50). • Threshold metrics let you configure at what point a load imbalance should cause connections to be redirected from one instance to another. The workload manager redirects connections when the load score difference (as a percent) between the target instance and the least loaded instance meets or exceeds the threshold value. The hysteresis value guards against redirection when the load score difference meets the threshold value, but the instance load scores (for example, 2 and 8) are so low that redirection is not appropriate.

Permissions

The permission checks for **sp_cluster** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage cluster` privilege or **ha_role**.
Disabled	With granular permissions disabled, you must be a user with **sa_role** or **ha_role**.

sp_clusterlockusage

(Cluster environments only) Reports on the free, used, and retained locks in the cluster .

Syntax

```
sp_clusterlockusage
```

Examples

- **Example 1 –** Reports the locks currently used in the cluster:

```
Lock Usage                         count    % of total
-------------------------  ------------  ------------
Total Locks                        95039           n/a
```

```
 Free Locks                          85807       90.29 %
 Used Locks                           9232        9.71 %
   Object Locks                       4032        4.24 %
   Physical Locks                      233        0.25 %
   Partition Locks                       9        0.00 %
   Table Locks                           0        0.00 %
   Page Locks                            0        0.00 %
   Row Locks                            17        0.02 %
   Others                              501        0.53 %
 Retention Used                          0        0.00 %
```

Usage

`Retention Used` reports on the number of locks that are not owned by any task, but are owned at the cluster level because of lock retention.

Permissions

The permission checks for **sp_clusterlockusage** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage cluster privilege` or a user with **ha_role**.
Disabled	With granular permissions disabled, you must be a user with **sa_role** or **ha_role**.

sp_cmp_all_qplans

Compares all abstract plans in two abstract plan groups.

Syntax

```
sp_cmp_all_qplans group1, group2 [, mode]
```

Parameters

- ***group1, group2* –** are the names of the two abstract plan groups.
- ***mode* –** is the display option. The modes and what information they report are:
  - **counts** – the default mode, this option reports plans that:
    - Are the same
    - Have the same association key, but different groups
    - Exist in one group, but not the other
  - **brief** – the information provided by counts, plus:

    - The IDs of the abstract plans in each group where the plans are different, but the association key is the same
    - The IDs of plans that are in one group, but not in the other.
- **same** – all counts, plus the IDs, queries, and plans for all abstract plans where the queries and plans match.
- **diff** – all counts, plus the IDs, queries, and plans for all abstract plans where the queries and plans are different.
- **first** – all counts, plus the IDs, queries, and plans for all abstract plans that are in the first plan group, but not in the second plan group.
- **second** – all counts, plus the IDs, queries, and plans for all abstract plans that are in the second plan group, but not in the first plan group.
- **offending** – all counts, plus the IDs, queries, and plans for all abstract plans that have different association keys or that do not exist in both groups. This is the combination of the **diff**, **first**, and **second** modes
- **full** – all counts, plus the IDs, queries, and plans for all abstract plans. This is the combination of **same** and **offending** modes.

Examples

- **Example 1 –** Generates a default report on two abstract plan groups:

```
sp_cmp_all_qplans dev_plans, prod_plans
```

```
If the two query plans groups are large, this might take some time.
Query plans that are the same
 count
 -----------
          49
Different query plans that have the same association key
 count
 -----------
           1
Query plans present only in group 'dev_plans':
 count
 -----------
           1
Query plans present only in group 'prod_plans':
 count
 -----------
           0
```

- **Example 2 –** Generates a report using the **brief** mode:

```
sp_cmp_all_qplans dev_plans, prod_plans, brief
```

Usage

There are additional considerations when using **sp_cmp_all_qplans**:

- Use **sp_cmp_all_qplans** to check for differences in abstract plans in two groups of plans.

- **sp_cmp_all_qplans** matches pairs of plans where the plans in each group have the same user ID and query text. The plans are classified as follows:
  - Plans that are the same
  - Plans that have the same association key in both groups, but have different abstract plans. The association key is the group ID, user ID and query text.
  - Plans that exist in one group, but do not exist in the other group
- To compare two individual abstract plans, use **sp_cmp_qplans**. To see the names of abstract plan groups, use **sp_help_qpgroup**.
- When a system administrator or database owner runs **sp_cmp_all_qplans**, it reports on all plans in the two groups. When another user executes **sp_cmp_all_qplans**, it reports only on plans that have the user's ID.

Permissions

The permission checks for **sp_cmp_all_qplans** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege or a user with `monitor qp performance` privilege. Any user can compare plans that they own.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**. Any user can compare plans that they own.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_cmp_qplans* on page 155
- *sp_help_qpgroup* on page 370

sp_cmp_qplans

Compares two abstract plans.

Syntax

```
sp_cmp_qplans id1, id2
```

Parameters

- ***id1, id2* –** are the IDs of two abstract plans.

Examples

- **Example 1 –** Compares abstract plan 411252620 to 1383780087:

  ```
  sp_cmp_qplans 411252620, 1383780087
  ```

  ```
  The queries are the same.
  The query plans are the same.
  ```

- **Example 2 –** Compares abstract plan 2091258605 to 647777465:

  ```
  sp_cmp_qplans 2091258605, 647777465
  ```

  ```
  The queries are the same.
  The query plans are different.
  ```

Usage

There are additional considerations when using **sp_cmp_qplans**:

- **sp_cmp_qplans** compares the queries, abstract plans, and hash keys of two abstract plans, and reports whether the queries are the same, and whether the plans are the same. It prints one of these messages for the query:
  - The queries are the same.
  - The queries are different.
  - The queries are different but have the same hash key.

  It prints one of these messages for the abstract plan:
  - The query plans are the same.
  - The query plans are different.
- **sp_cmp_qplans** also prints a return status showing the results of the comparison. The status values 1, 2 and 10 are additive. The status values and their meanings are:

    - 0 – The query text and abstract plans are the same.
    - +1 – The queries and hash keys are different.
    - +2 – The queries are different, but the hash keys are the same.
    - +10 – The abstract plans are different.
    - 100 – One or both of the plan IDs does not exist.
- To find the ID of a plan, use **sp_help_qpgroup** or **sp_find_qplan**. Plan IDs are also returned by **create plan** and are included in **showplan** output.

Permissions

The permission checks for **sp_cmp_qplans** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege or `monitor qp performance` privilege. Any user can compare plans that they own.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**. Any user can compare plans that they own.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_cmp_all_qplans* on page 152
- *sp_help_qpgroup* on page 370

sp_commonkey

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

- **Example 1 –** Defines a common key on `titles.titleid` and `titleauthor.titleid`:

  ```
  sp_commonkey titles, titleauthor, title_id, title_id
  ```

- **Example 2 –** Assumes two tables, `projects` and `departments`, each with a column named `empid`. This statement defines a frequently used join on the two columns:

  ```
  sp_commonkey projects, departments, empid, empid
  ```

Usage

There are additional considerations when using **sp_commonkey**:

- 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.
- You cannot use a Java datatype with **sp_commonnkey**.

See also **alter table**, **create table**, **create trigger** in *Reference Manual: Commands*.

Permissions

You must be the table owner to execute **sp_commonkey**. Permission checks do not differ based on the granular permissions settings

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_dropkey* on page 275
- *sp_foreignkey* on page 349
- *sp_helpjoins* on page 415
- *sp_helpkey* on page 417
- *sp_primarykey* on page 589

sp_companion

Performs cluster operations such as configuring SAP ASE as a secondary companion in a high availability system and moving a companion server from one failover mode to another. **sp_companion** is run from the secondary companion.

Syntax

```
sp_companion
    [server_name
    {, configure
        [, {with_proxydb | NULL}]
        [, srvlogin]
        [, server_password]
        [, cluster_login]
        [, cluspassword]]
    | drop
    | suspend
    | resume
    | prepare_failback
    | do_advisory}
        {, all
        | help
        | group_attribute_name
        | base_attribute_name}
```

Parameters

- ***server_name*** **–** is the name of the SAP ASE server on which you are performing a cluster operation.
- **configure –** configures the server specified by *server_name* as the primary companion in a failover configuration.
- **drop –** permanently drops a companion from failover configuration. After the command has completed, the servers are in single-server mode.
- **suspend –** temporarily removes the companions from a failover configuration. After the command is completed, the companions are in suspended mode.
- **resume –** reverses the **suspend** command and resumes normal companion mode between the companions.
- **prepare_failback –** prepare the secondary companion to relinquish the primary companion's resources so it can failback.
- **do_advisory –** verifies that the secondary companion is compatible for successfully performing the primary companion's functions during failover mode.
    - **all** – causes **do_advisory** the investigate all the parameters.
    - **help** – displays information and syntax about the **do_advisory** parameter.

    - *group_attribute_name* – is the name of the group attribute upon which **sp_companion** reports
    - *base_attribute_name* – is the name of the base attribute upon which you want **sp_companion do_advisory** reports.
- **with_proxydb –** creates proxy databases on the secondary companion for all database other than the system databases – and all subsequent databases that are added – when this parameter is included in the initial configuration of the companion servers. By default, **with_proxydb** is disabled.
- ***srvlogin* –** is a user's login to access the companion server. By default, the value of **srvlogin** is "sa".
- ***srvpassword* –** is the user's password to access the companion server. By default, the value of **srvpassword** is null.
- ***cluster_login* –** is the user's login to log into the cluster. By default, the value of **cluster_login** is "sa".
- ***cluspassword* –** is the user password you must provide to log into the cluster. By default, the value of **cluspassword** is null.

Examples

- **Example 1 –** Configures the SAP ASE MONEY1 as the primary companion:

  ```
  sp_companion "MONEY1", configure
  ```

- **Example 2 –** Configures the SAP ASE MONEY1 as the primary companion and creates proxy databases on the secondary companion:

  ```
  sp_companion "MONEY1", configure, with_proxydb, "sa", "sapsswd"
  ```

- **Example 3 –** Drops the SAP ASE PERSONEL1 from the failover configuration. After the command has completed, both the primary companion and the secondary companion are in single-server mode:

  ```
  sp_companion "PERSONEL1", "drop"
  ```

- **Example 4 –** Resumes normal companion mode for the companion server (in this example, MONEY1):

  ```
  sp_companion "MONEY1", "resume"
  ```

- **Example 5 –** Prepares the primary companion (in this example, PERSONEL1) to change to normal companion mode and resume control of the SAP ASE server that failed over:

  ```
  sp_companion "PERSONEL1", "prepare_failback"
  ```

- **Example 6 –** Checks to make sure a cluster operation with the PERSONEL1 companion is successful. Because **do_advisory** in this example uses the **all** parameter, it checks all the **do_advisory** attributes of PERSONEL1 to make sure that none of them prevent a successful cluster operation, and that the secondary companion can successfully perform the primary companion's operations after failover is complete:

```
sp_companion "PERSONEL1", do_advisory, "all"
```

- **Example 7 –** Checks to make sure that none of the attributes for the Component Integration Services (CIS) on the companion server is compatible with the local server:

  ```
  sp_companion "PERSONEL1", do_advisory, "CIS"
  ```

Usage

sp_companion performs cluster operations such as configuring SAP ASE as a secondary companion in a high availability system. **sp_companion** also moves companion servers from one failover mode to another (for example, from failover mode back to normal companion mode). **sp_companion** is run from the secondary companion.

sp_companion is installed with the `installhasvss` (`insthasv` on Windows), not the `installmaster` script. `installhasvss` is located in the `scripts` subdirectory in `$SYBASE_ASE`.

sp_companion automatically disables SAP's mirroring. You should use a third-party mirroring software to protect your data from disk failures.

For complete information, see *Using Failover in A High Availability System*. Before running the **do_advisory** command, make sure to read the configuration chapter of this book as well as the **do_advisory** chapter.

Permissions

You must be user with **ha_role** to execute **sp_companion**. Permission checks do not differ based on the granular permissions settings

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_compatmode

Verifies whether full **compatibility mode** can be used.

Syntax

```
sp_compatmode
```

Examples

- **Example 1 –** Verifies whether full compatibility mode can be used:

  ```
  1> sp_compatmode
  -----------------
  Compatibility mode is enabled.
  WARNING: Compatibility mode may not be used when
  statement cache and literalautoparam are enabled.
  WARNING: The configuration option 'histogram tuning
  factor' is configured with value '20', which is not the
  default value in ASE 12.5. This may lead to different
  accuracy of statistics and different query plans.

  (return status = 0)
  1>
  ```

Usage

This query reports whether **compatibility mode** is enabled or not. You see a warning if there are conflicts with the use of **enable compatibility mode**.

For more information, see the *Migration Technology Guide*.

Permissions

Any user can execute **sp_compatmode**. Permission checks do not differ based on the granular permissions settings.

sp_config_dump

Allows you to list, add, or change dump configurations.

Syntax

```
sp_config_dump
    [@config_name = ‘configuration_name’
    [, {
        [@stripe_dir = ’stripe_dir_name’,]
        [@ext_api = ’external_api’,]
        [@num_stripes = ’number_of_stripes’,]
```

```
        [@retry = 'number_of_retries',]
        [@blocksize = 'number_of_bytes',]
        [@compression = 'compression_level',]
        [@retaindays = 'number_of_days',]
        [@init = '[noinit | init]',]
        [@verify = '[header | full]',]
        [@notify = '[client | operator_console]',]
        [@backup_srv_name = backup_server_name',]
        } | ['delete']
    ] ]
```

Parameters

- **@config_name = '*configuration_name*' –** is a unique dump configuration name that is required for adding or changing any specific dump configuration. The SAP ASE server lists all dump configurations when you do not include **'*configuration_name*'**. Additional parameters, when supplied, are changed to new values.
- **@stripe_dir = '*stripe_dir_name*' –** is a file system directory in which files are archived during the dump operation. Archived files are typically named using this format:

  ```
  database_name.dump_type.date-timestamp.stripeID
  ```

  @stripe_dir defaults to the directory where the Backup Server is started.

  @stripe_dir cannot be a tape device.
- **@ext_api = '*external_api*' –** is the name of the external API (byte stream device) used for the dump operation. By default, this parameter is unused. Provide *external_api* in this format:

  ```
  external_API_name::additional_options
  ```
- **@num_stripes = '*number_of_stripes*' –** is the number of stripe devices used during the dump operation. The default is 1.
- **@retry = '*number_of_retries*' –** is the number of times the server tries the dump operation for nonfatal errors. Valid values are 0 to 5; the default is 0 (which indicates no retry).
- **@blocksize = '*number_of_bytes*' –** is the block size for the dump device, overriding the default block size for the device. The value must be at least 1 database page (2048 bytes for most systems), and an exact multiple of the database page size. For optimal performance, specify **blocksize** as a power of 2 (such as 65,536, 131,072, or 262,144).
- **@compression = '*compression_level*' –** is the compression level for compressed dumps. By default, compression is disabled.
- **@retaindays = '*number_of_days*' –**

  is the number of days that Backup Server prevents a dump from being overwritten. Backup Server requires you to confirm any overwrite requests on an unexpired volume. By default, value is 0, meaning dumps can be overwritten.
- **@init = '[noinit | init]' –** specifies whether to initialize the volume. The default is **noinit**.

- **@verify = '[header | full]' –** specifies whether you want Backup Server to perform a minimal page-header or full structural row check on the data pages as they are copied to archives. There is no structural check made to global allocation map (GAM), object allocation map (OAM), allocation pages, indexes, text, or log pages. By default, there is no verification of data pages during archiving.
- **@notify = ’[client | operator_console]' –**

  specifies whether Backup Server routes messages to the client terminal that initiated the dump, or to the operator-console terminal where the Backup Server is running.
- **@backup_srv_name = '*backup_server_name*’ –** specifies the network name of the remote Backup Server running on the machine to which the dump device is attached. Do not use **backup_server_name** to dump to SYB_BACKUP, the default Backup Server. You can specify up to 32 remote Backup Servers using this option.

  For platforms that use interfaces files, the Backup Server name must appear in the interfaces file.
- **'delete' –** specifies the dump configuration to be deleted.

Examples

- **Example 1 –** Lists all dump configurations:

```
sp_config_dump
go

Configuration name
-------------------
dmp_cfg1
dmp_cfg2
dmp_cfg3
```

- **Example 2 –** Lists parameter values for a dump configuration called `dmp_cfg1`:

```
sp_config_dump 'dmp_cfg1'
go

Dump configuration: dmp_cfg1

Option name           Option value
-----------           -------------
compression          5
num_stripes          3
stripe_dir           /work/dump_dir
```

- **Example 3 –** Creates a new dump configuration called `dmp_cfg2` that specifies that a dump operation creates 5 stripes in the `/work1/dmp_dir` stripe directory, and that retries once if it fails with a nonfatal error:

```
sp_config_dump 'dmp_cfg2',
   @stripe_dir='/work1/dmp_dir', @num_stripes='5',
   @retry='1'
```

- **Example 4 –** Changes the stripe directory of an existing dump configuration:

```
sp_config_dump 'dmp_cfg2',
   @stripe_dir='/work2/dmp_dir'
```

- **Example 5 –** Deletes a dump configuration:

```
sp_config_dump 'dmp_cfg2', 'delete'
```

Usage

The **sp_config_dump** procedure does not support tape devices.

See also:

- **dump**, **load**, **genddlonly** in *Reference Manual: Commands*
- For information about dump operations, see the *System Administration Guide*.

Permissions

The permission checks for **sp_config_dump** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage dump configuration` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role** or **oper_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	exec_procedure
Audit option	
Command or access audited	Execution of procedure
Information in `extrainfo`	• *Roles* – current active roles. • *Keywords or options* – NULL. • *Previous value* – NULL. • *Current value* – NULL. • *Other information* – all input parameters. • *Proxy information* – original login name, if **set proxy** is enabled.

sp_confighistory

Creates the `ch_events` view and displays changes made to SAP ASE configuration.

Syntax

```
sp_confighistory create_view
    begin_date[, end_date]]
    last[items_num]
    {area | type | target | element}[, item_name]
    help
```

Parameters

- **create_view –** indicates you are creating the `ch_events` view.
- ***begin_date*, [*end_date* –** displays all items from *begin_date* value to the *end_date* value.
- **last –** displays the latest configuration history items.
- ***items_num* –** number of items to show from the list of latest configuration history items.
- **area | type | target | element –** displays items from the specified area:
    - **area** – area in which the auditable event occurs. One of:
        - `server` – server-level events.
        - `database` – database-level events.
        - `cache` – cache-level events.
        - `traceflag` – **dbcc traceflag** and **set switch** events.
        - `SUSD` – startup/shutdown.
        - `audit` – auditing state changes.
    - **type** – type of auditable event. One of:
        - **sp_configure**
        - **sp_serveroption**
        - **sp_dboption**
        - **sp_cacheconfig**
        - **sp_poolconfig**
        - **create thread pool**
        - **alter thread pool**
        - **drop thread pool**
        - **dbcc traceflag**
        - **set switch**
        - configuration file change
        - startup
        - shutdown

    - shutdown with wait
    - shutdown with nowait
    - abrupt shutdown
    - global auditing
    - config history auditing
  - **target** – name of the target objects to which the change applies (for example, server, cache, thread pool, and database names, traceflag number, and so on).
  - **element** – configuration or other option name (for example, "`enable monitoring`", "`config pool: 4K, option: wash size`", and so on).
- **help –** displays usage information for **sp_confighistory**.

Permissions

- Only the system administrator (users with sa_role) can use this procedure to create the `ch_events` view.
- Only the system administrator (users with sa_role) and users with mon_role can use this procedure to query the `ch_events` view.

The permission checks differ, based on your granular permission settings:

Setting	Description
Enabled	Only users with: • `select any audit table` permission can query against the `ch_events` view. • `manage auditing` permission can change the option state of configuration history auditing • `select any audit table` permission can query against the `ch_events` view. • `select any audit table` permission can query the audit tables.
Disabled	Only: • System security officers (users with sso_role) can change the option state of configuration history auditing • only system administrators (users with sa_role) and users with mon_role can query against the ch_events view.

sp_configure

Displays configuration parameters by group, their current values, their non-default value settings, the value to which they have most recently been set, and the amount of memory used by this setting. Displays only the parameters with a display level that is the same as or below that of the user.

Syntax

```
sp_configure [configname [, configvalue] | group_name |
    non_unique_parameter_fragment] 'drop instance'
    [, instance_name] [display_nondefault_settings]
```

```
sp_configure "configuration file", 0, {"write" | "read" | "verify" |
"restore"}
    "file_name"
```

Parameters

- ***configname* –** displays the current value, default value, most recently changed value, and amount of memory used by the setting for all parameters matching *parameter*.
- ***configvalue* –** resets *configname* to *configvalue* and displays the 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*.
- ***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.
- ***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.
- **drop instance –** allows you to drop an instance-specific configuration setting
- ***instance_name* –** in cluster environments – indicates the instance for which you are setting the instance-specific options.
- **display_nondefault_settings –** displays configuration options for which the configuration or run value is different from the default value.
- **write –** 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.
- **read –** performs validation checking on values contained in *file_name* and reads those values that pass validation into the server. If any parameters are missing from *file_name*, the current running values for those parameters are used.
- **verify –** performs validation checking on the values in *file_name*.
- **restore –** 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.
- ***file_name* –** is the name of the file you want to use **sp_configure** on.

Examples

- **Example 1 –** 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:

  ```
  sp_configure
  ```

- **Example 2 –** Displays all configuration parameters that include the word “identity”:

  ```
  sp_configure "identity"
  ```

  ```
  Configuration option is not unique.

  Parameter Name     Default Memory Used Config Value Run Value Unit
  Type
  --------------      ------- ----------- ------------ ---------
  ------ ----
  identity burning
  set     1         0           1        1     id    static
  identity grab size      0        0          0       0    id   dyna
  size of auto
  identit    10         0           10       10     bytes dyna
  . . .
  ```

- **Example 3 –** Sets the system **recovery interval in minutes** to 3 minutes:

  ```
  sp_configure "recovery interval in minutes", 3
  ```

  ```
  Parameter Name     Default Memory Used Config Value Run Value Unit
  Type
  --------------       ------- ----------- ------------ ---------
  ------ ----
  recovery interval   5       0         3           3        min   dyn

  Configuration option changed. The SQL Server need not be rebooted
  since the option is dynamic.
  ```

- **Example 4 –** Resets the value for **number of devices** to the SAP ASE default:

  ```
  sp_configure "number of device", 0, "default"
  ```

- **Example 5 –** Configures four databases to be recovered concurrently, enter:

  ```
  sp_configure "max concurrently recovered db", 4
  ```

- **Example 6 –** Starts four checkpoint tasks, enter:

  ```
  sp_configure "number of checkpoint tasks", 4
  ```

- **Example 7 –** Captures Query Processing metrics (qp metrics) at the server level:

  ```
  sp_configure "enable metrics capture", 1
  ```

- **Example 8 –** Performs validation checking on the values in the file `srv.config` and reads the parameters that pass validation into the server. Current run values are substituted for values that do not pass validation checking:

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

- **Example 9 –** Runs validation checking on the values in the file `restore.config`:

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

- **Example 10 –** Creates the file `my_server.config` and writes the current configuration values the server is using to that file:

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

- **Example 11 –** Performs a validation check on the values in `$SYBASE/backup_config.cfg`:

  ```
  sp_configure "configuration file", 0, "verify",
      "backup_config.cfg"
  ```

Usage

- 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.
- **sp_configure** allows you to specify the value for configuration paramters in unit specifiers. The unit specifiers are p or P for pages, m or M for megabytes, g or G for gigabytes, and t or T for terabytes. If you do not specify a unit, and you are configuring a parameter that controls memory, the SAP ASE server uses the logical page size for the basic unit.
- Files created with **sp_configure** have restricted permissions.
- 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 the SAP ASE 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 SAP ASE 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 in kilobytes. 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 the SAP ASE server. It changes after you modify a parameter's value with **sp_configure** and, for static parameters, after you restart the SAP ASE server. This is the value stored in `syscurconfigs.value`.

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

- 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 – *default display level*. Displays all parameters, including the most complex ones. This level is appropriate for users who do highly detailed server tuning.

    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
  ```
- **sp_configure** can run in sessions using chained transaction mode if there are no open transactions.
- For information on the individual configuration parameters, see the *System Administration Guide*.

See also:

- **set** in *Reference Manual: Commands*
- For more information on **max concurrently recovered db** and **number of checkpoint tasks**, see *Backing up and Restoring User Databases* in the *System Administration Guide*.

Permissions

The permission checks for **sp_configure** differ based on your granular permissions settings. Any user can display information about parameters and their current values.

Setting	Description
Enabled	With granular permissions enabled: • Only a user with `manage security configuration` privilege can execute **sp_configure** to modify values for parameters in **table** <*table number*>. • You must have the manage server configuration privilege to execute sp_configure to modify values for other configuration parameters.

Setting	Description
Disabled	With granular permissions disabled: • Only user with **sso_role** can execute **sp_configure** to modify values for parameters in **table** <*table number*>. • You must have **sa_role** to execute **sp_configure** to modify values for other configuration parameters: With granular permissions disabled, you must have **sa_role** to execute **sp_configure** to modify values for other configuration parameters: • **allow procedure grouping** • **allow remote access** • **allow sendmsg** • **allow updates to system tables** • **audit queue size** • **auditing** • **automatic master key access** • **check password for digit** • **curread change w/ open cursors** • **current audit table** • **enable encrypted columns** • **enable granular permissions** • **enable ldap user auth** • **enable logins during recovery** • **enable pam user auth** • **enable predicated privileges** • **enable ssl** • **FIPS login password encryption** • **log audit logon failure** • **log audit logon success** • **maximum failed logins** • **minimum password length** • **msg confidentiality reqd** • **msg integrity reqd** • **net password encryption reqd** • **restricted decrypt permission** • **secure default login** • **select on syscomments.text** • **SQL Perfmon Integration** • **suspend auditing when device full** • **syb_sendmsg port number** • **systemwide password expiration** • **unified login required**

Setting	Description
	• **use security services**

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

Information	Values
Event	73
Audit option	Automatically audited event nto controlled by an option.
Command or access audited	Turning the **auditing** parameter on with **sp_configure**
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – NULL • *Proxy information* – Original login name, if **set proxy** in effect

Information	Values
Event	74
Audit option	Automatically audited event nto controlled by an option.
Command or access audited	Turning the **auditing** parameter off with **sp_configure**

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – NULL • *Proxy information* – Original login name, if **set proxy** in effect

Information	Values
Event	82
Audit option	**security**
Command or access audited	**sp_configure**
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – Name of the configuration parameter • *Previous value* – Old parameter value if command is setting a new value • *Current value* – New parameter value if command is setting a new value • *Other information* – Number of configuration parameter, if a parameter is being set; name of configuration file, if a configuration file is being used to set parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_dboption* on page 193
- *sp_displaylevel* on page 232
- *sp_helpconfig* on page 382
- *sp_monitorconfig* on page 532

The sp_configure number of checkpoint tasks Parameter

The `number of checkpoint tasks` parameter configures parallel checkpoints.

Parallel checkpoints depend on the layout of the databases and performance of underlying I/O sybsystems. Tune this parameter depending on the number of active databases and the ability of the I/O subsystem to handle writes.

This configuration parameter is dynamic. When the value for this parameter is reduced, checkpoint tasks drain out, and when the value is increased, additional tasks are created.

The sp_configure max concurrently recovered db Parameter

The **max concurrently recovered db** parameter determines the degree of parallelism during database recovery:

When the SAP ASE server is not in recovery, this configuration parameter takes effect statically. However when the SAP ASE server is in recovery, a system administrator can force serial recovery dynamically.

The effectiveness of **max concurrently recovered db** depends on the database layout and the performance of underlying I/O subsystem.

Setting Configuration Parameters for Clusters Using sp_configure

Considerations for configuring clusters.

- If you do not specify a configuration option or instance name, the information displayed depends on the **system_view** setting.
- If you do not specify a configuration option but specify the instance name, the SAP ASE server displays all instance-specific configuration settings for the specified instance.
- If you specify the configuration option, but not the configuration value and instance name, the SAP ASE server displays the current settings for the specified option for all instances under the “cluster” view. If you specify the instance name, the SAP ASE server displays configuration information for the specified instance.
- If you specify the configuration option and value, but not the instance, the SAP ASE server configures the cluster-wide setting for the option. If, however, you specify the instance name, the SAP ASE server sets the configuration value only for the instance. The syntax is:

  ```
  sp_configure configuration_name, config_value, NULL,
  instance_name
  ```
- You cannot set configuration options from inside a local temporary database.
- If an instance already has instance-specific setting for a configuration parameter set, you can reconfigure this parameter for a cluster-wide setting.
- A user can reconfigure only those instances to which they are connected.

sp_copy_all_qplans

Copies all plans for one abstract plan group to another group.

Syntax

```
sp_copy_all_qplans src_group, dest_group
```

Parameters

- ***src_group* –** is the name of the source abstract plan group.
- ***dest_group* –** is the name of the abstract plan group to which the plans are to be copied.

Examples

- **Example 1 –** Copies all of the abstract plans in the `dev_plans` group to the `ap_stdin` group:

```
sp_copy_all_qplans dev_plans, ap_stdin
```

Usage

There are additional considerations when using **sp_copy_all_qplans**:

- The destination group must exist before you can copy plans into it. It may contain plans.
- **sp_copy_all_qplans** calls **sp_copy_qplan** for each plan in the source group. Each plan is copied as a separate transaction, so any problem that keeps **sp_copy_all_qplans** from completing does not affect the plans that have already been copied.
- **sp_copy_qplan** prints messages when it cannot copy a particular abstract plan. You also see these messages when running **sp_copy_all_qplans**.
- If the query text for a plan in the destination group exactly matches the query text in the source group and the user ID is the same, the plan is not copied, and a message giving the plan ID is sent to the user, but the copying process continues with the next plan in the source group.
- Copying a very large number of abstract plans can take considerable time, and also requires space on the `system` segment in the database and space to log the changes to the database. Use **sp_spaceused** to check the size of `sysqueryplans`, and **sp_helpsegment** for the `system` and `logsegment` to check the space available.

Permissions

The permission checks for **sp_copy_all_qplans** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege. Any user can execute **sp_copy_all_qplans** to copy an abstract plan that they own.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**. Any user can execute **sp_copy_all_qplans** to copy an abstract plan that they own.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_copy_qplan* on page 178
- *sp_help_qpgroup* on page 370

sp_copy_qplan

Copies one abstract plan to an abstract plan group.

Syntax

```
sp_copy_qplan src_id, dest_group
```

Parameters

- ***src_id* –** is the ID of the abstract plan to copy.
- ***dest_group* –** is the name of the destination abstract plan group.

Examples

- **Example 1 –** Copies the abstract plan with ID 2140534659 to the ap_stdin abstract plan group:

  ```
  sp_copy_qplan 2140534659, ap_stdin
  ```

Usage

There are additional considerations when using **sp_copy_qplan**:

- The destination group must exist before you can copy an abstract plan into it. You do not need to specify a source group, since plans are uniquely identified by the plan ID.
- A new plan ID is generated when the plan is copied. The plan retains the ID of the user who created it, even if the system administrator or database owner copies the plan. To assign a different user ID, a system administrator or database owner can use **sp_export_qpgroup** and **sp_import_qpgroup**.
- If the query text for a plan in the destination group exactly matches the query text in the source group and the user ID, the plan is not copied, and a message giving the plan IDs is sent to the user.
- To copy all of the plans in an abstract plan group, use **sp_copy_all_qplans**.

Permissions

The permission checks for **sp_copy_qplans** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege. Any user execute **sp_copy_qplan** to copy a plan that they own.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**. Any user execute **sp_copy_qplan** to copy a plan that they own.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_copy_all_qplans* on page 176
- *sp_help_qpgroup* on page 370
- *sp_help_qplan* on page 373
- *sp_import_qpgroup* on page 451

sp_countmetadata

Displays the number of indexes, objects, or databases in the SAP ASE server.

Syntax

```
sp_countmetadata "configname" [, dbname]
```

Parameters

- ***configname* –** is either "**number of open databases**", "**number of open objects**", or "**number of open indexes**", or "**number of open partitions**".
- ***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

- **Example 1 –** Reports on the number of user objects in the SAP ASE server. Use this value to set the number of objects allowed in the database, plus space for additional objects and temporary tables:

  ```
  sp_configure "number of open objects", 310
  ```

  ```
  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.
  ```

- **Example 2 –** Reports on the number of indexes in the SAP ASE server:

  ```
  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.
  ```

Usage

There are additional considerations when using **sp_countmetadata**:

- **sp_countmetadata** displays the number of indexes, objects, databases, or partitions in the SAP ASE server, including the number of system databases such as `model` and `tempdb`.
- Avoid running **sp_countmetadata** during SAP ASE peak times. It can cause contention on the `sysindexes`, `sysobjects`, `sysdatabases`, and `syspartitions` 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, databases, or partitions, 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 an SAP ASE server for Windows could have a different **sp_countmetadata** result than the same database on Sun Solaris. Information on the number of user indexes, objects, databases, or partitions 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, open partitions** value to accommodate temporary tables.
- If you specify a nonunique fragment of "**open indexes**", "**open objects**", "**open databases**", or "**open partitions**" 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
```

Permissions

The permission checks for **sp_countmetadata** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_configure* on page 167
- *sp_helpconfig* on page 382
- *sp_monitorconfig* on page 532

sp_cursorinfo

Reports information about a specific cursor or all execute cursors that are active for your session.

Syntax

```
sp_cursorinfo [{cursor_level | null}] [, cursor_name]
```

Parameters

- ***cursor_level* | null –** is the level at which the SAP ASE 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. The SAP ASE server reports information about all active cursors that use this name at the *cursor_level* you specify. If you omit this parameter, the SAP ASE server reports information about all the cursors at that level.

Examples

- **Example 1 –** Displays the information about the cursor named `c` at level 0:

```
1> declare c cursor
2> for select au_id,au_lname, au_fname from authors
3> go
1> sp_cursorinfo
2> go
```

```
Cursor name 'c' is declared at nesting level '0'.
The cursor is declared as NON-SCROLLABLE cursor.
The cursor id is 917505.
The cursor has been successfully opened 0 times.
The cursor will remain open when a transaction is
committed or rolled back.
The number of rows returned for each FETCH is 1.
The cursor is updatable.
This cursor is using 5389 bytes of memory.

(return status = 0)
```

- **Example 2 –** Displays information on the cursor's scrollability and sensitivity, in this case a semi-sensitive scrollable cursor **css**:

```
sp_cursorinfo 0, cursor_css
```

```
-------------

Cursor name 'css' is declared at nesting level '0'.
The cursor is declared as SEMI_SENSITIVE SCROLLABLE cursor.
The cursor id is 786434.
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 on a row.
There have been 1 rows read, 0 rows updated and 0 rows deleted
through this
cursor.
The cursor will remain open when a transaction is committed or
rolled back.
The number of rows returned for each FETCH is 1.
The cursor is read only.
This cursor is using 19892 bytes of memory.
There are 2 columns returned by this cursor.
The result columns are:
```

```
Name = 'c1', Table = 't1', Type = INT, Length = 4 (not updatable)
Name = 'c2', Table = 't1', Type = INT, Length = 4 (not updatable)
```

Usage

There are additional considerations when using **sp_cursorinfo**:

- If you do not specify either *cursor_level* or *cursor_name*, the SAP ASE server displays information about all active cursors. Active cursors are those declared by you and allocated by the SAP ASE server.
- The SAP ASE 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 closes if the cursor's current position is deleted.
  - Whether the cursor remains 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.

  The output from **sp_cursorinfo** varies, depending on the status of the cursor. In addition to the information listed, **sp_cursorinfo** displays the **showplan** output for the cursor. For more information about **showplan**, see the *Performance and Tuning Guide*.

See also:

- **declare cursor**, **set** in *Reference Manual: Commands*

Permissions

Any user can execute **sp_cursorinfo**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38

Information	Values
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_dbextend

Allows you to:

- Install automatic database expansion procedures on database/segment pairs and devices.
- Define site-specific policies for individual segments and devices.
- Simulate execution of the database expansion machinery, to study the operation before engaging large volume loads.

These policies are stored in the `sysattributes` table in `master` database.

All arguments are string arguments:

Syntax

```
sp_dbextend ’help’[, command]
```

```
sp_dbextend [ ['set', ['threshold', dbname, segmentname, freespace |
    'database', dbname, segmentname {[ [, growby ] [, maxsize ] ]} |
    'device', devicename { [ [, growby ] [, maxsize ] ] }] |
    'clear', 'threshold', dbname, segmentname
```

```
sp_dbextend 'clear', 'database' [, dbname [, segmentname ] ]
```

```
sp_dbextend 'clear', 'device' [, devicename ]
```

```
sp_dbextend 'modify', 'database', dbname, segmentname,
    { 'growby' | 'maxsize' }, newvalue
```

```
sp_dbextend 'modify', 'device', devicename, { ’growby’ | ' maxsize
' },
    newvalue
```

```
sp_dbextend { 'list' | 'listfull' } [, 'database' [, dbname [,
segmentname
    [, order_by_clause ] ] ] ]
```

```
sp_dbextend { 'list' | 'listfull' } [, 'device' [, devicename [,
order_by_clause ] ] ]
```

```
sp_dbextend { 'list' | 'listfull' }, [ 'threshold' [ , @dbname
    [ , @segmentname ] ] ]
```

```
sp_dbextend 'check', 'database' [, dbname [, segmentname ] ]
```

```
sp_dbextend { 'simulate' | 'execute' }, dbname, segmentname [,
iterations ]
```

```
sp_dbextend 'trace', {'on' | 'off' }
```

```
sp_dbextend 'reload [defaults]'
```

```
sp_dbextend { 'enable' | 'disable' }, 'database' [, dbname [,
segmentname ] ]
```

```
sp_dbextend 'who' [, 'spid' | 'block' | 'all' ]
```

Parameters

- **set –** sets the threshold at which a database, segment, or device should fire. The arguments are:
  - **'threshold',** ***dbname*****,** ***segmentname*****,** ***freespace*** – specifies the free space level at which to install the threshold on a specified database and segment.

    You should always specify *freespace* in size unit specifiers, such as megabytes. If you specify no size units, the value of *freespace* is treated as the number of kilobytes in the segment.
  - **'database',** ***dbname*****,** ***segmentname*** **{[[,** ***growby*** **] [,** ***maxsize*** **]]}** – specifies the name of the database/segment pair, the size by which to alter the database, and the maximum size of the database, at which the expansion process stops.
    - *growby* – is the rate, in unit specifiers or percentage values, at which the database grows at each expansion attempt.
    - *maxsize* is the maximum size of the segment, after which no further expansion occurs. Both are optional parameters.
  - **'device',** ***devicename*** **{ [[,** ***growby*** **] [,** ***maxsize*** **]] }]** – defines the growth rate and maximum size of a device, in unit specifiers or percentage values, at which the device can grow. *maxsize* in devices is subject to OS disk limitations.
- **clear –** clears any previously set rules of expansion for a specified database and segment or for a specified device.
- **modify –** modifies previously set site-specific policies, such as *growby* and *maxsize*, for a database and segment.

  Use *newvalue* to specify the new value you set for automatic expansion.
- **list –** lists briefly existing rules for a specified database, segment, device, or thresholds on specified segments, and presents the data from `master.dbo.sysattributes` in a readable format. Allows you to view rules on a per-database or per-device basis.

  Presents the current rules in effect.

Use *order_by_clause* to generate listings in a different order from the default ordering of name, type.

Use **threshold** to display all the thresholds that are currently installed on the specified database (using the ***@dbname***) and segment (using ***@segment*** name).

- **listfull –** lists fully the site-specific policy rules, and includes a `comment` column in the `sysattributes` table that displays a `datetime` stamp for when the rule was set, and when it was last modified.
- **check –** examines current policies and verifies that they are consistent with the current space layout in each segment. If any policy settings appear redundant, ineffective, or incorrect, a warning message appears.
- **simulate –** simulates executing the database or device expansion schemes executed at runtime, according to the set of current policies implemented by the **set** command.

  iterations specifies the number of times you simulate the expansion.
- **execute –** performs the actual database/segment, or device, expansion, using the current set of policies.
- **reload [defaults] –** reinitializes `sysattributes` with the system-supplied defaults for *growby* and *maxsize* in all databases, segments, and devices, and reverts the databases or devices to the original default behavior.

  reload does not delete user-specified policies.
- **help –** provides help information for all command parameters, such as **set** or **list**, or help information for any single command.
- **trace –** traces the threshold procedure execution logic in all expansion processes.
- **enable, disable –** enables or disables the automatic expansion procedures on a specified database segment or device.
- **who –** shows any active expansion processes running currently. '<spid>' restricts the output for a particular spid. Use:
  - *block* – shows tasks that currently cause blocking of the expansion process.
  - *all* – shows all currently active tasks.
- ***freespace* –** specifies the free space value at which the threshold procedure is installed on the specified segment. Always use size unit specifiers, such as megabytes, to specify *freespace*.
- ***dbname* –** is the name of the database in which the threshold is being installed.
- ***segmentname* –** is the segment contained in database *dbname*.
- ***devicename* –** is the logical name of the affected device.
- ***newvalue* –** specifies the new value you set for automatic expansion when you modify a policy for a database/segment pair or device.
- ***order_by_clause* –** generates listings in a different order from the default ordering in the *list* command. The default order is name, type.
- ***iterations* –** specifies the number of times an expansion is simulated or executed.

- ***growby*** **–** specifies the rate, in unit specifiers or percentage values, at which a specified database segment or device grows each time the threshold procedures are attempted.
- ***maxsize*** **–** is the maximum size of a segment/database pair or device, the size at which automatic expansion must stop.

  maxsize is the maximum size of the segment at which the automatic expansion process stops, not the maximum size of the database.

  You can set *maxsize* to a value larger than the total amount of disk space available on the device, but actual expansion is limited to the available disk space at the time expansion is attempted.

Examples

- **Set Thresholds –** Installs the space expansion threshold on a log segment in the database `pubs2` at 100MB:

  ```
  sp_dbextend 'set', 'thresh', pubs2, logsegment, '100m'
  ```

- **Set Database –** Installs a policy for the `logsegment` segment, at a growth rate of 100MB per expansion attempt:

  ```
  sp_dbextend 'set', 'database', pubs2, logsegment, '100m'
  ```

- **Set Device –** Expands this device until either the OS disk space limitation or the device size of 32GB is reached:

  ```
  sp_dbextend 'set', 'device', pubs2-datadev1, '100m'
  ```

- **Clear –** Shows how to clear all space-expansion thresholds previously installed in `pubs2`, `logsegment`:

  ```
  sp_dbextend 'clear', 'thresh', pubs2, logsegment
  ```

  You can also the space-expansion threshold for segment `dataseg1` in `pubs2`, installed at a free space of 200MB:

  ```
  sp_dbextend 'clear', 'thresh', pubs2, dataseg1, '200m'
  ```

- **Modify –** Defines the rate of growth as 5% of current value, in each expansion attempt:

  ```
  sp_dbextend 'modify', 'da', pubs2, logsegment, 'growby', '5%'
  ```

  A command can fail when *maxsize* is not previously defined:

  ```
  sp_dbextend 'modify', 'device', pubs2_log_dev, 'maxsize', '2.3g'
  ```

- **List –** Lists briefly the rules for all databases and devices:

  ```
  sp_dbextend 'list'
  ```

  This lists rules for all databases with names similar to ‘`pubs%`’:

  ```
  sp_dbextend 'list', 'database', 'pubs%'
  ```

- **Listfull –** Lists the rules for all databases and devices, including a `comment` column showing a `datetime` stamp:

  ```
  sp_dbextend 'listfull'
  ```

- **List Threshold –** When issued from the `pubs2` database, this lists the thresholds setup on various segments in the `pubs2` database:

  ```
  sp_dbextend 'list', 'threshold'
  ```

  To examine the thresholds on a particular segment, use as:

  ```
  sp_dbextend 'list', 'threshold', pubs2, 'logsegment'
  ```

- **Simulate –** Simulates an expansion twice, without tripping the thresholds:

  ```
  sp_dbextend 'simulate', pubs2, logsegment, '2'
  ```

- **Execute –** Executes an automatic expansion procedure:

  ```
  sp_dbextend 'execute', pubs2, logsegment
  ```

- **Help –** Obtains help for a specific command:

  ```
  sp_dbextend help, 'set'
  ```

Usage

There are additional considerations when using **sp_dbextend**:

- You can only set one automatic expansion threshold on any given database/segment pair. If you try to install another instance of the threshold procedure, even at a different free space value, an error is raised.
- You cannot set system-supplied defaults, only modify them. After you modify system defaults you can reset them by re-running the `installdbextend` script, or by using the **reload defaults** command.
- To disallow any automatic growth in a particular segment, either specify 0 for *growby* or *maxsize*, or do not install the threshold procedure at all. If you specify NULL for this parameter, defaults to the system-specified default *growby* rate is used.
- By default, if the size of the device is greater than 40MB, the size of the database is increased by 10 percent. If your datebase is smaller than 40MB, the size of the database is increased by 4MB. However, you can specify database resizing limits that match the needs of your site
- There is no system-specified maximum size for the `default` database. If no *maxsize* value is specified, the size of the database is limited only by the physical limitations of the database device.
- To turn off the automatic growth feature on a particular device, specify 0 for *growby* or *maxsize*. If you do not specify a value for *growby*, the default expansion rate is used.
- When you use this stored procedure to clear a threshold, *dbname* and *segmentname* are required arguments.

- When you use this stored procedure to clear a database, and provide no *dbname* and *segmentname*, all policy rules—that is, all the relevant rows in `master.dbo.sysattributes`—for the current database and all segments in it are deleted. This is a good way to reverse all settings to default and restart.
- When you use this stored procedure to clear a device, if you do not provide a value for *devicename*, no policy rules are cleared. You can clear out the policy rules for a single device by providing *devicename* or using "%" to clear policies for all devices.
- You can specify *dbname*, *devicename*, and *segmentname* using patterns, so that names with patterns that match the specified pattern are considered for the **clear**, **enable**, **disable**, and **list** operations.
- You must have **set** a value or property before you can modify it. **modify** fails if no value was previously set. *growby* and *maxsize* are modified to the new value specified by *newvalue*
- The new value specified in *newvalue* remains in effect throughout subsequent attempts to expand either the database or device. Even if *newvalue* is less than the current size of the database, segment, or device, the object does not shrink. *newvalue* specifies only future expansion, and does not affect current sizes.
- Provide *dbname* and *segmentname* to obtain policy rules for individual databases and for the segments inside them.
- When you use **list** for a database and provide no *dbname* or *segmentname*, all the policy rules (that is, rows in `master.dbo.sysattributes`) for all segments in the current database are listed.
- When you use **list** for a device name and provide no *devicename*, default policy rules for all devices are listed. You can filter this to list the policy rules for a single device by providing *devicename* or use pattern specifiers for the *devicename*.
- You can simulate the expansion of only one database/segment pair at a time. Both *dbname* and *segmentname* are required arguments. You cannot use wildcard patterns in *dbname* or *segmentname* for **execute** or **simulate** commands.
- The maximum size of a device is 32Gb.
- Use *reload* to re-initialize your databases and devices after using **modify** and **simulate**. *reload* deletes any existing rows in `master.dbo.sysattributes` that describe system default behavior, and loads new rows.
- **trace** turns the trace facility on or off throughout the server. If **trace** is on, messages appear in the server error log when a threshold fires. Use **trace** only for troubleshooting.

See also **alter database**, **create database**, **disk init**, **disk resize** in *Reference Manual: Commands*.

Permissions

If the automatic expansion procedures are installed on a segment by a database owner without **sa_role** privilege, the devices do not expand, because the user cannot run the **disk resize** command. A user with **sa_role** privilege should run the **set threshold** command when installing the threshold procedure.

The following permission checks for **sp_dbextend** differ based on your granular permissions settings

Setting	Description
Enabled	With granular permissions enabled: • **sp_dbextend clear database** – You must be a user with `own any database` privilege, or for the specified database, you must be the database owner or a user with `own database` privilege on the database. • **sp_dbextend clear device** – You must be a user with `manage disk` privilege . • **sp_dbextend clear threshold** – You must be the database owner or a user with `own database` privilege on the database. • **sp_dbextend execute** – You must be the database owner or be a user with `own database` privilege on the specified database, and you must have `manage disk` privilege. • **sp_dbextend simulate** – You must be the database owner or a user with `own database` privilege. • **sp_dbextend enable/disable** – You must be a user with `own any database` privilege or the database owner or have the `own database` privilege on the specified database. • **sp_dbextend list database** – You must be a user with `own any database` privilege when **%** pattern is specified. • **sp_dbextend list @ verbose=2** – You must be a user with `own any database` privilege. • **sp_dbextend modify database** – You must be the database owner or a user with `own database` privilege on the specified database or a user with `own any database` privilege for **sp_dbextend 'modify'**, **'database'**, **'default'**. • **sp_dbextend modify device** – You must be the database owner or a user with `manage disk` privilege. • **sp_dbextend reload defaults** – You must be a user with `own any database` privilege. • **sp_dbextend set database** – You must be the database owner or a user with `own database` privilege on the specified database. • **sp_dbextend set device** – You must be a user with `manage disk` privilege. • **sp_dbextend set threshold** – You must be the database owner or a user with `own database` on the specified database and you must have the `manage disk` privilege. • **sp_dbextend trace** – You must be a user with `set switch` privilege.

Setting	Description
Disabled	With granular permissions disabled: • **sp_dbextend clear database** – You must be the database owner or a user with **sa_role**. • **sp_dbextend clear device** – You must be a user with **sa_role**. • **sp_dbextend clear threshold** – You must be the database owner or a user with sa_role. • **sp_dbextend execute** – You must be a user with **sa_role**. • **sp_dbextend simulate** – You must be the database owner or a user with **sa_role**. • **sp_dbextend enable/disable** – You must be the database owner or a user with **sa_role**. • **sp_dbextend list database** – You must be a user with **sa_role** permission when **%** pattern is specified. • **sp_dbextend list @ verbose=2** – You must be a user with **sa_role**. • **sp_dbextend modify database** – You must be the database owner or a user with **sa_role** if `dbname` equals default. • **sp_dbextend modify device** – You must be the database owner or a user with **sa_role**. • **sp_dbextend reload defaults** – You must be the database owner or a user with **sa_role**. • **sp_dbextend set database** – You must be the database owner or a user with **sa_role**. • **sp_dbextend set device** – You must be the database owner or a user with **sa_role**. • sp_dbextend set threshold – You must be the database owner or a user with **sa_role**. • **sp_dbextend trace** – You must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_dropthreshold* on page 291
- *sp_modifythreshold* on page 520

sp_dboption

Displays or changes database options, and enables the asynchronous log service feature.

Syntax

```
sp_dboption [dbname, optname, optvalue [, dockpt]]
```

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, 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. The SAP ASE 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.

  You can turn on more than one database option at a time. You cannot change database options inside a user-defined transaction.
- ***optvalue* –** is the value of the setting. **true** turns the option on, and **false** turns it off.
- ***dockpt* –** specifies whether **sp_dboption** performs the **checkpoint** command on *dbname*. The default value is 1, which automatically performs **checkpoint**. You can run **checkpoint** on the *dbname* by manually executing the **checkpoint**.

Examples

- **Example 1 –** Displays list of database options:

  ```
  sp_dboption
  ```

  ```
  Settable database options
  ```

  ```
   database_options
   ------------------------
  ```

```
abort tran on log full
allow incremental dumps
allow nulls by default
allow wide dol rows
async log service
auto identity
dbo use only
ddl in tran
deallocate first text page
deferred table allocation
delayed commit
enforce dump tran sequence
erase residual data
full logging for all
full logging for alter table
full logging for reorg rebuild
full logging for select into
identity in nonunique index
no chkpt on recovery
no free space acctg
read only
scratch database
select into/bulkcopy/pllsort
single user
trunc log on chkpt
trunc. log on chkpt.
unique auto_identity index
```

- **Example 2 –** Makes database `pubs2` read-only:

  ```
  1> use pubs2
  2> go
  1> master..sp_dboption pubs2, "read", true
  2> go
  ```

  ```
  Database option 'read only' turned ON for database 'pubs2'.
  Running CHECKPOINT on database 'pubs2' for option 'read only' to
  take effect.
  (return status = 0)
  ```

  The **read** string uniquely identifies the **read only** option from among all available database options. Note the use of quotes around the keyword **read**.

- **Example 3 –** Makes the database `pubs2` writable again, but by specifying 0 for the *dockpt* option, you see "Run the CHECKPOINT command in the database that was changed":

  ```
  1> use pubs2
  2> go
  1> master..sp_dboption pubs2, "read", false, 0
  2> go
  ```

  ```
  Database option 'read only' turned OFF for database 'pubs2'.
  Run the CHECKPOINT command in the database that was changed.
  (return status = 0)
  ```

  To manually perform a checkpoint on `pubs2`, enter:

```
1> checkpoint
2> go
```

- **Example 4 –** Allows **select into**, **bcp**, parallel sort operations on tables in `pubs2`. The **select into** string uniquely identifies the **select into/ bulkcopy** option from among all available database options:

```
use pubs2
go
master..sp_dboption pubs2, "select into", true
go
```

Note: Quotes are required around the option because of the embedded space.

- **Example 5 –** Automatically defines 10-digit IDENTITY 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:

```
use mydb
go
master..sp_dboption mydb, "auto identity", true
go
```

- **Example 6 –** Automatically includes an IDENTITY column in the index keys of `mydb` tables, provided these tables already have an IDENTITY column. All indexes created on the tables are internally unique:

```
use master
go
sp_dboption mydb, "identity in nonunique index", true
go
use mydb
go
```

- **Example 7 –** Automatically includes IDENTITY With unique, nonclustered index for new tables in `pubs2`:

```
use master
go
sp_dboption pubs2, "unique auto_identity index", true
go
use pubs2
go
```

- **Example 8 –** Sets asynchronous log service (ALS) in a specified database, enabling the user log cache and the log writer threads.

```
sp_dboption "mydb", "async log service", true
use mydb
```

- **Example 9 –** Disables ALS in a specified database:

```
sp_dboption "mydb", "async log service", false
use mydb
```

- **Example 10 –** Enforces a dump transaction sequence for `big_db`:

  ```
  sp_dboption 'big_db', 'enforce dump tran sequence',
  true
  ```

- **Example 11 –** Enables full logging for **select into** and **alter table** in `mydb`:

  - The **create database** command creates `mydb`:

    ```
    create database mydb on datadev=20 log on logdev=10
    go
    ```

    ```
    CREATE DATABASE: allocating 10240 logical pages (20.0
    megabytes) on disk
    'datadev' (10240 logical pages requested).
    CREATE DATABASE: allocating 5120 logical pages (10.0 megabytes)
    on disk
    'logdev' (5120 logical pages requested).
    Database 'mydb' is now online.
    ```

  - Turns on the full-logging option for **select into** in `mydb`:

    ```
    sp_dboption "mydb", "full logging for select into", "true"
    go
    ```

    ```
    Database option 'full logging for select into' turned ON for
    database
    'mydb'.
    Running CHECKPOINT on database 'mydb' for option 'full logging
    for select
    into' to take effect.
    (return status = 0)
    ```

  - Turns on the full-logging option for **alter table** in `mydb`:

    ```
    sp_dboption "mydb", "full logging for alter table", "true"
    go
    ```

    ```
    Database option 'full logging for alter table' turned ON for
    database
    'mydb'.
    Running CHECKPOINT on database 'mydb' for option 'full logging
    for alter
    table' to take effect.
    (return status = 0)
    ```

  - Running **sp_helpdb** shows the settings for `mydb`:

    ```
    sp_helpdb mydb
    go
    ```

    ```
     name db_size owner dbid created      durability status
     ---- ------- ----- ---- ------------ ---------- --------------
    ---------
     mydb 30.0 MB  sa      5 Dec 16, 2010 full      full logging for
    select
                                                     into/alter table

    (1 row affected)
     device_fragments size    usage      created             free
    kbytes
    ----------------- ------- ---------- ------------------- -----
    ```

```
---------
 datadev          20.0 MB data only  Dec 16 2010 6:08PM  18696
 logdev           10.0 MB log only   Dec 16 2010 6:08PM  not
applicable

---------------------------------------------------------------
log only free kbytes = 10184
(return status = 0)
1>
```

- **Example 12 –** Enables back-up and restoration of cumulative dumps:

```
sp_dboption mydb, "allow incremental dumps", true
```

- **Example 13 –** Enables deferred table creation for `pubs2`:

```
sp_dboption pubs2, 'deferred table allocation', true
```

- **Example 14 –** The syntax to enable the removal of residual data at the database level for these two examples is:

```
sp_dboption dbname, "erase residual data", true
```

The following examples use these two tables:

- `create table t1 (col1 int) with "erase residual data" on`
- `create table t2 (col1 int) with "erase residual data" off`

The option to erase residual data is turned on for table `t1` because it is set at the database level, so that both the **drop table** and **truncate table** commands for `t1` result in the cleanup of all residual data from its pages.

Table `t2`, however, has the **erase residual data** option turned off explicitly, as it was created with the "**erase residual data off**" clause. Residual data is not removed, even though the "`erase residual data`" option is set to `true` at the database level. As a result, residual data remains, even after running **drop table** and **truncate table** on `t2`:

```
create database db1
go
sp_dboption db1, "erase residual data", true
go
use db1
go
create table t1 (col int)
go
insert t1 values ...
go
create table t2 (col1 int, col2 char(10)) with "erase residual
data" off
go
truncate table t1
go
drop table t1
go
truncate table t2
go
```

```
drop table t2
go
```

In the second example:

- Table `t1` does not have "`erase residual data off`" set explicitly, but does have it set at the database level, resulting in the removal of residual data from `t1` when you run **truncate table t1**.
- Table `t2` has the "`erase residual data`" option set at creation, because the option was set at the database level. This results in the removal of residual data from `t2` when you run **truncate table t2**.
- Table `t3` is marked with "`erase residual data off`" explicitly, so that even though **sp_dboption** sets "`erase residual data`" to true, residual data is not removed when SAP ASE runs **truncate table t3**.

```
create database db1
go
use db1
go
create table t1 (col int)
go
sp_dboption db1, "erase residual data", true
go
create table t2 (col1 int, col2 char(10))
go
create table t3 (col1 int, col2 char(10)) with "erase residual
data" off
go
truncate table t1
go
truncate table t2
go
truncate table t3
go
```

- **Example 15 –** Deallocate the first text page after a NULL update:

```
sp_dboption mydb, "deallocate first text page", true
```

Usage

- When you enable the "`erase residual data`" setting at the database level, any operation that results in deallocation is followed by the cleaning of its pages. By default, this option is disabled
- You cannot change `master` database option settings.
- If you enter an ambiguous value for *optname*, an error message appears. For example, two of the database options are **dbo use only** and **read only**. Using “only” for the *optname* parameter generates a message because it matches both names. The complete names that match the string supplied are printed out so that you can see how to make the *optname* more specific.

- 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 **no chkpt on recovery** option disables the **trunc log on chkpt** option when both are set with **sp_dboption** for the same database. This conflict is especially possible in the **tempdb** database which has **trunc log on chkpt** set to **on** as the default.
- 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.

See also:

- **alter table**, **checkpoint**, **create default**, **create index**, **create procedure**, **create rule**, **create schema**, **create table**, **create trigger**, **create view**, **drop default**, **drop index**, **drop procedure**, **drop rule**, **drop table**, **drop trigger**, **drop view**, **grant**, **revoke**, **select** in *Reference Manual: Commands*
- See the *System Administration Guide* for more information on database options.
- **bcp** in the *Utility Guide*

Permissions

The permission checks for **sp_dboption** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner or a user with `own database` privilege on the database. Any user can display database options.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**. Any user can display database options. A user aliased to the database owner cannot execute **sp_dboption** to change database options.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**

Information	Values
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_configure* on page 167
- *sp_helpdb* on page 394
- *sp_helpindex* on page 409
- *sp_helpjoins* on page 415

Full Logging and sp_dboption

By default, **select into**, certain types of **alter table**, and **reorg rebuild** are run in minimally logged mode. Before executing these commands, first set the **select into**/**bulk copy** database option to **true** to allow the SAP ASE server to break the dump sequence—that is, to perform operations that prevent the ability to use **dump transaction**.

When you use the "**full logging for [select into | alter table | reorg rebuild | all]**" option, the command is run with full logging. Any previously set value of **select into**/**bulk copy** becomes irrelevant for any of the now-fully logged commands.

Full logging for fast **bcp** and parallel sort is not supported, and cannot take place unless you set **select into**/**bulk copy** to true.

Once the operation is set to run with full logging, you can run **dump transaction**/**load transaction** and recovery for these operations, just like any other fully logged operation.

The syntax to fully log commands that are, by default, minimally logged is:

```
sp_dboption dbname, “full logging for
    [select into | alter table | reorg rebuild | all]”,
    true | false
```

where:

- **full logging for select into** – in order to have a **select into proxy table** fully logged, set the "**full logging for select into**" option to true on the remote server that hosts the actual table. If you set the **full logging for select into** option to false on the server that hosts the actual table, the command is then executed with minimal logging in that database and the dump transaction sequence breaks.

- **pll create index** – enables full logging when a parallel sort is done. Parallel sorting is required when you create a clustered index on a round-robin-partitioned table
- **full logging for alter table** – enables full logging for these versions of **alter table** that require data movement:
  - **alter table add *column* not null**
  - **alter table drop *column* not null**
  - **alter table modify *datatype* of not null *column***
  - **alter table partition**

  Other variants of **alter table** are already executed in fully logged mode.

  Note: Changing the locking scheme between an allpages-locked table and a datapages-locked/data rows-locked table by **alter table lock** requires data movement, however, this behavior is not supported by **full logging for alter table**.
- **full logging for reorg rebuild** – involves table data movement. This has no impact on the **reorg rebuild** index command, which is already fully logged. This parameter enables **full logging for reorg rebuild** table statements. When you do not set this option (or set this option to false), the SAP ASE server executes the **reorg rebuild** table command with minimal logging.
- **full logging for all** – enables all the above full logging options. Setting **all** to false disables all the full logging options.

Note: The syntax requires that you specify what you want to fully log; "**full logging**" by itself is not a valid option.

When you use any of the **full logging for** option, the command is run with full logging. Any previously set value of **select into/bulk copy/pllsort** becomes irrelevant for any of the now-fully logged commands. Full logging for fast **bcp** and parallel sort is not supported and cannot take place unless you set **select into/bulk copy/pllsort** to true.

Once the operation is set to run with full logging, you can run **dump transaction**/**load transaction** and recovery for these operations, just like any other fully logged operation.

The **dboption** is "**full logging for all**" and not just "**full logging**" on its own.

Note: The execution of a fully logged **select into**, **alter table**, or **reorg rebuild** command may require a significant amount of log space to accommodate the transaction log.

Shrinking the Log

Issuing **select into**, **alter table**, and **reorg rebuild** when full logging is enabled can greatly increase the demand for log space, particularly for large tables. You may need to increase the size of the log. Once you have completed the command, you may remove the extra log space using the **alter database log off** command.

See **alter database** and *Shrinking Log Space* in *System Administration Guide Volume I*.

You cannot set full logging for **select into**, **alter database**, or **reorg rebuild** for:

- The `master` database
- In-memory databases

You can change the settings of:

- Any database that has mixed log and data segments, but the option is ignored until such time as the database is altered to no longer have mixed log and data segments.
- A database that does not have a durability level of full, but the option is ignored until the database is altered to have full durability.

These restrictions apply because none of the databases allow you to execute a **dump transaction** command. The use of fully recoverable DDLs enables **dump transaction**.

Allowing Wide Rows Using sp_dboption

allow wide dol rows configures databases to allow wide, variable-length data-only locked (DOL) rows.

- You must enable **allow wide dol rows** separately for each database.
- You can set the **allow wide dol rows** database option in user databases only. You cannot set the **allow wide dol rows** database option for the `master` database.
- Enabling **allow wide dol rows** in an the SAP ASE server configured with page size of 8K or less has no effect.
- Disabling **allow wide dol rows** prevents SAP ASE from creating wide, variable-length DOL rows; it does not prevent you from selecting data that includes such rows. However, until you enable **allow wide dol rows**, you cannot change rows that contain wide data, unless the change produces rows that no longer contain wide data.
- Temporary databases cannot use wide DOL worktables until you enable their **allow wide dol rows** setting. If you use `tempdb` groups, enable **allow wide dol rows** either for all databases in the group or for none of them, so worktable and query processing behavior is consistent across the group, regardless of the `tempdb` to which a particular user session is bound.

Asynchronous Log Service (ALS) Options

Enabling **async log service** (ALS) allows for greater scalability in the SAP ASE server, providing higher throughput in logging subsystems for high-end symmetric multiprocessor systems.

- The ALS option is disabled by default.
- You cannot enable the ALS option in system databases, such as `master` or `model`.
- The ALS option is persistent; once you enable ALS on a specified database, you can dump and reload the database without disabling ALS. To disable this feature, you must use **sp_dboption** to set the parameter to **false**.

Considerations for Using enforce dump tran sequence

enforce dump tran sequence prevents operations that disallow a subsequent dump transaction.

- **false** – (the default) does not affect operations that interfere with dump transactions.
- **true** – disallows operations that prevent a dump transaction.

You can set this option to true, only if the database:

- Is a dedicated log database.
- Is not an archive database.
- Is not a local or global temporary database.
- Is not read-only.
- Was not brought online for standby access.
- Has full durability. Databases with **at_shutdown** and **no_recovery** durability are not allowed.
- Has **select into/bulk copy/pllsort** or **trunc log on chkpt** set to false. If any of these options are true, they automatically reset to false.
- Does not need a **dump database** due to one of the following reasons. Perform a dump database before setting this database option to true.
  - A partially logged update has been done, for example, **select into**, **alter table modify**, **reorg rebuild**, **fast bcp**, and **writetext**.
  - The transaction log was truncated.
  - It is a newly created or upgraded database.

  If the database option **enforce dump tran sequence** is true, you cannot:
  - Set **select into/bulk copy/pllsort** to true. Commands with partial logging are not allowed.
  - Set **trunc log on chkpt** to true. The log cannot be truncated by the **checkpoint** process.
  - Execute **dump tran with truncate_only** or **dump tran with no_log**. The log cannot be truncated without dumping it to an archive device.
  - Mark the database as read-only.
  - Change durability from **full** to **at_shutdown** or **no_recovery**.
  - Change to be a **mixed-log-and-data** database. In cases like **load database** and **dbcc findstranded** where the database may be changed to mixed log and data.

Database Options and sp_dboption

There are additional considerations when using the database options of **sp_dboption**.

- 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.

  You cannot use **allow nulls by default** to change the nullibility of a column during **select into** statements. Instead, use **convert** to specify the nullibility of the resulting columns.
- 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 (DDL) 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 DDL commands on `tempdb` within transactions may cause your system to grind to a halt. Always leave **ddl in tran** set to **false** in `tempdb`.
- You can use these commands inside a user-defined transaction when the **ddl in tran** option is set to **true**:
  - **alter table** – clauses other than **partition** and **unpartition** are allowed
  - **create default**
  - **create index**
  - **create procedure**
  - **create rule**
  - **create schema**

    - **create table**
    - **create trigger**
    - **create view**
    - **drop default**
    - **drop index**
    - **drop procedure**
    - **drop rule**
    - **drop table**
    - **drop trigger**
    - **drop view**
    - **grant**
    - **revoke**
- You can never use these commands inside a user-defined transaction:
    - **alter table**
    - **alter table...lock**
    - **alter table...partition**
    - **alter table...unpartition**
    - **create database**
    - **disk init**
    - **dump database**
    - **dump transaction**
    - **drop database**
    - **load database**
    - **load transaction**
    - **select into**
    - **truncate table**
    - **update statistics**

  In addition, system procedures that create temporary tables or change the `master` database cannot be used inside user-defined transactions.
- You may enable **deferred table allocation** for the `model` database, but not for any other system databases, including `master`, `sybsystemprocs`, `sybsystemdb`, or for any temporary databases.
- **identity in nonunique index** automatically includes an IDENTITY column in a table's index keys, so that all indexes created on the table are unique. This database option makes logically nonunique indexes internally unique, and allows 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 is 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.

Do not confuse the **identity in nonunique index** option with **unique auto_identity index**, which is used to add an IDENTITY column with a unique, nonclustered index to new tables.

For a report on indexes in a particular table that includes the IDENTITY column, execute **sp_helpindex**.

- **no free space acctg** suppresses free-space accounting and execution of threshold actions for data segments. Setting **no free space acctg** to **true** speeds recovery time because speeds recovery time because the free-space counts are not recomputed for data 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 the SAP ASE server. This checkpoint, which ensures that you need not re-run the recovery mechanism 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.
- **select into/bulkcopy/pllsort** must be set to **on** to perform operations that do not keep a complete record of the transaction in the log, which include:
  - Using the **writetext** utility.
  - Doing a **select into** a permanent table.
  - Doing a "fast" *bulk copy* with **bcp**. By default, fast **bcp** is used on tables that do not have indexes.
  - Performing a parallel sort.

  A transaction log dump cannot recover these minimally logged operations, so **dump transaction** to a dump device is prohibited. However, you can still use **dump transaction...with no_log** and **dump transaction...with truncate_only**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.

Setting **select into/bulkcopy/pllsort** does not block log dumping, but making minimally logged changes to data does block the use of a regular **dump transaction**. .

By default, **select into/bulkcopy/pllsort** is turned off in newly created databases. To change the default, turn this option on in the `model` database.

- 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.

  trunc log on chkpt is off in newly created databases. To change the default, turn this option on in the `model` database.

  Warning! If you set **trunc log on chkpt** on in `model`, and you need to load a set of database and transaction logs into a newly created database, be sure to turn the option off in the new database.

- The **delayed commit** option is disabled by default. When this is enabled, all local transactions use delayed commits. That is, at the time of commit, control returns to the client without waiting for the I/O on the log pages to complete, and the I/O is not issued on the last log buffer for delayed commit transactions. Delayed commits are not used when both **delayed commit** and ALS options are enabled for a database.

- 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 IDENTITY 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.

  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.

  The **unique auto_identity index** option provides a mechanism for creating tables that have an automatic IDENTITY column with a unique index that can be used with updatable cursors. The unique index on the table ensures that the cursor is positioned at the correct row after a **fetch**. (If you are using isolation level 0 reads and need to make logically nonunique indexes internally unique so that they can process updatable cursors, use the **identity in nonunique index** option.)

In some cases, the **unique auto_identity index** option can avoid the Halloween problem for the following reasons:

- Users cannot update an IDENTITY column; hence, it cannot be used in the cursor update.
- The IDENTITY column is automatically created with a unique, nonclustered index so that it can be used for the updatable cursor scan.

For more information about the Halloween problem, IDENTITY columns, and cursors, see *Cursors: Accessing Data* in the *Transact-SQL Users Guide* and *Optimization for Cursors* in *Performance and Tuning Series: Query Processing and Abstract Plans*.

Do not confuse the **unique auto_identity index** option with the **identity in nonunique index** option, which is used to make all indexes in a table unique by including an IDENTITY column in the table's index keys.

sp_dbrecovery_order

Specifies the order in which user databases are recovered and lists the user-defined recovery order of a database or all databases.

Syntax

```
sp_dbrecovery_order [database_name [, rec_order [, force [, relax |
strict ]]]]
```

Parameters

- ***database_name* –** The name of the database being assigned a recovery order or the database with a user-defined recovery order that is to be listed.
- ***rec_order* –** The order in which the database is to be recovered. A *rec_order* of `-1` deletes a specified database from the user-defined recovery sequence.
- **force –** allows the user to insert a database into an existing recovery sequence without putting it at the end.
- **relax –** specifies that the databases are made as they recover (default).

  The default is **relax**, which means that databases are brought online immediately when recovery has completed.
- **strict –** specifies that the databases are specified by the recovery order.

Examples

- **Example 1 –** Makes the `pubs2` database the first user database to be recovered following a system failure:

```
sp_dbrecovery_order pubs2, 1
```

- **Example 2 –** Inserts the `pubs3` database into third position in a user-defined recovery sequence. If another database was initially in third position, it is moved to fourth position, and all databases following it are moved accordingly:

  ```
  sp_dbrecovery_order pubs3, 3, force
  ```

- **Example 3 –** Removes the `pubs2` database from the user-defined recovery sequence. Subsequently, `pubs2` is recovered after all databases with a user-specified recovery order have recovered:

  ```
  sp_dbrecovery_order pubs2, -1
  ```

- **Example 4 –** Lists the current recovery order of all databases with a recovery order assigned through **sp_dbrecovery_order**:

  ```
  sp_dbrecovery_order
  ```

Usage

There are additional considerations when using **sp_dbrecovery_order**:

- You must be in the `master` database to use **sp_dbrecovery_order** to enter or modify a user-specified recovery order. You can list the user-defined recovery order of databases from any database.
- To change the user-defined recovery position of a database, use **sp_dbrecovery_order** to delete the database from the recovery sequence, then use **sp_dbrecovery_order** to insert it into a new position.
- System databases are always recovered before user databases. The system databases and their recovery order are:
  1. `master`
  2. `model`
  3. `tempdb`
  4. `sybsystemdb`
  5. `sybsecurity`
  6. `sybsystemprocs`
- If no database is assigned a recovery order through **sp_dbrecovery_order**, all user databases are recovered in order, by database ID, after system databases.
- If *database_name*:
  - Is specified but no ***rec_order*** is given – **sp_dbrecovery_order** shows the user-defined recovery position of the specified database.
  - Is not specified – **sp_dbrecovery_order** lists the recovery order of all databases with a user-assigned recovery order.
- The order of recovery assigned through **sp_dbrecovery_order** must be consecutive, starting with **1** and containing no gaps between values. The first database assigned a

recovery order must be assigned a *rec_order* of **1**. If three databases have been assigned a recovery order of **1**, **2**, and **3**, you cannot assign the next database a recovery order of **5**.

Permissions

The permission checks for **sp_dbrecovery_order** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege
Disabled	With granular permissions disabled, you must be a user with **sa_role**. .

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_dbremap

Forces the SAP ASE server to recognize changes made by **alter database**. Run this procedure only when instructed to do so by an SAP ASE message.

Syntax

```
sp_dbremap dbname
```

Parameters

- ***dbname* –** is the name of the database in which the **alter database** command was interrupted.

Examples

- **Example 1 –** An **alter database** command changed the database `sample_db`. This command makes the changes visible to the SAP ASE server:

  ```
  sp_dbremap sample_db
  ```

Usage

There are additional considerations when using **sp_dbremap**:

- If an **alter database** statement issued on a database that is in the process of being dumped is interrupted, the SAP ASE 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 the SAP ASE server until the next restart.

See also:

- **alter database**, **dump database**, **dump transaction** in *Reference Manual: Commands*

Permissions

The permission checks for **sp_dbremap** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner or a user with `own database` privilege on the database.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_defaultloc

(Component Integration Services only) Defines a default storage location for objects in a local database.

Syntax

```
sp_defaultloc dbname, defaultloc, 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*.The valid values are as follows. Enclose the *defaulttype* value in quotes:
  - **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

- **Example 1 – sp_defaultloc** defines the remote storage location `pubs.dbo.` in the remote server named MYSERVER. 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`:

  ```
  sp_defaultloc pubs, MYSERVER.pubs.dbo., table
  create table pubs.dbo.book1 (bridges char(15))
  ```

- **Example 2 –** Removes the mapping of the database `pubs` to a remote location:

  ```
  sp_defaultloc pubs, NULL
  ```

- **Example 3 –** 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:

  ```
  sp_defaultloc ticktape, wallst.nasdaq.dbo., rpc
  create existing table sybase (bestbuy integer)
  ```

Usage

There are additional considerations when using **sp_defaultloc**:

- **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.

See also **create existing table**, **create table** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_defaultloc**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addobjectdef* on page 37
- *sp_addserver* on page 46
- *sp_helpserver* on page 434

sp_deletesmobj

(Only when the TSM is licensed at your site) Deletes specified backup objects from the IBM Tivoli Storage Manager (TSM).

Syntax

```
sp_deletesmob “syb_tsm“, “server_name"{, “database_name”,
“object_type”, “dump_type", “until_time", “bs_name"}
```

Parameters

- **syb_tsm –** is the keyword that invokes the libsyb_tsm.so module that enables communication with TSM.
- ***server_name* –** is the name of the SAP ASE server associated with the TSM backup objects to be deleted.
- ***database_name* –** is the name of the database associated with the TSM backup objects to be deleted. An asterisk (*) indicates all databases.
- ***object_name* –** is the name of theTSM backup object as provided in the **dump database** or **dump transaction** command. If this parameter is omitted, all backup objects are deleted. An asterisk (*) indicates all backup objects.
- ***dump_type* –** is the backup object type to be deleted. Values are:
  - **DB** – database backup objects created by the **dump database** command.
  - **XACT** – database backup objects created by the **dump transaction** command.
  - ***** – all database backup objects. This is the default.
- ***until_time* –** is the date timestamp field. All backup objects matching the specified criteria and created before the *until_time* date are deleted.
- ***bs_name* –** is the name of the remote Backup Server. If *bs_name* is omitted, the default is SYB_BACKUP.

Examples

- **Example 1 –** Removes all TSM backup objects created at the SAP ASE “svr1.”

  ```
  sp_deletesmobj "syb_tsm", "svr1"
  ```

- **Example 2 –** Removes all backup objects of the `testdb` database created by “svr1” before May 20, 2009, 10:51:43:866am. The backup object name is “obj1.dmp.”

  ```
  sp_deletesmobj "syb_tsm", "svr1", "testdb", "obj1.dmp",
      "*", "may 20, 2009 10:51:43:866am"
  ```

- **Example 3 –** Removes all backup objects of the “testdb” database created by “svr1” of **dump database** type before May 21, 2009, 10:51:43:866 a.m.

```
sp_deletesmobj "syb_tsm", "svr1", "’testdb", "*", "DB",
    "may 21, 2009 10:51:43:866am"
```

- **Example 4 –** Removes all backup objects of “testdb” created by “svr1” of **dump transaction** type before May 20, 2009, 10:51:43:866 a.m.

```
sp_deletesmobj "syb_tsm", "svr1", "testdb",
    "/tmp/obj1.dmp", "*", "XACT", "may 21, 2009
    10:51:43:866am"
```

Usage

See also *Using Backup Server with IBM Tivoli Storage Manager*.

Permissions

The permission checks for **sp_dbremap** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `dump any database` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_rol**e or **oper_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• Roles – current active roles. • Keywords or options – NULL. • Previous value – NULL. • Current value – NULL. • Other information – all input parameters. • Proxy information – original login name, if **set proxy** in effect.

See also

- *sp_querysmobj* on page 595

sp_depends

Displays information about database object dependencies—the views, triggers, user-defined functions, procedures, and predicates—in the database that depend on a specified table or view, the tables and views in the database on which the specified view, trigger, procedure, or predicate depends, and multiple triggers associated with a table. Predicates cannot be granted in a view.

Also displays information about table column dependencies—the indexes, defaults, check constraints, rules, precomputed result sets, referential integrity constraints, and predicates—defined in either the column specified, if *column_name* is provided, or on all the columns in the table, if *column_name* is not provided.

Syntax

```
sp_depends objname[, column_name]
```

Parameters

- ***objname*** **–** is the name of the table, view, Transact-SQL stored procedure, SQLJ stored procedure, SQLJ function, 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.
- ***column_name*** **–** is the name of the column to be examined for dependencies.

Examples

- **Objects Dependent on a Table –** Lists the database objects that depend on the table `sysobjects`:

  ```
  sp_depends sysobjects
  ```

- **Objects Dependent on a View –** Lists the database objects that depend on the `titleview` view, and the database objects on which the `titleview` view depends:

  ```
  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
  ```

- **Objects Dependent on a Specific Table –** 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:

  ```
  sp_depends "mary.titles"
  ```

- **Precomputed Result Sets –** The following examples assume that `prs1` and `view1` are created with the following dependency structure:

  - `prs1` is defined on base table `tab1` (with unique constraint on column `c1`) and `view1` is defined on prs1
  - `prs1` is configured for immediate refresh

  This example displays the precomputed result sets that include dependencies for column `c1`:

  ```
  sp_depends prs1,c1
  Things the object references in the current database.
  object       type           updated    selected
  ------------ -------------- ---------- ---------
  dbo.tab1     user table     no         no

  Things inside the current database that reference the object.
  object             type
  ------------------ --------
  dbo.view1          view

  Dependent objects that reference column c1.
  Columns referenced in stored procedures, views or triggers are
  not
  included in this report.
  Type  Property   Object Names or Column Names Also see/Use
  command
  ----- ---------- ----------------------------
  --------------------
  index constraint prs1_10240036482 (c1)        sp_helpindex,
                                                drop index,
                                                sp_helpconstraint,
                                                alter table drop
                                                constraint
  ```

  ```
  sp_depends prs1,c1
  Things the object references in the current database.
  object       type           updated    selected
  ------------ -------------- ---------- ---------
  dbo.tab1     user table     no         no

  Things inside the current database that reference the object.
  object             type
  ------------------ --------
  dbo.view1          view

  Dependent objects that reference column c1.
  Columns referenced in stored procedures, views or triggers are
  not
  ```

```
included in this report.
Type  Property   Object Names or Column Names Also see/Use
command
----- ---------- ----------------------------
--------------------
index constraint prrs1_10240036482 (c1)        sp_helpindex,
                                              drop index,
                                              sp_helpconstraint,
                                              alter table drop
                                              constraint
```

- **Dependencies Between Predicate and Table –** Displays the dependencies between predicate `pred1` and any tables it references:

```
sp_depends pred1
```

```
Things the object references in the current database.
object           type                 updated        selected
---------------- -------------------- --------------
----------------
dbo.tab1         user table           no             no
dbo.tab2         user table           no             no
```

- **Dependencies Between Predicate, Table, and Column –** Displays the dependencies between predicates and table `tab1` and column `col1`:

```
sp_depends tab1, col1
```

```
Things inside the current database that reference the object.
object             type
------------------ ------------------
dbo.pred1          predicate
Dependent objects that reference column col1.
Columns referenced in stored procedures, views or triggers are not
included in this report.
Type                 Property
    Object Names or Column Names
       Also see/Use command
-------------------- --------------------
    -----------------------------------------------------------------
----
       ------------------------------------------------------
permission           permission
    column permission
       sp_helprotect, grant/revoke
```

- **Column-Level Dependencies –** Shows the column-level dependencies for all columns of the `sysobjects` table:

```
sp_depends sysobjects

Things inside the current database that reference the object.
object                                    type
----------------------------------------- ----------------
dbo.sp_dbupgrade                          stored procedure
dbo.sp_procxmode                          stored procedure

Dependent objects that reference all columns in the table. Use
```

```
sp_depends
on each column to get more information.
Columns referenced in stored procedures, views or triggers are not
included
in this report.

Column            Type         Object Names or Column Names
----------------------- ------------
------------------------------
cache             permission   column permission
ckfirst           permission   column permission
crdate            permission   column permission
deltrig           permission   column permission
expdate           permission   column permission
id                index        sysobjects (id)
id              logical RI   From syscolumns (id) To sysobjects (id)
id                logical RI   From syscomments (id) To sysobjects
(id)
id              logical RI   From sysdepends (id) To sysobjects (id)
id              logical RI   From sysindexes (id) To sysobjects (id)
id              logical RI   From syskeys (depid) To sysobjects (id)
id                logical RI   From syskeys (id) To sysobjects (id)
id               logical RI   From sysobjects (id) To sysprocedures
(id)
id                logical RI   From sysobjects (id) To sysprotects
(id)
id                logical RI   sysobjects (id)
id                permission   column permission
indexdel          permission   column permission
instrig           permission   column permission
loginame          permission   column permission
name              index        ncsysobjects (name, uid)
name              permission   column permission
objspare          permission   column permission
schemacnt         permission   column permission
seltrig           permission   column permission
sysstat           permission   column permission
sysstat2          permission   column permission
type              permission   column permission
uid               index        ncsysobjects (name, uid)
uid               logical RI   From sysobjects (uid) To sysusers
(uid)
uid               permission   column permission
updtrig           permission   column permission
userstat          permission   column permission
versionts         permission   column permission
```

- **Detailed Column-Level Dependencies –** Shows more details about the column-level dependencies for the `id` column of the `sysobjects` table:

```
sp_depends sysobjects, id

Things inside the current database that reference the object.
object                                              type
------------------------------------              -------------
dbo.sp_dbupgrade                                  stored procedure
dbo.sp_procxmode                                  stored procedure
```

```
Dependent objects that reference column id.
Columns referenced in stored procedures, views or triggers are not
included
in this report.
Type          Property     Object Names or Column Names
                           Also see/Use command
----------   ---------     ----------------------------------
                           ----------------------------------
index         index       sysobjects (id)
                          sp_helpindex, drop index,
                          sp_helpconstraint, alter table drop
constraint
logical RI    primary     sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    foreign     From syskeys (id) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From syscolumns (id) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From sysdepends (id) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From sysindexes (id) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From syskeys (depid) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common     From syscomments (id) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common     From sysobjects (id) To sysprotects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From sysobjects (id) To sysprocedures
(id)
                          sp_helpkey, sp_dropkey
permission    permission  column permission
                          sp_helprotect, grant/revoke
```

- **Column-Level Dependencies for All Columns –** Shows the column-level dependencies for all columns of the user-created table, `titles`:

```
1> sp_depends titlesThings inside the current database that
reference the object.
object                               type
-----------------------------------  ---------------
dbo.deltitle                          trigger
dbo.history_proc                      stored procedure
dbo.title_proc                        stored procedure
dbo.titleid_proc                      stored procedure
dbo.titleview                         view
dbo.totalsales_trig                   trigger

Dependent objects that reference all columns in the table. Use
sp_depends
on each column to get more information.
Columns referenced in stored procedures, views or triggers are not
included
in this report.

Column     Type         Object Names or Column Names
------     -----        ----------------------------------------
```

```
pub_id     logical RI   From titles (pub_id) To publishers (pub_id)
pubdate     default      datedflt
title       index        titleind (title)
title       statistics   (title)
title_id    index        titleidind (title_id)
title_id    logical RI   From roysched (title_id) To titles
(title_id)
title_id    logical RI   From salesdetail (title_id) To titles
(title_id)
title_id    logical RI   From titleauthor (title_id) To titles
(title_id)
title_id    logical RI   titles (title_id)
title_id    rule         title_idrule
title_id    statistics   (title_id)
type        default      typedflt
```

- **Column-Level Dependencies for a Specific Column –** Shows more details about the column-level dependencies for the `pub_id` column of the user-created `titles` table:

```
sp_depends titles, pub_id

Things inside the current database that reference the object.
object                                  type
------------------------------------    ----------------
dbo.deltitle                            trigger
dbo.history_proc                        stored procedure
dbo.title_proc                          stored procedure
dbo.titleid_proc                        stored procedure
dbo.titleview                           view
dbo.totalsales_trig                     trigger
Dependent objects that reference column pub_id.
Columns referenced in stored procedures, views or triggers are not
included in this report.
Type          Property    Object Names or Column Names
                          Also see/Use command
----------    ---------   ---------------------
                          ----------------------------
logical RI    foreign     From titles (pub_id) To publishers
(pub_id)
                          sp_helpkey, sp_dropkey
```

Usage

- 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.
- Before you modify or drop a column, use **sp_depends** to determine if the table contains any dependent objects that could be affected by the modification. For example, if you modify a column to use a new datatype, objects tied to the table may need to be redefined to be consistent with the column's new datatype.

- 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.
- 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.

sp_depends follows these SAP ASE 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*.

See also **create procedure**, **create table**, **create view**, **execute** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_depends**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_help* on page 358

Java Methods

SQLJ functions and SQLJ stored procedures are Java methods wrapped in SQL wrappers.

- SQLJ functions and SQLJ stored procedures are database objects for which you can list dependencies. The only dependencies of SQLJ stored procedures and SQLJ functions are Java classes.
- If *objname* is a SQLJ stored procedure or SQLJ function, **sp_depends** lists the Java class in the routine's external name declared in the create statement, not classes specified as the return type or datatypes in the parameter list.
- SQLJ stored procedures and SQLJ functions can be listed as dependencies of other database objects.

See *Java in Adaptive Server Enterprise* for more information.

sp_deviceattr

(UNIX platforms only) Changes the device parameter settings of an existing database device file.

Syntax

```
sp_deviceattr  logicalname, optname, optvalue
```

Parameters

- ***logicalname* –** is the logical name of an existing database device. The device can be stored on either an operating system file or a raw partition, but the **dsync** setting is ignored for raw partitions.
- ***optname* –** name of the attribute to change. Valid values are **directio** or **dsync**:
  - **directio** – enables the SAP ASE server to write directly to disk, bypassing the operating system's buffer system. The SAP ASE server passes the device options to Backup

Server, which enables Backup Server to access the database device with the appropriate directio option.

- **dsync** – enables updates to the device take place directly on the storage media, or are buffered by the UNIX file system

Note: The **directio** and **dsync** options are mutually exclusive; you cannot specify “true” for both at the same time.

- ***optvalue* –** can be either “true” or “false.”

Examples

- **Example 1 –** Sets **dsync** on for the device named “file_device1”:

```
sp_deviceattr file_device1, dsync, true
```

Usage

There are additional considerations when using **sp_deviceattr**:

- For database devices stored on UNIX files, **dsync** determines whether updates to the device take place directly on the storage media, or are buffered by the UNIX file system.

  When **dsync** is on, writes to the database device occur directly to the physical storage media, and the SAP ASE server can recover data on the device in the event of a system failure.

  When **dsync** is off, writes to the database device may be buffered by the UNIX file system. The UNIX file system may mark an update as being completed, even though the physical media has not yet been modified. In the event of a system failure, there is no guarantee that requests to update data have ever taken place on the physical media, and the SAP ASE server may be unable to recover the database.
- (UNIX only) On raw devices, you cannot set **directio** or **dsync** via the **sp_deviceattr** stored procedure to true.

  Note: For HPUX, only the **dsync** option applies.

  Doing so returns a message such as:

  ```
  You cannot set option dsync for raw device 'dev/raw/raw235'
  ```

  or

  ```
  You cannot set attribute dsync for raw device 'myrawdisk1'
  ```
- After using **sp_deviceattr** to change the **dsync** or **directio** setting, you must restart the SAP ASE server before the change takes affect.
- **sp_deviceattr** displays a warning message if the **dsync** option is disabled for a database device file.

  directio setting, you must restart the SAP ASE server before the change takes affect.

- **dsync** is always on for the master device file. You cannot change the **dsync** setting for a master device file with **sp_deviceattr**.Therefore, you cannot set the **directio** option for the master device.
- Turn off the **dsync** value only when the databases on the device does not need to be recovered after a system failure. For example, you may consider turning **dsync** off for a device that stores only the `tempdb` database.
- The SAP ASE server ignores the **dsync** setting for devices stored on raw partitions; updates to those devices are never buffered, regardless of the **dsync** setting.
- **dsync** is not used on the Windows platform.

Permissions

The permission checks for **sp_deviceattr** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage disk` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_helpdevice* on page 402

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.

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 SAP ASE is first installed).

Examples

- **Example 1 –** The master device is no longer used by **create database** or **alter database** for default storage of a database:

  ```
  sp_diskdefault master, defaultoff
  ```

Usage

There are additional considerations when using **sp_diskdefault**:

- 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 SAP ASE, 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**.

See also **alter database**, **create database**, **disk init** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_diskdefault** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage disk` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_helpdevice* on page 402

sp_displayaudit

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 are 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 are 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. The parameters and their values are:

  - **procedure –** Procedure or trigger name
  - **object –** Table or view name
  - **login –** User login
  - **database –** Database name
  - **global –** Global audit option
  - **default_object –** Database name
  - **default_procedure –** Database name

  You cannot specify a value for *name* unless you first specify an object type parameter.

Examples

- **Example 1 –** Displays the status of each category and all auditing options when you do not specify a parameter:

  ```
  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
keycustodian_role              off
navigator_role                 off
oper_role                      off
replication_role               off
rpc                            off
sa_role                        off
security                       off
sso_role                       off
```

- **Example 2 –** Displays the status of all procedure audit options when you do not specify a procedure name:

```
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
```

- **Example 3 –** Displays only the status of the procedure when you specify a name for a procedure:

```
sp_displayaudit "procedure", "sp_who"
```

```
Procedure/Trigger Audit Option    Value Database
----------------- --------------- ----- ----------------------
dbo.sp_who        exec_procedure  on    sybsystemprocs
```

- **Example 4 –** Displays the status of all global audit options when you do not specify a global audit option:

```
sp_displayaudit "global"
```

```
Option Name                    Value
------------------------------ ------------------------------
adhoc                          off
dbcc                           off
disk                           off
errors                         off
```

```
login                         off
logout                        off
keycustodian_role             off
navigator_role                off
oper_role                     off
replication_role              off
rpc                           off
sa_role                       off
security                      off
sso_role                      off
```

Usage

The valid auditing options for each parameter are:

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**, **encryption_key**, **func_dbaccess**, **grant**, **load**, **revoke**, **setuser**, **truncate**, **unbind**
global	**adhoc**, **dbcc**, **disk**, **errors**, **login**, **logout**, **navigator_role**, **oper_role**, **replication_role**, **rpc**, **keycustodian_role**, **sa_role**, **security**, **sso_role**
default_object	**delete**, **func_obj_access**, **insert**, **reference**, **select**, **update**
default_procedure	**exec_procedure**, **exec_trigger**

See also:

- See the *System Administration Guide* for information on setting up auditing.
- **bcp** in the *Utility Guide*

Permissions

The permission checks for **sp_displayaudit** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage auditing` privilege.
Disabled	With granular permissions disabled, you must be a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_audit* on page 63

sp_displaylevel

Sets or shows which SAP ASE configuration parameters appear in **sp_configure** output.

Syntax

```
sp_displaylevel [loginame [, level]]
```

Parameters

- ***loginame*** **–** is the SAP ASE login of the user for whom you want to set or show the display level.
- ***level*** **–** sets the display level to one of the following:
  - “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

- **Example 1 –** Shows the current display level for the user who invoked **sp_displaylevel**:

  ```
  sp_displaylevel
  ```

  ```
  The current display level for login 'sa' is 'comprehensive'.
  ```

- **Example 2 –** Shows the current display level for the user “jerry”:

  ```
  sp_displaylevel jerry
  ```

  ```
  The current display level for login 'jerry' is 'intermediate'.
  ```

- **Example 3 –** Sets the display level to “comprehensive” for the user “jerry”:

  ```
  sp_displaylevel jerry, comprehensive
  ```

  ```
  The display level for login 'jerry' has been changed to
  'comprehensive'.
  ```

Usage

See the *System Administration Guide* for details about display levels and configuration parameters.

Permissions

The permission checks for **sp_displaylevel** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server configuration` privilege. Any user can execute **sp_displaylevel** to set and show their own configuration parameters.
Disabled	With granular permissions disabled, you must be a user with **sso_role**. Any user can execute **sp_displaylevel** to set and show their own configuration parameters.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_configure* on page 167

sp_displaylogin

Displays information about a login account. By using a wildcard expression (%), you can also obtain information about matching logins. Also displays the encryption versions of the login password stored on disk.

Syntax

```
sp_displaylogin ['user_id' | '[loginame | wildcard]'
```

Parameters

- ***user_id*** **–** is the server user ID.
- ***loginame*** **–** is the user login account about which you want information. You must be a system security officer or system administrator to get information about someone else's login account.
- ***wildcard*** **–** is the wildcard expression you use to obtain information about login accounts.

Examples

- **Display Information About Server Login Account –** The password expiration is set to 0, indicating the password never expires:

```
1> sp_displaylogin 'sa'
2> go
```

```
Suid: 121
Loginame: sa
Fullname:
Configured Authorization:
        sa_role (default ON)
        sso_role (default ON)
        oper_role (default ON)
        sybase_ts_role (default ON)
Locked: NO
```

```
Date of Last Password Change: Aug 10 2010 11:17AM
Password expiration interval: 0
Password expired: NO
Minimum password length: 6
Maximum failed logins: 0
Current failed login attempts:
Login password encryption: SYB-PROP, SHA-256
Last login date : Aug 17 2010 5:55PM
Login Profile :emp_lp
```

- **Display Information About Login Account "susanne" –** The information displayed varies, depending on the role of the user executing **sp_displaylogin**. There is not password expiration set for user "susanne", so the password does not expire.

```
sp_displaylogin susanne
```

```
Suid: 12
Loginame: susanne
Fullname:
Configured Authorization:
    supervisor (default OFF)
Locked: NO
Date of Last Password Change: July 26 2010 10:42AM
Login Profile :emp_lp
```

- **Display Login Security-Related Parameters Configured for a Login –** Displays the login security-related parameters configured for a login, as well as a specified authentication mechanism. The password expires on November 29, 2010 at 3:46PM, and expires five days later, on December 5, 2010 at 3:46PM.

```
sp_displaylogin joe
```

```
Suid: 294
Loginame: joe
Fullname: Joseph Resu
Configured Authorization:
    intern_role (default OFF)
Locked: NO
Date of Last Password Change: Nov 24 2010 3:46PM
Password expiration interval : 5
Password expired : NO
Minimum password length:4
Maximum failed logins : 10
Current failed logins : 3
Login password encryption: SHA-256
Login Profile :emp_lp
```

- **Display Information About Login Account With Server User ID 1 –**

```
sp_displaylogin '1'
-------------
Suid: 1
Loginame: sa
Fullname:
Configured Authorization:
       sa_role (default ON)
       sso_role (default ON)
       oper_role (default ON)
       sybase_ts_role (default ON)
Locked: NO
Date of Last Password Change: Dec 18 2010
Password expiration interval: 0
Login Profile :emp_lp
```

- **Use Wildcard to Indicate Any Server Login Account –** You can use a wildcard to indicate any server login account, as opposed to your own server login account.

```
sp_displaylogin '%'
--------------------
```

```
Suid  Loginname  Fullname  Locked  Date of Last Password Change
Password expiration interval  Password expired  Minimum password
length  Maximum failed logins  Current failed login
attempts  Authenticate with  Login Profile  Configured
Authorization---- --------- --------- ---------------
---------------- ----------------- --------
---------------------------
---------------------------------------
--------------------------- ----------------------------------
--------------------------------
----------------------------------------
----------------------------
-----------------------------------------------------------------
---------------------------

2 probe NULL sybsystemdb NULL NULL NO Jan  8 2010 7:13AM 1 NO 6 0 0
NONE
NULL
1 sa NULL master NULL NULL NO Jan  8 2010 6:46AM 1 NO 6 0 0 NONE
```

- **Display Encrypted and Stored On-disk Login Password –** The on-disk login password is encrypted and stored, using both the old Sybase proprietary encryption algorithm and the SHA-256 algorithm:

```
1> sp_displaylogin 'mylogin'
2> go
```

```
Suid: 121
Loginame: mylogin
Fullname:
Configured Authorization:
        sa_role (default ON)
        sso_role (default ON)
        oper_role (default ON)
        sybase_ts_role (default ON)
Locked: NO
Date of Last Password Change: Aug 10 2006 11:17AM
Password expiration interval: 0
Password expired: NO
Minimum password length: 6
Maximum failed logins: 0
Current failed login attempts:
Login password encryption: SYB-PROP, SHA-256
Last login date : Aug 17 2010 5:55PM
```

```
Login Profile :emp_lp

(return status = 0)
```

When the login password is stored on disk using the SHA-256 algorithm only, the output of **sp_displaylogin** has the line “**Login password encryption: SHA-256**”:

```
1> sp_displaylogin 'mylogin'
2> go
```

```
Suid: 121
Loginame: mylogin
 ...
Authenticate with: NONE
Login password encryption: SHA-256
Last login date : Aug 17 2010 5:55PM
Login Profile :emp_lp

(return status = 0)
```

When a login has not occurred after upgrade from SAP ASE versions earlier than 15.0.2, the previous style of encryption is still in place, and the output of **sp_displaylogin** has the line “**Login password encryption: SYB-PROP**”:

```
1> sp_displaylogin 'mylogin'
2> go
```

```
Suid: 121
Loginame: mylogin
 ...
Authenticate with: NONE
Login password encryption: SYB-PROP
Last login date : Aug 17 2006 5:55PM
(return status = 0)
```

When a login has been locked, **sp_displaylogin** shows the date, reason, and login that locked the account. The **lastlogindate** value is also displayed:

```
1> sp_displaylogin 'mylogin'
2> go
```

```
Suid: 121
Loginame: mylogin
Fullname:
Configured Authorization:
        sa_role (default ON)
        sso_role (default ON)
        oper_role (default ON)
        sybase_ts_role (default ON)
Locked: YES
        Date when locked: Aug 18 2010 9:15AM
        Reason: Account locked by SAP ASE due to failed login
attempts reaching max failed logins.
        Locking suid: mylogin
Date of Last Password Change: Aug 10 2010 11:17AM
Password expiration interval: 0
Password expired: NO
Minimum password length: 6
Maximum failed logins: 3
Current failed login attempts: 3
Login password encryption: SYB-PROP, SHA-256
Last login date : Aug 17 2010 5:55PM
Login Profile :emp_lp
(return status = 0)
```

- **Display Encryption Versions Used for a Login –** Displays the encryption versions used for a login; this output includes information about the on-disk login password encryption the SAP ASE server uses:

  ```
  sp_displaylogin sa
  go
  ```

  ```
  Suid: 1
  Loginame: sa
  Fullname:
  Configured Authorization:
      sa_role (default ON)
      sso_role (default ON)
      oper_role (default ON)
      sybase_ts_role (default ON)
  Locked: NO
  Date of Last Password Change: Mar  8 2010 3:04PM
  Password expiration interval: 0
  Password expired: NO
  Minimum password length: 6
  Maximum failed logins: 0
  Current failed login attempts:
  Login Password Encryption: SHA-256
  Login Profile :emp_lp
  ```

  If the SAP ASE server uses encryption algorithms from SAP ASE versions earlier than 15.0.2 or the current release during a downgrade period, **sp_displaylogin** displays the earlier Sybase proprietary encryption algorithm and the new algorithm, SHA-256:

  ```
  Login password encryption: SYB-PROP, SHA-256
  ```

- **Display Login and Password Policy Options of Current Login Account –**

  ```
  sp_displaylogin
  go
  ```

  ```
  Suid: 5
  Loginame: tammi
  Fullname:
  Configured Authorization:
      sa_role (default ON)
      sso_role (default ON)
      oper_role (default ON)
      sybase_ts_role (default ON)
  Locked: NO
  Date of Last Password Change: Mar  8 2010 3:04PM
  Password expiration interval: 0
  Password expired: NO
  Minimum password length: 6
  ```

```
Maximum failed logins: 0
Current failed login attempts:
Authenticate with: ANY
Login Password Encryption: SHA-256
```

```
Exempt inactive lock: 0
```

```
Login Profile: emp_lp
```

- **Display Login Account for User with Suid 56 –**

  ```
  sp_displaylogin '56'
  ```

- **Display Login Account Information for All Users With Logins Begin With “st” –**

  ```
  sp_displaylogin 'st%'
  ```

Usage

There are additional considerations when using **sp_displaylogin**:

- The **sp_passwordpolicy** security options are taken into consideration when displaying login information related to password expiration, maximum failed logins, and password length.
- **sp_displaylogin** displays the encryption version(s) used for a login. For example, when both old and new encryption is used during the password downgrade period, the output of **sp_displaylogin** has the new line “**Password encryption**.”
- **sp_displaylogin** displays configured roles, so even if you have made a role inactive with the **set** command, it is displayed.
- Login triggers associated with the login in question are specified through a login profile. For more information, see *Managing Login Accounts and Login Profiles* in the *System Administration Guide*.
- 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, login profile, full name, any roles that have been granted to you, date of last password change, 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.

Permissions

The permission checks for **sp_displaylogin** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any login` privilege or `manage sever` privilege. Any user can execute **sp_displaylogin** to display information about their own login account.
Disabled	With granular permissions disabled, you must be a user with **sa_role** or **sso_role**. Any user can execute **sp_displaylogin** to display information about their own login account.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_activeroles* on page 4
- *sp_displayroles* on page 240
- *sp_helprotect* on page 425

sp_displayroles

Displays all roles granted to another role, login or login profile, the entire hierarchy tree of roles in table format, and other login security-related parameters configured for the specified role, including the date when the role was locked, its reason, and the login server user ID (suid) that locked the role. For password-protected roles, also displays the role password encryption version.

Displays roles granted to logins through an associated login profile. A `grantee` column in the output displays the login profile name as applicable. This column is only displayed if the login has an associated login profile with roles granted to the login. The login profile association could be direct or through a default login profile.

Syntax

```
sp_displayroles [grantee_name [, mode]]
```

Parameters

- ***grantee_name* –** is the login name of a user or login profile name with roles that you want information about, or the name of a role you want information about.
- ***mode* –** is one of the following:
  - **expand_up** – shows the role hierarchy tree for the parent levels
  - **expand_down** – shows the role hierarchy tree for the child levels
  - **display_info** – shows the login security-related parameters configured for the specified role

Examples

- **Example 1 –** Displays all roles granted to the user issuing the command:

  ```
  sp_displayroles
  ```

  ```
  Role Name
  ------------------------------
  supervisor_role
  ```

- **Example 2 –** Displays all roles granted to `supervisor_role`:

  ```
  sp_displayroles "supervisor_role"
  ```

  ```
  Role Name
  ------------------------------
  clerk
  ```

- **Example 3 –** Displays the roles granted to login “susanne” and the roles below it in the hierarchy:

  ```
  sp_displayroles susanne, expand_down
  ```

  ```
  Role Name          Parent Role Name       Level
  ----------------- ---------------------- ------
  supervisor_role   NULL                     1
  clerk_role       supervisor_role          2
  ```

- **Example 4 –** Displays the roles granted to `intern_role` and the roles above it in the hierarchy:

  ```
  sp_displayroles "intern_role", expand_up
  ```

- **Example 5 –** Shows the login security-related parameters configured for the specified role:

```
sp_displayroles physician_role, "display_info"
```

```
Role name = physician_role
Locked : YES
    Date when locked: Jul 14 2007 9:15AM
    Reason: Role locked by SAP ASE due to failed login
    attempts reaching max failed logins.
    Locking suid: dr_john
Date of Last Password Change : Oct 31 1999 3:33PM
Password expiration interval = 5
Password expired : NO
Minimum password length = 4
Maximum failed logins = 10
Current failed logins = 3
Password encryption version: SHA-256
```

- **Example 6 –** Displays the roles granted to login “tom,” which is associated with the login profile named “sec_profile”:

```
grant role sec_role to sec_profile

create login tom with password C0mp13x login profile sec_profile

grant role emp_role to tom
go
sp_displayroles tom
go
```

```
Role Name                          Grantee
-------------------------------------------------
emp_role                           tom
sec_role                           sec_profile
```

Usage

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.

The `Grantee` column displays only when a login has an associated login profile, or the default login profile is applicable to the login with role(s) granted to it.

See also:

- **alter role**, **create role**, **drop role**, **grant**, **revoke**, **set** in *Reference Manual: Commands*
- *User-Defined Login Security* in the *System Administration Guide* for more information.

Permissions

The permission checks for **sp_displayroles** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage roles` or `manage server` privilege. Any can execute **sp_displayroles** to see the roles granted to themselves.
Disabled	With granular permissions disabled, you must be a user with **sso_role**. Any can execute **sp_displayroles** to see the roles granted to themselves.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_activeroles* on page 4
- *sp_displaylogin* on page 234
- *sp_helpprotect* on page 425

sp_downgrade

(`master` database only) Validates readiness for downgrade to a version of SAP ASE earlier than the current version you are running. Also downgrades the system catalog changes that were modified with the current version of SAP ASE.

Syntax

```
sp_downgrade @cmd = {'prepare' | 'downgrade' | 'help',}
    @toversion = 'n'[, @verbose = 0 | 1][, @override = 0 | 1]
```

Parameters

- **prepare –** is used first to validate readiness of SAP ASE 15.0.2 for downgrade.
- **downgrade –** is used after **prepare** parameter when ready to proceed with the act of downgrading to a previously installed 15.x version of SAP ASE. Server must be in single user mode. (started with **-m** option)
- **@toversion –** can be 15.0, 15.0.1, or 15.0.2. The format can be written as “150” or “15.0”, “1501” or “15.0.1”
- **@verbose –** specifies verbosity. Valid options are 0 (for no) or 1 (for yes).
- **@override –** specifies whether to skip databases that are not writable at this time. Valid options are 0 (for no) or 1 (for yes).

Examples

- **Example 1 –** Shows output from running **sp_downgrade**.

```
00:0006:00000:00006:2011/06/29 02:16:44.35 server  Preparing ASE
downgrade from
15.7.0.0 to 15.5.0.0.
00:0006:00000:00006:2011/06/29 02:16:44.37 server  Starting
downgrading ASE.
00:0006:00000:00006:2011/06/29 02:16:44.37 server  Downgrade :
Marking stored
procedures to be recreated from text.
00:0006:00000:00006:2011/06/29 02:16:45.34 server  Downgrade :
Removing full logging
modes from sysattributes.
00:0006:00000:00006:2011/06/29 02:16:45.34 server  Downgrade :
Downgrading data-only
locked table rows.
00:0006:00000:00006:2011/06/29 02:16:45.34 server  Downgrade :
Removing full logging
modes from sysattributes.
00:0006:00000:00006:2011/06/29 02:16:45.34 server  Downgrade :
Removing column
sysoptions.number.
00:0006:00000:00006:2011/06/29 02:16:45.34 server  Downgrade :
Removing srvprincipal
column from sysservers system table
00:0006:00000:00006:2011/06/29 02:16:45.34 server  Downgrade :
Removing 'automatic
master key access' configuration parameter.
00:0006:00000:00006:2011/06/29 02:16:45.35 server  Downgrade :
Removing DualControl
sysattribute rows
00:0006:00000:00006:2011/06/29 02:16:45.35 server  Downgrade :
Downgrading
sysattributes system table.
00:0006:00000:00006:2011/06/29 02:16:45.37 server  Downgrade :
Downgrading
syscomments system table.
00:0006:00000:00006:2011/06/29 02:16:45.42 server  Downgrade :
Truncated role
```

```
password, locked role and removed columns locksuid, lockreason,
lockdate from
syssrvroles
00:0006:00000:00006:2011/06/29 02:16:45.43 server  Downgrade :
Removing catalog
changes for RSA Keypair Regeneration Period and Login Profile
00:0006:00000:00006:2011/06/29 02:16:45.43 server  Downgrade :
Turning on database
downgrade indicator.
00:0006:00000:00006:2011/06/29 02:16:45.43 server  Downgrade :
Resetting database
version indicator.
00:0006:00000:00006:2011/06/29 02:16:45.43 server  ASE downgrade
completed.
```

- **Example 2 –** Checks the databases for downgrade readiness:

```
1> sp_downgrade 'prepare','15.5',1
2> go
```

```
Downgrade from 15.7.0.0 to 15.5.0.0 (command: 'prepare')

Checking databases for downgrade readiness.

There are no errors which involve encrypted columns.
sp_downgrade 'prepare' completed.
(return status = 0)
```

- **Example 3 –** Downgrades SAP ASE from version 15.7 to 15.5:

```
1> sp_downgrade 'downgrade','15.5',1
2> go
```

```
Downgrade from 15.7.0.0 to 15.5.0.0 (command: 'downgrade')

Checking databases for downgrade readiness.

There are no errors which involve encrypted columns.

Executing downgrade step 2 [dbcc markprocs(@dbid)] for :
- Database: master (dbid: 1)
sql comman is: dbcc markprocs(@dbid)

DBCC execution completed. If DBCC printed error messages, contact
a user with System
Administrator (SA) role.
- Database: tempdb (dbid: 2)
sql comman is: dbcc markprocs(@dbid)

DBCC execution completed. If DBCC printed error messages, contact
a user with System
Administrator (SA) role.
- Database: model (dbid: 3)
sql comman is: dbcc markprocs(@dbid)

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

```
- Database: sybsystemdb (dbid: 31513)
sql comman is: dbcc markprocs(@dbid)

DBCC execution completed. If DBCC printed error messages, contact
a user with System
Administrator (SA) role.
- Database: sybsystemprocs (dbid: 31514)
sql comman is: dbcc markprocs(@dbid)

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

Executing downgrade step 17 [delete sysattributes where class =
38] for :
- Database: master (dbid: 1)
sql comman is: delete sysattributes where class = 38


Executing downgrade step 18 [declare @ret int select @ret =
dol_downgrade_check(':DBNAME:', @toversid) print
"Database :DBNAME: table downgrade
status: %1!", @ret if @ret != 0 begin   print "*** Tables in
database ':DBNAME:' cannot
be downgraded."   print "*** See the server error log for
details."   select
@exec_error_count = @exec_error_count + 1 end] for :
- Database: master (dbid: 1)
sql comman is: declare @ret int select @ret =
dol_downgrade_check('master', @toversid)
print "Database master table downgrade status: %1!", @ret if
@ret != 0 begin   print
"*** Tables in database 'master' cannot be downgraded."   print
"*** See the server
error log for details."   select @exec_error_count =
@exec_error_count + 1 end

Database master table downgrade status: 0
- Database: tempdb (dbid: 2)

sql comman is: declare @ret int select @ret =
dol_downgrade_check('tempdb', @toversid)
print "Database tempdb table downgrade status: %1!", @ret if
@ret != 0 begin   print
"*** Tables in database 'tempdb' cannot be downgraded."   print
"*** See the server
error log for details."   select @exec_error_count =
@exec_error_count + 1 end

Database tempdb table downgrade status: 0
- Database: model (dbid: 3)
sql comman is: declare @ret int select @ret =
dol_downgrade_check('model', @toversid)
print "Database model table downgrade status: %1!", @ret if @ret !
= 0 begin   print
"*** Tables in database 'model' cannot be downgraded."   print
```

```
"*** See the server
error log for details."   select @exec_error_count =
@exec_error_count + 1 end

Database model table downgrade status: 0
- Database: sybsystemdb (dbid: 31513)
sql comman is: declare @ret int select @ret =
dol_downgrade_check('sybsystemdb',
@toversid) print "Database sybsystemdb table downgrade status:
%1!", @ret if @ret !=
0 begin   print "*** Tables in database 'sybsystemdb' cannot be
downgraded."   print
"*** See the server error log for details."   select
@exec_error_count =
@exec_error_count + 1 end

Database sybsystemdb table downgrade status: 0
- Database: sybsystemprocs (dbid: 31514)
sql comman is: declare @ret int select @ret =
dol_downgrade_check('sybsystemprocs',
@toversid) print "Database sybsystemprocs table downgrade status:
%1!", @ret if @ret
!= 0 begin   print "*** Tables in database 'sybsystemprocs' cannot
be downgraded."
print "*** See the server error log for details."   select
@exec_error_count =
@exec_error_count + 1 end

Database sybsystemprocs table downgrade status: 0

Executing downgrade step 19 [delete sysattributes where class =
38] for :
- Database: master (dbid: 1)
sql comman is: delete sysattributes where class = 38


Executing downgrade step 20 [delete syscolumns where id =
object_id('sysoptions') and
name='number'] for :
- Database: master (dbid: 1)
sql comman is: delete syscolumns where id =
object_id('sysoptions') and name='number'


Executing downgrade step 21 [delete syscolumns where id =
object_id('sysservers') and
name = 'srvprincipal'] for :
- Database: master (dbid: 1)
sql comman is: delete syscolumns where id =
object_id('sysservers') and name = 'srvprincipal'


Executing downgrade step 22 [delete sysconfigures where config =
503] for :
- Database: master (dbid: 1)
sql comman is: delete sysconfigures where config = 503
```

```
Executing downgrade step 23 [delete sysattributes where class = 25
and attribute in
(2, 3)] for :
- Database: master (dbid: 1)
sql comman is: delete sysattributes where class = 25 and attribute
in (2, 3)


Executing downgrade step 24 [update :DBNAME:..sysattributes set
object_cinfo2 = null,
object_datetime = null where object_cinfo2 is not null or
object_datetime is not null
delete :DBNAME:..syscolumns where id = 21 and name in
('object_cinfo2',
'object_datetime')] for : - Database: master (dbid: 1)
sql comman is: update master..sysattributes set object_cinfo2 =
null, object_datetime
= null where object_cinfo2 is not null or object_datetime is not
null delete
master..syscolumns where id = 21 and name in ('object_cinfo2',
'object_datetime')

- Database: tempdb (dbid: 2)
sql comman is: update tempdb..sysattributes set object_cinfo2 =
null, object_datetime
= null where object_cinfo2 is not null or object_datetime is not
null delete
tempdb..syscolumns where id = 21 and name in ('object_cinfo2',
'object_datetime')

- Database: model (dbid: 3)
sql comman is: update model..sysattributes set object_cinfo2 =
null, object_datetime
= null where object_cinfo2 is not null or object_datetime is not
null delete
model..syscolumns where id = 21 and name in ('object_cinfo2',
'object_datetime')

- Database: sybsystemdb (dbid: 31513)
sql comman is: update sybsystemdb..sysattributes set object_cinfo2
= null,
object_datetime = null where object_cinfo2 is not null or
object_datetime is not null
delete sybsystemdb..syscolumns where id = 21 and name in
('object_cinfo2',
'object_datetime')

- Database: sybsystemprocs (dbid: 31514)
sql comman is: update sybsystemprocs..sysattributes set
object_cinfo2 = null,
object_datetime = null where object_cinfo2 is not null or
object_datetime is not null
delete sybsystemprocs..syscolumns where id = 21 and name in
('object_cinfo2',
'object_datetime')


Executing downgrade step 25 [update :DBNAME:..syscomments set
encrkeyid = null where
encrkeyid is not null delete:DBNAME:..syscolumns where id = 6 and
name = 'version'
delete :DBNAME:..syscolumns where id = 6 and name ='encrkeyid']
for :
- Database: master (dbid: 1)
sql comman is: update master..syscomments set encrkeyid = null
where encrkeyid is not
null delete master..syscolumns where id = 6 and name = 'version'
delete
master..syscolumns where id = 6 and name ='encrkeyid'

- Database: tempdb (dbid: 2)
sql comman is: update tempdb..syscomments set encrkeyid = null
where encrkeyid is not
null delete tempdb..syscolumns where id = 6 and name = 'version'
delete tempdb..syscolumns where id = 6 and name ='encrkeyid'

- Database: model (dbid: 3)
sql comman is: update model..syscomments set encrkeyid = null
where encrkeyid is not
null delete model..syscolumns where id = 6 and name = 'version'
delete
model..syscolumns where id = 6 and name ='encrkeyid'

- Database: sybsystemdb (dbid: 31513)
sql comman is: update sybsystemdb..syscomments set encrkeyid =
null where encrkeyid
is not null delete sybsystemdb..syscolumns where id = 6 and name =
'version' delete
sybsystemdb..syscolumns where id = 6 and name ='encrkeyid'

- Database: sybsystemprocs (dbid: 31514)
sql comman is: update sybsystemprocs..syscomments set encrkeyid =
null where encrkeyid
is not null delete sybsystemprocs..syscolumns where id = 6 and
name = 'version' delete
sybsystemprocs..syscolumns where id = 6 and name ='encrkeyid'


Executing downgrade step 26 [delete statistics
syssrvroles(password) if exists
(select 1 from syssrvroles where password is not null) begin print
"Truncating
password and locking following role(s)" select name from
syssrvroles where password
is not null update syssrvroles set password = null, status =
(status | @lockrole)
where password is not null end update syscolumns set length = 30
where id =
object_id('syssrvroles') and name = 'password' update syssrvroles
set locksuid = null,
lockreason = null, lockdate = null where locksuid is not null or
lockreason is not
null or lockdate is not null delete syscolumns where id =
object_id('syssrvroles')
and name in ('locksuid', 'lockreason', 'lockdate')] for :
- Database: master (dbid: 1)
sql comman is: delete statistics syssrvroles(password) if exists
(select 1 from
syssrvroles where password is not null) begin print "Truncating
password and locking
following role(s)" select name from syssrvroles where password is
not null update
syssrvroles set password = null, status = (status | @lockrole)
where password is not
null end update syscolumns set length = 30 where id =
object_id('syssrvroles') and
name = 'password' update syssrvroles set locksuid = null,
lockreason = null, lockdate
= null where locksuid is not null or lockreason is not null or
lockdate is not null
delete syscolumns where id = object_id('syssrvroles') and name in
('locksuid',
'lockreason', 'lockdate')

Truncating password and locking following role(s)
 name
 ------------------------------------------------------------
 doctor_role

Executing downgrade step 27 [delete sysattributes where class = 35
delete
sysattributes where class = 39 update syslogins set lpid = null,
crsuid = null where
lpid is not null or crsuid is not null delete syscolumns where id
=
object_id('syslogins') and name in ('lpid', 'crsuid') delete
syslogins where (status
& @lp_status) = @lp_status update syslogins set status = status &
~(@exempt_lock)
where (status & @exempt_lock) = @exempt_lock] for :
- Database: master (dbid: 1)
sql comman is: delete sysattributes where class = 35 delete
sysattributes where class
= 39 update syslogins set lpid = null, crsuid = null where lpid is
not null or crsuid
is not null delete syscolumns where id = object_id('syslogins')
and name in ('lpid',
'crsuid') delete syslogins where (status & @lp_status) =
@lp_status update syslogins
set status = status & ~(@exempt_lock) where (status &
@exempt_lock) = @exempt_lock


Executing downgrade step 998 [declare @d int, @stat4 int select
@stat4=convert(int,
dbinfo_get('master','status4')) select @d=dbinfo_update(1,
'status4', 32 | @stat4)]
for :
- Database: master (dbid: 1)
sql comman is: declare @d int, @stat4 int select
@stat4=convert(int,
dbinfo_get('master','status4')) select @d=dbinfo_update(1,
'status4', 32 | @stat4)


Executing downgrade step 999 [declare @d int select
@d=dbinfo_update(@dbid,
'ASEvers', 15500)] for :
- Database: master (dbid: 1)
sql comman is: declare @d int select @d=dbinfo_update(@dbid,
'ASEvers', 15500)

- Database: tempdb (dbid: 2)
sql comman is: declare @d int select @d=dbinfo_update(@dbid,
'ASEvers', 15500)

- Database: model (dbid: 3)
sql comman is: declare @d int select @d=dbinfo_update(@dbid,
'ASEvers', 15500)

- Database: sybsystemdb (dbid: 31513)
sql comman is: declare @d int select @d=dbinfo_update(@dbid,
'ASEvers', 15500)

- Database: sybsystemprocs (dbid: 31514)
sql comman is: declare @d int select @d=dbinfo_update(@dbid,
'ASEvers', 15500)

(return status = 0)
```

Usage

Use to revert to the previously installed SAP ASE 15.0.x release.

When downgrading to a version of SAP ASE earlier than 15.7, the **sp_downgrade** system procedure internally calls **sp_passwordpolicy [prepare | downgrade]** along with the SAP ASE version number to downgrade.

When you execute **sp_downgrade**, the SAP ASE server performs these tasks:

- Truncates role passwords and locks roles.
- Removes newly added attributes in `sysattributes` under class 35.
- Removes newly added class 35 in `sysattributes`.
- Removes the new `locksuid`, `lockreason`, and `lockdate` columns from `syssrvroles`.

Note: When you downgrade SAP ASE version 15.7 to a pre-15.0.2 version, both role and login passwords are downgraded. When downgrading to version 15.0.2, however, the SAP ASE server truncates and locks only role passwords.

For more information about downgrading role passwords, see the downgrade section of the installation guide for your platform.

Permissions

The permission checks for **sp_downgrade** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with **sa_serverprivs_role** and **sybase_ts_role**.
Disabled	With granular permissions disabled, you must be a user with **sso_role** and **sa_role** and **sybase_ts_role**.

sp_dropalias

Removes the alias user name identity established with **sp_addalias**.

Syntax

```
sp_dropalias loginame [, force]
```

Parameters

- ***loginame* –** is the name (in `master.dbo.syslogins`) of the user who was aliased to another user.
- **force –** allows you to drop an alias even if it owns database objects.

Examples

- **Example 1 –** 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:

  ```
  sp_dropalias victoria
  ```

- **Example 2 –** Drops the alias “harry,” which owns a procedure **namelist**. The SAP ASE server drops the alias but issues a warning message:

  ```
  sp_dropalias harry, force
  Warning: You have forced the drop of the alias for login 'harry'
  which owns objects in the database. This may result in errors when
  those objects are accessed from or contain references to another
  database.
  Alias user dropped.(return status = 0)
  ```

Usage

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.

You can drop the alias of a user who owns objects in the database. You do not need to first drop the objects before dropping the login.

Permissions

The permission checks for **sp_dropalias** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any user` privilege.
Disabled	With granular permissions disabled, you must be the database owner, a user with **sso_role**, or a user with sa_role.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addalias* on page 14
- *sp_adduser* on page 59
- *sp_droplogin* on page 278
- *sp_dropuser* on page 293

- *sp_helpuser* on page 446

sp_drop_all_qplans

Deletes all abstract plans in an abstract plan group.

Syntax

```
sp_drop_all_qplans name
```

Parameters

- ***name* –** is the name of the abstract plan group from which to drop all plans.

Examples

- **Example 1 –**

  ```
  sp_drop_all_qplans dev_test
  ```

Usage

To drop individual plans, use **sp_drop_qplan**.

To see the names of abstract plan groups in the current database, use **sp_help_qpgroup**.

sp_drop_all_qplans silently drops all plans in the group that belong to the specified user, or all plans in the group, if it is executed by a system administrator or database owner.

Permissions

The permission checks for **sp_drop_all_qplans** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege. Any user can execute **sp_drop_all_qplans** to drop plans that they own.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**. Any user can execute **sp_drop_all_qplans** to drop plans that they own.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_drop_qplan* on page 256
- *sp_drop_qpgroup* on page 255

sp_drop_qpgroup

Drops an abstract plan group.

Syntax

```
sp_drop_qpgroup group
```

Parameters

- ***group* –** is the name of the abstract plan group to drop.

Examples

- **Example 1 –** Drops the abstract plan group "dev_test":

  ```
  sp_drop_qpgroup dev_test
  ```

Usage

You cannot:

- Drop the default groups, `ap_stdin` and `ap_stdout`.
- Drop a group that contains plans. To drop all of the plans in a a group, use **sp_drop_all_qplans**. To see a list of groups and the number of plans they contain, use **sp_help_qpgroup**.

- Run **sp_drop_qpgroup** in a transaction.

Permissions

The permission checks for **sp_drop_qpgroup** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_drop_all_qplans* on page 254
- *sp_help_qpgroup* on page 370

sp_drop_qplan

Drops an abstract plan.

Syntax

```
sp_drop_qplan id
```

Parameters

- ***id* –** is the ID of the abstract plan to drop.

Examples

- **Example 1 –** The abstract plan with the specified ID is dropped:

```
sp_drop_qplan 1760009301
```

Usage

To find the ID of a plan, use **sp_help_qpgroup**, **sp_help_qplan**, or **sp_find_qplan**. Plan IDs are also returned by **create plan** and are included in **showplan** output.

To drop all abstract plans in a group, use **sp_drop_all_qplans**.

See also **create plan** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_drop_qplans** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege. Any user can execute **sp_drop_qplans** to drop plans that they own.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**. Any user can execute **sp_drop_qplans** to drop plans that they own.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_drop_all_qplans* on page 254
- *sp_find_qplan* on page 337
- *sp_help_qpgroup* on page 370
- *sp_help_qplan* on page 373

sp_drop_resource_limit

Removes one or more resource limits from the SAP ASE server.

Syntax

```
sp_drop_resource_limit { name, appname }
    [, rangename, limittype, enforced, action, scope]
```

Parameters

- ***name*** **–** is the SAP ASE 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 SAP ASE 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 *scope*, without regard to *rangename*.
- ***limittype*** **–** is the type of resource being limited. This must be one of the following:
  - **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.

    - **tempdb_space** – drops only the limits of the number of `tempdb` database pages that a single session used or can have.
    - **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 valid values for each limit type are:
    - **1** – drops only limits for which action is taken when the estimated cost of execution exceeds the specified limit.
    - **2** – drops only limits for which action is taken when the actual row count, elapsed time, or cost of execution exceeds the specified limit.
    - **3** – drops only limits for which action is taken when either the estimated cost (1) or the actual cost (2) exceeds the specified limit.
    - **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, and must be one of these:
    - **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, and must be one of the following:
    - **1** – drops only limits that apply to queries.
    - **2** – drops only limits that apply to query batches.
    - **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

- **Example 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:

  ```
  sp_drop_resource_limit joe, payroll, friday_afternoon, io_cost,
  2, 4, 1
  ```

  Note: If no resource limit matches these selection criteria, **sp_drop_resource_limit** returns without error.

- **Example 2 –** Drops all limits that apply to joe's use of the *payroll* application:

```
sp_drop_resource_limit joe, payroll
```

- **Example 3 –** Drops all limits that apply to the user “joe”:

```
sp_drop_resource_limit joe
```

- **Example 4 –** Drops all resource limits that apply to the *payroll* application:

```
sp_drop_resource_limit NULL, payroll
```

- **Example 5 –** Drops all resource limits on the *payroll* application with an action that is to kill the session:

```
sp_drop_resource_limit NULL, payroll, NULL, NULL, NULL, 4, NULL
```

Usage

To determine which resource limits apply to a given user, application, or time of day, use **sp_help_resource_limit**.

When you use **drop login** to drop an SAP ASE 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.

See the *System Administration Guide* for more information on resource limits.

Permissions

The permission checks for **sp_drop_resource_limit** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage resource limit` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_add_resource_limit* on page 7
- *sp_droplogin* on page 278
- *sp_help_resource_limit* on page 368
- *sp_modify_resource_limit* on page 510

sp_drop_time_range

Removes a user-defined time range from the SAP ASE server.

Syntax

```
sp_drop_time_range name
```

Parameters

- ***name* –** is the name of the time range to be dropped.

Examples

- **Example 1 –** Removes the “evenings” time range:

  ```
  sp_drop_time_range evenings
  ```

Usage

There are additional considerations when using **sp_drop_time_range**:

- 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 the *System Administration Guide*.

Permissions

The permission checks for **sp_drop_time_range** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage resource limit` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_add_resource_limit* on page 7
- *sp_add_time_range* on page 11
- *sp_modify_time_range* on page 513

sp_dropdevice

Drops an SAP ASE 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

- **Example 1 –** Drops the device named `tape5` from SAP ASE:

  ```
  sp_dropdevice tape5
  ```

- **Example 2 –** Drops the database device named `fredsdata` from SAP ASE. The device must not be in use by any database:

  ```
  sp_dropdevice fredsdata
  ```

Usage

There are additional considerations when using **sp_dropdevice**:

- **sp_dropdevice** drops a device from SAP ASE, 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 SAP ASE. To delete a file after using **sp_dropdevice**, use operating system commands.

See also **drop database** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_dropdevice** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage disk` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addumpdevice* on page 57
- *sp_helpdb* on page 394
- *sp_helpdevice* on page 402

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.

Considerations for process mode

sp_dropengine does not run in threaded mode.

Syntax

```
sp_dropengine engine_number [,engine_group] [,instance_id]
```

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.
- ***instance_id*** **–** (Cluster environments only) Is the ID of the instance from which you are dropping an engine or engine group.

Examples

- **Example 1 –** Drops engine number 2 from the group called `DS_GROUP`. If it is the last engine in the group, the group is also dropped:

  ```
  sp_dropengine 2, DS_GROUP
  ```

- **Example 2 –** (Cluster environments only) Drops engine number 5 from instance id 8:

  ```
  sp_dropengine 5, 8
  ```

Usage

There are additional considerations when using **sp_dropengine**:

- You can invoke **sp_dropengine** only from the `master` database.
- If *engine_number* is the last engine in *engine_group*, SAP ASE also drops *engine_group*.
- (Cluster Edition only) If you set **sp_cluster set *system_view*** to:
  - **cluster** – you can drop an engine or engine group from any instance in the cluster.
  - **instance** – you can drop an engine or engine group only from a local instance.
- **sp_dropengine** can run in sessions using chained transactions after you use **sp_procxmode** to change the transaction mode to **anymode**.
- The *engine_number* you specify must exist in *engine_group*.

Permissions

The permission checks for **sp_dropengine** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any execution class` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addengine* on page 21

sp_dropexeclass

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

- **Example 1 –** This statement drops the user-defined execution class `DECISION`:

  ```
  sp_dropexeclass 'DECISION'
  ```

Usage

An execution class helps define the execution precedence used by the SAP ASE server to process tasks. See the *Performance and Tuning Guide* for more information on execution classes and execution attributes.

classname must not be bound to any client application, login, stored procedure, or default execution class. Unbind the execution class first, using **sp_unbindexeclass**, then drop the execution class, using **sp_dropexeclass**.

You cannot drop system-defined execution classes.

Permissions

The permission checks for **sp_dropexeclass** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any execution class` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addexeclass* on page 23
- *sp_bindexeclass* on page 83
- *sp_showexeclass* on page 652
- *sp_unbindexeclass* on page 716

sp_dropextendedproc

Removes an extended stored procedure.

Syntax

```
sp_dropextendedproc esp_name
```

Parameters

- ***esp_name* –** is the name of the extended stored procedure to be dropped. *esp_name* is case-sensitive, and must precisely match the name with which the extended stored procedure was created.

Examples

- **Example 1 –** Removes `xp_echo`:

  ```
  sp_dropextendedproc xp_echo
  ```

Usage

You can execute **sp_dropextendedproc** only from the `master` database.

Permissions

The permission checks for **sp_dropextendedproc** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any ESP` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also
- *sp_addextendedproc* on page 25
- *sp_freedll* on page 351
- *sp_helpextendedproc* on page 404

sp_dropexternlogin

(Component Integration Services only) Drops the definition of a remote login previously defined by **sp_addexternlogin**.

Syntax

```
sp_dropexternlogin server [, loginame [, rolename ] ]
```

Parameters

- ***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.
- ***loginame* –** is a login account known to the local server. If *loginame* is not specified, the current account is used. *loginame* must exist in the `master.dbo.syslogins` table.
- ***rolename* –** is the SAP ASE user's assigned role.

Examples

- **Example 1 –** 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":

  ```
  sp_dropexternlogin CIS1012, bobj
  ```

- **Example 2 –** Drops the definition of an external login to the remote server SSB from users with the sa_role:

  ```
  sp_dropexternlogin SSB, NULL, sa_role
  ```

Usage

There are additional considerations when using **sp_dropexternlogin**:

- **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 **sp_addalias**, **create login**, and **drop login**.

Permissions

The permission checks for **sp_dropexternlogin** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any remote login` privilege. **sp_dropexternlogin** can be executed by users bound to *loginname*.
Disabled	With granular permissions disabled, you must be a user with *sa_role*. **sp_dropexternlogin** can be executed by users bound to *loginname*.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addexternlogin* on page 27
- *sp_helpexternlogin* on page 406
- *sp_addlogin* on page 35
- *sp_droplogin* on page 278

sp_dropglockpromote

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

- **Example 1 –** Removes the lock promotion values from `titles`. Lock promotion for `titles` now uses the database or server-wide values:

```
sp_dropglockpromote "table", titles
```

Usage

There are additional considerations when using **sp_droplockpromote**:

- 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 use the server-wide values.
- When a table's values are dropped, the SAP ASE 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.

Permissions

The permission checks for **sp_dropglockpromote** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage lock promotion threshold` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_setpglockpromote* on page 631

sp_ dropglockpromote_ptn

Removes partition lock promotion values.

Syntax

The syntax for dropping server-wide partition lock promotion settings is:

```
sp_dropglockpromote_ptn "server"
```

The syntax for dropping the partition lock promotion threshold at the database or table level is:

```
sp_dropglockpromote_ptn {"database" | "table"}, objname
```

Parameters

- **server –** removes server-wide values for the partition lock promotion thresholds.
- **"database" | "table" –** specifies whether to remove the partition lock promotion thresholds for a database or table. These are Transact-SQL keywords and therefore, require quotes.
- ***objname* –** is the name of the table or database from which to remove the partition lock promotion thresholds.

Examples

- **Example 1 –** Removes the partition lock promotion values from *titles*. Lock promotion for *titles* now uses the database or server-wide values:

  ```
  sp_ dropglockpromote_ptn "table", titles
  ```

Usage

There are additional considerations when using **sp_dropglockpromote_ptn**:

- Use **sp_dropglockpromote_ptn** to drop partition lock promotion values set with **sp_setpglockpromote_ptn**.
- When you drop a database’s partition lock promotion thresholds, tables that do not have partition lock promotion thresholds configured use the server-wide values.
- When a table’s values are dropped, the SAP ASE server uses the database’s lock promotion thresholds if they are configured or the server-wide values if they are not.
- When you drop server-wide partition lock promotion thresholds, partition lock promotion threshold values set at the table level will be used. Otherwise, partition lock promotion threshold values set at the database level will be used. If partition lock promotion threshold values are not set at either database or table level, then partition lock promotion is disabled. It can be enabled again using **sp_setrowlockpromote_ptn**.

Permissions

The permission checks for **sp_dropglockpromote_ptn** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage lock promotion threshold` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_dropgroup

Drops a group from a database.

Syntax

```
sp_dropgroup grpname
```

Parameters

- ***grpname* –** is the name of a group in the current database.

Examples

- **Example 1 –** 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:

```
sp_changegroup accounting, martha
sp_changegroup "public", george
sp_dropgroup purchasing
```

Usage

Executing **sp_dropgroup** drops a group name from a database's `sysusers` table.

You cannot drop a group if it has members. To drop the group, execute **sp_changegroup** for each member first.

Permissions

The permission checks for **sp_dropgroup** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any user` privilege.
Disabled	With granular permissions disabled, you must be the database owner, a user with **sa_role** , or a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addgroup* on page 30
- *sp_changegroup* on page 103

- *sp_helpgroup* on page 408

sp_dropkey

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

- **Example 1 –** Drops the primary key for the `employees` table. Any foreign keys that were dependent on the primary key for `employees` are also dropped:

  ```
  sp_dropkey primary, employees
  ```

- **Example 2 –** Drops the common keys between the `employees` and `projects` tables:

  ```
  sp_dropkey common, employees, projects
  ```

- **Example 3 –** Drops the foreign key between the `titleauthor` and `titles` tables:

  ```
  sp_dropkey foreign, titleauthor, titles
  ```

Usage

There are additional considerations when using **sp_dropkey**:

- 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**.

Permissions

You must be the table owner to execute **sp_dropkey**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_commonkey* on page 157
- *sp_foreignkey* on page 349
- *sp_helpkey* on page 417
- *sp_primarykey* on page 589

sp_droplanguage

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 SAP ASE system messages in *language*. You cannot drop a language with associated system messages without also dropping its messages.

Examples

- **Example 1 –** This example drops French from the available alternate languages, if there are no associated messages:

  ```
  sp_droplanguage french
  ```

- **Example 2 –** This example drops French from the available alternate languages, if there are associated messages:

  ```
  sp_droplanguage french, dropmessages
  ```

Usage

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.

Permissions

The permission checks for **sp_droplanguage** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addlanguage* on page 32
- *sp_helplanguage* on page 419

sp_droplogin

Deprecated by SAP ASE 15.7. To drop a login account in SAP ASE, use the **drop login** command. See Reference Manual: Commands.

sp_dropmessage

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.

  When you include the optional *language* parameter, 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.

Examples

- **Example 1 –** Removes the French version of the message with the number 20002 from `sysusermessages`:

  ```
  sp_dropmessage 20002, french
  ```

Permissions

The permission checks for **sp_dropmessage** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the user who created the message or the database owner, or a user with `own database` privilege on the current database.
Disabled	With granular permissions disabled, you must be the user who created the message, the database owner, or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addmessage* on page 35
- *sp_getmessage* on page 353

sp_dropobjectdef

(Component Integration Services only) Deletes the external storage mapping provided for a local object.

Syntax

```
sp_dropobjectdef tablename
```

Parameters

- ***tablename* –** has the form `dbname.owner.object`, where:
  - *dbname* is the name of the database containing the object with a storage location that 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 with a storage location that 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

- **Example 1 –** 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`:

  ```
  sp_dropobjectdef "personnel.dbo.colleges"
  ```
- **Example 2 –** 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`:

  ```
  sp_dropobjectdef "andrea.fishbone"
  ```

Usage

There are additional considerations when using **sp_dropobjectdef**:

- **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 *tablename* can be in any of these forms:
  - *object*
  - ***owner.object***
  - ***dbname..object***
  - ***dbname.owner.object***

See also **create existing table**, **create table**, **drop table** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_dropobjectdef** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the object owner or a user with `drop any table` privilege.
Disabled	With granular permissions disabled, you must be the object owner, the database owner, or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addobjectdef* on page 37

sp_dropremotelogin

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

- **Example 1 –** Drops the entry for the remote server named GATEWAY:

  ```
  sp_dropremotelogin GATEWAY
  ```

- **Example 2 –** Drops the entry for mapping remote logins from the remote server GATEWAY to the local user named “churchy”:

  ```
  sp_dropremotelogin GATEWAY, churchy
  ```

- **Example 3 –** Drops the login for the remote user “pogo” on the remote server GATEWAY that was mapped to the local user named “churchy”:

  ```
  sp_dropremotelogin GATEWAY, churchy, pogo
  ```

Usage

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 commands **create login** and **drop login**.

Permissions

The permission checks for **sp_dropremotelogin** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any remote login` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**

Information	Values
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addlogin* on page 35
- *sp_addremotelogin* on page 40
- *sp_addserver* on page 46
- *sp_droplogin* on page 278
- *sp_helpremotelogin* on page 424
- *sp_helpserver* on page 434

sp_droprowlockpromote

Removes row lock promotion threshold values from a database or table.

Syntax

```
sp_droprowlockpromote {"database" | "table"}, objname
```

Parameters

- **database | table –** specifies whether to remove the row lock promotion thresholds from a database or table.
- ***objname* –** is the name of the database or table from which to remove the row lock promotion thresholds.

Examples

- **Example 1 –** Removes the row lock promotion values from the `sales` table. Lock promotion for `sales` now uses the database or server-wide values:

  ```
  sp_droprowlockpromote "table", "sales"
  ```

Usage

There are additional considerations when using **sp_droprowlockpromote**:

- Use **sp_droprowlockpromote** to drop row lock promotion values set with **sp_setrowlockpromote**.
- When you drop a database's row lock promotion thresholds, datarows-locked tables that do not have row lock promotion thresholds configured use the server-wide values. Use **sp_configure** to check the value of the row lock promotion configuration parameters.
- When a table's row lock promotion values are dropped, the SAP ASE server uses the database's row lock promotion thresholds, if they are configured, or the server-wide values, if no thresholds are set for the database.
- To change the lock promotion thresholds for a database, you must be using the `master` database. To change the lock promotion thresholds for a table in a database, you must be using the database where the table resides.
- You can change server-wide values with **sp_setrowlockpromote**. Since this changes the values in the row lock promotion configuration parameters, there is no corresponding server option for **sp_droprowlockpromote**.

Permissions

The permission checks for **sp_droprowlockpromote** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage lock promotion threshold` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_setrowlockpromote* on page 636

sp_droprowlockpromote_ptn

Removes partition lock promotion threshold values at server, database, or table levels.

Syntax

The syntax for dropping server-wide partition lock promotion settings is:

```
sp_droprowlockpromote_ptn "server"
```

The syntax for dropping the partition lock promotion threshold at the database or table level is:

```
sp_droprowlockpromote_ptn {"database" | "table"}, objname
```

Parameters

- **server –** removes server-wide values for the partition lock promotion thresholds.
- **"database" | "table" –** specifies whether to remove the partition lock promotion thresholds for a database or table. These are Transact-SQL keywords and therefore, require quotes.
- ***objname* –** is the name of the table or database from which to remove the partition lock promotion thresholds.

Examples

- **Example 1 –** Removes the partition lock promotion values from the `sales` table. Partition lock promotion for `sales` now uses the database or server-wide values:

  ```
  sp_droprowlockpromote_ptn "table", "sales"
  ```

Usage

There are additional considerations when using **sp_droprowlockpromote_ptn**:

- Use **sp_droprowlockpromote_ptn** to drop partition lock promotion values set with **sp_setrowlockpromote_ptn**.
- When you drop a database's partition lock promotion thresholds, datarows-locked tables that do not have partition lock promotion thresholds configured at table level use the server-wide values. Use **sp_configure** to check the value of the partition lock promotion configuration parameters.
- When a table's partition lock promotion values are dropped, the SAP ASE server uses the database's partition lock promotion thresholds, if they are configured, or the server-wide values, if no thresholds are set for the database.

- To change the partition lock promotion thresholds for a database, you must be using the `master` database. To change the partition lock promotion thresholds for a table in a database, you must be using the database where the table resides.
- When you drop server-wide partition lock promotion thresholds, partition lock promotion threshold values set at the table level will be used. Otherwise, partition lock promotion threshold values set at the database level will be used. If partition lock promotion threshold values are not set at either database or table level, then partition lock promotion is disabled. It can be enabled again using **sp_setrowlockpromote_ptn**.

Permissions

The permission checks for **sp_droprowlockpromote_ptn** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage lock promotion threshold` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_dropsegment

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

- **Example 1 –** This command drops the segment `indexes` from the `pubs2` database.

  ```
  sp_dropsegment indexes, pubs2
  ```

- **Example 2 –** This command unmaps the segment `indexes` from the database device `dev1`:

  ```
  sp_dropsegment indexes, pubs2, dev1
  ```

Usage

There are additional considerations when using **sp_dropsegment**:

- You can drop a segment if it is not referenced by any table, index, or partition in the specified database.
- If you:
  - Do not supply *device* – the segment is dropped from the specified database.
  - Supply *device* – 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.
- You can only execute **sp_dropsegment** for the `logsegment` system segment in single-user mode.

  Note: This command may take a long time to complete in in very large databases.

- When you unmap a segment from one or more devices, the SAP ASE server drops any thresholds that exceed the total space on the segment. When you unmap the `logsegment` from one or more devices, the SAP ASE 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.

Permissions

The permission checks for **sp_dropsegment** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage database` privilege.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addsegment* on page 43
- *sp_addthreshold* on page 49
- *sp_helpsegment* on page 430
- *sp_helpthreshold* on page 445

- *sp_placeobject* on page 577

sp_dropserver

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

- **Example 1 –** This command drops the remote server GATEWAY:

  ```
  sp_dropserver GATEWAY
  ```

- **Example 2 –** Drops the entry for the remote server RDBAM_ALPHA and drops all remote logins and external logins for that server:

  ```
  sp_dropserver RDBAM_ALPHA, droplogins
  ```

Usage

There are additional considerations when using **sp_dropserver**:

- 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.

Permissions

The permission checks for **sp_dropserver** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege. When **droplogins** is specified, you must be a user with `manage any remote login` privilege. SAP ASE high availablility – You must be a user with `manage server` privilege and **ha_role**. When **droplogins** is specified, you must be a user with `manage any remote login` privilege. SAP ASE shared-disk cluster – You must be a user with `manage server` and `manage cluster` privileges. When **droplogins** is specified, you must be a user with `manage any remote login` privilege.
Disabled	With granular permissions disabled, you must be a user with **sso_role**. SAP ASE high availablility – You must be a user with **sso_role** permission and **ha_role**. SAP ASE shared-disk cluster – You must be a user with **sso_role** and **sa_role** permission.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addserver* on page 46
- *sp_dropremotelogin* on page 281
- *sp_helpremotelogin* on page 424
- *sp_helpserver* on page 434

sp_dropthreshold

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 with free space that 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

- **Example 1 –** Removes a threshold from `segment1` of `mydb`. You must specify the database, segment, and amount of free space to identify the threshold:

  ```
  sp_dropthreshold mydb, segment1, 200
  ```

Usage

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.

Permissions

The permission checks for **sp_dropthreshold** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage database` privilege.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addthreshold* on page 49
- *sp_dboption* on page 193
- *sp_helpthreshold* on page 445
- *sp_thresholdaction* on page 700

sp_droptype

Drops a user-defined datatype.

Syntax

```
sp_droptype typename
```

Parameters

- ***typename* –** is the name of a user-defined datatype that you own.

Examples

- **Example 1 –** Drops the user-defined datatype named `birthday`:

  ```
  sp_droptype birthday
  ```

Usage

Executing **sp_droptype** deletes a user-defined datatype from `systypes`.

You cannot drop a user-defined datatype if it is referenced by tables or another database object.

See also *User-Defined Datatypes* in *Reference Manual: Building Blocks*.

Permissions

The permission checks for **sp_droptype** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the datatype owner or a user with `manage database` privilege.
Disabled	With granular permissions disabled, you must be datatype owner or database owner.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addtype* on page 54
- *sp_rename* on page 605

sp_dropuser

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

- **Example 1 –** Drops the user "albert" from the current database. The user "albert" can no longer use the database:

  ```
  sp_dropuser albert
  ```

Usage

There are additional considerations when using **sp_dropuser**:

- **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.
  - A user who has granted permissions to other users.
  - 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**.
- **sp_dropuser** drops all key copies from `sysencryptkeys` for the specified user in the current database. **sp_dropuser** fails if the user owns an encryption key in any database. See the *Encrypted Columns Users Guide*.

See also **grant**, **revoke**, **use** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_dropuser** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any user` privilege.
Disabled	With granular permissions disabled, you must be the datatype owner, a user with **sa_role**, or a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addalias* on page 14
- *sp_adduser* on page 59
- *sp_bindexeclass* on page 83
- *sp_droplogin* on page 278

sp_dump_history

Allows you to purge records from the dump history file.

The original dump history file is copied to a new file named `original_name.`*XXX*, where *XXX* represents an increasing numerical value (001, 002, and so on).

Syntax

```
sp_dump_history [  @operation = {'list' | 'purge' | 'listfiles' |
'help'}]
    [, @until_time = 'date']
    [, @database_name = 'database_name']
    [, @dump_type =
        {'DATABASE' | 'TRAN[SACTION]' | 'CONFIG[URATION]'}]
    [, @status = {'success' | 'fail'}]
    [, @file = 'filename']
```

Parameters

- **@operation = {'list' | 'purge' | 'listfiles' | 'help'} –** is a required parameter that include these options:
  - **list** – displays the records from the dump history file. By default, the list includes:

    - Database name – the server configuration name
    - Dump type
    - Total number of dump stripes
    - Compression level
    - Remote Backup Server name
    - Dump creation time
    - Error number – on SAP ASE
  - **purge** – purges records from the dump history file. The records to be purged are selected based on criteria specified using the other **sp_dump_history** parameters.
  - **listfiles** – displays the list of dump history file names.
  - **help** – shows the syntax for **sp_dump_history**.
- **@until_time = '*date*' –** (optional) allows you to specify a date and time, with Backup Server purging all dump objects created before that time. By default, all dump records are purged.
- **@database_name = '*database_name*' –** (optional) is the name of the database that has dump records from the dump history file that are listed or purged. By default, all database dump records are included.
- **@dump_type' –** (optional) specifies the type of dump record to be listed or purged. Valid types are:

  - **'DATABASE'** – database dump objects created by **dump database**.
  - **'TRAN[SACTION]'** – transaction dump objects created by **dump transaction**.
  - **'CONFIG[URATION]'** – server configuration objects created by **dump configuration**.

  By default, all types of dump records are listed or purged.
- **@status = {'success' | 'fail'} –** (optional) specifies with to list or purge the successful or failed dump records. By default, only successful dump records are included.
- **@file = '*filename*' –** (optional) specifies the name of the dump history file. You must specify the the path, or location, of the file as part of *filename*. The default location of the dump history file is `$SYBASE/$SYBASE_ASE` (`%SYBASE%\%SYBASE_ASE%` in Windows).

Examples

- **Example 1 –** Lists all dump records from the dump history file:

```
sp_dump_history 'list'
go
```

- **Example 2 –** Lists dump records of a specified database created before a specified time:

```
sp_dump_history 'list', "mar 20, 2010 10:51:43:866am",
   'testdb'
go
```

- **Example 3 –** Lists the transaction dump objects from the `model` database, specifying a full path for the dump history file:

```
sp_dump_history @operation='list', @database_name = "model",
    @dump_type='TRAN', @status = 'success',
    @file = '/john_machine/john/ASE/ASE-16_0/dumphist'
```

Usage

The default behavior for **sp_dump_history** with no parameters is to display the output from its **list** parameter.

The output for database and transaction dumps differs from that of configuration files.

See also:

- For information about dump operations, see the *System Administration Guide*.
- **dump configuration**, **dump database**, **load database** in *Reference Manual: Commands*
- **sp_config_dump**

Permissions

The permission checks for **sp_dump_history** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage dump configuration` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role** or **oper_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	
Audit option	
Command or access audited	
Information in `extrainfo`	• *Roles*– current actve roles. • *Keywords or options* – NULL. • *Previous value* – NULL. • *Current value* – NULL. • *Other information* – all input parameters. • *Proxy information* – original login name, if **set proxy** in effect.

sp_dump_info

The **sp_dump_info** system procedure displays the size of data and log that an uncompressed cumulative dump would contain at a specific point in time.

The size is reported in units of KB, MB, or GB, as appropriate. The size reported may be slightly smaller than the actual size of the archive file (or files, if using multiple stripes), because the archive contains some additional information by way of the labels, header, trailer and runlist pages. **sp_dump_info** can also only assume that an uncompressed dump is done; if a compressed dump is done, the size of the archive will clearly be smaller than that reported by **sp_dump_info**.

You cannot use **sp_dump_info**:

- Unless you allow incremental dumps of your database by using the **allow incremental dumps** parameter of **sp_dboption**.
- If the database has not yet been fully dumped since you enabled incremental dumps for your database.

Syntax

```
sp_dump_info database_name
```

Parameters

- **database_name –** is the name of the database.

Examples

- **Data and log size –** Displays the size of data and log that the cumulative dump of the `test` database contains

  ```
  sp_dump_info test
  go
   Data     Log    Database percentage  Allocation threshold
   -------  -----  -------------------  --------------------
   4368 KB  2 KB                     2                    40
  (return status = 0) (return status = 0)
  ```

  The output indicates that if a cumulative dump were taken at this point in time, it would contain approximately 4,368KB of data and a single log page, which represents 2 percent of the total database size.

  Compare this with the size if you performed a cumulative dump at this time:

  ```
  dump database test cumulative to "c:/tmp/test.dmp"
  go
  Backup Server: 4.171.1.1: The current value of 'reserved pages
  threshold' is 85%.
  Backup Server: 4.171.1.2: The current value of 'allocated pages
  ```

```
threshold' is 40%.
Backup Server session id is: 10. Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any
volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'test122480F0EF ' section
number 1 mounted on disk file 'c:/tmp/test.dmp'
Backup Server: 4.188.1.1: Database test: 4328 kilobytes (3%)
DUMPED.
Backup Server: 3.43.1.1: Dump phase number 1 completed.
Backup Server: 3.43.1.1: Dump phase number 2 completed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.188.1.1: Database test: 4370 kilobytes (3%)
DUMPED.
Backup Server: 3.42.1.1: DUMP is complete (database test).
```

The corresponding size of the archive is 4,487,168 bytes, or 2191 pages. This differs from the estimate given by **sp_dump_info** by 29 pages (14 KB), which is the result of 8 pages for the dump labels, 1 page for the dump header, 1 page for the dump trailer and 19 pages containing run lists. The size of the dump labels, header and trailer are independent of the numbers of pages dumped, while the number of pages used by run lists is dependent on the numbers of pages dumped.

- **Error message –** Displays an error message when incremental dumps are not enabled on `master`

```
 sp_dump_info mydb
go
Msg 17154, Level 16, State 1:
Procedure 'sp_dump_info', Line 32:
Incremental dumps are not enabled in database mydb.
(return status = 1)
```

Usage

sp_dump_info fails if you do not allow incremental dumps, or you have not enabled incremental dumps for your database.

Permissions

Any user can execute **sp_dump_info**.

sp_dumpoptimize

Specifies the amount of data dumped by Backup Server during a **dump database** operation.

Syntax

```
sp_dumpoptimize [ 'archive_space = {maximum | minimum | default }' ]
```

```
sp_dumpoptimize [ 'reserved_threshold = {nnn | default }' ]
```

```
sp_dumpoptimize [ 'allocation_threshold = {nnn | default }' ]
```

Parameters

- **archive_space –** specifies the amount of the database you want dumped.
- **maximum –** dumps the whole database without determining which pages are allocated or not. The total space used by the archive image or images is equal to the size of the database. Using this option has the same effect as using the options **reserved_threshold=0** and **allocation_threshold=0**.
- **minimum –** dumps only the allocated pages, which results in the smallest possible archive image. This option is useful when dumping to archive devices for which the throughput is much smaller than that of the database devices such as QIC tape drives. Using this option has the same effect as using the options **reserved_threshold=100** and **allocation_threshold=100**.
- **default –** specifies that default values should be used. When used with:
  - **archive_space** – this option dumps the database with the **reserved_threshold** and **allocation_threshold** options set to their default values. Use this to reset Backup Server to the default configuration.
  - **reserved_threshold** – **default** specifies 85 percent.
  - **allocation_threshold** – **default** specifies 40 percent.
- **reserved_threshold –** dumps all the pages belonging to the database in a database disk if the percentage of reserved pages in the disk is equal to or greater than *nnn*. For example, if you specify *nnn* as **60** and if a database disk has a percentage of reserved pages equal to or greater than 60 percent, then the entire disk is dumped without determining which pages within that disk are allocated. The default for this option is 85 percent.
- ***nnn* –** an integer value between 0 and 100 that represents the value of the threshold. It is used to determine how much data to dump.

  When used with **reserved_threshold**, if the percentage of reserved pages in the disk is greater than the value specified, all the pages of the database in a database disk are dumped.

  When used with **allocation_threshold**, if the percentage of allocated pages in an allocation unit is greater than the percentage specified for **allocation_threshold**, all the pages within an allocation unit are dumped.
- **allocation_threshold –** dumps all the pages in the allocation unit if the percentage of allocated pages in the unit is equal to or greater than *nnn*. For example, if *nnn* is specified as **70** and if the percentage of allocated pages in an allocation unit is equal to or greater than 70 percent, then the entire allocation unit is dumped without determining whether pages within that allocation unit are allocated or not. If the **reserved_threshold** setting causes the whole disk to be dumped, the **allocation_threshold** setting is ignored for the disk. The default for this option is 40 percent.

Examples

- **Example 1 –** Dumps the whole database:

  ```
  sp_dumpoptimize 'archive_space=maximum'
  ```

```
Backup Server: 4.172.1.1: The value of 'reserved pages threshold'
has been set to 0%.
Backup Server: 4.172.1.2: The value of 'allocated pages threshold'
has been set to 0%.
```

- **Example 2 –** Dumps only the allocated pages, thereby resulting in the smallest archive image:

  ```
  sp_dumpoptimize 'archive_space=minimum'
  ```

  ```
  Backup Server: 4.172.1.1: The value of 'reserved pages threshold'
  has been
  set to 100%.
  Backup Server: 4.172.1.2: The value of 'allocated pages threshold'
  has been
  set to 100%.
  ```

- **Example 3 –** Sets the reserved threshold to 85 percent and the allocation threshold to be set to 40 percent:

  ```
  sp_dumpoptimize 'archive_space=default'
  ```

  ```
  Backup Server: 4.172.1.1: The value of 'reserved pages threshold'
  has been
  set to 85%.
  Backup Server: 4.172.1.2: The value of 'allocated pages threshold'
  has been
  set to 40%.
  ```

- **Example 4 –** Dumps disks in the database with a percentage of reserved pages that is greater than or equal to 60 percent without reading allocation pages on this disk. For the remaining disks, the allocation pages are read, and the last set value for the **allocation_threshold** is used. If the **allocation_threshold** was not set after Backup Server was started, default **allocation_threshold** of 40 percent is used:

  ```
  sp_dumpoptimize 'reserved_threshold=60'
  ```

  ```
  Backup Server: 4.172.1.3: The value of 'reserved pages threshold'
  has been
  set to 60%.
  ```

- **Example 5 –** Causes the reserved threshold to be set to 85 percent. It does not affect the allocation page threshold:

  ```
  sp_dumpoptimize 'reserved_threshold=default'
  ```

  ```
  Backup Server: 4.172.1.3: The value of 'reserved pages threshold'
  has been
  set to 85%.
  ```

- **Example 6 –** Reads allocation pages for those disks with a reserved page percentage that is less than the last set value for the **reserved_threshold** and if an allocation unit has 80 percent or more pages allocated, then the whole allocation unit is dumped:

  ```
  sp_dumpoptimize 'allocation_threshold=80'
  ```

```
Backup Server: 4.172.1.4: The value of 'allocated pages threshold'
has been
set to 80%.
```

- **Example 7 –** This example causes the allocation page threshold to be set to the default of 40 percent. It does not affect the reserved pages threshold:

  ```
  sp_dumpoptimize 'allocation_threshold=default'
  ```

  ```
  Backup Server: 4.172.1.4: The value of 'allocated pages threshold'
  has been
  set to 40%.
  ```

- **Example 8 –** Dumps disks in the database with a percentage of reserved pages that is greater than or equal to 60 percent without reading allocation pages on this disk. For the remaining disks, the allocation pages are read and if an allocation unit has 30 percent or more pages allocated, then the whole allocation unit is dumped:

  ```
  sp_dumpoptimize 'reserved_threshold=60',
  'allocation_threshold=30'
  ```

  ```
  Backup Server: 4.172.1.3: The value of 'reserved pages threshold'
  has been
  set to 60%.
  Backup Server: 4.172.1.4: The value of 'allocated pages threshold'
  has been
  set to 30%.
  ```

- **Example 9 –** Displays the current value of the thresholds:

  ```
  sp_dumpoptimize
  ```

  ```
  Backup Server: 4.171.1.1: The current value of 'reserved pages
  threshold'
  is 60%
  Backup Server: 4.171.1.2: The current value of 'allocated pages
  threshold'
  is 30%.
  ```

Usage

- When you set a threshold using **sp_dumpoptimize**, this threshold acts on each individual device that the database resides on.
- When you set values with **sp_dumpoptimize**, those values are immediately in affect without the need to restart Backup Server. However, the changes are effective only until the Backup Server is restarted. When Backup Server is restarted, the default values are used.
- If you issue **sp_dumpoptimize** multiple times, the thresholds specified by the last instance are used by later dumps. For example, if you first set the **reserved_threshold** value, and later issue **archive_space=maximum**, then that value overwrites the previous value you set for **reserved_threshold**.
- Dumps of different databases can use different thresholds by changing the **sp_dumpoptimize** values before each database dump.

- The optimal threshold values can vary from one database to another. Therefore, the performance of a dump depends on both the I/O configuration and the amount of used space in the database. The DBA can determine the appropriate configuration for a database by experimenting with dumps using different values and choosing the one that results in the shortest dump time.
- You can use **sp_dumpoptimize** for both local and remote dumps.
- **sp_dumpoptimize** has no effect on the performance of a transaction log dump or a load. Therefore, it need not be issued before **dump transaction**, **load database** or **load transaction** operations.
- If **sp_dumpoptimize** is issued without any parameters, the current value of the thresholds is displayed on the client.
- On configurations in which the archive device throughput is equal to or higher than the cumulative throughput of all the database disks, using **archive_space=maximum** may result in a faster dump. However, on configurations in which the archive device throughput is less than the cumulative throughput of all the database disks, using this option may result in a slower dump.
- The option names and the values for this procedure can be abbreviated to the unique substring that identifies them. For example, **ar = ma** is sufficient to uniquely identify the option **archive_space=maximum**.
- There can be zero or more blank space characters around the equal sign (**=**) in the option string.
- The option names and their values are case insensitive.

See also:

- **dump database**, **dump transaction**, **load database**, **load transaction** in *Reference Manual: Commands*
- See the *System Administration Guide* for information on allocation pages.

Permissions

The permission checks for **sp_dumpoptimize** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `dump database` privilege on the database you are dumping.
Disabled	With granular permissions disabled, you must be the datatype owner, a user with **sa_role**, or a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

Thresholds and sp_dumpoptimize

The default values for the thresholds are: Reserved pages: 85%; allocation pages: 40%

If the device fragment of the database has a reserved pages percentage that is:

- Greater than or equal to the reserved threshold – then all the blocks on this device that pertain to this database are dumped.
- Less than the reserved threshold – then Backup Server starts checking each allocation unit on this device for the allocation percentage. If the cumulative allocation percentage is:
  - Less than the allocation threshold – then it would only dump those pages with data written on it
  - Greater than the allocation threshold – the entire allocation unit would be dumped.

sp_encryption

Reports encryption information.

Syntax

```
sp_encryption help | helpkey
```

```
sp_encryption help | helpkey [, key_name | wildcard]
    [, all_dbs | key_copy | display_cols]
```

```
sp_encryption help | 'helpkey',
    {'master'|'dual master'} [, 'display_keys' | 'all_dbs'
```

```
sp_encryption 'help'[, 'servicekeyname '[, 'display_objs']]
```

```
sp_encryption 'helpextpasswd'
```

```
sp_encryption helpcol [, table_name | column_name ]
sp_encryption helpuser [, user_name | wildcard ][, key_copy |
login_passwd_check ]
```

```
sp_encryption 'mkey_startup_file'[, {'new_path' | 'default_location'
| 'null'}
    [, {sync_with_mem | sync_with_qrm}]]
```

```
sp_encryption 'downgrade_kek_size' [, 'true'|'false']
```

```
sp_encryption system_encr_passwd, 'newpasswd' [,'oldpasswd']
```

Parameters

- **helpkey –** lists encryption key properties, including:
    - Whether the database contains encryption keys.
    - The following, when run by a user with sso_role, key custodian, or DBO: keyname, keyowner, key length, key algorithm, key type, pad, initialization vector, type of password used to encrypt the key, whether key recovery has been enabled and count of key copies. The output is sorted on owner.key_name. When run by a non-privileged user, this command lists keyname, keyowner and keytype.
- **help –** included for backward compatibility. Includes the same output as **helpkey**
- ***key_name* –** is the name of the key you are investigating. Lists the properties defined for *key_name*. If *key_name* is omitted, lists properties for all keys.
- ***wildcard* –** lists the properties for keys matching the wildcard pattern in the current database. See the *Reference Manual: Building Blocks* for information about using wildcards.
- **all_dbs –** lists information on encryption keys in all available databases. Only the SSO can run **all_dbs**.
- **key_copy –** lists all user copies for the specified key in the current database. The output is sorted by ***key_owner.key_name***. Includes information about:
    - The base key owner.
    - If the key copy is a recovery key copy.
    - The user to whom a copy belongs.
    - If the copy is encrypted with a user-encryption password, a login password, or the system encryption password for login association (indicated by Login Access).
- **login_passwd_check –** indicates if the key copies assigned to the matched users are well synchronized with the user's login password. That is, the last update date of the key copy is newer than the date of the login password. The key copies are encrypted with the user's login password or login association.
- **display_keys –** is used with *system_encr_passwd* to display the keys and key copies that are encrypted using the system encryption password. Used with **master** or **dual master** to display keys and key copies encrypted using the master key or the dual master key.

You must be the system security officer, key custodian, or the database owner can run `sp_encryption helpkey, master | 'dual master', display_keys` to display encryption keys protected by the master or dual master key.

- **display_cols –** displays the key name, all keys (or matching wildcard keys) in the current database and the columns the key encrypts. When SSO includes **display_cols**, it displays columns encrypted by the keys across all available databases. When a user without the sso_role runs **display_cols**, only those columns encrypted by the key in the current database are displayed. Data is sorted by *key_name*, *key_owner*, *database_name*, *table_owner*, *table_name*, and *column_name*.
- **master –** reports information about the master key.
- **dual master –** reports information about the dual master key.
- **servicekeyname –** is set to **syb_extpasswdkey** or **syb_syscommkey%**. Use with **display_objs** to display objects encrypted by the service key.
- **display_objs –** displays object owners.

  You must be the system security officer, key custodian, or the database to run **sp_encryption helpkey, keyname, display_objs** to display objects in current database protected by the syb_extpasswdkey or syb_syscommkey service keys.
- **helpextpasswd –** displays the encryption status of external passwords in the status column. The encryption status is one of:

  - FIPS Encryption – the password is protected by the `syb_extpasswdkey` service key using a FIPS compliant cryptography algorithm
  - Needs Reset – indicates the system removed the password, and you must manually reset it.
  - Legacy Encryption – the password is protected with an algorithm from a version of SAP ASE earlier than 15.7.

  You must be the system security officer to run **sp_encryption helpextpasswd** to check the status of external passwords.
- **helpcol *column_name* –** displays the column name and the key used to encrypt the column. If the SSO includes **helpcol**, it prints the key name even if the key is not present in the current database. If a non-SSO user includes **helpcol**, the SAP ASE server prints the `keyid` of the key if it is not present in the current database, omitting the *key_name*. The output includes: ***owner.table.column***, ***database.owner.keyname***. The information is sorted by ***owner.table.column***.
- **helpuser –** displays the keys owned by or assigned to a user in the current database.
- **mkey_startup_file –** displays or sets the master key startup file name and path. **sp_encryption** sets the master key startup file to *new_path* or the default location. If you specify **null**, or no location, **sp_encryption** displays the current master key startup file name and path.
- **sync_with_mem –** (Cluster Edition only) writes the master key encryption key that exists in server memory to the master key startup file. Replaces the current master key encryption key, if it exists. If **automatic master key access** is set to off, **sync_with_mem** is also disabled.

You must be the system security officer display, set, or sync the master key startup file.

- **sync_with_qrm –** updates the local master key startup file with the version in the quorum device.

  You must be the system security officer display, set, or sync the master key startup file.
- **downgrade_kek_size –** displays or sets the **downgrade kek size** configuration. **true** indicates that the SAP ASE server is in **downgrade kek size** mode, **false** disables this mode.

  If you specify no argument, **sp_encryption** displays the current value for **downgrade_kek_size**.

  You must be the system security officer or the key custodian to run this command.
- **system_encr_passwd –** displays the keys and key copies encrypted using the system encryption password in the current database.
- **system_encr_passwd, all_dbs –** displays the properties of the system encryption password in every database where it has been set. The output is sorted by database name. Only the system security officer an run this command. If the system encryption password has not been set for all databases, the SAP ASE server generates Message 19782:

```
The system encryption password has not been set for all
available databases
```

Examples

- **Display Key Information for Fully Encrypted Database**

  This example shows a key type called "`database encryption key`" to indicate that the database is fully encrypted.

```
1> create encryption key key1 as default for database encryption
2> go
1> sp_encryption helpkey, key1

Key Name    Key Owner    Key Length   Key Algorithm
   Key Type                          Pad    Initialization Vector
  Protected By     Key Recovery
     # of Key Copies
----------- ------------ ------------ -------------
    -------------------------------   ------
---------------------
  -------------- ----------------    -----------------
key1        dbo          256          AES
   symmetric database encryption key      0                     1
  master key       0                             0
1> create encryption key key2 for database encryption with master
key
2> create encryption key key3 for database encryption with
dual_control
3> go
1> sp_encryption helpkey, 'key%'
Key Name    Key Owner    Key Length   Key Algorithm
   Key Type                          Pad    Initialization Vector
  Protected By     Key Recovery
     # of Key Copies
----------- ------------ ------------ -------------
    --------------------------------   ------
---------------------
  -------------- ----------------    -----------------
key1        dbo          256          AES
    symmetric database encryption key     0                    1
  master key       0                           0
key2        dbo          256          AES
    symmetric database encryption key     0                    1
  master key       0                           0
key3        dbo          256          AES
    symmetric database encryption key     0                    1
  dual_control(master key + dual master key)  0                 0

1> create database encr_db1 encrypt with key1
2> create database encr_db2 encrypt with key2
3> create database encr_db3 encrypt with key3
4> go
1> sp_encryption helpkey, '%', "display_dbs"
Key Name Key Owner Encrypted Database
-------------------- --------------------------------------
key1 dbo encr_db1
key1 dbo encr_db2
key3 dbo encr_db3
```

- **Display Key Information in Current Database –** Use the **helpkey** parameter to display key information in the current database. You can get information on all keys or specific keys. The second parameter to **sp_encryption** supplies the key name and may include SQL pattern-matching characters. If you are not the database owner and do not have **sso_role** or **keycustodian_role**, **sp_encryption** displays fewer columns.

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

```
sp_encryption helpkey
```

```
Key Name    Key Owner  Key Length  Key Algorithm  Key
Type                 Pad
     Init Vector  Protected By    Key Recovery  # of Key Copies
----------  ---------  ----------  --------------  --------
        ----
     -----------  --------------------------    ------------  ---
------------
tinnap_key  tinnap     128         AES            symmetric
key          0
     1           system encryption
password              0               0
tinnap_key1 tinnap    128        AES          symmetric default
key  0
     1           user
Passwd                        1                3
sample_key1 dbo        192         AES            symmetric
key          1
```

```
     1            login
Passwd                           1                2
```

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

```
sp_encryption helpkey
```

```
Key Name          Key Owner   Key Type
--------------     ---------   ------------
tinnap_key         tinnap      symmetric key
tinnap_key1        tinnap      symmetric default key
sample_key1        dbo         symmetric key
```

If you are not the system security officer, or do have **keycustodian_role**, the query displays all base keys you own in the current database. If you do not specify a *user_name* as the second parameter, the query displays the base keys you own.

- **Display Properties of Base Encryption Key When Run by SSO –** Displays the properties of base encryption key `sample_key1` when run by the SSO, key custodian, or DBO in the current database:

```
sp_encryption helpkey, sample_key1
```

```
Key Name   Key Owner  Key Length  Key Algorithm         Key Type
   Pad  Init Vector   Protected By      Key Recovery     # of Key
Copies
---------  --------    ---------  --------------         --------
------
   ---  -----------   ------------      -------------    --------
-------
sample_key1    dbo         192           AES           symmetric
Key
    1          1         Login                1                 2
```

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

```
sp_encryption helpkey, sample_key1
```

```
Key Name       Key Owner    Key Type
-------------  -----------  ------------
sample_key1            dbo  ymmetric key
```

- **Display Properties of All Base Encryption Keys in All Available Databases –** Only the SSO can run this command:

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

```
Db.Owner.Keyname         Key Length   Key Algorithm  Key Type
Pad       Init Vector   Protected By       Key Recovery    #of Key
Copies
------------------------   ----------   -----------  ------------
--------
-----  ----------   -----------------    --------------  --------
--------
keydb.dbo.cc_key          256        AES       symmetric default
```

```
key
   1          1   system encr passwd             0           0
keydb.dbo.sample_key1      128          AES         symmetric key
   0          0   system encr passwd             1           4
keydb1.tinnap.tinnap_key   128          AES         symmetric key
   0          1   system encr passwd             0           0
keydb1.tinnap.tinnap_key1  128          AES         symmetric
default key
   0          1       user password              1           3
keydb1.dbo.sample_key1     192          AES         symmetric key
    1           1        login
passwd                1             2
```

- **Display Properties of All Base Encryption Keys Similar to %key in All Available Databases – all_dbs** indicates that information on keys across all databases is required. You must have **sso_role** to use the **all_dbs** parameter.

```
sp_encryption helpkey, '%key', all_dbs
```

```
Db.Owner.Keyname         Key Length   Key Algorithm  Key Type
Pad   Init Vector   Protected By       Key Recovery    #of Key
Copies
------------------------   ----------   -----------  ------------
--------
-----  ----------   ---------------    --------------  ----------
------
keydb.dbo.cc_key              256     AES      symmetric default
key
   1          1           system encr passwd     0            0
keydb1.tinnap.tinnap_key         128     AES          symmetric
key
    0           1             system encr
passwd    0               0
```

- **Display Properties of All Base Encryption Keys With Names Similar to "tinnap%" in Database Run by SSO –** Displays properties of all base encryption keys with names similar to "tinnap%" in the current database when run by SSO, key custodian, or DBO:

```
sp_encryption helpkey, "tinnap%"
```

```
Key Name  Key Owner  Key Length   Key Algorithm          Key Type
   Pad  Init Vector   Protected By        Key Recovery     # of Key
Copies
---------  --------    ---------  --------------  ---------------
--------
   ---  -----------   -------------------  ------------    ------
---------
tinnap_key   tinnap          128            AES   symmetric key
    0         1   system encr passwd           0              0
tinnap_key1  tinnap         128           AES   symmetric default
key         0             1           user
passwd               1                  3
```

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

```
sp_encryption helpkey, "tinnap%"
```

```
Key Name           Key Owner  Key Type
------------------  ---------  ------------
tinnap_key          tinnap     symmetric key
tinnap_key1         tinnap     symmetric default key
```

- **Display Information on Key Copies Using key_copy as Third Parameter –** Displays information on key copies using **key_copy** as the third parameter. Enter null instead of value for *keyname* for the second parameter to see information on all key copies. You can use pattern-matching characters in *keyname* (see the previous example):

  ```
  sp_encryption helpkey, tinnap_key1, key_copy
  ```

  ```
  Owner.Keyname        Assignee      Protected by      Key Recovery
  ------------------     ---------       ----------------       ---
  -------
  tinnap.tinnap_key1     joesmp          user passwd           0
  tinnap.tinnap_key1     samcool         user passwd           1
  tinnap.tinnap_key1     billyg          user passwd           0
  ```

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

  ```
  sp_encryption helpkey, tinnap_key1, key_copy
  ```

  ```
  Owner.Keyname         Assignee       Protected by          Key
  Recovery
  ------------------     ---------       ----------------      ----
  --------
  tinnap.tinnap_key1        joesmp             user
  passwd                 0
  ```

- **Display All Encrypted Columns in All Available Databases Encrypted by Keys from Database –** Use the **display_cols** parameter to show all encrypted columns in all available databases encrypted by keys from the current database. If you do not have the **sso_role**, the query displays only the encrypted columns in the current database encrypted by keys from the current database.

  You can use pattern matching characters or *key_name* for the second parameter. If you use pattern matching characters for *key_name* as **sso_role**, the query displays all encrypted columns in all available databases encrypted by the pattern matching *key_name*. If you use *key_name* for the second parameter and have the **sso_role**, displays all encrypted columns in all available databases encrypted by the specified *key_name*:

  ```
  sp_encryption helpkey, null, display_cols
  ```

  ```
  Key Name     Key Owner  Database Name  Table Owner  Table
  Name  Column Name
  ----------   ---------  -------------  ----------   ----------  -
  ---------
  tinnap_key      tinnap        testdb1      tinnap          t3
        c3
  tinnap_key1     tinnap        testdb1      tinnap          t4
        c4
  sample_key1        dbo          coldb         dbo          t1
        c1
  ```

```
sample_key1        dbo          coldb      billyg            t2
      c2
```

- **Display All Keys, Key Copies Encrypted With System Encryption Password in Database –** Displays all keys and key copies encrypted with the system encryption password in the current database. If you do not have these privileges, the query displays the keys owned by or assigned to the user which are encrypted with the system encryption password:

```
sp_encryption helpkey, system_encr_passwd, display_keys
```

```
Owner.Keyname                      Assignee
---------------                     -------------
dbo.cc_key                                    NULL
dbo.sample_key1                               NULL
dbo.sample_key1                             tinnap
```

- **Display All Base Keys Owned by Users in Database –** When run by the database owner or a user with **keycustodian_role** or **sso_role**, the **helpuser** parameter displays all base keys owned by users in the current database.

```
sp_encryption helpuser
```

```
Owner.Keyname              Protected by
---------------               -------------------
tinnap.tinnap_key            system encr passwd
tinnap.tinnap_key1                  user passwd
dbo.sample_key1                     login passwd
```

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

```
sp_encryption helpuser
```

```
Owner.Keyname              Protected by
---------------               -------------------
tinnap.tinnap_key            system encr passwd
tinnap.tinnap_key1            user passwd
```

- **Display Key Copies Assigned to One or More Users –** The database owner or a user with **keycustodian_role** or **sso_role** can use the **key_copy** parameter with the **helpuser** parameter to display key copies assigned to one or more users in the current database. You can use pattern-matching characters for the *user* parameter. This shows the key copies of all users in the current database:

```
sp_encryption helpuser, NULL, key_copy
```

```
Owner.Keyname             Assignee           Protected by      Key
Recovery
---------------------      -----------      ----------------  ---
--------
dbo.sample_key1                  tinnap          login
passwd           0
tinnap.tinnap_key1               joesmp           user
passwd           0
dbo.sample_key1                  joesmp          login
passwd           1
```

```
tinnap.tinnap_key1              samcool           user
passwd           1
tinnap.tinnap_key1               billyg            user
passwd          0
```

If you are not the database owner and do not have **keycustodian_role** or **sso_role**, this query displays the copies of any keys you own and the key copies that other key owners have assigned to you. For example, when user "tinnap" runs this query:

```
sp_encryption helpuser, NULL, "key_copy"
```

```
Owner.Keyname            Assignee         Protected by       Key
Recovery
---------------------     -----------      ----------------   ---
------
dbo.sample_key1              tinnap        login passwd         0
tinnap.tinnap_key1           joesmp         user passwd         0
tinnap.tinnap_key1             samcool          user
passwd           1
tinnap.tinnap_key1               billyg            user
passwd         0
```

- **Display All Encrypted Columns in Database and Keys Used to Encrypt Columns –** If you are the database owner or a user with **keycustodian_role** or **sso_role**, **helpcol** displays all encrypted columns in the current database and the keys used to encrypt the columns. If you do not have these privileges, **helpcol** displays `keyid` instead of the *key_name* if the encryption key is in a different database:

```
sp_encryption helpcol
```

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

- **Display All Encrypted Columns or Specific Encrypted Column in a Table –** Include the **helpcol** parameter with the *table_name* and *column_name* parameters to display all encrypted columns or a specific encrypted column in a given table. When run by a user with **sso_role**, the query below displays all encrypted columns in table `t3` in the current database and the keys used to encrypt the columns across all available databases. When run by a user without `sso_role`, this query displays the key's ID instead of its name if the key is not in the current database. The second parameter can have a combination of [*database_name*.][*table_name*.][*column_name*]:

```
sp_encryption helpcol, t3
```

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

- **Display System Encryption Password Properties for Each Database –** Displays the system encryption password properties for each database (you must have **sso_role to** run this query):

```
sp_encryption helpkey, system_encr_passwd, all_dbs
```

```
Database  Type of system_encr_passwd   Last modified
by            Date
--------  --------------------------    ---------------   -------
---------
master                  persistent              sa  Aug 26 2008
10:05AM
```

- **Display All Encryption Keys Encrypted With Master Key in Database –** Displays all encryption keys encrypted with the master key in the current database (you must have sso_role, keycustodian_role, or be the database owner to run this query):

```
sp_encryption helpkey,'master',display_keys
```

```
Owner.Keyname      Assignee
---------------- ----------
user1.key_dual          NULL
user1.key_mst           NULL
user4.key_dC_pwd        NULL
user4.key_dC_pwd       user5
user4.key_dC_pwd       user6
user4.key_dC_pwd     KC_tdb1
```

- **Display Name and Location of Current Master Key Start-Up File –** Displays the name and location of the current master key start-up file configured for the current server:

```
sp_encryption mkey_startup_file
```

```
Msg 19956, Level 16, State 1: Procedure 'sp_encryption', Line 298:
The current master key startup file is:'/sybase/release/ASE-150/
init/ase_encrcols_mk_l157.dat'.
```

- **Display Encrypted Stored Procedures –** Displays three stored procedures that are encrypted with key `syb_syscommkey_123456`, and are owned by user1 and user2:

```
sp_encryption  helpkey, "syb_syscommkey%", display_objs
```

```
Key Name                            Key Owner     Database Name
        Object owner           Object Name
------------                         -------       -----------
        ------------            -----------
syb_syscommkey_1234567890ab              dbo           testdb
              user1            sp_mysproc1
syb_syscommkey_abcdefghijkl123456       dbo           testdb
              user1            sp_mysproc2_
syb_syscommkey_ABCDEF123456              dbo           testdb
              user2            sp_mysproc3
```

Usage

- When a database is fully encrypted, **sp_encryption** reports a key type called "`database encryption key`".
- The privileges granted to the user who runs **sp_encryption** determines the output. See for more information.

- If you run **sp_encryption helpkey** and no keys are present in the database, you see an informational message.
- You must specify the *key_copy* parameter to get information about key copies. If you do not specify the *key_copy* parameter, **sp_encryption** returns information only about base keys.
- If *keyname* is NULL in **sp_encryption helpkey, keyname, key_copy**, lists all the key copies in the current database for a SSO, key custodian, or DBO. If it is run by a user without privileges, it lists all the key copies assigned to the user in the current database and all key copies of the keys owned by the user in the current database.
- For **sp_encryption helpcol**, *column_name* uses the form ***name.name.name***, where:
  - *name* – if **sp_encryption** finds no tables of this name, it looks for all columns of that name.
  - ***name.name*** – is ***owner.table***. If **sp_encryption** finds no tables of this name, it looks for a single column named *table.column*.
  - ***name.name.name*** – is ***owner.table.name***.

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

  The output for **sp_encryption helpcol**, *column_name* is ***owner.table.column*** and ***db.owner.keyname***. The *keyname* is expressed as ***database.keyid*** when run by non-SSO users, and the key is present in a different database from the encrypted column. The result set is sorted by ***owner.table.column***.

The restrictions for **sp_encryption** are:

- Only an SSO can run **sp_encryption helpkey [, *keyname* | wildcard], all_dbs** to get the properties of keys in all databases. If a user without the **sso_role** runs this command, they receive an "unauthorized user" error message. If no keys qualify the keyname or wildcard, the SAP ASE server returns a message stating `'There are no encryption keys (key copies) like keyname in all databases'.`
- When the SSO runs **sp_encryption helpkey, *keyname*, *display_cols***, it lists all columns across all available databases encrypted by *keyname*. If it is run by a user without privileges, it lists the columns in the current database encrypted by *keyname*.

  If the SSO runs **sp_encryption helpkey, *keyname*, *display_cols*** and the *keyname* value is NULL, it displays all encrypted columns across all available databases. When run by a user without privileges, it displays all encrypted columns in the current database.
- If an SSO, key custodian, or DBO runs **sp_encryption helpuser, *user_name*, key_copy** without specifying a *user_name* and *key_copy* for the **helpuser** parameter, it lists all the base keys owned by all users in the current database. If **sp_encryption** is run by a user without privileges without specifying a *user_name* or *key_copy*, it displays the base keys owned by the current user.

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

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

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

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

Permissions

The permission checks for **sp_encryption** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled: • **downgrade_kek_size** – You must be a user with `manage security configuration` privilege. • **help/help_key system_encr_passwd, display_keys** – You must be a user with `manage column encryption key` privilege. Any user can see their own key. • **help/help_key system_encr_passwd** – You must be a user with `manage column encryption key` privilege. • **help/help_key master key/ dual master key, display_keys** – You must be a user `manage master key` privilege. • **help/help_key keyname/wild card, display_cols** – You must be a user with `use any database` privilege for cross database check. Any user for the current database. • **help/help_key service keyname, display_objs** –You must be a user with `manage service key` privilege. • **help/help_key keyname/wild card, all_dbs** – You must be a user with the following privilege depending the key type: • • column encryption key – `manage column encryption key` • master key – `manage master key` • service key – `manage service key` • For cross-database checks, one of the above three, and `use any database` permission. • **help/help_key keyname wildcard** – You must be a user with the following privilege depending the key type: • • column encryption key – `manage column encryption key` • master key – `manage master key` • service key – `manage service key` For non-privileged users, limited encyption key information is displayed. • **help/help_key keyname wildcard, key_copy** – You must a user with the following privilege depending on the key type: • column encryption key – `manage column encryption key` • master key – `manage master key` • helpcol – You must be a user with `use any database` privilege for cross database checks. • **helpextpassword** – You must be a user with `manage service key` privilege. • **helpuser username/wildcard, [key_copy/login_passwd_check]** – You must be a user with `manage any encryption key` privilege. Non-privilege users can see their own key. • **mkey_startup_file** – You must be a user with with `manage security configuration` privilege.

Setting	Description
	• **system_encr_passwd** – You must be a user with `manage column encryption key` privilege. • **verify_downgrade** – You must be a user with `manage security configuration` privilege.
Disabled	With granular permissions disabled: • **downgrade_kek_size** – You must be a user with **sso_role** or **keycustdian_role**. • **help/help_key system_encr_passwd, display_keys** – You must be the database owner, a user with **sso_role**, or a user with **keycustdian_role**. Any user can see their own key. • **help/help_key system_encr_passwd** – You must be the database owner, a user with **sso_role**, or a user with **keycustdian_role**. • **help/help_key master key/ dual master key, display_keys** – You must be the database owner, a user with **sso_role**, or a user with **keycustdian_role**. • **help/help_key keyname/wild card, display_cols** – You must be a user with **sso_role** for cross database check. Any user for the current database. • **help/help_key service keyname, display_objs** – You must be a user with **sso_role** or a user with **keycustodian_role**. • **help/help_key keyname/wild card, all_dbs** – You must be a user with **sso_role**. • **help/help_key keyname wildcard** – You must be the database owner, a user with **sso_role**, or a user with **keycustdian_role**. • **help/help_key keyname wildcard, key_copy** – You must be the database owner, a user with **sso_role**, or a user with **keycustdian_role**. • For: • Non-privileged users – displays only **key_copy** information • For privileged users – displays the encryption key and key_copy information for all users in the database • helpcol – You must be a user with **sso_role**. • **helpextpassword** – You must be a user with **sso_role**. • **helpuser username/wildcard, [key_copy/login_passwd_check]** – You must be the database owner, a user with **sso_role**, or a user with **keycustdian_role**. Non-privilege users can see their own keys. • **mkey_startup_file** – You must be a user with **sso_role**. • **system_encr_passwd** – You must be a user with **sso_role** or **keycustdian_role**. • **verify_downgrade** – You must be a user with **sso_role** or **keycustdian_role**.

sp_engine

Enables you to bring an engine online or offline. In threaded mode, use **alter thread pool** to bring engines online.

Syntax

```
sp_engine {"online" | [offline | can_offline] [, engine_id] |
["shutdown", engine_id]}
```

Parameters

- **"online" –** bring an engine online. The value of **sp_configure "max online engines"** must be greater than the current number of engines online. Because **"online"** is a reserved keyword, you must use quotes.

  In threaded mode, **online** increases the thread count for `syb_default_pool` by 1.
- **offline –** bring an engine offline. You can also use the *engine_id* parameter to specify a specific engine to bring offline.

  In threaded mode, **offline** decreases the thread count for `syb_default_pool` by 1.
- **can_offline –** returns information on whether an engine can be brought offline. **can_offline** returns the SAP ASE tasks with an affinity to this engine (for example, during Omni or java.net tasks) if its state is **online**. If you do not specify an *engine_id*, the command describes the status of the engine in `sysengines` with the highest *engine_id*.

  In threaded mode, can_offline succeeds only if the total number of engines is less than the total number of threads in `syb_default_pool` and the total number of threads in `syb_default_pool` is greater than or equal to 2.
- ***engine_id* –** the ID of the engine. The *engine_id* parameter is optional. If you do not specify an *engine_id*, **sp_engine** uses the incremented or decremented value for *engine_id* for the value of engine found within `sysengines`. That is, if your system uses engines 0, 1, 2, and 3, and you do not specify an engine ID, **sp_engine** takes engine ID 3 offline, then engine ID 2, and so on.

  This parameter is ignored in threaded mode.
- **"shutdown" –** Forces an engine offline. If there are any tasks with an affinity to this engine, they are killed after a five-minute wait. You must use quotes, as **shutdown** is a reserved keyword.

Examples

- **Example 1 –** Brings engine 1 online. Messages are platform specific (this example uses Sun Solaris):

```
sp_engine "online", 1
```

```
02:00000:00000:2001/10/26 08:53:40.61 kernel  Network and device
connection
limit is 3042.
02:00000:00000:2001/10/26 08:53:40.61 kernel  SSL Plus security
modules
loaded successfully.
02:00000:00000:2001/10/26 08:53:40.67 kernel  engine 2, os pid
8624  online
02:00000:00000:2001/10/26 08:53:40.67 kernel  Enabling Sun Kernel
asynchronous disk I/O strategy
00:00000:00000:2001/10/26 08:53:40.70 kernel  ncheck: Network
fc0330c8
online
```

- **Example 2 –** Describes the steps in taking an engine offline that is currently running tasks with an affinity for this engine:

  ```
  select engine, status from sysengines
  ```

  ```
  engine    status
  ------    ------
  0         online
  1         online
  2         online
  3         online
  ```

  If you bring engine 1 offline:

  ```
  sp_engine offline, 1
  ```

  ```
  The following task(s) will affect the offline process:
  spid: 19 has outstanding ct-lib connections.
  ```

  And then run the same query as above, it now shows that engine 1 is in an offline state:

  ```
  select engine, status from sysengines
  ```

  ```
  engine    status
  ------    ------
  0         online
  1         in offline
  2         online
  3         online
  ```

  As soon as the task that has an affinity to engine 1 finishes, the SAP ASE server issues a message similar to the following to the error log:

  ```
  02:00000:00000:2001/10/26 09:02:09.05 kernel  engine 1, os pid
  8623  offline
  ```

- **Example 3 –** Determines whether engine 1 can be brought offline:

  ```
  sp_engine can_offline, 1
  ```

- **Example 4 –** Takes engine 1 offline:

  ```
  sp_engine offline, 1
  ```

  The SAP ASE server eventually returns a message similar to the following:

```
01:00000:00000:2001/11/09 16:11:11.85 kernel  Engine 1 waiting
for
affinitated process(es) before going offline
01:00000:00000:2001/11/09 16:11:11.85 kernel  Process 917518 is
preventing
engine 1 going offline
00:00000:00000:2001/11/09 16:16:01.90 kernel  engine 1, os pid
21127  offline
```

- **Example 5 –** Shuts down engine 1 :

```
sp_engine shutdown, 1
```

Usage

- As **sp_engine** works only in process mode, the SAP ASE server issues an error message if you run **sp_engine** in threaded mode. Use **alter thread pool** in threaded mode.
- You cannot take offline or shut down engine 0.
- You can determine the status of an engine, and which engines are currently online with the following query:

```
select engine, status from sysengines
where status = "online"
```

- **online** and **shutdown** are keywords and must be enclosed in quotes.
- Engines can be brought online only if **max online engines** is greater than the current number of engines with an **online** status, and if enough CPU is available to support the additional engine.
- **sp_engine** can run in sessions using chained transaction mode if there are no open transactions.
- An **engine offline** command may fail or may not immediately take effect if there are server processes with an affinity to that engine.

Permissions

The permission checks for **sp_engine** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

Using sp_engine “offline” Versus sp_engine “shutdown”

Sometimes when you use **sp_engine “offline"**, the engine does not immediately go offline, and instead appears to be in “dormant” state in the engine table. This is caused by processes that are attached to your engine that cannot be migrated to other engines. When this happens, the engine does not take new work, and consumes minimal CPU cycles. When the process preventing the completion of **engine offline** either end or become available for migration, the engine moves from dormant to fully offline, and disappears from the engine table.

sp_engine "shutdown" is a more aggressive version of the **offline** command. **sp_engine "shutdown"** actively kills any processes that are preventing the engine from going offline, forcing it to shut down.

However, if you use **sp_engine "shutdown"** on an engine that has Client Library™ or Java connections, you see:

```
Engine has outstanding ct-lib/java connections and
cannot be offlined.
```

When this happens, repeat the command again every few minutes until the connections are no longer there, and the engine can shut down.

sp_errorlog

Dynamically changes the path of the error log.

Syntax

```
sp_errorlog “change log“, “new_path“ [,{“jslog true“ | “jslog
false“}]
```

```
sp_errorlog “help”, “change log”
```

Parameters

- ***new_path* –** new path of the error log. Maximum length of *new_path* is 255 characters.
- **jslog true –** the default option. If the Job Scheduler is running, **change log** attempts to change the Job Scheduler Agent log to the directory where the new SAP ASE error log resides. Both logs indicate error messages, if any.
- **jslog false –** do not change the location of the Job Scheduler Agent log.

Examples

- **Example 1 –** To change the SAP ASE error log to use a new location without changing the location of the Job Scheduler log, use:

  ```
  sp_errorlog 'change log',
  '$SYBASE/$SYBASE_ASE/install/new.log', 'jslog false'
  ```

  The SAP ASE error log location is changed to `$SYBASE/$SYBASE_ASE/install/new.log`. However, the location of the Job Scheduler Agent log is not changed.

- **Example 2 –** This example changes the error log location to `$SYBASE/$SYBASE_ASE/install/new.log`.

  If the Job Scheduler Agent is running, the agent log location is also changed to `$SYBASE/$SYBASE_ASE/install/new.log`.

  If the Job Scheduler Agent is not running, SAP ASE does not change the agent log location.You see a message that the agent log location is unchanged.

  ```
  sp_errorlog 'change log',
  '$SYBASE/$SYBASE_ASE/install/new.log', 'jslog true'
  ```

- **Example 3 –** This example changes the SAP ASE error log to `$SYBASE/$SYBASE_ASE/install/new.log`.

  If the Job Scheduler Agent is running, the agent log is also changed to `$SYBASE/$SYBASE_ASE/install/new.log`.

  If the Job Scheduler Agent is not running, SAP ASE does not change the path of the Job Scheduler Agent log. You do not see a message that the agent log location is unchanged.

  ```
  sp_errorlog 'change log',
  '$SYBASE/$SYBASE_ASE/install/new.log'
  ```

Usage

sp_errorlog returns 0 if the switch to the new location is successful. A non-zero return value implies an error.

Note: To pick up the new location of the error log when the server is restarted, update the `-e` argument in the runserver file.

See *Logging Error Messages and Events* in the *Configuration Guide* for information on the `runserver` file.

Permissions

The permission checks for **sp_errorlog** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Use the **errorlog** auditing option to audit changes to the error log administration.

When the **errorlog** audit option is enabled, any change to the error log generates an audit record with `event=127`. To view the event from `sybsecurity`, issue:

```
select * from sybsecurity..sysaudits01 where event=127
```

To enable the errorlog audit option enter:

```
sp_audit 'errorlog', 'all', 'all', 'on'
```

sp_estspace

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, page_size
```

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 contains.
- ***fill_factor*** **–** is the index fillfactor. The default is null, which means that the SAP ASE 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.
- ***pagesize* –** allows you to estimate the space required for a given table—and all of its indexes—if you migrate the table to a server of the specified page size. You can either specify a page size (2048, 4096, 8192, 16384, or 2K, 4K, 8K, 16K) or NULL to use your current page size. If you do not use "K" as a unit specifier, the default for *pagesize* is bytes. Because page allocation allocates the same size page for various objects, the *page_size* value applies to all page types (index, data, text and so on).

Examples

- **Example 1 –** 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:

```
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
```

- **Example 2 –** 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:

```
declare @i int
select @i = avg(datalength(pic)) from au_pix
exec sp_estspace au_pix, 1000, null, null, 16, @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
```

- **Example 3 –** Calculates the size of the `titles` table with 50,000 rows, using defaults for all other values:

```
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
```

- **Example 4 –** Runs after adding a clustered index to the `blurbs` table:

```
declare @i int
select @i = avg(datalength(copy)) from blurbs
exec sp_estspace blurbs, 6, null, null, 16, @i, "16k"
```

```
name                      type        idx_level Pages        Kbytes
------------------------ ----------- --------- ---------    ------
blurbs                    data                0         8       128
blurbs                    text/image          0         6        96
blurbs_ind                clustered           0         1        16
blurbs_ind                clustered           1         1        16

Total_Mbytes
-----------------
0.25

name                      type         total_pages  time_mins
------------------------ ------------ ------------ ------------
blurbs_ind                clustered              10            0
blurbs                    data                    6            0
```

This example is run on a 2K server, and indicates that the `blurbs` table would require .25MB after it is migrated to a 16K server. Below is the same query run on a 16K server, which verifies the .25MB space requirement:

```
declare @i int
select @i = avg(datalength(copy)) from blurbs
exec sp_estspace blurbs, 6, null, null, 16, @i, "16k"
```

```
name                     type         idx_level Pages     Kbytes
------------------------ ------------ --------- --------- ------
blurbs                   data                 0         8    128
blurbs                   text/image           0         6     96
blurbs_ind               clustered            0         1     16
blurbs_ind               clustered            1         1     16

Total_Mbytes
-----------------
0.25

name                    type         total_pages  time_mins
----------------------- ------------ ------------ ----------
blurbs_ind              clustered              10          0
blurbs                  data                    6          0
```

- **Example 5 –** Estimates that, if the `blurbs` table had a thousand rows in it on a 2K server, it would require 1.99MB of space:

```
declare @i int
select @i = avg(datalength(copy)) from blurbs
exec sp_estspace blurbs, 1000, null, null, 16, @i, "2k"
```

```
name                    type         idx_level Pages        Kbytes
----------------------- ------------ --------- ------------ -----
-
blurbs                  data                 0     16           32
blurbs                  text/image           0   1000         2000
blurbs_ind              clustered            0      1            2
blurbs_ind              clustered            1      1            2

Total_Mbytes
------------
1.99

name                    type         total_pages  time_mins
----------------------- ------------ ------------ ---------
blurbs_ind              clustered              18         0
blurbs                  data                 1000         0
```

Usage

To estimate the amount of space required by a table and its indexes:

1. Create the table.
2. Create all indexes on the table.
3. Run **sp_estspace**, giving the table name, the estimated number of rows for the table, and the optional arguments, as needed.

For information about tables or columns, use **sp_help *tablename***.

See also **create index**, **create table** in *Reference Manual: Commands*.

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.

Permissions

Any user can execute **sp_estspace**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_dboption* on page 193
- *sp_help* on page 358

Estimating the Extra Space Required by a Column

If the **auto identity** option is set in a database, the SAP ASE 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:

1. In the `master` database, use **sp_dboption** to turn on the **auto identity** option for the database.
2. Create the table.
3. Run **sp_estspace** on the table and record the results.
4. Drop the table.
5. Turn the **auto identity** option off for the database.

6. Re-create the table.
7. Rerun **sp_estspace** on the table, and record the results.

sp_export_qpgroup

Exports all plans for a specified user and abstract plan group to a user table.

Syntax

```
sp_export_qpgroup usr, group, tab
```

Parameters

- ***usr* –** is the name of the user who owns the abstract plans to be exported.
- ***group* –** is the name of the abstract plan group that contains the plans to be exported.
- ***tab* –** is the name of a table into which to copy the plans. It must be a table in the current database. You can specify a database name, but not an owner name, in the form ***dbname..tablename***. With large identifiers, the total length must be no more than 255 characters.

Examples

- **Example 1 –** Creates a table called `moveplans` containing all the plans for the user “freidak” that are in the `ap_stdout` group:

  ```
  sp_export_qpgroup freidak, ap_stdout, "tempdb..moveplans"
  ```

Usage

sp_export_qpgroup copies plans from an abstract plan group to a user table. With **sp_import_qpgroup**, it can be used to copy abstract plans groups between servers and databases or to assign user IDs to copied plans.

The user table name that you specify cannot exist before you run **sp_export_qpgroup**. The table is created with a structure identical to that of `sysqueryplans`.

sp_export_qpgroup uses **select...into** to create the table to store the copied plans. You must use **sp_dboption** to enable **select into/bulkcopy/pllsort** in order to use **sp_export_qpgroup**, or create the table in `tempdb`.

Permissions

The permission checks for **sp_export_qpgroup** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_copy_all_qplans* on page 176
- *sp_copy_qplan* on page 178
- *sp_dboption* on page 193
- *sp_import_qpgroup* on page 451

sp_extendsegment

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

- **Example 1 –** Extends the range of the segment `indexes` for the database `pubs2` on the database device `dev2`:

  ```
  sp_extendsegment indexes, pubs2, dev2
  ```

Usage

There are additional considerations when using **sp_extendsegment**:

- You cannot extend a segment on a device that already has an exclusive segment, and you cannot extend an exclusive segment on a device that has another segment.

  For example, if you attempt to extend segment `orders_seg` on a device `orders.dat`, which already has an exclusive segment, you see an error message similar to:

  ```
  A segment with a virtually hashed table exists on
  device orders.dat.
  ```

  If you attempt to extend exclusive segment `orders_seg` on device `orders.dat`, which has other segments, you see an error message similar to:

  ```
  You cannot extend a segment with a virtually hashed
  table on device orders.dat, because this device has
  other segments.
  ```
- A segment can be extended over several database devices.
- You can only execute **sp_extendsegment** for the `logsegment` system segment in single-user mode.
- If the `logsegment` segment is extended, any other segments on the device are dropped and the device is used for the log segment exclusively.
- When you extend the `logsegment` segment, the SAP ASE server recalculates its last-chance threshold.
- 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.
- 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.

See also **alter database**, **create index**, **create table** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_extendsegment** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage database` privilege.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addsegment* on page 43
- *sp_dropsegment* on page 287
- *sp_helpdb* on page 394
- *sp_helpdevice* on page 402
- *sp_helpsegment* on page 430
- *sp_placeobject* on page 577

sp_extengine

(Only when you have a valid SAP ASE EJB Server site license at your site) Starts and stops EJB Server. Displays status information about EJB Server.

Syntax

```
sp_extengine 'ejb_server', '{ start | stop | status }'
```

Parameters

- ***ejb_server*** **–** the logical name of the EJB Server.
- **start –** starts the EJB Server.
- **stop –** shuts down the EJB Server.
- **status –** displays status information about the EJB Server.

Examples

- **Example 1 –** Informs user that the EJB Server SYB_EJB is running:

  ```
  sp_extengine 'SYB_EJB', 'status'
  ```

  ```
  Enterprise java bean server is up and running.
  ```

- **Example 2 –** Shuts down the EJB Server SYB_EJB:

  ```
  sp_extengine 'SYB_EJB', 'stop'
  ```

Usage

See the *EJB Server User's Guide*.

Permissions

The permission checks for **sp_extenengine** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_extrapwdchecks

A custom stored procedure that can contain user-defined logic for password complexity checks. You can configure **sp_extrapwdchecks** according to your security needs. Install **sp_extrapwdchecks** in the `master` database.

Syntax

```
sp_extrapwdchecks caller_password, new_password, login_name
```

Parameters

- ***caller_password* –** specifies the current password.
- ***new_password* –** specifies the new password being set.
- ***login_name* –** specifies the login name associated with the password being changed or added.

Usage

- **sp_extrapasswordchecks** must use **raiserror** to signal a failure to the SAP ASE server. Use **sp_addmessage** to add error message for this failure in the SAP ASE server.

  Note: Do not use **raiserror** to get the expected behaviour. **raiserror** updates the ***@@error*** global variable. ***@@error*** is also updated each time you execute a T-SQL statement, including **print** and **if**. If **raiserror** is followed by any T-SQL statement, ***@@error*** gets overwritten, and **sp_extrapwdchecks** fails to return an error for a failed password if **raiserror** is followed by any TSQL statement.

- **sp_extrapwdchecks** allows NULL values for **caller_password** and **loginame** parameters. The **caller_password** parameter is NULL when :

- The system security officer creates a new login account using **create login** command.
- The system security officer modifies the login account's password using **alter login … modify password** command.

The **loginame** parameter is NULL when:

- The system security officer creates a new login account using the **create login** command.

sp_familylock

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.
- ***fpid2* –** is the SAP ASE process ID number for another lock.

Examples

- **Example 1 –** Displays information about the locks held by all members of the family with an `fid` of 5:

```
sp_familylock 5
fid spid locktype      table_id  partitionid page dbname
class           context
--- ---- ----------    --------- -----------
---- ------ --------------- ------------------------
  5   5  Sh_intent     176003658          0    0 userdb Non cursor
lock Sync-pt duration request
  5   5  Sh_intent-blk 208003772          0    0 userdb Non cursor
lock Sync-pt duration request
  5   6  Sh_page      208003772          0 3972 userdb Non cursor
lock Sync-pt duration request
  5   7  Sh_page      208003772          0 3973 userdb Non cursor
lock Sync-pt duration request
  5   8  Sh_page      208003772          0 3973 userdb Non cursor
lock Sync-pt duration request
```

- **Example 2 –** Displays information about partition-level locks:

```
sp_familylock
spid  locktype            table_id     partitionid     page   row…
----  ------------------  -----------  -------------   -----  ----
0     Ex_partition          576002052      576004423       0     0
```

```
0       Sh_partition_intent   1417053053     1417053053
0       0
```

Table lock and fine-grained lock values for `partitionid` are 0. `partitionid` is populated only for partition-level locks.

Usage

There are additional considerations when using **sp_familylock**:

- **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 SAP ASE 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:
  - 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:
  - "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 holds 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.

See also **kill**, **select** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_familylock**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_lock* on page 483
- *sp_who* on page 736

sp_find_qplan

Finds an abstract plan, given a pattern from the query text or plan text.

Syntax

```
sp_find_qplan pattern [, group ]
```

Parameters

- ***pattern* –** is a string to find in the text of the query or abstract plan.
- ***group* –** is the name of the abstract plan group.

Examples

- **Example 1 –** Reports on all abstract plans that have the string “from titles” in the query:

```
sp_find_qplan "%from titles%"
```

```
gid id          text
--- ----------- ------------------------------------------------------
-
2   921054317 select count(*) from titles
2   921054317
         ( plan
        ( i_scan t_pub_id_ix titles )
        ( )
)
( prop titles
        ( parallel 1 )
        ( prefetch 16 )
        ( lru )
)
5   937054374 select type, avg(price) from titles group by type
5   937054374
         ( plan
        ( store Worktab1
                ( i_scan type_price titles )
        )
        ( t_scan ( work_t Worktab1 ) )
)
( prop titles
        ( parallel 1 )
        ( prefetch 16 )
        ( lru )
```

- **Example 2 –** Finds all plans that include a table scan operator:

```
sp_find_qplan "%t_scan%"
```

- **Example 3 –** Uses the range pattern matching to look for strings such as “table1”, “table2”, and so forth, in plans in the `dev_plans` group:

```
sp_find_qplan "%table[0-9]%", dev_plans
```

Usage

There are additional considerations when using **sp_find_qplan**:

- Use **sp_find_qplan** to find an abstract plan that contains a particular string. You can match strings from either the query text or from the abstract plan text.
- For each matching plan, **sp_find_qplan** prints the group ID, plan ID, query text and abstract plan text.
- If you include a group name, **sp_find_qplan** searches for the string in the specified group. If you do not provide a group name, **sp_find_plan** searches all queries and plans for all groups.

- You must supply the “%” wildcard characters, as shown in the examples, unless you are searching for a string at the start or end of a query or plan. You can use any Transact-SQL pattern matching syntax, such as that shown in Example 3.
- The text of queries in `sysqueryplans` is broken into 255-byte column values. **sp_find_qplan** may miss matches that span one of these boundaries, but finds all matches that are less than 127 bytes, even if they span two rows.

Permissions

The permission checks for **sp_find_qplan** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege or `monitor qp performance` privilege. Any user can execute **sp_find_qplan** to find and display report plans that they own.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**. Any user can execute **sp_find_qplan** to find and display report plans that they own.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_help_qpgroup* on page 370
- *sp_help_qplan* on page 373

sp_fixindex

sp_fixindex repairs corrupt indexes on system tables. It can rebuild a specified index or all indexes on the table. **sp_fixindex** rebuilds the data layer if the target table has a placement or clustered index (it reclaims the unused space in the data layer while working on the placement or clustered index of a system table).

Syntax

```
sp_fixindex database_name, table_name [, index_id | null]
    [, index_name | null]  [, force_option]
```

Parameters

- ***dbname* –** is the database name
- ***tabname* –** is the table name
- ***indiex_id* –** is the ID of the index you want to fix
- ***index_name* –** indicates the index that needs to be processed. If a NULL value is used, the index associated with *index_id* is rebuilt. If *index_id* is also a NULL value, all the indexes in the system table are rebuilt
- **force_option –** forces the SAP ASE server to rebuild the system table index in `tempdb`. **sp_fixindex** without the **force_option** forces the database specified by *database_name* to be in single-user mode, which is not possible for `tempdb`. Although the **force_option** allows you to rebuilt system catalogs in `tempdb`, it should not be used for user databases.

Examples

- **Example 1 –** Repairs the clustered index on the `sysprocedures` table of the `pubs2` database:

  ```
  sp_fixindex pubs2, sysprocedures, 1
  ```

- **Example 2 –** Rebuilds the index with an index ID of 2 on `testdb..sysprocedures`:

  ```
  sp_fixindex 'testdb', 'sysprocedures', 2
  ```

- **Example 3 –** Rebuilds the index `csysprocedures` in the `testdb..sysprocedures` system table:

  ```
  sp_fixindex 'testdb', 'sysprocedures', null, 'csysprocedures'
  ```

- **Example 4 –** Rebuilds all available indexes on the `sysprocedures` table in `testdb`. If the table has clustered or placement index, **sp_fixindex** reclaims the unused space by removing the garbage present in data pages (that is, it rebuilds the data pages):

  ```
  sp_fixindex 'testdb', 'sysprocedures'
  ```

- **Example 5 –** Rebuilds the index with an with an index ID of 2 on `tempdb..sysprocedures`:

  ```
  sp_fixindex 'tempdb', 'sysprocedures', 2, null, 1
  ```
- **Example 6 –** Rebuilds the index `csysprocedures` for the table `tempdb..sysprocedures`:

  ```
  sp_fixindex 'tempdb', 'sysprocedures', null,
      'sysprocedures', 1
  ```
- **Example 7 –** Rebuilds all indexes on `sysprocedures` in `tempdb`:

  ```
  sp_fixindex 'tempdb', 'sysprocedures', null, null, 1
  ```

Usage

Before you run **sp_fixindex**, make sure your database is in single-user mode, and is reconfigured to allow updates to system tables.

After you run **sp_fixindex**:

- Use the **dbcc checktable** command to verify that the corrupted index has been fixed
- Disallow updates to system tables using **sp_configure**
- Turn off single-user mode

Do not run **sp_fixindex** on user tables.

Warning! You cannot use **sp_fixindex** against the clustered index on `sysindexes`. If you do, **sp_fixindex** returns the following error message:

```
Cannot re-create index on this table.
```

For more information on **sp_fixindex**, see:

- *Encyclopedia of Tasks* in the *Troubleshooting and Error Message Guide*.
- *Indexing for Performance* in the *Performance and Tuning Guide: Basics*.

Permissions

The permission checks for **sp_fixindex** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the databse owner or a user with `own database` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_flushstats

Flushes statistics from in-memory storage to the `systabstats` and `sysstatistics` system tables.

Syntax

```
sp_flushstats [objname]
```

Parameters

- ***objname* –** is the name of a table.

Examples

- **Example 1 –** Flushes statistics for the `titles` table:

  ```
  sp_flushstats titles
  ```

Usage

There are additional considerations when using **sp_flushstats**:

- When you do not specify a table with the *objname* parameter, **sp_flushstats** acts at the database level.
- Some statistics in the `systabstats` table are updated in in-memory storage locations and flushed to `systabstats` periodically, to reduce overhead and contention on `systabstats`.
- If you query `systabstats` using SQL, executing **sp_flushstats** guarantees that in-memory statistics are flushed to `systabstats`.

- The **optdiag** command always flushes in-memory statistics before displaying output.
- The statistics in `sysstatistics` are changed only by data definition language commands and do not require the use of **sp_flushstats**.
- The in-memory datachange counters are persistently stored in `sysstatistics`. These are flushed to disk when **sp_flushstats** is executed.

Permissions

The permission checks for **sp_flushstats** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `monitor qp performance` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_forceonline_db

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

- **Example 1 –** Allows the system administrator access to all suspect pages in the `pubs2` database:

  ```
  sp_forceonline_db pubs2, "sa_on"
  ```

- **Example 2 –** Revokes access to all suspect pages in the `pubs2` database from the system administrator. Now, no one can access the suspect pages in `pubs2`:

  ```
  sp_forceonline_db pubs2, "sa_off"
  ```

- **Example 3 –** Allows all users access to all pages in the `pubs2` database:

  ```
  sp_forceonline_db pubs2, "all_users"
  ```

Usage

There are additional considerations when using **sp_forceonline_db**:

- A page that is forced online is not necessarily repaired. Corrupt pages can also be forced online. The SAP ASE 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**.

Permissions

The permission checks for **sp_forceonline_db** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner or a user with `own database` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_forceonline_page* on page 347
- *sp_listsuspect_db* on page 475
- *sp_listsuspect_page* on page 478
- *sp_setsuspect_granularity* on page 640
- *sp_setsuspect_threshold* on page 643

sp_forceonline_object

Provides access to an index previously marked suspect by recovery.

Syntax

```
sp_forceonline_object dbname, objname, indid,
    {sa_on | sa_off | all_users} [, no_print]
```

Parameters

- ***dbname* –** is the name of the database containing the index to be brought online.
- ***objname* –** is the name of the table.
- ***indid* –** is the index ID of the suspect index being brought online.
- **sa_on –** allows only users with the **sa_role** to access the specified index.
- **sa_off –** revokes access privileges created by a previous invocation of **sp_forceonline_object** with **sa_on**.
- **all_users –** allows all users to access the specified index.

- **no_print** – skips printing a list of other suspect objects after the specified object is brought online.

Examples

- **Example 1 –** Allows a system administrator to access the index with `indid 3` on the `titles` table in the `pubs2` database:

  ```
  sp_forceonline_object pubs2, titles, 3 , sa_on
  ```

- **Example 2 –** Revokes access to the index from the system administrator. Now, no one has access to this index:

  ```
  sp_forceonline_object pubs2, titles, 3, sa_off
  ```

- **Example 3 –** Allows all users to access the index on the `titles` table in the `pubs2` database:

  ```
  sp_forceonline_object pubs2, titles, 3, all_users
  ```

Usage

There are additional considerations when using **sp_forceonline_object**:

- If an index on a data-only-locked table has suspect pages, the entire index is taken offline during recovery. Offline indexes are not considered by the query optimizer. Indexes on allpages-locked tables are not taken completely offline during recovery; only individual pages of these indexes are taken offline. Use **sp_forceonline_page** to bring these pages online.
- Use **sp_listsuspect_object** to see a list of databases that are offline.
- To repair a suspect index, use **sp_forceonline_object** with **sa_on** access. Then, drop and re-create the index.

  Note: If the index is on `systabstats` or `sysstatistics` (the only data-only-locked system tables) call Sybase Technical Support.

- **sp_forceonline_object** with **all_users** cannot be reversed. When an index has been brought online for all users, you cannot take it offline again.
- An index that is forced online is not necessarily repaired, as corrupt indexes can be forced online. The SAP ASE server does not perform any consistency checks on indexes that are forced online.
- **sp_forceonline_object** cannot be used in a transaction.
- **sp_forceonline_object** 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 and indexes online in a single command, use **sp_forceonline_db**.

For more information on recovery fault isolation, see the *System Administration Guide*.

Permissions

The permission checks for **sp_forceonline_object** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner or a user with `own database` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_listsuspect_object* on page 477
- *sp_setsuspect_granularity* on page 640

sp_forceonline_page

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

- **Example 1 –** Allows a system administrator access to page 312 in the `pubs2` database:

  ```
  sp_forceonline_page pubs2, 312, "sa_on"
  ```

- **Example 2 –** Revokes access to page 312 in the `pubs2` database from the system administrator. Now, no one has access to this page:

  ```
  sp_forceonline_page pubs2, 312, "sa_off"
  ```

- **Example 3 –** Allows all users access to page 312 in the `pubs2` database:

  ```
  sp_forceonline_page pubs2, 312, "all_users"
  ```

Usage

There are additional considerations when using **sp_forceonline_page**:

- **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. The SAP ASE server does not perform any consistency checks on pages that are forced online.
- You cannot use **sp_forceonline_page** 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**.

Permissions

The permission checks for **sp_forceonline_page** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner or a user with `own database` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_forceonline_db* on page 343
- *sp_listsuspect_db* on page 475
- *sp_listsuspect_page* on page 478
- *sp_setsuspect_granularity* on page 640
- *sp_setsuspect_threshold* on page 643

sp_foreignkey

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

- **Example 1 –** 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`:

  ```
  sp_foreignkey titles, publishers, pub_id
  ```

- **Example 2 –** 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`:

  ```
  sp_foreignkey orders, parts, part, subpart
  ```

Usage

There are additional considerations when using **sp_foreignkey**:

- **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**.
- You cannot use a Java datatype with **sp_foreignkey**.

See also

alter table, **create table**, **create trigger** in *Reference Manual: Commands*.

Permissions

You must be the owner of the table or view to execute **sp_foreignkey**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_commonkey* on page 157
- *sp_dropkey* on page 275
- *sp_helpjoins* on page 415
- *sp_helpkey* on page 417
- *sp_primarykey* on page 589

sp_freedll

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

- **Example 1 –** Unloads the `sqlsrvdll.dll` DLL:

  ```
  sp_freedll "sqlsrvdll.dll"
  ```

Usage

There are additional considerations when using **sp_freedll**:

- You cannot execute 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.
- You cannot use to update an ESP function in a DLL without shutting down XP Server or the SAP ASE server.
- If you use **sp_freedll** to unload a DLL that is in use, **sp_freedll** succeeds, causing the ESP currently using the DLL to fail.

Permissions

The permission checks for **sp_freedll** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any ESP` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addextendedproc* on page 25
- *sp_dropextendedproc* on page 267

- *sp_helpextendedproc* on page 404

sp_getmessage

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`, `unichar`, `nchar`, `varchar`, `univarchar`, 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

- **Example 1 –** Retrieves message number 20001 from `sysusermessages`:

  ```
  declare @myvar varchar(200)
  exec sp_getmessage 20001, @myvar output
  ```

- **Example 2 –** Retrieves the French language version of message number 20010 from `sysusermessages`:

  ```
  declare @myvar varchar(200)
  exec sp_getmessage 20010, @myvar output, french
  ```

Usage

Any application can use **sp_getmessage**, and any user can read the messages stored in `sysmessages` and `sysusermessages`.

See also

print, **raiserror** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_getmessage**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addmessage* on page 35
- *sp_dropmessage* on page 278

sp_grantlogin

(Windows only) Assigns SAP ASE roles or **default** permissions to Windows 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 user.
- ***group_name* –** is the Windows group name.
- ***role_list* –** is a list of the SAP ASE 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.

Examples

- **Example 1 –** Assigns the SAP ASE **oper_role** to the Windows user "jeanluc":

  ```
  sp_grantlogin jeanluc, oper_role
  ```

- **Example 2 –** Assigns the **default** value to the Windows user "valle". User "valle" receives any permissions that were assigned to her via the **grant** command:

  ```
  sp_grantlogin valle
  ```

- **Example 3 –** Assigns the SAP ASE **sa_role** and **sso_role** to all members of the Windows administrators group:

  ```
  sp_grantlogin Administrators, "sa_role sso_role"
  ```

Usage

There are additional considerations when using **sp_grantlogin**:

- You must create the Windows login name or group before assigning roles with **sp_grantlogin**. See your Windows documentation for details.
- **sp_grantlogin** is active only when the SAP ASE server is running in Integrated Security mode or Mixed mode when the connection is Named Pipes. If the SAP ASE server is running under Standard mode or Mixed mode with a connection other than Named Pipes, use **grant** 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 SAP ASE role. It specifies that the user or group should receive any permissions that were assigned to it via the **grant** command.
- Using **sp_grantlogin** with an existing *login_name* or *group_name* overwrites the user's or group's existing roles.

See also

grant, **setuser** in *Reference Manual: Commands*

Permissions

The permission checks for **sp_grantlogin** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage roles` privilege.

Setting	Description
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addlogin* on page 35
- *sp_displaylogin* on page 234
- *sp_droplogin* on page 278
- *sp_locklogin* on page 488
- *sp_logininfo* on page 497
- *sp_modifylogin* on page 515
- *sp_revokelogin* on page 616
- *sp_role* on page 617

sp_ha_admin

Performs administrative tasks on SAP ASE servers configured with Failover in a high availability system. **sp_ha_admin** is installed with the `installhavss` script on UNIX platforms or the `insthasv` script on Windows.

Syntax

```
sp_ha_admnin [cleansessions | help]
```

Parameters

- **cleansessions –** removes old entries from `syssessions`. Old `syssessions` entries are typically left behind because either the SAP ASE server failed to clean up `syssessions` during a restart, or because a client failed to connect to the SAP ASE server.
- **help –** displays the syntax for **sp_ha_admin**.

Examples

- **Example 1 –** Removes old entries from `syssessions` left by a client connection that did not exit correctly:

```
sp_ha_admin cleansessions
(return status = 0)
```

- **Example 2 –** Displays the syntax for **sp_ha_admin**:

```
sp_ha_admin "help"
```

```
sp_ha_admin Usage: sp_ha_admin command [, option1 [, option2]]
sp_ha_admin commands:
sp_ha_admin 'cleansessions'
sp_ha_admin 'help'
(return status = 0)
```

Usage

There are additional considerations when using **sp_ha_admin**:

- **sp_ha_admin** performs administrative tasks on the SAP ASE server that are configured for Failover in a high availability system. **sp_ha_admin** is not installed using the `installmaster` script; instead, use the `installhavss` script that installs and configures for Failover (`insthasv` on Windows).
- **sp_ha_admin** returns a 0 if it successfully cleaned up `syssessions`, and returns a 1 if it encounters an error.
- **sp_ha_admin** enters a message in the errorlog if it could not remove any entries from `syssessions` (for example, if it could not get a lock on `syssessions`).
- To view all the current entries in `syssessions`, enter:

```
select * from syssessions
```

Permissions

You must be a user with **ha_role** to execute **sp_ha_admin**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_help

Reports information about a database object (any object listed in `sysobjects`) and about system or user-defined datatypes, as well as user-defined functions, computed columns and function-based indexes. Column displays **optimistic_index_lock**.

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, precomputed result sets, stored procedures, logs, rules, defaults, triggers, referential constraints, encryption keys, predicates, and check constraints, but refers to tables when you enable **optimistic_index_lock**. Use owner names if the object owner is not the user running the command and is not the database owner.

Examples

- **Example 1 –** 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):

  ```
  sp_help
  ```

- **Example 2 –** Displays information about a partitioned `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.

```
sp_help
publishersName                    Owner              Object_Type
  Create_date
--------------------- ---------------    -----------     --------
---------
publishers           dbo              user table      Oct 7 2005
11:14AM
Column_name Type  Length  Prec  Scale  Nulls  Default_name  Rule_
name
Access_Rule_name  Computed_Column_object    Identity
----------  ----  ------  ----- -----  -----  ------------  -----
------
----------------  ----------------------    --------
pub_id    char      4   NULL   NULL    0  NULL        pub_idrule
            NULL                     NULL          0
pub_name   varchar   40    NULL    NULL      1          NULL
           NULL                    NULL          0
city       varchar   20    NULL    NULL      1          NULL
           NULL                    NULL          0
state       char      2    NULL    NULL      1          NULL
           NULL                    NULL          0

Object does not have any indexes.
keytype  object      related_objs  object_keys           related_
keys
-------  ------      ------------  -----------           --------
-------
primary  publishers  -- none
--    pub_id,*,*,*,*,*,*,*  *,*,*,*,*,*,*,*

name        type       partition_type  partitions  partition_keys
----------  --------   --------------  ----------  --------------
publishers  base table  roundrobin              3  NULL

partition_name         partition_id   pages  segment  create_date
------------------      -------------- -----  -------  ----------
-------
publishers_608002166   608002166        1  default  Oct 13 2005
11:18AM
publishers_1116527980  1116527980       1  default  Oct 13 2005
11:18AM
publishers_1132528037  1132528037       1  default  Oct 13 2005
11:19AM

Partition_Conditions
--------------------
NULL

Avg_pages  Max_pages  Min_pages  Ratio(Max/Avg)  Ratio(Min/Avg)
---------  ---------  ---------  --------------  --------------
        1          1          1  1.0000000       1.0000000

Lock scheme Allpages
The attribute 'exp_row_size" is not applicable to tables with
allpages lock scheme.
```

```
exp_row  reservepagegap  fillfactor  max_rows_per_page  identity_
gap
-------  --------------  ----------  -----------------  ---------
---
      0               0           0                  0          0

concurrency_opt_threshold  optimistic_index_lock  dealloc_first_t
xtpg
-------------------------  ---------------------  ---------------
----
                        0                      0                0
```

- **Example 3 –** Displays information about a partitioned `titles` table:

```
sp_help
titlesName           Owner              Object_Type    Create_date
---------------- ---------------    -----------    --------------
-------
titles          db               user table    Oct 7 2005 11:14AM
(1 row
affected)Column_name Type Length Prec Scale Nulls Default_name Ru
le_name    Access_Rule_name      Identity
----------- ---- ------ ---- ----- ----- ------------
---------    -----------------
      --------
title_id    tid       6
NULL NULL      0 NULL         title_idrule NULL
       0
title   varchar     80 NULL NULL     0 NULL        NULL        NULL
        0
type      char     12 NULL NULL     0 typedflt    NULL        NULL
        0
pub_id    char      4 NULL NULL     1 NULL        NULL        NULL
        0
price    money      8 NULL NULL     1 NULL        NULL        NULL
        0
advance  money      8 NULL NULL     1 NULL        NULL        NULL
        0
total_sales int       4 NULL NULL      1 NULL         NULL
  NULL
        0
notes   varchar     200 NULL NULL      1 NULL         NULL
  NULL
        0
pubdate datetime       8 NULL NULL      0 datedflt     NULL
  NULL
        0
contract   bit      1 NULL NULL     0 NULL        NULL        NULL
        0
index_name      index_description                      index_keys

Object has the following indexes

index_name  index_keys      index_description  index_max_rows_per
_page
index_fillfactor    index_reservepagegap  index_created      inde
x_local
```

```
----------  ----------     -----------------  ------------------
------
----------------    ---------------------  ------------      ----
------
title_idx   total_sales     clustered                              0
             0                      0  Oct 13 2005 5:20PM  Local
Index

index_ptn_name        index_ptn_seg
--------------------  ---------------
p1                     default
p2                     default
p3                     default
title_idx_98505151     default

keytype   object       related_object  object_keys

related_keys
--------  ---------  --------------------------------------------
---
---------------------
foreign   roysched    titles         title_id, *, *, *, *, *, *, *
title_id, *, *, *, *, *, *, *
foreign   salesdetail  titles         title_id, *, *, *, *, *, *,
* title_id, *, *, *, *, *, *
foreign   titleauthor  titles         title_id, *, *, *, *, *, *,
* title_id, *, *, *, *, *, *
foreign   titles      publishers      pub_id, *, *, *, *, *, *, *
pub_id, *, *, *, *, *, *, *
primary   titles      -- none --      title_id, *, *, *, *, *, *,
*
*, *, *, *, *, *, *, *

name    type        partition_type  partitions  partition_keys
----    ----------  --------------  ----------  --------------
titles  base table  range                    4  pubdate

partition_name  partition_id  pages  segment  create_date
--------------  ------------  -----  -------  -------------------
q1              937051343         1  default  Oct 13 2005 5:20PM
q2              953051400         1  default  Oct 13 2005 5:20PM
q3              969051457         1  default  Oct 13 2005 5:20PM
q4              985051514         1  default  Oct 13 2005 5:20PM


Partition_Conditions
--------------------
VALUES <= ("3/31/2006")
VALUES <= ("6/30/2006")
VALUES <= ("9/30/2006")
VALUES <= ("12/31/2006")
VALUES <= ("3’31’2006")

Avg_pages  Max_pages  Min_pages  Ratio(Max/Avg)  Ratio(Min/Avg)
---------  ---------  ---------  --------------  --------------
        1          1          1  1.000000        1.000000
```

```
Lock scheme Allpages
The attribute 'exp_row_size" is not applicable to tables with
allpages lock scheme.

exp_row  reservepagegap  fillfactor  max_rows_per_page  identity_
gap
-------  --------------  ----------  -----------------  ---------
---
      0               0           0                  0          0

concurrency_opt_threshold  optimistic_index_lock  dealloc_first_t
xtpg
-------------------------  ---------------------  ---------------
----
                       0                      0               0
```

- **Example 4 –** Displays information about the trigger `marytrig` owned by user “mary”. The quotes are needed, because the period is a special character:

```
sp_help "mary.marytrig"
```

```
Name         Owner              Object_type
------------ ------------------ ----------------
marytrig     mary              trigger

Data_located_on_segment When_created
----------------------- --------------------------
not applicable          Mar 20 2002  2:03PM
```

- **Example 5 –** Displays information about the system datatype `money`:

```
sp_help money
```

```
Type_name  Storage_type  Length  Prec   Scale  Nulls  Defaul_name
---------  ------------  ------- -----  -----  -----  -----------
Rule_name  Access_Rule_name  Identity
---------  ----------------  --------
money      money              8   NULL    NULL      1         NULL
NULL                   NULL          0
```

- **Example 6 –** 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:

```
sp_help identype
```

```
Type_name  Storage_type  Length  Prec   Scale  Nulls  Defaul_name
---------  ------------  ------- -----  -----  -----  -----------
Rule_name  Access_Rule_name  Identity
---------  ----------------  --------
identype        numeric       4   NULL    NULL      1         NULL
NULL                   NULL          1
```

- **Example 7 –** Shows a new column, indicating whether optimistic index locking is enabled. 1 indicates that the option is enabled; 0 indicates that it is not.

```
sp_help "mytable"
```

```
------------
exp_row_size   reserve   pagegap   fillfactor   max_rows_per_page
--------------------------------------------------------
           1         0         0            0                    0
concurrency_opt_threshold    optimistic_index_lock
-------------------------------------------------
                        0                        1
```

- **Example 8 –** Shows a virtual computed column:

```
alter table authors add fullname as au_fname + ' ' + au_lname
sp_help authors
```

```
Object has the following computed columns

Column_Name Property
----------- --------
fullname    virtual

Text
------------------------------
AS au_fname + ' ' + au_lname
```

- **Example 9 –** Shows a virtual computed column to a materialized computed column:

```
alter table authors modify fullname materialzied
sp_help authors
```

```
Object has the following computed columns

Column_Name Property
----------- ------------
fullname    materialized

Text
-------------------------------------------
AS  au_fname + ' ' + au_lname
MATERIALIZED
```

- **Example 10 –** The result set for **sp_help** ***table_name*** includes the `Decrypt_Default_name` column, which indicates the decrypt default name for the column. For example, if you run the following:

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

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

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

```
    NULL                          0         1 encr_table_col1_103
6527695
```

- **Example 11 –** Displays the `Name`, `Owner`, `Object_type`, `Object_status`, and `Create_date` of the predicate object:

```
grant select on tab1 where col1 = 5 as pred1 to robert
sp_help pred1
```

```
Name  Owner  Object_type  Object_status  Create_date
----- ------ ------------- -------------- ------------
pred1 dbo    predicate     -- none --      Feb 9 2010 12:49PM
```

- **Example 12 –** For this precomputed result set:

```
create table numtrips (source int, destination int, count_trip
int)
create precomputed result set frequent_trips unique (source,
destination)
as
select * from numtrips where count_trip > 100
```

sp_help numtrips returns the following:

```
Name              Owner       Object_type
Object_status
Create_date
---------------- ---------- ----------------------
------------------------------------------------------------
--------------------------------------
numtrips         dbo        user table
precomputed result set defined
May 11 2012  6:46AM
. . .
```

sp_help frequent_trips returns:

```
Name                         Owner
Object_type
Object_status
Create_date
---------------------------- ----------
--------------------------------------------
-----------------------------------------------------------------
-----
--------------------------------------
frequent_trips               dbo
precomputed result set
immediate, enabled, enabled for QRW
May 11 2012  6:46AM
. . .
```

- **Example 13 – sp_help** displays **execute as owner** or **execute as caller** in the `Object` status field as follows:

```
create proc p1 with execute as owner asselect 1gosp_help
p1Name  Owner  Object_type  Object_statuse  Create_date
----  -----  -----------  --------------  -----------
```

```
p1       dbo        stored procedureexecute as ownerJun  8 2012
10:05AM
(1 row
affected)Column_name  Type Length Prec Scale Nulls Not_compressed
  Default_name
Rule _name  Access_Rule_name  Computed_Column_object  Identity
-----------------------------------------------------------------
---------(return status = 0) Rule_name
```

Usage

- For virtually hashed table, **sp_help** reports:
  - That a table is virtually-hashed with this message:

    ```
    Object is Virtually Hashed
    ```

  - The `hash_key_factors` for the table with a message using this syntax:

    ```
    column_1:hash_factor_1,
    column_2:hash_factor_2...,
    max_hash_key=max_hash_value
    ```

    For example:

    ```
    attribute_class         attribute       int_value
    char_value                                comments

    ---------------------   ---------------  ----------------------
    ---
    ------------------------------------    -----------
    hash clustered tables   hash key factors                   NULL
    id:10.0, id2:1.0, max_hash_key=1000.0           NULL
    ```
- **sp_help** looks for an object in the current database only.
- **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.
- **sp_help** displays the following new settings:
  - The locking scheme, which can be set with **create table** and changed with **alter table**
  - The expected row size, which can be set with **create table** and changed with **sp_chgattribute**
  - The reserve page gap, which can be set with **create table** and changed with **sp_chgattribute**
  - The row lock promotion settings, which can be set or changed with **sp_setpglockpromote** and dropped with **sp_droprowlockpromote**
- **sp_help** includes the report from:

- **sp_helpindex** – showing the order of the keys used to create the index and the space management properties
- **sp_helpartition** – showing the partition information of the table
- **sp_helpcomputedcolumn** – showing the computed column information of the table

- When Component Integration Services is enabled, **sp_help** displays information on the storage location of remote objects.
- **sp_help** displays information about encryption keys. When a key name is specified as the parameter to **sp_help**, the command lists the key's name, owner, object type, and creation date.
- **sp_help** ***tablename*** indicates if a column is encrypted, including the name of the decrypt default on the column, if one exists.
- **sp_help** *predicate_name* displays information about the predicated privilege.

See also:

- **alter table**, **create table** in *Reference Manual: Commands*
- *Java in Adaptive Server Enterprise* for more information about SQLJ routines.

Permissions

Any user can execute **sp_help**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_chgattribute* on page 121
- *sp_droprowlockpromote* on page 283

- *sp_helppartition* on page 375
- *sp_helpcomputedcolumn* on page 381
- *sp_helpconstraint* on page 390
- *sp_helpindex* on page 409
- *sp_setpglockpromote* on page 631

Rules for Finding Objects

sp_help follows the SAP ASE 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*.

Precomputed Result Sets and sp_help

sp_help displays information about precomputed result set objects in the `Object_type` column. The SAP ASE server treats precomputed result set objects internally as user tables. When you issue **sp_help** with a precomputed result set as the *objectname*, it reports all the relevant details about columns, partitions, keys, indexes, and so on, similar to when you run **sp_help** against a user table.

Additionally, the `Object_Status` column returns the following for precomputed result sets:

- For user tables – returns `precomputed result set defined` in the `Object_Status` column if any precomputed result set objects are defined on the user table
- For precomputed result set objects – returns the following in the `Object_Status` column for:
  1. The refresh mode – `immediate` or `manual`
  2. The precomputed result set state – `enabled` or `disabled`
  3. The query rewrite state – `enable for QRW` or `disabled for QRW`

sp_help_resource_limit

Reports on resource limits.

Syntax

```
sp_help_resource_limit [name [, appname [, limittime
    [, limitday [, scope [, action[, verbose]]]]]]]
```

Parameters

- ***name* –** is the SAP ASE 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 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 SAP ASE 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:
  - **1** – for help on all limits that govern queries
  - **2** – for help on all limits that govern query batches (one or more SQL statements sent by the client to the server)
  - **4** – for help on all limits that govern transactions
  - **6** – for help on all limits that govern both query batches and transactions
  - **NULL** – for help on all limits that govern 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:

    - **1** – for help on all limits that issue a warning
    - **2** – for help on all limits that abort the query batch
    - **3** – for help on all limits that abort the transaction
    - **4** – for help on all limits that kill the session
    - **NULL** – for help on all limits that govern the specified *name*, *appname*, *limittime*, *limitday,* and *scope*, without regard to the *action* they take
- **verbose –** when used, the output is displayed in the verbose mode, with value 1 or 0 (zero).

Examples

- **Example 1 –** Lists all resource limits stored in the `sysresourcelimits` system table:

```
sp_help_resource_limit
```

- **Example 2 –** Lists all limits for the user “joe_user”:

```
sp_help_resource_limit joe_user
```

- **Example 3 –** Lists all limits for the application *my_app*:

```
sp_help_resource_limit NULL, my_app
```

- **Example 4 –** Lists all limits enforced at 9:00 a.m.:

```
sp_help_resource_limit NULL, NULL, "09:00"
```

- **Example 5 –** An alternative way of listing the limits enforced at 9:00 a.m.:

```
sp_help_resource_limit @limittype = "09:00"
```

- **Example 6 –** Lists all limits enforced on Mondays:

```
sp_help_resource_limit NULL, NULL, NULL, Monday
```

- **Example 7 –** Lists any limit in effect for “joe_user” on Mondays at 9:00 a.m.

```
sp_help_resource_limit joe_user, NULL, "09:00", Monday
```

- **Example 8 –** To list all limits in verbose mode:

```
sp_help_resource_limit null,null,null,null,null,null,1
```

- **Example 9 –** To list all resource limits in verbose mode:

```
sp_help_resource_limit @verbose=1
```

Usage

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.

See the *System Administration Guide* for more information on resource limits.

Permissions

The permission checks for **sp_help_resource_limit** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage resource limit` privilege. Any user can execute **sp_help_resource_limit** to list their own resource limits.
Disabled	With granular permissions disabled, you must be a user with **sa_role**. Any user can execute **sp_help_resource_limit** to list their own resource limits.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_add_resource_limit* on page 7
- *sp_drop_resource_limit* on page 258
- *sp_modify_resource_limit* on page 510

sp_help_qpgroup

Reports information on an abstract plan group.

Syntax

```
sp_help_qpgroup [ group [, mode ]]
```

Parameters

- ***group* –** is the name of an abstract plan group.
- ***mode* –** is the type of report to print, and is one of the following:
  - **full** – returns the number of rows and number of plans in the group, the number of plans that use two or more rows, the number of rows and plan IDs for the longest plans, and number of hash keys and hash key collision information. This is the default report mode.
  - **stats** – returns all of the information from the "full" report, except hash key information.
  - **hash** – returns the number of rows and number of abstract plans in the group, the number of hash keys, and hash-key collision information.
  - **list** – returns the number of rows and number of abstract plans in the group, and the following information for each query/plan pair: hash key, plan ID, first few characters of the query, and the first few characters of the plan.
  - **queries** – returns the number of rows and number of abstract plans in the group, and the following information for each query: hash key, plan ID, first few characters of the query.
  - **plans** – returns the number of rows and number of abstract plans in the group, and the following information for each plan: hash key, plan ID, first few characters of the plan.
  - **counts** – returns the number of rows and number of abstract plans in the group, and the following information for each plan: number of rows, number of characters, hash key, plan ID, first few characters of the query.

Examples

- **Example 1 –** Reports summary information about all abstract plan groups in the database:

```
sp_help_qpgroup
```

```
Group                  GID        Plans
----------------------- ----------- -----------
ap_stdin                         1           0
ap_stdout                        2           0
dev_test                         3         209
```

- **Example 2 –** Reports on the `test_plans` group:

```
sp_help_qpgroup test_plans
```

```
Query plans group 'test_plans', GID 8

 Total Rows  Total QueryPlans
 ----------- ----------------
           6                3
sysqueryplans rows consumption, number of query plans per row
count

 Rows        Plans
 ----------- -----------
```

```
           2           3

 Hashkeys
 -----------
           3
There is no hash key collision in this group.
```

Usage

When used with an abstract plan group name, and no `mode` parameter, the default mode for **sp_help_qpgroup** is **full**.

Hash-key collisions indicate that more than one plan for a particular user has the same hash-key value. When there are hash key collisions, the query text of each query with the matching hash key must be compared to the user's query text in order to identify the matching query, so performance is slightly degraded.

Permissions

The permission checks for **sp_help_qpgroup** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**. Any user can execute **sp_help_qpgroup** for their own abstract plan group.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_help_qplan* on page 373

sp_help_qplan

Reports information about an abstract plan.

Syntax

```
sp_help_qplan id [, mode ]
```

Parameters

- ***id*** **–** is the ID of the abstract plan.
- ***mode*** **–** is the type of report to print, one of the following:
  - **full** – returns the plan ID, group ID, and hash key, and the full query and plan text.
  - **brief** – returns the same as full, but only prints about 80 characters of the query and plan, rather than the full query and plan. This is the default mode.
  - **list** – returns the hash key, ID, and first 20 characters of the query and plan.

  If you do not supply a value for the `mode` parameter, the default is **brief**.

Examples

- **Example 1 –** Prints the brief abstract plan report:

```
sp_help_qplan 800005881
```

```
gid         hashkey     id
 ----------- ----------- -----------
           5  2054169974   937054374

 query
--------------------------------------------------------------------
---------
 select type, avg(price) from titles group by type

 query_plan
--------------------------------------------------------------------
---------
 ( plan
    ( store Worktab1
        ( i_scan type_price titles )
    )
    ( t_scan ( ...
```

- **Example 2 –** Prints the full abstract plan report:

```
sp_help_qplan 784005824, full
```

Permissions

The permission checks for **sp_help_qplan** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege. Any user can execute **sp_help_qplan** for their own abstract plan.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**. Any user can execute **sp_help_qplan** for their own abstract plan.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_find_qplan* on page 337
- *sp_help_qpgroup* on page 370

sp_helpapptrace

Determines which sessions the SAP ASE server is tracing. Returns the server process IDs (spids) for all the sessions the SAP ASE server is tracing, the spids of the sessions tracing them, and the name of the tracefile.

Syntax

```
sp_helpapptrace
```

Examples

- **Example 1 –** Determines which sessions the SAP ASE server is tracing:

  ```
  sp_helpapptrace
  ```

  ```
   traced_spid  tracer_spid    trace_file
  -------------  ------------  ----------
  11               exited      /tmp/myfile1
  13               14          /tpcc/sybase.15_0/myfile2
  ```

Usage

sp_helpapptrace returns these columns:

- `traced_spid` – spid of the session you are tracing.
- `tracer_spid` – spid of the session that `traced_spid` is tracing. Prints “exited” if the `tracer_spid` session has exited.
- `trace_file` – full path to the tracefile.

If a session is tracing another session, but quits without disabling the tracing, the SAP ASE server allows a new session to rebind with the earlier trace. This means that a sa or sso is not required to finish every trace they start, but can start a trace session, quit, and then rebind to this trace session

Permissions

The permission checks for **sp_helpapptrace** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role** or **sso_role**.

sp_helpartition

Lists partition-related information of a table or index.

Syntax

```
sp_helpartition [ tabname [, { null | indexname | ‘all’ }[,
partitionname ] ] ]
```

Parameters

- ***tabname*** **–** is the name of a table in the current database.
- **null –** specifies that information about base table partitions is to be listed.
- ***indexname*** **–** is the name of an index in the current table. Information about this index displays.
- **'all' –** specifies that all index partition information is to be listed.
- ***partitionname*** **–** is the name of the partition in the base table or index.

Examples

- **Example 1 –** Returns summary and detailed information about the data partitions in the `titles` table.

```
sp_helpartition titles
go
```

```
name      type       partition_type partitions  partition_keys
--------- ---------- -------------- ----------- --------------
titles    base table range                    5  total_sales

(1 row affected)

partition_name partition_id  pages row_count segment   create_date
-------------- ------------  ----- --------- ---------
-------------------
smallsales     1440005130       1        5 titleseg1 Sep 26 2005
5:44PM
smallsales2    1456005187       1        0 titleseg2 Sep 26 2005
5:44PM
smallsales3    1472005244       1        2 titleseg3 Sep 26 2005
5:44PM
mediumsales4   1488005301       1        8 titleseg4 Sep 26 2005
5:44PM
bigsales5      1504005358       1        3 titleseg5 Sep 26 2005
5:44PM

Partition_Conditions
--------------------
VALUES <= (1000)
VALUES <= (2000)
VALUES <= (3000)
VALUES <= (10000)
VALUES <= (25000)

Avg_pages   Max_pages   Min_pages   Ratio(Max/Avg)   Ratio(Min/Avg)
----------- ----------- ----------- ------------------
--------------------
        1           1            1       1.000000         1.000000
(return status = 0)
```

- **Example 2 –** Returns summary partition information about the `titles` table and detailed information about the `smallsales` data partition.

```
sp_helpartition titles, null, smallsales
```

```
name      type       partition_type partitions  partition_keys
--------- ---------- -------------- ----------- --------------
titles    base table range                    5 total_sales

(1 row affected)

partition_name partition_id pages  row_count
segment    create_date
-------------- ------------ ------ ---------
---------  ------------------
smallsales    1440005130      1         5  titleseg1 Sep 26 2005
5:44PM

Partition_Conditions
--------------------
VALUES <= (1000)
(return status = 0)
```

- **Example 3 –** First, creates the nonclustered index `ncidx_local` on the `my_titles` table, then returns summary partition information about `my_titles` and detailed information on the partition `ncip4` on `ncidx_local`.

```
create nonclustered index ncidx_local on my_titles(title_id) local
index
    (ncip1, ncip2, ncip3, ncip4, ncip5)
go
sp_helpartition my_titles, ncidx_local, ncip4
go
```

```
name        type        partition_type partitions  partition_keys

----------- ----------- -------------- ----------- --------------
ncidx_local local index range                    5 total_sales

(1 row affected)

partition_name partition_id pages row_count segment create_date
-------------- ------------ ----- --------- -------
---------------------
ncip4         1584005643      1       8 default Sep 26 2005 6:06PM

Partition_Conditions
--------------------
VALUES <= (10000)
(return status = 0)
```

Usage

- **sp_helpartition** lists partition related information at the table, index, and partition level. The table- or index-level partition information includes index type (whether it is a local or global index), partition type, number of partitions, and partition keys, if applicable. For each partition, the information include partition name, ID, number of pages, segment name, create date, and the partition condition if applicable.

The summary information displays the number of pages per partition, the minimum and maximum number of pages, and the ratio between the average number of pages and the maximum or minimum number.

- If you do not supply a table name, **sp_helpartition** lists the owner, table name, number of partitions, and the partition type of all user tables in the current database.
- If you specify:
  - **'all'** instead of an index name or **null** – **sp_helpartition** lists the table- and index-level partition information for each index of the specified table and of the base table.
  - A particular index – **sp_helpartition** lists the index-level information for that index. If the partition name is:
    - Not specified – **sp_helpartition** displays the partition-level information for all partitions in the index, and summary information for the partitions.
    - Specified – **sp_helpartition** displays only the partition-level information for that partition.
  - Only the table name – **sp_helpartition** displays table-level index partition information for the base table and partition-level information for all partitions in the base table.
  - Null instead of an index name, and a partition name is specified – **sp_helpartition** displays table-level partition information for the base table and partition-level information for the named partition—with no summary information.
- Partitions are created using **create table**, **alter table**, and **select into**. See these commands for more information about partitioning.
- Use **sp_helpsegment** to display the number of used and free pages on the segment on which the partition is stored.

See also **alter table**, **create table**, **select into** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_helpartition**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_helpsegment* on page 430
- *sp_statistics* on page 763

Accuracy of Results and sp_helpartition

The values reported in the “pages” column may differ from 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 these commands to update the partition statistics:

- **dbcc checkdb**
- **dbcc checktable**
- **update all statistics**
- **update table statistics**

Then, rerun **sp_helpartition** for an accurate report.

See also

- *sp_statistics* on page 763

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.

Syntax

```
sp_helpcache {cache_name | "cache_size[P | K | M | G]" ,
      ‘instance instance_name’}
```

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**.
- ***instance_name* –** name of the instance with a cache that you are investigating.

Examples

- **Example 1 –** Displays information about items bound to `pub_cache`:

  ```
  sp_helpcache pub_cache
  ```

- **Example 2 –** Shows the amount of overhead required to create an 80MB data cache:

  ```
  sp_helpcache "80M"
  ```

- **Example 3 –** Displays information about all caches and all items bound to them:

  ```
  sp_helpcache
  ```

- **Example 4 –** (Cluster Edition) displays the overhead for the cache `C2` on instance “blade1” for size 10M:

  ```
  sp_helpcache 'C2', '10M', 'instance blade1'
  ```

Usage

There are additional considerations when using **sp_helpcache**:

- 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.
- (Cluster Edition) If you do not specify an *instance_name*, **sp_helpcache** displays information for all caches.
- 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.
- Although you can still use **sp_bindcache** on a system `tempdb`, the binding of the system `tempdb` is now non-dynamic. Until you restart the server, the changes do not take effect, and **sp_helpcache** reports a status of “P” for pending, unless you have explicitly bound the system `tempdb` to the default data cache, in which case the status as “V” for valid, because by default the system `tempdb` is already bound to the default datacache.

Permissions

Any user can execute **sp_helpcache.** Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_bindcache* on page 76
- *sp_cacheconfig* on page 90
- *sp_poolconfig* on page 582
- *sp_unbindcache* on page 710
- *sp_unbindcache_all* on page 713

sp_helpcomputedcolumn

Reports information on the computed columns in a specified table.

Syntax

```
sp_helpcomputedcolumn {tabname}
```

Parameters

- ***tabname* –** names the table that contains computed columns.

Examples

- **Example 1 –** This example reports the computed columns in the `mytitles` table:

  ```
  sp_helpcomputedcolumn mytitles
  ```

  ```
  Column_Name Property
  ----------- ------------
  sum_sales   materialized

  Text
  ------------------------------------
  AS price * total_sales materialized

  (return status = 0)
  ```

Permissions

Any user can execute **sp_helpcomputedcolumn**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_helpconfig

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

- **Example 1 –** Returns a report on all configuration options that start with “allow”:

```
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
```

- **Example 2 –** Returns a report on how much memory is needed to create a metadata cache for 421 object descriptors:

```
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.
```

- **Example 3 –** Returns a report on how many database descriptors would fill a 1MB database cache:

```
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.
```

- **Example 4 –** Returns a report on how many locks use 512K of memory:

```
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.
```

- **Example 5 –** Returns a report on the status of the **allow updates to system tables** configuration parameter:

```
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
```

Usage

- **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.

  Note: The “maximum value” setting refers to the largest number that the parameter’s datatype can accept, rather than to an actual configurable value.

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

- **cluster options** displays all strictly cluster-wide configuration options.
- If **system_view** is set to **cluster**, **sp_helpconfig** displays configuration information for all instances in the cluster.
- If **system_view** is set to **instance**, **sp_helpconfig** displays configuration information for the current instance.
- 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.
- **sp_helpconfig** accepts static, dynamic, and read-only options.
- **sp_helpconfig 'restricted decrypt permission'** returns the following display:

```
sp_helpconfig 'restricted decrypt permission'
```

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

Minimum Value  Maximum Value  Default Value  Current Value
    Memory Used  Unit     Type
-------------  -------------  -------------  --------------
    -----------  -------------
0                         1             0             0
              0  switch   dynamic
```

Permissions

Any user can execute **sp_helpconfig** except the following, which requires **sybase_ts_role**.

- **number of ccbs**
- **caps per ccb**
- **average cap size**

Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_configure* on page 167
- *sp_countmetadata* on page 180
- *sp_monitorconfig* on page 532

Planning Metadata Cache Configuration with sp_helpconfig

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
```

Estimate Memory Requirements for compression info pool size

Use the **estimate** parameter to determine the approximate amount of memory required for the **compression info pool size** configuration parameter.

The **estimate** parameter recommends a value to which you can set the indicated configuration parameter, based on the settings of other configuration parameters or user-specified values that override those settings:

```
sp_helpcofig "config_name"
   [, { "size" | "estimate [using_argument = value [, using_argument
= value ] [, ...] ] } ]
```

using_argument= value provides these additional arguments for the **estimate** parameter to override default values:

- `maxconcusers = value` – specifies the maximum **number of concurrent users**, as an integer, that can access compressed tables.
  For example, `maxconcusers = 0.7` indicates 70 percent of the configured value for **number of user connections**. An integer value of 1 or greater specifies an absolute number of concurrent users.
- `numcolumns = value` – specifies the average number of columns in a compressed table.
- `numcompobjs = value` – specifies the default number of open objects as an integer, or as a percentage, that require memory for compression metadata. For example, `numcompobjs = 0.2` indicates that 20 percent of the configured value for **number of open objects**. An integer value of 1 or greater specifies an absolute number of open objects.
- `numtables = value` – determines the average number of compressed tables accessed in a statement.

Issuing **sp_helpconfig** without arguments generates usage information, showing the subclauses you may specify, and some examples of typical usage.

This example shows the **sp_helpconfig ... estimate** parameter run from a system database (such as `master` or `tempdb`). In this example, **sp_helpconfig** performs the estimate using default values for factors that affect the required memory:

```
sp_helpconfig 'compression info pool', 'estimate'
The compression information pool size parameter indicates the
amount of memory currently available to store table compression
information.

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

```
-------------  -------------- -------------  -------------
     -----------     ----------------  -------
            0     2147483647          4096          4096
           8240      memory pages(2k)  dynamic

Estimated memory required for 600 concurrent users requesting
memory from this pool, accessing 500 compressed objects with
50 columns, on an average, per compressed table is 22600 KB.

Configuration parameter, 'compression info pool size', can be
configured to 21971 to fit in 44200K of memory.
```

This example overrides the defaults with site-specific parameters to estimate the memory and configuration value setting. **sp_helpconfig** is executed a second time from a system database (such as `master` or `tempdb`) to estimate the memory required for server-wide concurrent access to compressed objects, when these objects are accessed from multiple databases in the server:

```
sp_helpconfig 'compression info pool', 'estimate
 using numcompobjs=0.3, numtables=2.25, numcolumns=25,
 maxconcusers=0.85'

The compression information pool size parameter indicates the
amount of memory currently available to store table compression
information.

Minimum Value  Maximum Value  Default Value  Current Value
     Memory Used     Unit              Type
-------------  -------------- -------------  -------------
     -----------     ----------------  -------
            0     2147483647           4096          4096
         8240        memory pages(2k)  dynamic

Estimated memory required for 1020 concurrent users requesting
memory from this pool, accessing 150 compressed objects with 25
columns, on an average,
 per compressed table is 37020 KB.

Configuration parameter, 'compression info pool size', can be
configured to 18402 to fit in 37020K of memory.
```

This example shows **sp_helpconfig ... estimate** run against a user database with numerous compressed tables, which are used frequently by an application. The server is configured as:

```
sp_configure 'user connections', 900
sp_configure 'worker processes', 500
sp_configure 'max parallel degree', 5
```

In this example, **estimate** gathers metrics from the user database from which you issue the procedure for:

- The number of compressed objects
- The average number of columns in these compressed objects

Using these input values, **sp_helpconfig** estimates the memory required for `compression info pool` to store table compression information:

```
sp_helpconfig 'compression info pool size', 'estimate'

The compression information pool size parameter indicates
the amount of memory currently available to store table
compression information.

Minimum Value  Maximum Value  Default Value  Current Value
     Memory Used     Unit             Type
-------------  -------------- -------------  -------------
     -----------     ----------------  -------
            0      2147483647          4096          15396
           33384     memory pages(2k)  dynamic

Estimated memory required for 1400 concurrent users requesting
memory from this pool, accessing 78240 compressed objects
with 10 columns, on an average, per compressed table is 74850 KB.

Configuration parameter, 'compression info pool size', can be
configured to 34519 to fit in 74850K of memory.
```

This output indicates that a total of 1400 concurrent users are expected to simultaneously request memory. The database has slightly more than 78000 compressed objects, with each table having, on average, 10 columns. The estimated value for this configuration option is 34519.

However, if not all the objects are routinely accessed simultaneously, and not all the configured user connections are simultaneously active, you can refine the estimates by providing site-specific overrides with the **using** parameter subclause:

```
sp_helpconfig 'compression info pool size', 'estimate
using numcompobjs = 50000, maxconcusers=600'

The compression information pool size parameter indicates
the amount of memory currently available to store table
compression information.

Minimum Value  Maximum Value  Default Value  Current Value
     Memory Used     Unit             Type
-------------  -------------- -------------  -------------
     -----------     ----------------  -------
            0      2147483647          4096          15396
           33384     memory pages(2k)  dynamic

Estimated memory required for 1100 concurrent users requesting
memory from this pool, accessing 50000 compressed objects with
10 columns, on an average, per compressed table is 55225 KB.

Configuration parameter, 'compression info pool size', can be
configured to 25468 to fit in 55225K of memory.
```

In this output, `maxconcusers = 600` implies that 600 concurrent client connections are accessing compressed objects requesting memory. Because of the parallel configuration settings, **sp_helpconfig** estimates that a total of 1100 requesters may concurrently request memory. The estimated value for this configuration option is 25468.

Using sp_helpconfig with sybdiagdb (SAP Product Support Only)

sp_helpconfig includes several *configname* options that are intended only forSAP Product Support to use with the `sybdiagdb` database:

- *number of ccbs* – the number of configurable action point control blocks available to aid debugging.
- *caps per ccb* – the maximum number of configurable action points that can be configured at any one time within one configurable action point.
- *average cap size* – the estimated number of bytes of memory required to store the information associated with a typical configurable action point.

Note: SAP Technical Support may create the `sybdiagdb` database on your system for debugging purposes. This database holds diagnostic configuration data, and is for use by SAP Technical Support only.

For example:

```
sp_helpconfig "number of ccbs"
```

```
Minimum Value Maximum Value Default Value Current Value Memory Used
------------- ------------- ------------- ------------- -----------
0             100             0             0             0
```

```
sp_helpconfig "caps per ccb"
```

```
Minimum Value Maximum Value Default Value Current Value Memory Used
------------- ------------- ------------- ------------- -----------
5             500            50            50             0
```

```
sp_helpconfig "average cap size"
```

```
Minimum Value Maximum Value Default Value Current Value Memory Used
------------- ------------- ------------- ------------- -----------
100         10000           200           200             0
```

sp_helpconstraint

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

- **Example 1 –** 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`):

```
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
```

- **Example 2 –** Displays more detailed information about the `pubs3..salesdetail` constraints, including the constraint type and any constraint error messages:

```
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.
```

- **Example 3 –** Displays a listing of all tables in the `pubs3` database:

```
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)
```

Usage

There are additional considerations when using **sp_helpconstraint**:

- **sp_helpconstraint** truncates foreign keys and reference keys to 30 characters.
- **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.
- **sp_helpconstraint** displays sharable inline defaults similarily to how it displays regular inline defaults.
- 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, the SAP ASE 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, the SAP ASE server returns an error message. You can use **sp_helpconstraint** to determine the necessary number of scan descriptors. See the *System Administration Guide* or more information on the **number of aux scan descriptors** configuration parameter.
- A system security officer can prevent the source text of constraint definitions from being displayed to most users who execute **sp_helpconstraint**. To restrict **select** permission on the `text` column of the `syscomments` table to the object owner or a system administrator, use **sp_configure** to set the **select on syscomments.text column** parameter to **0**. This restriction is required to run the SAP ASE server in the evaluated configuration. See the *System Administration Guide* for more information about the evaluated configuration.

See also**alter table**, **create table** in *Reference Manual: Commands* .

Permissions

Any user can execute **sp_helpconstraint**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38

Information	Values
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_configure* on page 167
- *sp_help* on page 358
- *sp_helpdb* on page 394
- *sp_monitorconfig* on page 532

sp_helpdb

Reports information about a particular database or about all databases.

Syntax

```
sp_helpdb [dbname [, order]]
```

Parameters

- ***dbname*** **–** is the name of the database on which to report information. Without this optional parameter, **sp_helpdb** reports on all databases. *dbname* can include wildcard characters to return all databases that match the specified pattern.
- ***order*** **–** The default order of the output is by `lstart`, which is the order in which the databases were created or altered. Use **device_name** along with *dbname* to display the output of **sp_helpdb** ordered by `device_name`.

Examples

- **Example 1 –** Displays information about all the databases in the SAP ASE server.

```
sp_helpdb
```

```
name          db_size  owner  bid   created        status
------------- -------- ----- ----- --------------
--------------------
```

```
master        24.0 MB   sa  1      Jan 07, 2004  mixed log and data
model         8.0 MB    sa  3      Jan 07, 2004  mixed log and data
pubs2        8.0 MB    sa  4      Jan 21, 2004  trunc log on chkpt,
                                                 mixed log and data
sybsystemdb    8.0 MB    sa  31513  Jan 07, 2004  mixed log and data
sybsystemprocs 112.0 MB  sa  31514  Jan 07, 2004  trunc log on
chkpt, mixed
                                                     log and data
tempdb          8.0 MB   sa  2      Feb 24, 2004  select into/
bulkcopy/
                                          pllsort, trunc log on
                                         chkpt, mixed log| and
                                                     data
(1 row affected)
(return status = 0
```

- **Example 2 –** Issued from within `pubs2`, displays information about the `pubs2` database, and includes segment information:

```
1> use pubs2
2> go
1> sp_helpdb pubs2
2> go
```

```
name   db_size  owner  dbid  created       status
------ -------- ------ ----- -------------
------------------------------
pubs2  20.0 MB   sa    4    Apr 13, 2005  trunc log on chkpt, mixed
log
                                         and data
(1 row affected)
pubs2
device_fragments  size   usage       created          free kbytes
----------------- ------- ------------- -------------------
-------
master            10.0 MB  data and log  Apr 13 2005 10:29AM  2304
pubs_2_dev        10.0 MB  data and log  Apr 13 2005 10:33AM  9888

device         segment
-------------
-----------------------------------------------------
master         default
master         logsegment
master         system
pubs_2_dev     default
pubs_2_dev     logsegment
pubs_2_dev     system
pubs_2_dev     titleseg1
pubs_2_dev     titleseg2
pubs_2_dev     titleseg3
pubs_2_dev     titleseg4
pubs_2_dev     titleseg5
return status = 0)
```

- **Example 3 –** Not issued from within `pubs2`, displays information about the `pubs2` database:

```
sp_helpdb pubs2
```

```
name  db_size  owner  dbid    created      status
------------ ------------- ------------------------ -----
----------------
pubs2   20.0 MB  sa      4     Jan 21, 2004  trunc log on chkpt,
single user,
                                            mixed log and data
(1 row
affected)device_fragments  size    usage         created
     free kbytes
----------------  ----    -----         -------             ----
---------
master           10.0 MB  data and log  Apr 13 2005  10:29AM  2304
pubs_2_dev        10.0 MB  data and log  Apr 13 2005  10:33AM
9888
(return status = 0)
```

- **Example 4 –** Specifies `device_name` for the *order* parameter to display the device fragments for `mydb` in alphabetical order, overriding the default sort order of **sp_helpdb**.

```
sp_helpdb mydb, device_name
```

```
name             db_size  owner  dbid  created      status
---------------  -------  -----  ----  -----------  -----------
mydb           4.5 MB    sa     5     Feb 27, 2003   no options set

(1 row affected)
device_fragments  size     usage      created               free
kbytes
----------------  -----    ------      --------  ----------------
--
A                1.5 MB   data only   Feb 27 2003   7:50AM    1530
B                 1.0 MB   log only    Feb 27 2003   7:50AM not
applicable
C                 2.0 MB   data only   Feb 27 2003   7:50AM    846
```

- **Example 5 –** Displays the row lock promotion attributes set for the `pubtune` database:

```
sp_helpdb pubtune
```

```
name    attribute_class  attribute      int_value char_value  com
ments
----    ---------------  ---------      --------- ----------  ---
---
pubtune lock strategy   row lock promotion  NULL    PCT = 95, LWM =
300,
                                                          HWM =
300
```

- **Example 6 –** Displays whether or not a database is a user-created temporary database under the `status` column:

```
sp_helpdb "mytempdb3"
```

```
name    db_size owner dbid created    status
-------  ------- ----- ---- -------     -----
```

```
mytempdb 32.0 MB sa    7    Dec 2, 2001 select into/bulkcopy/
pllsort, trunc
                                  log on chkpt, user created temp
db
```

- **Example 7 –** Reports the status of database that is being encrypted:

```
>sp_helpdb
>go
name        db_size    owner dbid  created      durability
    lobcomplvl inrowlen
status
……
test_db     6.0 MB     sa       4  Aug 07, 2013 full
             0 NULL
    encryption in progress: 35%
……
```

- **Example 8 –** Reports the status of a partially encrypted database:

```
>sp_helpdb
>go
name        db_size    owner dbid  created      durability
    lobcomplvl inrowlen
status
……
test_db     6.0 MB     sa       4  Aug 07, 2013 full
             0 NULL
    encrypted partly

……
```

- **Example 9 –** Reports the status of a database that is partially decrypted:

```
>sp_helpdb
>go
name        db_size    owner dbid  created      durability
    lobcomplvl inrowlen
status
……
test_db     6.0 MB     sa       4  Aug 07, 2013 full
             0 NULL
    decrypted partly

……
```

- **Example 10 –** Displays information about the durability of a user-created temporary database. For this example, if you create the database:

```
create temporary database tempdb_explicit on default = 50
with durability = no_recovery
```

sp_helpdb displays this output:

```
sp_helpdb tempdb_explicit
 name db_size owner dbid created durability lobcomplvl inrowlen status
 ---- ------- ----- ---- ------- ---------- ---------- -------- ------

 tempdb_explicit  50.0 MB  sa  7    Dec 05, 2012 no_recovery  0  NULL
         select into/bulkcopy/pllsort, trunc log on chkpt,
        mixed log and data, user-created enhanced performance
        temp db, allow wide dol rows

(1 row affected)
device_fragments   size      usage         created          free  kbytes
---------------- ------ -------------- ------------------ -------------
 master          50.0 MB  data and log  Dec 5 2012 8:49PM    49216
(return status = 0)
```

Usage

There are additional considerations when using **sp_helpdb**:

- When you run **sp_helpdb** on a fully encrypted database, it reports its encryption status:
  - Encrypted
  - Encryption in progress
  - Decryption in progress

  If the database is being encrypted or decrypted, **sp_helpdb** reports the percentage of work that has completed.
- **sp_helpdb** reports on the specified database when *dbname* is given. If no value is supplied for *dbname*, **sp_helpdb** reports on all the databases listed in `master.dbo.sysdatabases`.
- **sp_helpdb** reports all database-specific properties and settings, such as: whether a database is offline, compression type, large object compression level, in-row large object length, row lock promotion thresholds (if any are defined for the database), and so on.
- If you enable asynchronous log service on a database, the `attribute` column in the **sp_helpdb** output displays "`async log srv`".

  For more information about asynchronous log service, see **sp_dboption**, and *Advanced Optmizing Tools* in *Performance and Tuning: Optimizer*.
- For log segment disk pieces in a dedicated log database, **sp_helpdb** issues "not applicable" for the free space field in its per-disk-piece report. **sp_helpdb** also includes a column titled `free pages`, which is the value for the number of free pages the log segment has.
- (Cluster Edition) **sp_helpdb** does not display device-related information if the specified database is a local temporary database owned by a remote instance.
- *dbname* can include wildcard characters to return all databases that match the specified pattern. See *Expressions, Identifiers, and Wildcard Characters* in *Reference Manual: Building Blocks* for details about using wildcard characters.
- 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.
- 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”.
- The **status** column of **sp_helpdb** includes these descriptions for database durability:
  - `user created temp db` – normal temporary database created by the user (that is, created without specifying the **durability** parameter).
  - `user-created enhanced performance temp db` – user-created temporary database created explicitly with the **no_durability** parameter. Because a database created with **no_durability** depends on licensing, it may not come online if the license expires.

See also:

- *Advanced Optmizing Tools* in *Performance and Tuning: Optimizer*, *Expressions, Identifiers, and Wildcard Characters* in *Reference Manual: Building Blocks*.
- **alter database**, **create database** in *Reference Manual: Commands*

Permissions

Any user can execute **sp_helpdb**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_configure* on page 167
- *sp_dboption* on page 193
- *sp_rename* on page 605

sp_helpdefrag

sp_helpdefrag reports defragmentation information for either all eligible objects for **reorg defrag** in the database whose context it is invoked from or for the given object if it is eligible for **reorg defrag**.

sp_helpdefrag uses the built-in **defrag_status()** on each of the required tables or on each of the required data partitions to get the information about defragmentation.

- If *table_name* is not specified, defragmentation information for all eligible tables for *reorg defrag* (that is, user tables with datarows or datapages locking scheme) is reported. Rows for tables on which *reorg defrag* is currently executing precede those for tables where *reorg defrag* is not currently executing. Among these two sets, rows are in ascending order of the **pct_defrag**.
- If *table_name* is specified, and if the table is eligible for *reorg defrag*, defragmentation information of the table as well as that of each data partition is reported. Rows are in the ascending order of percentage defragmented portion. Row for the table comes first and has `NULL` in partition column.
- If *partition_name* is specified, only that particular data partition's information is reported.

Syntax

The syntax is:

```
sp_helpdefrag [table_name][,partition_name]
```

Parameters

- ***table_name*** **–** is the name of the table.
- ***partition_name*** **–** is the name of the partition.

Examples

- **No parameters and before defragmentation –** If **sp_helpdefrag** is executed without parameters on database testdb with user data-only locking tables before defragmentation:

  ```
  sp_helpdefrag
  ```

  The output is:

```
table                frag_index      pct_defrag     executing      last_run
---------------- ------------- ------------- -------------- ----------
t1_forw                    0.01            0              0         NULL
mymsgs                     0.39            0              0         NULL
mymsgs_clone               0.57            0              0         NULL
t1                         0.66            0              0         NULL
myprocs                    0.86            0              0         NULL
mymsgs_ptnd                1.07            0              0         NULL
t1_clone                   1.98            0              0         NULL
```

```
myprocs_clone            2.16          0               0          NULL
t1_ptnd                  2.99          0               0          NULL
myprocs_ptnd             3.03          0               0          NULL

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

If you execute **sp_helpdefrag** after defragmentation, the output is:

```
table          frag_index    pct_defrag    executing         last_run
------------- ------------- ------------- ------------- --------------
t1_forw           0.01         100           0         Oct 10 2012  4:15PM
mymsgs            0.05         100           0         Oct 10 2012  4:15PM
mymsgs_clone      0.06        100            0         Oct 10 2012  4:15PM
t1                0.08        100           0          Oct 10 2012  4:15PM
myprocs           0.09         100           0         Oct 10 2012  4:15PM
mymsgs_ptnd       0.09        100            0         Oct 10 2012  4:15PM
t1_clone          0.10         100           0         Oct 10 2012  4:15PM
myprocs_clone     0.11        100            0         Oct 10 2012  4:15PM
t1_ptnd           0.12         100           0         Oct 10 2012  4:15PM
myprocs_ptnd       0.14         100          0         Oct 10 2012  4:15PM

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

- **On a specified table –** If **sp_helpdefrag** is executed on table t1 in database testdb:

```
sp_helpdefrag t1
```

The output is:

```
table   partition  frag_index   pct_defrag  executing         last_run
------- ---------- ----------- ------------- ---------- -----------------
t1       NULL        0.35          35          0      Oct 10 2012  4:33PM
t1        p2          0.50          0          0                     NULL
t1       p1          0.42          20          0      Oct 10 2012  4:33PM
t1       p3          0.42          20          0      Oct 10 2012  4:33PM
t1       p4          0.05         100          0      Oct 10 2012  4:33PM

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

If **reorg defrag** is currently processing, the output is:

```
table   partition  frag_index   pct_defrag  executing         last_run
------- ---------- ----------- ------------- ---------- -----------------
t1       NULL        0.48          13          1      Oct 10 2012  4:33PM
t1        p2         0.50           0         1                      NULL
t1       p4          0.60           0          1      Oct 10 2012  4:33PM
t1       p1          0.42          20          1      Oct 10 2012  4:33PM
t1       p3          0.42          20          1      Oct 10 2012  4:33PM

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

- **On a specified partition –** If **sp_helpdefrag** is executed on partition p1 in table t1:

```
sp_helpdefrag t1, p1
```

The output is:

```
table   partition  frag_index   pct_defrag  executing       last_run
------- ---------- ----------- ------------- ---------- -----------------
t1        p1         0.42          20           0      Oct 10 2012  4:33PM

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

sp_helpdevice

Reports information about a particular device or about all SAP ASE 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

- **Example 1 –** Displays information about all the devices on SAP ASE:

  ```
  1> sp_helpdevice
  2> go
  ```

  ```
  device_name physical_name         description
           status cntrltype vdevno       vpn_low         vpn_high
  ----------- ---------------------
  ----------------------------------------
           ------ --------- ----------- ----------- -----------
  dev1       d:\sybdata\RV150.dev1 special, dsync off, directio on,
  physical
  disk, 150.00 MB, Free: 0.00 MB
               2         0         2         0         76799
  dev2       d:\sybdata\RV150.dev2 special, dsync on, directio off,
  physical
  disk, 150.00 MB, Free: 130.00 MB
           16386         0         3         0         76799
  master     d:\sybdata\RV150.mas  special, dsync on, directio off,
  default
  disk, physical disk, 30.00 MB, Free: 0.50 MB
               3         0         0         0         15359
  sysprocsdev d:\sybdata\RV150.ssp  special, dsync on, directio off,
  physical
  disk, 120.00 MB, Free: 0.00 MB
           16386         0         1         0         61439
  tapedump1  \\.\TAPE0             disk, dump device
              16         2         0         0         20000
  tapedump2   \\.\TAPE1             tape,        625 MB, dump
  device             16         3         0         0         20000
  ```

```
(6 rows affected, return status = 0)
```

- **Example 2 –** Reports information about the dump device named `diskdump`:

  ```
  sp_helpdevice diskdump
  ```

Usage

There are additional considerations when using **sp_helpdevice**.

- **sp_helpdevice** displays the amount of unallocated space per device, indicated by the placeholder `Free` in the description column in the ouput

  Note: A small amount of space can remain unused on a device, especially for servers with larger page sizes. For example, the last 2MB of a 250MB device in a 16K server cannot be allocated, and sp_helpdevice reports this as free. This is because the size of an allocation unit in a 16K server is 4Mb, so only multiples of allocation units can be allocated.

- **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**.
- If you issue **sp_helpdevice** against a single device, it displays a list of allocated fragments on that device.
- The `description` column displays information about device types:
  - `block device`
  - `file system device`
  - `raw device`

  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 `vdevno` column is 0 for dump devices, 0 for the `master` database device, and 1 or higher for other database devices.

  The `vpn_low` and `vpn_high` columns represent virtual page numbers, each of which is unique among all the devices in SAP ASE.

See also **disk init**, **dump database**, **dump transaction**, **load database**, **load transaction** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_helpdevice**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addumpdevice* on page 57
- *sp_deviceattr* on page 224
- *sp_diskdefault* on page 227
- *sp_dropdevice* on page 262
- *sp_logdevice* on page 490

sp_helpextendedproc

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

- **Example 1 –** Lists the **xp_cmdshell** ESP and the name of the DLL file in which its function is stored:

  ```
  use sybsystemprocs
  go
  sp_helpextendedproc xp_cmdshell
  ```

  ```
  ESP Name    DLL Name
  ----------- ----------
  xp_cmdshell sybsyesp
  ```

- **Example 2 –** Lists all the ESPs in the current database, along with the names of the DLL files in which their functions are stored:

  ```
  sp_helpextendedproc
  ```

  ```
  ESP Name    DLL Name
  ----------- ----------
  xp_freedl    sybsyesp
  xp_cmdshell  sybsyesp
  ```

Usage

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.

See also **create procedure**, **drop procedure** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_helpextendedproc.**

Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addextendedproc* on page 25
- *sp_dropextendedproc* on page 267
- *xp_cmdshell* on page 771

sp_helpexternlogin

(Component Integration Services only) Reports information about external login names.

Syntax

```
sp_helpexternlogin [server[, loginame[, rolename]]]
```

Parameters

- ***server*** **–** is the name of the remote server that has been added to the local server with **sp_addserver**.
- ***loginame*** **–** is a login account on the local server.
- ***rolename*** **–** is the SAP ASE user's assigned role.

Examples

- **Example 1 –** Displays all remote servers, local login names, role names, and external logins:

  ```
  sp_helpexternlogin
  ```

- **Example 2 –** Displays local login names, role names, and external logins for the server named SSB:

  ```
  sp_helpexternlogin SSB
  ```

- **Example 3 –** Displays remote servers, local login names and external logins for the user named "milo":

  ```
  sp_helpexternlogin NULL, milo
  ```

- **Example 4 –** Displays external logins for remote server SSB where the local user name is "trixi":

  ```
  sp_helpexternlogin SSB, trixi
  ```

- **Example 5 –** Displays external logins for remote server SSB for local users with sa_role:

  ```
  sp_helpexternlogin SSB, NULL, sa_role
  ```

Usage

sp_helpexternlogin displays all remote servers, the user's local login name, role name, and the user's external login name.

Add remote servers with **sp_addserver**. Add local logins with **create login**.

Permissions

Any user can execute **sp_helpexternlogin**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addexternlogin* on page 27
- *sp_addlogin* on page 35
- *sp_addserver* on page 46
- *sp_dropexternlogin* on page 268
- *sp_helpserver* on page 434

sp_helpgroup

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

- **Example 1 –** Displays information about all groups in the current database:

  ```
  sp_helpgroup
  ```

  ```
  Group_name           Group_id
  ---------------      --------
  hackers              16384
  public                0
  ```

- **Example 2 –** Displays information about the group “hackers”:

  ```
  sp_helpgroup hackers
  ```

  ```
  Group_name       Group_id       Users_in_group      Userid
  -----------      ---------      --------------      ------
  hackers          16384          ann                 4
  hackers          16384          judy                3
  ```

Usage

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
-----------           ---------        --------------         ------
```

See also **grant**, **revoke** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_helpgroup**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addgroup* on page 30
- *sp_changegroup* on page 103
- *sp_dropgroup* on page 273
- *sp_helpprotect* on page 425
- *sp_helpuser* on page 446

sp_helpindex

Reports information about the indexes created on a table. Reports information on computed column indexes and function-based indexes.

Syntax

```
sp_helpindex objname
```

Parameters

- ***objname* –** is the name of a table in the current database.

Examples

- **Example 1 –** Displays the types of indexes on the `sysobjects` table:

  ```
  sp_helpindex sysobjects
  ```

  ```
  index_name index_keys index_description index_max_rows_per_page
         index_fillfactor index_reservepagegap index_created
  ```

```
       index_local

sysobjects    id        clustered, unique                    0
                   0                  0        Apr 12 2005  2:38PM
        Global Index
ncsysobjects  name, uid nonclustered, unique
                   0                  0        Apr 12 2005  2:38PM
        Global Index

(2 rows affected)
index_pt_name          index_ptn_seg
---------------------- --------------
sysobjects_1            system
ncsysobjects_1           system
```

- **Example 2 –** Displays information about the index on the `titles` table in the `pubs2` database. The `titles` table is partitioned, but the index `titleind` is not. `titleind` is a nonclustered (single-partitioned), global index.

```
sp_helpindex titles
```

```
index_name index_keys index_description index_max_rows_per_page
       index_fillfactor index_reservepagegap index_created
       index_local

titleind    title    nonclustered                            0
          Global Index

(1 row affected)
index_pt_name          index_ptn_seg
---------------------- --------------
titleind_1232004389     default
```

- **Example 3 –** Displays index information about the `mysalesdetail` table. `mysalesdetail` is partitioned by hash on the `ord_num` column. A clustered, local index, with three partitions, has also been created on `ord_num`.

```
sp_helpindex mysalesdetail
```

```
index_name index_keys index_description index_max_rows_per_page
       index_fillfactor index_reservepagegap index_created
index_local
---------- ---------- ----------------- -----------------------
       ---------------- --------------------
-------------       -----------
clust_idx  ord_num    clustered                               0
                  0                  0 Apr 12 2005 2:38PM  Local
Index
(1 row affected)
index_pt_name          index_ptn_seg
---------------------- --------------
clust_idx_1344004788   default
clust_idx_1360004845   default
clust_idx_1376004902   default
```

- **Example 4 –** Displays a function-based index:

```
create index sum_sales on mytitles (price * total_sales)
sp_helpindex mytitles
```

```
Object has the following indexes

index_name index_keys index_description index_max_rows_per_page
       index_fillfactor index_reservepagegap index_created
index_local
---------- ---------- ----------------- -----------------------
       ---------------- --------------------
-------------       -----------
sum_sales  sybfi2_1   nonclustered                            0
      0                             0 Oct 12 2005 3:34PM  Global
Index

(1 row affected)
index_ptn_name       index_ptn_seg
-------------------- -------------
sum_sales_1724867646 default

(1 row affected)

Object has the following functional index keys

Internal_Index_Key_Name
-----------------------
sybfi2_1

(1 row affected)

Expression
-------------------
price * total_sales

(return status = 0)
```

Usage

There are additional considerations when using **sp_helpindex**:

- **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:
  - Partition information for each index.
  - Whether the index is local or global, clustered or nonclustered.
  - The **max_rows_per_page** setting of the indexes.
  - Information about clustered indexes on data-only locked tables.

    The index ID (`indid)` of a clustered index in data-only locked tables is not equal to 1.

- The column order of the keys, to indicate whether they are in ascending or descending order.
- Space manage property values.
- The key column name followed by the order. Only descending order is displayed. For example, if there is an index on column a ASC, b DESC, c ASC, “index_keys” shows “a, b DESC, c”.

See also **create index**, **drop index**, **update statistics** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_helpindex**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_help* on page 358
- *sp_helpkey* on page 417
- *sp_helpartition* on page 375

sp_helpjava

Displays information about Java classes and associated JARs that are installed in the database.

Syntax

```
sp_helpjava ["class"[, java_class_name[, “detail” | “depends”]] |
   "jar", jar_name[, “depends”]]]
```

Parameters

- **"class" | "jar" –** specifies whether to display information about a class or a JAR. Both “class” and “jar” are keywords, so the quotes are required.
- ***java_class_name* –** the name of the class about which you want information. The class must be a system class or a user-defined class that is installed in the database.
- **detail –** specifies that you want to see detailed information about the class.
- **depends –** lists all the database objects that depend on the specified class or classes in the JAR, including SQLJ functions, SQLJ stored procedures, views, Transact-SQL stored procedures, and tables.
- ***jar_name* –** the name of the JAR for which you want to see information. The JAR must be installed in the database using **installjava**.

Examples

- **Example 1 –** Displays the names of all classes and associated JAR files installed in the database:

  ```
  sp_helpjava
  ```

- **Example 2 –** Displays the name of all classes:

  ```
  sp_helpjava "class"
  ```

- **Example 3 –** Displays detailed information about the **Address** class:

  ```
  sp_helpjava "class", Address, detail
  ```

  ```
  Class
  --------------------------------------------------
  Address

  (1 row affected)
  Class Modifiers
  --------------------------------------------------
   public synchronized

   Implemented Interfaces
   --------------------------------------------------
   java.io.Serializable
  ```

```
 Extended Superclass
 --------------------------------------------------
 java.lang.Object

 Constructors
 --------------------------------------------------
 public Address()
 public Address(java.lang.String,java.lang.String)

 Methods
 --------------------------------------------------
 public final native java.lang.Class java.lang.Object.getClass()
 public native int java.lang.Object.hashCode()
 public boolean java.lang.Object.equals(java.lang.Object)
 public java.lang.String java.lang.Object.toString()
 public final native void java.lang.Object.notify()
 public final native void java.lang.Object.notifyAll()
 public final native void java.lang.Object.wait(long) throws
java.lang.InterruptedException
public final void java.lang.Object.wait(long,int) throws
java.lang.InterruptedException
 public final void java.lang.Object.wait() throws
java.lang.InterruptedException
 public java.lang.String Address.display()
 public void Address.removeLeadingBlanks()

 Fields
 --------------------------------------
 public java.lang.String Address.street
 public java.lang.String Address.zip
```

Usage

The **depends** parameter lists dependencies of a class or classes if the class is listed in the **external name** clause of a create statement for a SQLJ routine or is used as a datatype of a column in the database.

See also:

- **remove java** in *Reference Manual: Commands*
- See *Java in Adaptive Server Enterprise* for more information about Java in the database.
- **extractjava**, **installjava** in the *Utility Guide*

Permissions

Any user can execute **sp_helpjava**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_helpjoins

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

- **Example 1 –** Displays a list of columns that are likely join candidates in the `sales` and `salesdetail` tables:

```
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
```

- **Example 2 –** Displays a list of columns that are likely join candidates in the `sysobjects` and `syscolumns` system tables:

```
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
```

Usage

The column pairs that **sp_helpjoins** displays come from either of two sources. **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, 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 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.

Permissions

Any user can execute **sp_helpjoins**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_commonkey* on page 157
- *sp_foreignkey* on page 349
- *sp_helpkey* on page 417
- *sp_primarykey* on page 589

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.

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

- **Example 1 –** 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:

  ```
  sp_helpkey
  ```

  ```
  keytype object     related_object object_keys       related_keys
  ------- -------     -------------- ---------------    ----------
  ----
  primary authors     -- none --     au_id,*,*,*,*,*,*,*
  *,*,*,*,*,*,*,*,*
  foreign titleauthor authors        au_id,*,*,*,*,*,*,* au_id,*,*,
  *,*,*,
                                                                *,*
  ```

Usage

There are additional considerations when using **sp_helpkey**:

- **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 SAP ASE 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".

See also **create trigger** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_helpkey**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_commonkey* on page 157
- *sp_foreignkey* on page 349
- *sp_primarykey* on page 589

sp_helplanguage

Reports information about a particular alternate language or about all languages.

Syntax

```
sp_helplanguage [language]
```

Parameters

- ***language* –** is the name of the alternate language for which to display information about.

Examples

- **Example 1 –** Displays information about the alternate language, “french”:

  ```
  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
  ```

- **Example 2 –** Displays information about all installed alternate languages:

  ```
  sp_helplanguage
  ```

Usage

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.

Permissions

Any user can execute **sp_helplanguage**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addlanguage* on page 32
- *sp_droplanguage* on page 276
- *sp_setlangalias* on page 629

sp_helplog

Reports the name of the device that contains the first page of the transaction log.

Syntax

```
sp_helplog
```

Examples

- **Example 1 –** Reports “master” as the name of the device:

  ```
  sp_helplog
  ```

  ```
  In database 'master', the log starts on device 'master'.
  ```

Usage

See also **alter database**, **create database** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_helplog**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_helpdevice* on page 402
- *sp_logdevice* on page 490

sp_helpmaplogin

Displays mapping information.

Syntax

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

Parameters

- ***authentication_mech* –** is one of the valid values specified for the **authenticate with** option in **create login** and **alter login**.
- ***client_username* –** is an external username.

Examples

- **Example 1 –** Displays information about all logins:

```
sp_helpmaplogin
```

```
authentication   client name   login name
--------------   -----------   ------------------
NULL             jsmith        guest
LDAP             NULL          create login
```

Usage

If you do not include any parameters, **sp_helpmaplogin** displays login information about all users currently logged in to the SAP ASE server. Restrict the output to specific sets of client user names or authentication mechanists by using the parameters.

See also

- *sp_maplogin* on page 503

sp_helpobjectdef

(Component Integration Services only) Reports owners, objects, and type information for remote object definitions.

Syntax

```
sp_helpobjectdef [objname]
```

Parameters

- ***objname* –** is the name of the object as it is defined in the `sysattributes` table. The *objname* 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 *objname* in quotes.

Examples

- **Example 1 –** Displays all remote object definitions in the current database:

```
sp_helpobjectdef
```

- **Example 2 –** Displays remote object definitions for the `tb1` table owned by the database owner:

```
sp_helpobjectdef "dbo.tb1"
```

Usage

If no *objname* is supplied, **sp_helpobjectdef** displays all remote object definitions.

A server name is not permitted in the *objname* parameter.

See also **create table**, **create existing table**, **drop table** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_helpobjectdef**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addobjectdef* on page 37
- *sp_dropobjectdef* on page 279
- *sp_helpserver* on page 434

sp_helpremotelogin

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

- **Example 1 –** Displays information about all the remote users of the remote server GATEWAY:

  ```
  sp_helpremotelogin GATEWAY
  ```

- **Example 2 –** Displays information about all the remote users of all the remote servers known to the local server:

  ```
  sp_helpremotelogin
  ```

Usage

sp_helpremotelogin reports on the remote logins for the specified server, when *remoteserver* is given, or on all servers, when no parameter is supplied.

Permissions

Any user can execute **sp_helpremotelogin**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addremotelogin* on page 40
- *sp_dropremotelogin* on page 281
- *sp_helpserver* on page 434

sp_helprotect

Reports on permissions for database objects, users, groups, or roles.

Syntax

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

Parameters

- ***name*** **–** is either the name of the table, view, stored procedure, SQLJ stored procedure, SQLJ function, user-defined function, name of a user, 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 the name of the user, group, or role in the current database.
- **grant –** displays the privileges granted on *name* to *username* with **grant** option. If *username* is null, **sp_helprotect** lists all privileges granted with grant option on *name*.
- **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*** **–** lists privileges granted through *role_name*.
- ***permission_name*** **–** allows **sp_helprotect** to provide information (grantor name, grantee name, table/column name, grantability) for any specific permission granted in a given database.

The value of this parameter can be any value from the `sysprotects.action` column.

Examples

- **Example 1 –** This series of **grant** and **revoke** statements, executing **sp_helprotect titles** results in this display:

```
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
go
sp_helprotect titles
```

```
grantor grantee type  action   object   column
       predicate  grantable
------- ------  -----   ------   ------   ------
     ---------   ---------
dbo    judy     Grant   Select   titles   All
        0           FALSE
dbo    judy     Grant   Update   titles   advance
    0           FALSE
dbo    judy     Grant   Update   titles   notes
      0           FALSE
dbo    judy     Grant   Update   titles   pub_id
     0           FALSE
dbo    judy     Grant   Update   titles   pubdate
    0           FALSE
dbo    judy     Grant   Update   titles   title
      0           FALSE
dbo    judy     Grant   Update   titles   title_id
   0           FALSE
dbo    judy     Grant   Update   titles
  total_sales 0           FALSE
dbo    judy    Grant   Update   titles   type       0          FALSE
dbo    judy     Grant   Select   titles   all
        0           TRUE
```

- **Example 2 –** Issuing the following **grant** statement results in **sp_helprotect** displaying the following:

```
grant select, update on titles(price, advance)
    to mary
    with grant option
go
sp_helprotect titles
```

```
grantor   grantee   type    action   object   column   predicate
grantable
-------   -------   ------  -------  ------   ------   ---------
 --------
dbo       mary      Grant   Select   titles   advance  0
 TRUE
dbo      mary     Grant   Select   titles   price    0         TRUE
dbo       mary      Grant   Update   titles   advance  0
```

```
 TRUE
dbo      mary     Grant   Update   titles   price    0         TRUE
```

- **Example 3 –** Displays all the permissions that “judy” has in the database:

```
sp_helprotect judy
```

- **Example 4 –** Displays any permissions that “csmith” has on the `sysusers` table, as well as whether “csmith” has **with grant option** which allows “csmith” to grant permissions to other users:

```
sp_helprotect sysusers, csmith, "grant"
```

```
grantor  grantee  type    action     object    column  predicate
grantable
-------- -------  ------  ---------  --------  ------  ---------
--------
dbo     doctor   Grant  Delete    sysusers  All    0        FALSE
dbo     doctor   Grant  Insert    sysusers  All    0        FALSE
dbo      doctor    Grant  References sysusers  All     0
FALSE
```

- **Example 5 –** Displays information about the permissions that the doctor role has in the database:

```
sp_helprotect doctor
```

```
grantor  grantee  type   action     object   column   predicate
grantable
-------  ------  -----  ---------  --------  ------   -------   -
---------
dbo     doctor  Grant  Delete    sysusers  All     0        FALSE
dbo     doctor  Grant  Insert    sysusers  All     0        FALSE
dbo     doctor  Grant  References sysusers  All     0        FALSE
```

- **Example 6 –** Displays information on all roles granted to “csmith”:

```
sp_helprotect csmith, null, null, "granted"
```

```
grantor grantee   type   action     object    column  predicate
 grantable
------- --------  ------ ---------- --------
------- ---------- ---------
dbo    csmith   Grant  Update    sysusers  All    0        FALSE
dbo    doctor   Grant  Delete    sysusers  All    0        FALSE
dbo    doctor   Grant  Insert    sysusers  All    0        FALSE
dbo     doctor    Grant  References sysusers  All     0
FALSE

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

- **Example 7 –** Displays information on all active roles granted to “rpillai”:

```
sp_helprotect rpillai, null, null, "enabled"
```

```
grantor grantee   type   action object         column  predicate
grantable
------- --------  ------ ------ -------------  ------- ---------
```

```
-------
dbo     public    Grant  Select sysattributes  All     0
 FALSE

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

- **Example 8 –** Advises that SQLJ function access is public:

```
sp_helprotect function_sqlj
```

```
Implicit grant to public for SQLJ functions.
```

- **Example 9 –** Uses the action “Decrypt” from `sysprotects.action`:

```
sp_helprotect @permission_name = "Decrypt"
```

```
grantor  grantee  type   action   object     column   predicate
grantable
-------  -------- -----  -------  -------    ------   ---------
---------
sa1     hr_login Grant  Decrypt  employee   ssn      0         TRUE
sa1     hr_role  Grant  Decrypt  employee   ssn     0         FALSE
```

- **Example 10 –**

  Displays the name of the predicated privilege in the output:

```
grant select, update, on tab1 where col1 = 8 as pred1 to robert
grant select, delete on tab1 where col1 = 9 to robert, joffrey
grant select, delete, update on tab1 where col2 = 10 as pred2 to
role1,
group1
```

```
sp_helprotect tab1
```

```
grantor  grantee type  action object column predicate         gr
antable
-----------------------------------------------------------------
------
 dbo    joffrey Grant Delete tab1   All    tab1_fdoIidqcSKLm  FALSE
 dbo    joffrey Grant Select tab1   All    tab1_fdoIidqcSKLm  FALSE
 dbo     group1  Grant Delete tab1   All    pred2             FALSE
 dbo     group1  Grant Select tab1   All    pred2             FALSE
 dbo     group1  Grant Update tab1   All    pred2             FALSE
 dbo     role1   Grant Delete tab1   All    pred2             FALSE
 dbo     role1   Grant Select tab1   All    pred2             FALSE
 dbo     role1   Grant Update tab1   All    pred2             FALSE
 dbo    robert  Grant Delete tab1   All    tab1_fdoIidqcSKLm  FALSE
 dbo     robert  Grant Select tab1   All    pred1             FALSE
 dbo    robert  Grant Select tab1   All    tab1_fdoIidqcSKLm  FALSE
 dbo     robert  Grant Update tab1   All    pred1             FALSE
```

Usage

- **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, a role, or a permission name. 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*, the SAP ASE server returns information on all roles activated by the current specified user.
- If the specified user is not the current user, the SAP ASE 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.

sp_helprotect reports information on encrypted columns, encryption keys, and users as follows:

- Tables and columns – reports who has been granted **decrypt** permission and on which columns.
- Encryption keys – reports who has been granted **select** permission.
- Users – indicates users who have been granted **create encryption key** permission.

sp_helprotect reports information on predicated privileges by listing the name of the predicated privilege, if any, as an extra column in the output.

See also **grant**, **revoke** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_helprotect**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_activeroles* on page 4
- *sp_displayroles* on page 240

sp_helpsegment

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

- **Example 1 –** Reports information about all segments in the current database:

  ```
  sp_helpsegment
  ```

  ```
  segment name                           status
  ------- ------------------------------ ------
        0 system                              0
        1 default                             1
        2 logsegment                          0
        3 seg1                                0
        4 seg2                                0
        5 seg3                                0
        6 seg4                                0
  ```

- **Example 2 –** Reports information about the segment named `order_seg`. This includes database tables and indexes that bond to this segment—the tables/indexes currently having this segment specified at the table/index level—as well as the objects currently on this segment (partitions that are actually located on this segment). In addition, this example reports the total number of pages, free pages, used pages, and reserved pages on this segment:

```
sp_helpsegment seg1
```

```
segment name                           status
------- ------------------------------ ------
      3 seg1                                0

device                 size           free_pages
---------------------- -------------- -----------
pubs_dev1              2.0MB                240

Objects on segment ‘seg1’:

table_name      index_name          indid   partition_name
----------- ------------- ------ ---------------
fictionsales    fictionsales            0   q1
pb_fictionsales pb_fictionsales         0   lov

Objects currently bound to segment ‘seg1’:
```

```
table_name  index_name  indid
----------  ----------  -----
new_titles  new_titles      0

total_size    total_pages    free-pages    used_pages    reserved
pages
----------    -----------    ----------    ----------    --------
------
2.0MB         256            240           16            0
```

- **Example 3 –** Reports information about the `default` segment. The keyword **default** must be enclosed in quotes. The output has been abridged due to length.

```
sp_helpsegment "default"
```

```
segment  name    status
-------  ------  ------
      1  default       1

device    size      free_pages
------    ----      -----------
master    14.0MB            303
pubs_dev1  2.0MB            240
pubs_dev2  2.0MB            232
pubs_dev3  2.0MB            232
pubs_dev4  2.0MB            240

Objects on segment ‘default’:
```

```
table_name     index_name     indid  partition_name
----------     ----------     -----  --------------
au_pix         au_pix             0  au_pix_864003078
au_pix         tau_pix            0  tau_pix_864003078
...
titles         title_idx           0  p1
titles          title_idx          0  p2
titles          title_idx          0  p3
titles          title_idx          0  title_idx_985051514

Objects currently bound to segment ‘default’:

table_name     index_name     indid
----------     ----------     -----
au_pix         au_pix             0
...
titleauthor    titleidind         3
titles         title_idx          1

total_size   total_pages   free_pages   used_pages   reserved_pages
----------   -----------   ----------   ----------   --------------
22.0MB        2816          1247         1569          0
```

- **Example 4 –** Reports information about the segment on which the transaction log is stored:

```
1> sp_helpsegment "logsegment"
2> go
```

```
 segment name       status
 ------- ---------- ------
       2 logsegment      0

 device      device size
 ------      ------
 master      14.0MB
 pubs_dev1    2.0MB
 pubs_dev2    2.0MB
 pubs_dev3    2.0MB
 pubs_dev4    2.0MB

 free_pages
 -----------
        1239

Objects on segment 'logsegment':

 table_name index_name indid  partition_name
 ---------- ---------- ------ --------------
 syslogs    syslogs        0  syslogs_8

Objects currently bound to segment 'logsegment':

 table_name index_name indid
 ---------- ---------- ------
 syslogs    syslogs        0

total_size    total_pages    free_pages    used_pages    reserved
_pages
------------- -------------- ------------- ------------- --------
-------
22.0MB        2816           1239          13            0
```

```
(return status = 0)
```

Usage

There are additional considerations when using **sp_helpsegment**:

- **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, the SAP ASE server automatically creates the `system`, `default`, and `logsegment` segments. Use **sp_addsegment** to add segments to the current database.
- If you specify a log segment from a dedicated log database for the *segname* parameter, **sp_helpsegment** reports the number of free pages in the log segment.
- 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.

See also **create index**, **create table** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_helpsegment**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addsegment* on page 43
- *sp_dropsegment* on page 287
- *sp_extendsegment* on page 330
- *sp_helpdb* on page 394
- *sp_helpdevice* on page 402
- *sp_placeobject* on page 577

sp_helpserver

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

- **Example 1 –** Displays information about the remote server GATEWAY:

  ```
  sp_helpserver GATEWAY
  ```

- **Example 2 –** Displays information about the local Backup Server:

  ```
  sp_helpserver SYB_BACKUP
  ```

  ```
  name        network_name    security_mechanism    server_principal
             class
     status
  id cost
  ----------   --------------  --------------------  --------------
  ----
             -----
  -----------------------------------------------------------------
  ```

```
----
      -----
SYB_BACKUP      SYB_BACKUP                NULL             NULL
           NULL
timeouts, no net password encryption, writable, enable login
redirection 1
       NULL
```

- **Example 3 –** Displays information about all the remote servers known to the local server:

```
sp_helpserver
```

Usage

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 (CIS) is installed, **sp_helpserver** lists the security mechanism, server principal name, and server class for each server.

Permissions

Any user can execute **sp_helpserver**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addserver* on page 46
- *sp_dropserver* on page 289
- *sp_helpremotelogin* on page 424
- *sp_serveroption* on page 623

sp_helpsort

Displays the SAP ASE server's default sort order and character set.

Syntax

```
sp_helpsort
```

Parameters

- **None. –**

Examples

- **Example 1 –** 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:

```
sp_helpsort
```

```
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 { | } ~
```

- **Example 2 –** 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
```

- **Example 3 –** 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.
```

- **Example 4 –** For case-insensitive character sets, the name and sort order ID of available case-insensitive sort orders is listed:

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

Usage

Binary sort order is the default.

Permissions

Any user can execute **sp_helpsort**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_helptext

Displays the source text of a compiled object, as well as the text for user-defined functions, computed columns, or function-based index definitions.

Syntax

```
sp_helptext objname[,grouping_num][, numlines[, printopts]]]
```

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.
- ***grouping_num*** **–** is an integer identifying an individual procedure, when *objname* represents a group of procedures. This parameter tells **sp_helptext** to display the source text for a specified procedure in the group.

  This parameter also specifies the start line number from which to generate the SQL text, when the *printops* argument is used.

  Note: Views, defaults, and other non-procedural objects are never grouped; use *number* only for groups of procedures.
- ***numlines*** **–** specifies the numbers of lines for which to generate SQL text. If the argument *printopts* is also used with **showsql**, *numlines* specifies the number of lines of SQL text to display; if *printopts* is used with **context**, *numlines* is treated as the context block width surrounding the starting line number.
- ***printopts*** **–** supports various comma-separated properties of the output format. One or more of these print options can be specified, in any order, as a comma-separated string:
  - **showsql** – generates formatted SQL output for the compiled object. If **showsql** does not appear in the *printopts* list, this property is not invoked.
  - **linenumbers** – produces line numbers for each line of SQL output.
  - **comments** – produces the line numbers as a comment field (`/*<nnn>*/`), so that the generated SQL can still recreate the compiled object, without furter edits, if necessary.

- **context** – produces a context block of output around a specified starting line number. If no, or null, *numlines* parameter is called, a default context block of five lines, generated before and after the line number of interest, is supplied.
- **noparams** – suppresses the automatically generated parameter information. Use this print option to produce only the relevant portion of SQL output for the compiled object.
- **ddlgen** – generates the SQL text as a DDL script, prefacing the output with a **use *database*** command and a **drop *object*** command. This allows you to reproduce almost exactly the SQL required to recreate most compiled objects, such as procedures, triggers, views, defaults, and rules.

The print options **ddlgen** and **context** are mutually exclusive specifiers. Used together, they raise an error. To get line numbers when you are displaying a context block of SQL text, use the **context** and **linenumbers** specifiers.

Examples

- **Example 1 –** Displays the source text of `pub_idrule`. Since this rule is in the `pubs2` database, execute this command from `pubs2`:

```
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]"
```

- **Example 2 –** Displays the source text of **sp_helptext**. Since system procedures are stored in `sybsystemprocs`, execute this command from `sybsystemprocs`:

```
sp_helptext sp_helptext
```

- **Example 3 –** Displays the source text of the **myproc** group behavior where you specify no *number* argument. The number of the procedure displays beside the text:

```
sp_helptext myproc
```

```
# Lines of Text
---------------
2
number
text
---------------
1
create procedure myproc; as select 1
2
```

```
create procedure myproc;2 as select 2
(2 rows affected)
```

- **Example 4 –** Displays the source text of **myproc**, specifying a procedure in the *myproc* group but displaying no grouping number.

```
sp_helptext myproc, 2
```

```
# Lines of Text
---------------
1
text
----------------
create procedure myproc;2 as select 2
```

- **Example 5 –** Generates text for **sp_help**:

```
sp_helptext sp_help,NULL,NULLM 'showsql'
```

- **Example 6 –** To generate text for **sp_help**, producing line numbers:

```
sp_helptext sp_help, NULL,NULL,'showsql,linenumbers'
```

- **Example 7 –** To generate the text for **sp_help**, in a context block of seven lines starting at line 25, with output generated in a comment block:

```
sp_helptext sp_help,25,7,'showsql,comments,context'
```

- **Example 8 –** Generates the text for **sp_droptabledef**, producing the output as a stand-alone DDL script that you can use to recreate the procedure:

```
sp_helptext sp_droptabledef,NULL,NULL,'showsql,ddlgen'
-------------
use sybsystemprocs
```

```
-------------
IF EXISTS (SELECT 1 FROM sysobjects
WHERE name = 'sp_droptabledef'
AND type = 'P'
DROP PROCEDURE sp_droptabledef
--------------
/*Sccsud="%Z%generic/sproc/src/%M%%I%%G%"*/
/*
**Omni only
*/
create procedure sp_droptabledef
  @tablename varchar(92) /*tablename*/
as begin
  declare @status int
  exec @status = sp_dropobjectdef @tablename
  return(@status)
end
----------
(return status = 0)
```

- **Example 9 –** Uses **sp_helptext** on a view created with delimited identifiers. You do not need **set quoted_identifier on** to extract the SQL defining the view. You do need it **ON** to create objects using delimited identifiers.

```
set quoted_identifier ON
---------
create table "t one"
        (c1 int,
        "c two" varchar(10),
        "c three int)
---------
create table "t two"
        ("t2 one" int,
        "t2 two" varchar(10),
        t2_three int)
------------
create view "v one" as
    select * from "t one"
    UNION
    select "t2 one","t2 two",t2_three
    from "t two"
-----------------
```

- **Example 10 –** The SAP ASE server displays the text for predicates. **sp_helptext** can be supplied the predicate's user-defined name, if there is one, or its internal name. For example:

```
sp_helptext pred1
```

```
# Lines of Text
 ---------------
          1
text ---------------------------------------------------
 grant select on tab1 where col1 = 5 as pred1 to robert
```

Usage

There are additional considerations when using **sp_helptext**:

- **sp_helptext** truncates trailing spaces when displaying the source text from `syscomments`
- **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.
- The source-text is displayed using `char(255)`, so trailing spaces are present in the displayed text. The text stored in `syscomments` may not include these trailing spaces. `syscomments` stores the text "as supplied," so another application or tool may not have included these trailing spaces. Because of this, you should not use **sp_helptext** to get a copy of the text stored. Instead, use other tools like **defncopy**.
- **sp_helptext** looks for the source text in the `syscomments` table in the current database.
- You can encrypt the source text with **sp_hidetext**.
- When **sp_helptext** operates on a group of procedures, it prints the number column from **syscomments** in addition to the source text.
- A system security officer can prevent the source text of compiled objects from being displayed to most users who execute **sp_helptext**. To restrict **select** permission on the

`text` column of the `syscomments` table to the object owner or a system administrator, use **sp_configure** to set the **select on syscomments.text column** parameter to **0**. This restriction is required to run SAP ASE in the evaluated configuration. See the *System Administration Guide* for more information about the evaluated configuration.

- Even when you use **sp_helptext** in **ddlgen** mode, the **showsql** print option is required.
- The object with text that you want to retrieve must reside in the database where the procedure is executed.
- If the text is either hidden or not in `syscomments`, an error message is raised. If, however, you request a context block output, and the text is missing or hidden, a message reporting the missing text is printed, but no error is raised.
- Text generated using the **ddlgen** print option may still fail to create a compiled object correctly if it contains references to other objects, such as temporary tables, that do not already exist when the generated script is executed.
- If the compiled object contains a `select *` statement, it usually reflects the entire column list of the table this statement references.
- You can generate SQL text for compiled objects created with quoted identifiers, but if the compiled object contains a `select *` statement, the expanded column list appears with bracketed identifiers after the SAP ASE server writes the text to `syscomments`. For example:

  ```
  [this column], [column name with space]
  ```

  It is not necessary to **set quoted_identifier ON** when generating text for compiled objects that are themselves, or use, delimited identifiers.

Permissions

The permission checks for **sp_helptext** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the object owner, the database owner, or a user with `own database` privilege.
Disabled	With granular permissions disabled, you must be the object owner, database owner, or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38

Information	Values
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_checksource* on page 119
- *sp_configure* on page 167
- *sp_hidetext* on page 449

sp_helpthread

Displays the current thread pool configuration.

Syntax

```
sp_helpthread [pool_name]
```

Parameters

- ***pool_name* –** name of the pool to show. If *pool_name* is null, **sp_helpthread** displays configuration information about all pools.

Examples

- **Example 1 –** Displays information about all pools:

```
sp_helpthread
```

```
name                type    size    idle_timeout
         description
-----------------  ------  ----    ------------
         --------------------------------------------
pubs_pool           Engine     2            100
                                                 NULL
syb_blocking_pool      RTC     4              0
         A pool dedicated to executing blocking calls
syb_default_pool    Engine     1            100
         The default pool to run query sessions
```

```
syb_system_pool       RTC     4               0
          The I/O and system task pool
```

- **Example 2 –** Displays information about the pubs_pool:

```
sp_helpthread pubs_pool
```

```
 name       type  size  idle_timeout  description
---------  ------  ----  ------------  ------------
pubs_pool  Engine     2           100         NULL

thread_id  osthread_id  state  affinity  instance_id
---------  -----------  -----  --------  -----------
       12   1248065856   IDLE      NULL            0
       13   1237576000   IDLE      NULL            0
```

Usage

sp_helpthread gathers information for its reports from the `monThread` monitoring table.

sp_helpthread produces output only in threaded mode.

Permissions

Any user can issue **sp_helpthread**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

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.

Syntax

```
sp_helpthreshold [segname]
```

Parameters

- ***segname*** **–** is the name of a segment in the current database.

Examples

- **Example 1 –** Shows all thresholds on the log segment:

  ```
  sp_helpthreshold logsegment
  ```

- **Example 2 –** Shows all thresholds on all segments in the current database:

  ```
  sp_helpthreshold
  ```

- **Example 3 –** Shows all thresholds on the default segment. Note the use of quotes around the reserved word “default”:

  ```
  sp_helpthreshold "default"
  ```

Usage

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.

Permissions

Any user can execute **sp_helpthreshold**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addthreshold* on page 49
- *sp_dropthreshold* on page 291
- *sp_helpsegment* on page 430
- *sp_modifythreshold* on page 520
- *sp_thresholdaction* on page 700

sp_helptrigger

sp_helptrigger lists all triggers created on the table specified by tablename; which command (**insert**, **update**, or **delete**) fires the trigger, and the trigger's order number.

Syntax

```
sp_helptrigger tablename
```

Parameters

- ***tablename* –** is the name of the table.

Permissions

Any user can execute **sp_helptrigger**.

sp_helpuser

Reports information about a particular user, group, or alias, or about all users, in the current database. Also identifies objects and user-defined datatypes owned by a users.

Syntax

```
sp_helpuser [name_in_db [, display_object]]
```

Parameters

- ***name_in_db* –** is **null** or name of a valid user in the current database.
- ***display_object* –** lists all objects and user-defined datatypes owned by *name_in_db* in the current database. If *name_in_db* is **null**, the objects and user-defined datatypes owned by the caller are listed. The output for objects includes object_name, object_type, and create_date, sorted by object_type and object_name. The output for user-defined datatype includes user type name.

Examples

- **Example 1 –** Displays information about all users in the current database:

  ```
  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
  ```

- **Example 2 –** Displays information about the database owner (user name “dbo”):

  ```
  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
  ```

- **Example 3 –** Displays objects owned by the user bill:

  ```
  sp_helpuser bill, display_object
  ```

  ```
  Object_name                Object_type                Create_date
  -----------                ------------               ---------
  proc_update_titles        stored procedures          Apr 28 2007
  04:47PM
  author                    user table                 Apr 27 2007
  04:47PM
  publisher                 user table                 Apr 27 2007
  05:47PM
  titles                    user table                 Apr 27 2007
  06:47PM
  ```

```
vw_author_in_ca          view                       Apr 27 2007
05:47PM
```

- **Example 4 –** Displays objects owned by the database owner (DBO):

  ```
  sp_helpuser 'dbo', display_object
  ```

  ```
  Object_name               Object_type             Create_date
  -----------             -------------           --------------
  enter_key                 encryption key            Sep 7 2007
  03:37PM
  sysalternatives           system table              Jul 17 2007
  09:25AM
  sysattributes             system table              Jul 17 2007
  09:25AM
  syscolumns                system table              Jul 17 2007
  09:25AM
  .....                     ......                    ...........
  sysquerymetrics           view                      Jul 17 2007
  09:25AM
  ```

Usage

sp_helpuser reports information about all users of the current database. If you specify a *name_in_db*, **sp_helpuser** reports information only on the specified user.

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.

Permissions

Any user can execute **sp_helpuser**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_adduser* on page 59
- *sp_dropuser* on page 293
- *sp_helpgroup* on page 408

sp_hidetext

Hides the source text for the specified compiled object, as well as the text of computed columns, predicates, and function-based index keys. **sp_hidetext** also encrypts the text for user-defined functions.

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

- **Example 1 –** Hides the source text of all compiled objects in the current database:

  ```
  sp_hidetext
  ```

- **Example 2 –** Hides the source text of the user-defined stored procedure, **sp_sort_table**, that is owned by Mary:

  ```
  sp_hidetext @objname = "sp_sort_table",
      @username = "Mary"
  ```

- **Example 3 –** Hides the source text of the stored procedure **pr_phone_list**:

  ```
  sp_hidetext "pr_phone_list"
  ```

- **Example 4 –** Hides the source text of all check constraints, defaults, and triggers defined on the table `my_tab`:

  ```
  sp_hidetext @tabname = "my_tab"
  ```

- **Example 5 –** Hides the source text of the view `my_vu` and all check constraints, defaults, and triggers defined on the table `my_tab`:

  ```
  sp_hidetext "my_vu", "my_tab"
  ```

- **Example 6 –** Hides the source text of all compiled objects that are owned by Tom:

  ```
  sp_hidetext @username = "Tom"
  ```

Usage

There are additional considerations when using **sp_hidetext**:

- **sp_hidetext** hides the source text for the specified compiled object.

  Warning! Before executing **sp_hidetext**, make sure you have a backup of the source text. The results of executing **sp_hidetext** are not reversible.

- If you do not provide any parameters, **sp_hidetext** hides the source text for all compiled objects in the current database.
- **sp_helprotect ... expand_predicate** prints a null predicate if text has been hidden.
- Hidden `syscomments.text` is not available for use by **sp_helprotect**.
- The SAP ASE server allows the predicate owner or the SSO to hide the text of a predicate. Hidden `syscomments.text` is not available for use by **sp_helprotect**. Users must be warned that the **expand_predicate** option of **sp_helprotect** prints a null predicate if text has been hidden.
- If you use **sp_hidetext** followed by a cross-platform **dump** and **load**, you must manually drop and re-create all hidden objects.

See also:

- **dump database**, **dump transaction**, **load database**, **load transaction** in *Reference Manual: Commands*
- *Transact-SQL User's Guide* for more information about hiding source text.

Permissions

The permission checks for **sp_hidetext** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage database` privilege. Any user can execute **sp_hidetext** to hide the source text of their own compiled objects.

Setting	Description
Disabled	With granular permissions disabled, you must be the datatype owner or a user with **sa_role**. Any user can execute **sp_hidetext** to hide the source text of their own compiled objects.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_checksource* on page 119

sp_import_qpgroup

Imports abstract plans from a user table into an abstract plan group.

Syntax

```
sp_import_qpgroup tab, usr, group
```

Parameters

- ***tab*** **–** is the name of a table from which to copy the plans. You can specify a database name, but not an owner name, in the form ***dbname..tablename***. The total length can be up to 255 characters long.
- ***usr*** **–** is the name of the user whose ID should be assigned to the abstract plans when they are imported.
- ***group*** **–** is the name of the abstract plan group that contains the plans to be imported.

Examples

- **Example 1 –** Copies plans from the table `moveplans` to the `new_plans` group, giving them the user ID for the database owner:

  ```
  sp_import_qpgroup moveplans, dbo, new_plans
  ```

Usage

There are additional considerations when using **sp_import_qpgroup**:

- **sp_import_qpgroup** copies plans from a user table to an abstract plan group in `sysqueryplans`. With **sp_export_qpgroup**, it can be used to copy abstract plan groups between servers and databases, or to copy plans belonging to one user and assign them the ID of another user.
- **sp_import_qpgroup** creates the abstract plan group if it does not exist when the procedure is executed.
- If an abstract plan group exists when **sp_import_qpgroup** is executed, it cannot contain any plans for the specified user. **sp_import_qpgroup** does not check the query text to determine whether queries already exist in the group. If you need to import plans for a user into a group where some plans for the user already exist:
  - Use **sp_import_qpgroup** to import the plans into a new plan group.
  - Use **sp_copy_all_qplans** to copy the plans from the newly-created group to the destination group. **sp_copy_all_qplans** does check queries to be sure that no duplicate plans are created.
  - If you no longer need the group you created for the import, drop the plans in the group with **sp_copy_all_qplans**, then drop the group with **sp_drop_qpgroup**.
- To create an empty table in order to bulk copy abstract plans, use:

  ```
  select * into load_table
  from sysqueryplans
  where 1 = 2
  ```

See also **create plan** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_import_qpgroup** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege.
Disabled	With granular permissions disabled, you must be the datatype owner or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_copy_all_qplans* on page 176
- *sp_copy_qplan* on page 178
- *sp_drop_all_qplans* on page 254
- *sp_drop_qpgroup* on page 255
- *sp_export_qpgroup* on page 329
- *sp_help_qpgroup* on page 370

sp_indsuspect

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

- **Example 1 –** Checks the table `newaccts` for indexes marked as suspect:

  ```
  sp_indsuspect newaccts
  ```

Usage

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.

See also **dbcc** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_indsuspect**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_jreconfig

Manages the Java PCA/JVM. Enables or disables arguments and directives, changes configuration values, and reports configuration values.

Note: You can safely change the **pca_jvm_module_path**, **pca_jvm_work_dir**, **pca_jvm_dbg_agent_port**, **pca_jvm_java_dbg_agent_suspend**, **pca_jvm_java_options**, and **pca_jvm_netio** arguments. Do not use **sp_jreconfig** to change other arguments or directives unless instructed to do so by Sybase Technical Support.

Syntax

```
sp_jreconfig {
    add array_arg, new_string |
    array_clear array_arg |
    array_enable array_arg |
    array_disable array_arg |
    delete array_arg, string_value |
    disable { directive | argument | array_arg, string_value } |
    enable { directive | argument | array_arg, string_value } |
    list { list_type [, formatted ] | units | units, units_type[,
formatted ] } |
    reload_config |
    report { directive[, formatted ] | directive, args[, formatted ]
        |argument[, formatted ] } |
    update { argument, old_value, new_value } }
```

Parameters

- **add –** adds a new argument to an argument array. Use **add** only with arguments where *units_type* is array.
- ***array_arg* –** is the name of an argument where *units_type* is array.
- ***new_string* –** is the string value for a new array element.
- **array_clear –** deletes all element in an argument array.
- **array_enable –** enables all elements in an argument array. Sets each array element to enabled.
- **array_disable –** disables, but does not delete, all elements in an argument array. Sets each element to disabled.
- **delete –** removes an existing element from an argument array. Use **delete** only with arguments where *units_type* is array.
- **disable –** disables the specified directive or argument.
- ***string_value* –** identifies an array element in the named argument array that is to be deleted, enabled, or disabled.
- **directive –** is the name of a valid directive.
- **argument –** is the name of a valid argument.
- **enable –** enables a directive or an argument.
- **list –** lists groups of related arguments as, for example, **sp_jreconfig list, directives** or **sp_jreconfig list, enabled**. Also, lists all arguments of a specific type as, for example, **sp_jreconfig list, units, string**. To see all current *units_types* values, use **sp_jreconfig list, units**.
- **formatted –** formats the displayed list for readability; longer values may be truncated.

  Note: In formatted reports, the process of improving readability may truncate wide columns. In addition, column headings may be overridden and may not match the actual table name. Do not format reports if the output is to be parsed or potential data truncation is not acceptable.

- ***list_type* –** specifies a type of list. Values are:
  - **directives** – list of directives
  - **enabled** – list of enabled arguments
  - **disabled** – list of disabled arguments
  - **argnames** – list of argument names, each argument's *units_type*, and the directive to which each belongs
- **units –** when used with **list**, generates a list of *units_type* currently in use.
- ***units_type* –** is a type of argument. Every argument has a *units_type* that identifies its type. Values are:
  - **switch**
  - **string**
  - **number**
  - **array**
- **reload_config –** reloads the configuration from the `sybpcidb` tables into memory. See *Restoring Default Configuration Values to sybpcidb* in *Java in Adaptive Server Enterprise*.
- **report –** creates a report based on arguments supplied. Usually used to generate a report for an argument to see its current value and whether or not it is enabled. Can also be used to generate a report for a directive or its arguments.
- ***directive* –** is any valid directive.
- **args –** is a keyword used with report to generate a list of argument names for the named directive. For example:

  ```
  sp_jreconfig report, "PCA_JVM", "args"
  ```
- **update –** modifies a string or numeric value for an argument where *units_type* is string, number, or array. You cannot modify an argument when *units_type* is switch.
- ***old_value* –** is a string or numeric value that identifies the existing argument or array element being updated.
- ***new_value* –** is a string or numeric value that defines the new argument or array element.

Examples

- **Generating Formatted Report –** Generates a formatted report for the **PCA_JVM_OPT** directive:

  ```
  sp_jreconfig "report", "PCA_JVM_OPT", "formatted"
  ```
- **Generating Report of Arguments –** Generates a report of the arguments of the **PCA_JVM_OPT** directive:

  ```
  sp_jreconfig "report", "PCA_JVM_OPT", "args"
  ```
- **Generating Report of Argument –** Generates a report for the argument **pca_jvm_netio**.

  ```
  sp_jreconfig "report", "pca_jvm_netio"
  ```

- **Generating Report of All Arguments –** Generates a report for all arguments that match "pca_jvm". A partial argument name generates a report for all matching arguments.

  ```
  sp_jreconfig "report", "pca_jvm_"
  ```

- **Generating Lists –** Displays a list of all directives and their state: enabled or disabled.

  ```
  sp_jreconfig "list", "directives"
  ```

- **Generating Lists –** Displays a list of all arguments, their units types, and directives.

  ```
  sp_jreconfig "list", "argnames", "formatted"
  ```

- **Generating Lists –** Displays a list of all currently enabled arguments.

  ```
  sp_jreconfig "list", "enabled"
  ```

- **Generating Lists –** Displays a formatted list of all array arguments.

  ```
  sp_jreconfig "list", "units", "array", "formatted"
  ```

- **Generating Lists –** Display a list of argument unit types. The report for this command is formatted by default. Using the “-formatted” option generates an error.

  ```
  sp_jreconfig "list", "units"
  ```

- **Enabling Directives and Arguments –** Enables the **PCA_JVM_WORK_DIR** directive. You can use a partial directive name as long as it includes sufficient information to uniquely identify the directive.

  ```
  sp_jreconfig "enable", "PCA_JVM_WORK_DIR"
  sp_jreconfig "enable", "WORK_DIR"
  ```

- **Enabling Directives and Arguments –** Enables the **pca_jvm_netio** argument.

  ```
  sp_jreconfig "enable", "pca_jvm_netio"
  ```

- **Disabling Directives and Arguments –** Disables the **WORK_DIR** directive. This example uses a partial directive name, which must include sufficient information to uniquely identify the directive.

  ```
  sp_jreconfig "disable", "WORK_DIR"
  ```

  Note: Disabling a directives causes its arguments to behave as disabled, but does not change their base states.

- **Disabling Directives and Arguments –** Disables the **pca_jvm_netio** argument.

  ```
  sp_jreconfig "disable", "pca_jvm_netio"
  ```

- **Disabling Directives and Arguments –** Disables array elements in **PCA_JVM_WORK_DIR**. The path, but not the permissions mask, is required. See *File and Network Access Using Java* in *Java in Adaptive Server Enterprise*.

  ```
  sp_jreconfig "disable", "pca_jvm_work_dir",
      "/some/path"
  ```

- **Updating String, Number, and Aray Arguments –** Updates a string argument. This example updates the file location of the **pca_jvm_log_filename** argument.

```
sp_jreconfig "update", "pca_jvm_log_filename", "/old/path/
filename.log",
    "/new/path/filename.log"
```

Note: The **update** option cannot be used with directives or switch argument as these items can not be modified.

- **Updating String, Number, and Array Arguments –** Updates a number argument. Numeric values must be enclosed in quotes (as strings) for the stored procedure. The SAP ASE server stores them as numeric values.

  ```
  sp_jreconfig "update", "pca_jvm_min_port", "1026", "2056"
  ```

- **Updating String, Number, and Array Arguments –** For the **PCA_JVM_WORK_DIR** directive, **work_dir** values consist of a path and an optional permission mask. Although the permission mask is optional, you must include the original string path to identify the **work_dir**. A permission mask is optional. If it is not supplied, the system uses a default mask with an octal equivalent of 0666. Example a does not set a permission mask; it uses the default mask. Examples b and c each set a permission mask of 0644.

  ```
  [a] sp_jreconfig "update", "pca_jvm_work_dir",
  "/old/path","/new/working/directory"

  [b] sp_jreconfig "update", "pca_jvm_work_dir",
   "/old/path", "/new/working/directory(u=rw,go=r)"

  [c] sp_jreconfig "update", "pca_jvm_work_dir",
   "/old/path", "/new/working/directory(u+w,ugo+r)"
  ```

- **Adding Array Elements –** Adds new elements to the **pca_jvm_work_dir** argument array in the **PCA_JVM_WORK_DIR** directive. Example a uses the default mask. Examples b and c each set a permissions mask of 0644. (The mask is evaluated from left to right.)

  ```
  [a] sp_jreconfig "add", "pca_jvm_work_dir",
   "/new/working/directory"
  ```

  ```
  [b] sp_jreconfig "add", "pca_jvm_work_dir",
   "/new/working/directory(u=rw,go=r)"
  ```

  ```
  [c] sp_jreconfig "add", "pca_jvm_work_dir",
   "/new/working/directory(u+w,ugo+r)"
  ```

- **Deleting Array Elements –** Deletes an array element in **pca_jvm_work_dir**.

  ```
  sp_jreconfig "delete", "pca_jvm_work_dir",
          "/new/working/directory"
  ```

  Note: To delete a an element in **pca_jvm_work_dir** in the **PCA_JVM_WORK_DIR** directive, you can specify a partial string if the string supplied identifies a unique record. The permission mask is not required; you only need to supply the path even if the **work_dir** element was originally defined with a specific permission mask.

- **Enabling or Disabling All Elements in an Array –** Disables all elements in the **pca_jvm_work_dir** array.

```
sp_jreconfig "array_enable", "pca_jvm_work_dir"
```

- **Enabling or Disabling All Elements in an Array –** Disables all elements in the **pca_jvm_work_dir** array.

  ```
  sp_jreconfig "array_disable", "pca_jvm_work_dir"
  ```

- **Clearing All Records in an Array –** Deletes all records in the **pca_jvm_work_dir** array and creates an empty array.

  ```
  sp_jreconfig "array_clear", "pca_jvm_work_dir"
  ```

- **Reloading Default Configuration Values –** Loads the configuration values stored in `sybpcidb` into memory.

  ```
  sp_jreconfig "reload_config"
  ```

Usage

There are additional considerations when using **sp_jreconfig**.

- Enabling and disabling a directive works like a toggle. When a directive is:
  - Enabled – the SAP ASE server uses the configured value (enabled or disabled) of each argument. This is the value stored in `sybpcidb`.
  - Disabled – the SAP ASE server disregards the configured value (enabled or disabled) of each argument and treats all arguments of the directive as disabled, although the base value of each argument is retained in `sybpcidb`.
- Arguments can be individually enabled or disabled. The types of arguments are:
  - Switch – these arguments turn a feature on or off. For example, if the argument for logging is enabled, a log file is generated; if the argument for logging is disabled, no log file is generated.
  - String – these arguments are for string and number values. Enabling a string or number argument ensures that the SAP ASE server uses the configured value. Disabling a string or number argument means that the SAP ASE server ignores the configured value and uses the default value. The configured and default values may or may not be the same.
  - Array – an array argument is a collection of related string arguments, each of which can be individually enabled or disabled. When an individual string argument (or element) is disabled, its value is ignored and the behavior is the same as if the element had been deleted. When enabled, the argument value is included in the collection and is active.

    Array arguments can be enabled or disabled at will; you do not have to delete a value and then re-enter it later on.

`pca_jvm_module_path` configures the path to the JVM shared-object library. If you use a JRE other than that supplied by SAP, you must configure this argument to point to a location accessible to the PCA/JVM. This can be an absolute path or a relative path that extends $SYBASE. If an absolute path, start the path with "/" on UNIX or "\" on Windows. Otherwise, the SAP ASE server assumes a relative path and looks under $SYBASE.

Table 4. pca_jvm_module_path

Argument	Units Type	Default Value	Default State	Description
pca_jvm_module_path	string	Platform specific	Enabled	The location of the JVM shared library using a relative path located under $SYBASE, or a fully qualified filename.

This table describes `pva_jvm_opt`.

Do not change default values unless instructed to do so by SAP Technical Support.

Table 5. pva_jvm_opt

Argument	Units Type	Default Value	Default State	Description
pca_jvm_abort	switch	On	Enabled	Abort abort(2) all on any failure (dangerous).
pca_jvm_allow_unchecked_sockops	switch	N/A	Disabled	Allow unchecked socket operations.
pca_jvm_debug	switch	N/A	Disabled	Report PCA_DEBUG requests.
pca_jvm_except	switch	N/A	Enabled	Report excepting PCA/VM JNI/JVM invocations.
pca_jvm_heap_ratio	string	0.3	Enabled	VM Heap / PCI memory ratio.
pca_jvm_jvmti	switch	N/A	Disabled	Java VM Tools Interface.
pca_jvm_min_port	number	1026	Enabled	Allow VM network support.
pca_jvm_netio	switch	N/A	Disabled	Allow VM network support.
pca_jvm_report	switch	N/A	Disabled	Report PCA/VM JNI/JVM invocations.
pca_jvm_security_manager_enabled	switch	N/A	Disabled	Enable the SecurityManager in the PCA/JVM.
pca_jvm_sigcache_density	number	100	Enabled	PCA/VM signature cache target density.
pca_jvm_sigcache_enabled	switch	N/A	Enabled	Enable PCA/VM signature cache.

Argument	Units Type	Default Value	Default State	Description
pca_jvm_sigcache_fixed_ratio	number	50	Enabled	PCA/VM signature cache size percentage fixed.
pca_jvm_sigcache_freeboard	number	30	Enabled	PCA/VM signature cache space recovery percentaga on cache sweeps.
pca_jvm_sigcache_size	number	512	Enabled	PCA/VM signature cache size in KBytes.
pca_jvm_sigcache_size_type	number	1	Enabled	PCA/VM signature cache size_type 0:AS_PCT 1:Kbyte 2:Mbyte.
pca_jvm_sigcache_washcycle	number	1000	Enabled	PCA/VM signature cache wash daemon cycle time (ms).
pca_jvm_sigcache_washdaemon	switch	N/A	Disabled	Enable PCA/VM signature cache wash daemon.
pca_jvm_strace	switch	N/A	Enabled	Produce stack traces on none emulated VM handles.

`pca_jvm_dir_options` configures the directory definitions used by the JVM for the ROOT and TEMP directories. Do not change these values unless you are a knowledgeable user or you have been directed to do so by SAP Technical Support.

Warning! Use this directive with care. The **pca_jvm_tmp_dir** in the **PCA_JVM_DIR_OPTIONS** directive should always point to the system temporary directory. Changing this location can be a serious security risk. The JVM allows files to be opened for reading and writing, and allows file creation in this directory.

Table 6. pca_jvm_dir_options

Argument	Units type	Default value	Default state	Description
pca_jvm_root_dir	string	Platform-specific	Enabled	Absolute path to the system root directory. Required for file I/O.
pca_jvm_tmp_dir	string	Platform-specific	Enabled	Absolute path to the system temporary directory. Required for file I/O.

`pca_jvm_work_dir` configures the JVM trusted directories. This argument consists of a collection of specific locations in your file system where your Java program classes can perform certain file I/O operations. Each directory can have an optional permission mask that defines which file I/O operations are allowed in each directory.

Table 7. pca_jvm_work_dir

Argument	Units Type	Default Value	Default State	Description
pca_jvm_work_dir	array	Platform-specific	Disabled	The absolute path (and optional permission mask) where the JVM is allowed to do file I/O. See *File and Network Access Using Java* in *Java in Adaptive Server Enterprise*.

`pca_jvm_min_jni_version` configures minimum backward compatible JNI version.

Table 8. pca_jvm_min_jni_version

Argument	Units Type	Default Value	Default State	Description
pca_jvm_min_jni_version	string	'JNI_VERSION_1_2'	Enabled	Minimum backward compatible JNI version.

`pva_jvm_logging` configures JRE/VM logging options.

Table 9. pva_jvm_logging

Argument	Units Type	Default Value	Default State	Description
pca_jvm_ase_logging	switch	N/A	Enabled	Configure SAP ASE logging.
pca_jvm_log_filename	string	'`/tmp/Java_vm.log1`'	Disabled	A fully qualified filen ame that the VM uses for logging.

`pca_jvm_ext_class_loader` configures global and database extension class loaders.

Table 10. pca_jvm_ext_class_loader

Argument	Units Type	Default Value	Default State	Description
pca_jvm_ext_class_loader_global	array	none	Disabled	Global Extension Class Loader.

Argument	Units Type	Default Value	Default State	Description
pca_jvm_ext_class_loader_dbase	array	none	Disabled	Database Extension Class Loader.

`pva_jvm_java_options` configures Java start-up options, both normal and extended.

Table 11. pva_jvm_java_options

Argument	Units Type	Default Value	Default State	Description
pca_jvm_java_options	array	"-Djava.awt.headless=true"	Enabled	Run Java in headless mode.
pca_jvm_java_options	array	"-Djava.compiler=JIT"	Enabled	Force JIT compilation and optimization.
pca_jvm_java_options	array	"-XX:+CITune:"	Disabled	Time spent in JIT Compiler (1.4 only).
pca_jvm_java_options	array	"-XX:+Use AltSigs"	Disabled	This option seems to crash the J2SE.
pca_jvm_java_options	array	"-XX:CodeCacheExpansionSize=512000"	Enabled	Code Cache extension size.
pca_jvm_java_options	array	"-Xbatch"	Disabled	Disabled background compilation.
pca_jvm_java_options	array	"-Xcheck:jni"	Enabled	Perform additional checks for JNI functions.
pca_jvm_java_options	array	"-Xfuture"	Disabled	Perform strict checks, anticipating future default.
pca_jvm_java_options	array	"-Xincgc"	Disabled	Enable incremental garbage collection.
pca_jvm_java_options	array	"-Xint"	Disabled	Interpreted mode execution only.
pca_jvm_java_options	array	"-Xloggc:./myGClog"	Disabled	Log GC status to a file with time stamps.
pca_jvm_java_options	array	"-Xmixed"	Disabled	Mixed mode execution (default).

Argument	Units Type	Default Value	Default State	Description
pca_jvm_java_options	array	"-Xms64m"	Disabled	Set initial Java heap size.
pca_jvm_java_options	array	"-Xmx64m"	Disabled	Set maximum Java heap size.
pca_jvm_java_options	array	"-XnoClassgc"	Disabled	Disable class garbage collection.
pca_jvm_java_options	array	"-Xprof"	Disabled	Output cpu profiling data.
pca_jvm_java_options	array	"-Xrs"	Disabled	Reduce use of OS signals by Java/VM.
pca_jvm_java_options	array	"-Xshare:auto"	Disabled	Configure shared class data (set to auto, off or on).
pca_jvm_java_options	array	"-Xss64m"	Disabled	Set Java thread stack size.
pca_jvm_java_options	array	-XX:MaxPermSize	Disabled	Sets the maximum size of the permanent heap
pca_jvm_java_options	array	"-enablesystemassertions"	Enabled	Enable Java/VM System Assertions - applies only to platforms using the Sun HotSpot (TM) JavaVM.
pca_jvm_java_options	array	"-esa"	Enabled	Enable All System Assertions - only applies to platforms using the Sun HotSpot (TM) JavaVM.
pca_jvm_java_options	array	"-verbose:class"	Disabled	Class loading within the JRE/VM.
pca_jvm_java_options	array	"-verbose:gc"	Disabled	Garbage Collection statistics.
pca_jvm_java_options	array	"-verbose:jni"	Disabled	Java Native Interface (JNI) invokations.

`pva_jvm_java_dbg_agent_port` configures the Java VM debug agent port number (used for debugging Java applications with a Java debugger). See *Java in Adaptive Server Enterprise* for more information.

Table 12. pva_jvm_java_dbg_agent_port

Argument	Units Type	Default Value	Default State	Description
pca_jvm_java_dbg_agent_port	number	8000	Disabled	Configure the port number and the Java VM Debug Agent.
pca_jvm_java_dbg_agent_suspend	switch	N/A	Disabled	Java VM Debug Agent starts suspended when enabled.

`pca_jvm_sys_device_path` configures platform-specific system device directories (required for Solaris).

Table 13. pca_jvm_sys_device_path

Argument	Unit Type	Default Value	Default State	Description
pca_jvm_sys_device_path	array	Platformspecific	Platform specific	Internal system option for Sun OS. DO NOT CHANGE.

Permissions

The permission checks for **sp_jreconfig** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server configuration` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_pciconfig* on page 573

sp_ldapadmin

Creates or lists an LDAP URL search string, verifies an LDAP URL search string or login, or specifies the access accounts and tunable LDAPUA-related parameters.

Syntax

```
sp_ldapadmin command [, option1 [, option2]]
```

Valid ***command* [, *option1* [, *option2*]]** options are:

```
    'set_primary_url', 'url'
    'set_secondary_url', 'url'
    'set_dn_lookup_url', 'url'
    'set_secondary_dn_lookup_url', 'url'
    'set_access_acct', 'distinguished_name', 'password'
    'set_secondary_access_acct', 'distinguished_name', 'password'
    'set_failback_interval', time_in_minutes
    'suspend', {'primary' | 'secondary'}
    'activate', {'primary' | 'secondary'}
    'list'
    'list_urls'
    'list_access_acct'
    'check_url', 'url'’
    'reinit_descriptors'
    'check_login', 'name'
    'set_timeout', timeout_in_milli_seconds
    'set_log_interval', log_interval_in_minutes
    'set_num_retries', num_retries
    'set_max_ldapua_native_threads', max_ldapua_native_threads
    'set_max_ldapua_desc', max_ldapua_desc
    'set_abandon_ldapua_when_full', {true|false}
    'starttls_on_primary', {true|false}
    'starttls_on_secondary', {true|false}
    'help’
```

Parameters

- **set_primary_url, '*ldapurl*' –** creates the specified search string *ldapurl*. Exactly one primary search string can be created.

  The syntax for *ldapurl* is:

  ```
  ldapurl::=ldap://host:port/node?attributes?base | one | sub?
  filter
  ```

  where:

  - *host* – is the host name of the LDAP server.
  - *port* – is the port number of the LDAP server.
  - *node* – specifies the node in the object hierarchy at which to start the search.
  - *attributes* – is a list of attributes to return in the result set. Each LDAP server may support a different list of attributes.
  - **base** – qualifies the search criteria, specifiying a search of the base node.
  - **one** – qualifies the search criteria. base specifies a search of the base node; one specifies a search of node and one sublevel below node; and sub specifies a search of node and all node sublevels.
  - **sub** – specifies a search of node and all node sublevels.
  - *filter* – specifies the attribute or attributes to be authenticated. The filter can be simple, such as "uid=*," or compound, such as "(uid=*)(ou=*group*)." The syntax is LDAP server dependent and uses a wildcard (*) to describe the login name.
- **set_secondary_url, { '*ldapurl*' | null } –** creates the specified secondary search string *ldapurl* or no secondary search string. Exactly one secondary search string can be created.
- **set_dn_lookup_url, *distinguished_name_url* –** uses the searched distinguished name algorithm to authenticate the login with an LDAP directory server when you set **set_dn_lookup_url** to a non-NULL value.

  distinguished_name_url has a maximum length of 255 characters and is used to search for a distinguished name associated with the login name.
- **set_secondary_dn_lookup_url, *distinguished_name_url* –** creates the specified secondary distinguished name algorithm to authenticate the login with an LDAP directory server when you set **set_secondary_dn_lookup_url** to a non-NULL value.

  distinguished_name_url has a maximum length of 255 characters and is used to search for a distinguished name associated with the login name.
- **set_access_acct, *account_distinguished_name*, *account_password* –** specifies the identity and password that the SAP ASE server uses to conduct searches and other read-only adminstrative actions. The identity is in the form of a distinguished name. Use *account_distinguished_name* to authenticate this user with the LDAP server. Both *account_distinguished_name* and *account_password* are limited to 255 characters each.
- **set_secondary_access_acct, *account_distinguished_name*, *account_password* –** creates the secondary identity and password that the SAP ASE server uses to conduct searches and other read-only adminstrative actions. The identity is in the form of a

distinguished name. Use *account_distinguished_name* to authenticate this user with the LDAP server. Both *account_distinguished_name* and *account_password* are limited to 255 characters each.

- **set_failback_interval –** sets the interval at which the SAP ASE housekeeper utility checks for failed LDAP servers.
- **suspend, {'primary' | 'secondary'} –** suspends the use of a primary or secondary URL for authentication.
- **activate, {'primary' | 'secondary'} –** enables using the set of primary or secondary URLs for authentication.
- **list –** displays LDAP search strings.
- **list_urls –** displays LDAP URL search strings.
- **list_urls –** displays LDAP URL search strings.
- **list_access_acct –** displays the LDAP access account distinguished name set.
- **check_url, *'ldapurl'* –** verifies an LDAP URL search string. Can also verify the existence of a user account, but it does not authenticate the user.
- **check_login, *login_name* –** verifies a user account for the existing LDAP URL search strings. It does not authenticate the user.
- **'set_timeout' *timeout_in_milli_seconds* –** sets the time in milliseconds that the SAP ASE server waits for a response from the LDAP server before abandoning the authentication request.

  The default value for **set_timeout** is 10,000 milliseconds (10 seconds.) Valid values are between 1 and 3,600,000 (one hour.)
- **'set_log_interval', *log_interval* –** sets the log interval, specified in minutes, from 0 to 480 minutes. The default value is 3 minutes. 0 implies that all messages are printed.
- **set_num_retries, *num_retries* –** sets the number of retries attempted after transient errors. The valid range for **set_num_retries** is 1 – 60, and the default is 3.
- **'set_max_ldapua_naptive_threads, *max_ldapua_native_threads* –** sets the maximum number of native threads that can be running concurrently in an engine processing an LDAP authentication request.

  The minimum value of **set_max_ldapua_native_threads** is 1. The maximum value is **max native threads** minus **number of dump threads** as specified using **sp_configure**. The default value is the same as the maximum value.

  sp_configure ensures that **max native threads** is sufficient for **set_max_ldapua_native_threads** and the value of the configuration parameter **number of dump threads**.
- **set_max_ldapua_desc, *max_ldapua_desc* –** sets the maximum number of LDAP descriptors per engine. The valid range for **set_max_ldapua_desc** is 1 – 20, and the default is 20.
- **set_abandon_ldapua_when_full', {true | false} –** allows you to seek alternative means of LDAP user authentication when the native threads per engine capacity is exceeded.

  When no more threads are available, the request is abandoned if **set_abandon_ldapua_when_full** is set to true. If **enable ldap user auth** is set to 1, the client is authenticated using SAP ASE `syslogins`. If **enable ldap user auth** is set to 2, the client login fails.

If **set_abandon_ldapua_when_full** is set to false, the authentication request is blocked until the LDAP descriptor can accept new authentication requests.

- **help –** displays usage information for **sp_ldapadmin**.
- **reinit_descriptors –** Unbinds all established LDAP server descriptors, and reinitializes the LDAP user-authentication subsystem. The syntax is:

  ```
  sp_ldapadmin 'reinit_descriptors'
  ```

  Whenever a certification authority trusted root file is modified, the system security officer must use **reinit descriptors** to reinitialize LDAP user authentication. For complete documentation, see **sp_ldapadmin** in the *Reference Manual: Procedures*.
- **set_log_interval, *log_interval* –** sets the time for the error message logging interval, in minutes. The valid range for **set_log_interval** is 0 – 480, and the default is 3.

Examples

- **Example 1 –** Creates an LDAP URL search string for the LDAP SunONE Directory Server.

  ```
  sp_ldapadmin set_primary_url,'ldap://voyager:389/
      ou=People,dc=MyCompany,dc=com??sub?uid=*'
  ```

  The search string identifies a directory server listening on host name “voyager,” port number 389 (the default LDAP protocol port), the base node to begin the search is within organizational unit (ou) “People,” and the domain is “MyCompany.com.” It returns all attributes that match the filter uid=*. The SAP ASE server replaces the wildcard with the SAP ASE login name that is to be authenticated.
- **Example 2 –** Creates an LDAP URL search string defined in OpenLDAP 2.0.25 using the criteria described in Example 1.

  ```
  sp_ldapadmin set_primary_url,'ldap://voyager:389/
      dc=MyCompany,dc=com??sub?cn=*'
  ```
- **Example 3 –** Sets the secondary LDAP URL search string to null, indicating no failover and no secondary LDAP server.

  ```
  sp_ldapadmin set_secondary_url, null
  ```
- **Example 4 –** Creates an LDAP URL search string with a compound filrer.

  ```
  sp_ldapadmin set_primary_url, 'ldap://voyager:389/
      ou=people,dc=siroe,dc=com??sub?(&(uid=*) (ou=accounting))
  ```
- **Example 5 –** Uses the default Microsoft Active Directory schema found on Windows 2000 controllers:

  ```
  1> sp_ldapadmin set_access_acct, 'cn=aseadmin, cn=Users,
  dc=mycompany,
        dc=com', 'aseadmin secret password'
  2> go
  ```

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

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

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

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

- **Example 6 –** Sets the maximum number of native threads to 12:

  ```
  sp_ldapadmin 'set_max_ldapua_native_threads', '12'
  ```

- **Example 7 –** sets the time that the SAP ASE server waits for a response from the LDAP server before abandoning the authentication request to 25,000 milliseconds:

  ```
  sp_ldapadmin, 'set_timeout', '25000'
  ```

- **Example 8 –** Disables the authentications requests until the LDAP descriptor can accept new authentication requests:

  ```
  sp_ldapadmin 'set_abandon_ldapua_when_full', 'false'
  ```

- **Example 9 –** Displays the current LDAP values:

  ```
  sp_ldapadminPrimary:
  URL:                   'ldap://linuxpuneeng1:50917/'
  DN Lookup URL:
  'ldap://linuxpuneeng1:50917/dc=sybase,dc=com??sub?uid=*'
  Access Account:       'cn=Directory Manager'
  Active:               'TRUE'
  Status:               'READY'
  Secondary:
  URL:                  ''
  DN Lookup URL:        ''
  Access Account:       ''
  Active:               'FALSE'
  Status:               'NOT SET'
  Timeout value:        '5000' milliseconds
  Log interval:         '1' minutes
  Number of retries:    '3'
  Maximum LDAPUA native threads per Engine: '400'
  Maximum LDAPUA descriptors per Engine: '3'
  ```

```
Abandon LDAP user authentication when full: 'false'(return status
= 0)
```

Usage

There are additional considerations when using **sp_ldapadmin**:

- The LDAP vendor determines the syntax of the search string. In all cases, the search string specifies the attribute name that uniquely identifies the user in the form "*`attribute=wildcard`*" as in "`cn=*`".
- The first attribute in a compound filter must define the Relative Distinguished Name (RDN). For example, "`...sub?(uid=*)(ou=group)`". Otherwise, the authentication fails.
- When a search string is added, the SAP ASE server verifies that it uses valid LDAP URL syntax and that it references an existing node. To ensure that the valid string returns expected values, carefully choose and verify the search string when configuring the SAP ASE server.
- The secondary URL search string enables failover to another LDAP server. The SAP ASE server uses the primary URL search string unless the LDAP Server is not active or the search string is invalid. In this event, the SAP ASE server uses the secondary URL search string for authentication.
- The login sequence of searched DN algorithm requires the SAP ASE server to bind to the LDAP server using the access account before it can perform searches. The SAP ASE server obtains an LDAP descriptor (handle) as a result of the bind. This descriptor is used for searching the DN of the login on the LDAP server.
- In order to access the server, users who are being authenticated with the LDAP server should either exist as a valid user in SAP ASE, or have a mapping defined.

See *Creating and Managing ASE Logins Using LDAP* in the *System Administration Guide* and **sp_maplogin**.

Permissions

The permission checks for **sp_ldapadmin** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage security configuration` privilege.
Disabled	With granular permissions disabled, you must be a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_maplogin* on page 503

sp_listener

Dynamically starts and stops SAP ASE listeners on any given port on a per-server basis.

Considerations for process mode

When executed in process mode, **sp_listener** dynamically starts and stops SAP ASE listeners on any given port on a per-engine basis.

Syntax

For threaded mode, the syntax is:

```
sp_listener “command”, “server_name | network”
```

Or:

```
sp_listener “command”, ‘[protocol:]machine:port:”CN=common_name”’
```

For process mode, the syntax is:

```
sp_listener “command”, “server_name | network”, engine | remaining
```

Or:

```
sp_listener “command”, ‘[protocol:]machine:port:”CN=common_name”’,
engine
```

Parameters

- ***command* –** can be any of:

- **start** – starts a listener on the specified ports on each of the specified servers.
- **stop** – terminates the specified listeners.
- **suspend** – prevents the listener from accepting any more connections.
- **resume** – instructs suspended listeners to resume listening.
- **status** – report on the state of the listeners specified by the parameters. The state is one of: active, stopped, or suspended.

  If your system is enabled for IPV6, the SAP ASE server encloses the listener name in brackets in the output.
- **help** – displays the **sp_listener** syntax.

- ***server_name* | *network* –** is the name of the SAP ASE server, as specified in the interfaces file, or the name of the network.
- ***engine* –** (Used only in process mode) specifies the number of the engine affected by this command. *engine* can be a single-engine number in quotes (“2”), a list (“3,5,6”), a range (“2 – 5”), or mix of all (“2,3 – 5,7”).

Note: Windows ignores the *engine* parameter.

- **remaining –** specifies that the command is to take effect on all engines on which it can be meaningfully applied (that is, where the listener is in a state in which the command is can take effect).
- ***protocol* –** the type of protocol; one of: afunix, tcp, tli, ssltcp, ssltli, winsock, sslnlwnsck, sslwinsock.
- ***machine:port* –** the machine name and port number (as specified in the interfaces file) to which the listener connects.
- **CN=*common_name* –** specifies a common name for the SSL certificate.

  Use **CN=common_name** only if you specify ssltcp as the protocol. The SAP ASE server validates the common_name you specify against the common_name in the SSL certificate. If you do not include **CN=*common_name***, the SAP ASE server uses *server_name* to validate against the common name in the SSL certificate. If you include a fully qualified domain name in the certificate, it must match **CN=common_name**.

Examples

- **Example 1 –** Start listeners for each master entry in the interfaces file corresponding to server orion:

```
sp_listener "start", "orion"
```

- **Example 2 –** Create TCP listeners for port number 4226:

```
sp_listener "start", "goldie:4226"
```

- **Example 3 –** Create listeners for all master entries in the interfaces file for server orion:

```
sp_listener "start", "orion", "remaining"
```

- **Example 4 –** Start TCP listeners on port 4226 on machine goldie for all engines not already listening to this port:

```
sp_listener "start", "goldie:4226", "remaining"
```

- **Example 5 –** Specify the common name `ase1.big server 1.com`:

  ```
  sp_listener 'start','ssltcp:blade1:17251:
  "CN=ase1.big server 1.com"','0'
  ```

- **Example 6 –** Stop the listener on port number 4226:

  ```
  sp_listener "stop", "tcp:goldie:4226"
  ```

- **Example 7 –** Stop all listeners on port number 4226. Because this command includes the **remaining** parameter, it does not fail if some engines are not listening to the port:

  ```
  sp_listener "stop", "tcp:goldie:4226", "remaining"
  ```

- **Example 8 –** Suspend Winsock listener on port 4226:

  ```
  sp_listener "suspend", "winsock:clouds:4226"
  ```

- **Example 9 –** Resume all active listeners on port number 4226:

  ```
  sp_listener "resume", "tcp:goldie:4226", "remaining"
  ```

- **Example 10 –** Start a named pipe listener using AF_UNIX communication.

  ```
  sp_listener "start", "afunix:big_server:/tmp/big_pipe"
  ```

Usage

There are additional considerations when using **sp_listener**:

- **sp_listener** uses either of two syntaxes, described in the syntax section, above. The first syntax affects all SAP ASE master ports listed in the interfaces file. The second allows you to manage listeners not listed in the interfaces file.
- The attribute name “CN” is case-insensitive (it can be “CN”, “cn” or “Cn”), but the attribute value for the common name is case-sensitive.
- **sp_listener** ignores the *engine* parameter if you include it while running in threaded mode.
- The semantics for **sp_listener** is atomic; if a command cannot be completed successfully, it is aborted.
- When the host component of a **sp_listener** command is an IPv6 address, it should be encloded in brackets. For example, `tcp:[2001:ec8:4008:1::123]:80`
- You can issue the **status** parameter by itself. The **status** parameter displays the state of all the listeners in the interfaces file.
- A listener can be in one of the following states: stopped, suspended, or active. **sp_listener** allows you to move listeners between these states. A request to move to a nonpermissible state results in failure (For example, requesting to stop a non existent listener). Use **sp_listener “status”** to determine the state of a listener.
- The **remaining** parameter specifies that, for the command you are running (**start**, **stop**, **resume**, and so on), the command runs successfully for all listeners that are in a state that allow the change (for example, moving states from start to stop). For example, if you attempt to start listeners on engines one through six, but engines one, four, and five are unavailable, **sp_listener... “remaining”** starts listeners on engines two, three, and six,

disregarding the offline engines. You cannot specify an engine number if you include the **remaining** parameter.
- The maximum number of listeners is 32. If you create an SAP ASE server with two master ports in the interfaces file, you can start at most 30 more listeners on other ports.

For limitations related to IPV6 in **sp_listener**, see *Specifying a Common Name* in the *Security Administration Guide*.

Permissions

The permission checks for **sp_listener** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_listsuspect_db

Lists all databases that currently have offline pages because of corruption detected on recovery, including the database name, number of suspect pages, and number of objects containing suspect pages.

Syntax

```
sp_listsuspect_db
```

Examples

- **Example 1 –** Lists the databases that have suspect pages:

```
sp_listsuspect_db
```

Usage

To identify suspect pages, use **sp_listsuspect_page**

Permissions

Any user can execute **sp_listsuspect_db**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_listsuspect_page* on page 478
- *sp_setsuspect_granularity* on page 640
- *sp_setsuspect_threshold* on page 643

sp_listsuspect_object

Lists all indexes in a database that are currently offline because of corruption detected on recovery.

Syntax

```
sp_listsuspect_object [dbname]
```

Parameters

- ***dbname* –** is the name of the database.

Examples

- **Example 1 –** Lists the suspect indexes in the current database:

  ```
  sp_listsuspect_object
  ```

- **Example 2 –** Lists the suspect indexes in the `pubs2` database:

  ```
  sp_listsuspect_object pubs2
  ```

Usage

There are additional considerations when using **sp_listsuspect_object**:

- If an index on a data-only-locked table has suspect pages, the entire index is taken offline during recovery. Offline indexes are not considered by the query optimizer.
- Use the system procedure **sp_forceonline_object** to bring an offline index online for repair.
- Indexes on allpages-locked tables are not taken completely offline during recovery; only individual pages of these indexes are taken offline. These pages can be brought online with **sp_forceonline_object**.
- **sp_listsuspect_object** lists the database name, object ID, object name, index ID, and access status for every suspect index in the specified database or, if *dbname* is omitted, in the current user database.
- A value of SA_ONLY in the `access` column means that the index has been forced online for system administrator use only. A value of BLOCK_ALL means that the index is offline for everyone.

See the *System Administration Guide* for more information on recovery fault isolation.

Permissions

Any user can execute **sp_listsuspect_object**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_forceonline_object* on page 345

sp_listsuspect_page

Lists all pages in a database that are currently offline because of corruption detected on recovery, including 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.

Syntax

```
sp_listsuspect_page [dbname]
```

Parameters

- ***dbname* –** is the name of the database.

Examples

- **Example 1 –** Lists the suspect pages in the current database:

  ```
  sp_listsuspect_page
  ```

- **Example 2 –** Lists the suspect pages in the `pubs2` database:

  ```
  sp_listsuspect_page pubs2
  ```

Usage

A value of SA_ONLY in 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.

Permissions

Any user can execute **sp_listsuspect_page**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_listsuspect_db* on page 475
- *sp_setsuspect_granularity* on page 640
- *sp_setsuspect_threshold* on page 643

sp_lmconfig

Configures license management-related information on SAP ASE.

Syntax

```
sp_lmconfig
    [
    [ 'edition' [, edition_type ]]
    | [ 'license type' [, license_type_name ]]
    | [ 'smtp host' [, smtp_host_name ]]
    | [ 'smtp port' [, smtp_port_number ]]
    | [ 'email sender' [, sender_email_address ]]
    | [ 'email recipients' [, email_recipients ]]
    | [ 'email severity' [, email_severity ]]
    ]
```

Parameters

- **sp_lmconfig –** without parameters displays the following license status information:
  - Server Name
  - License Name
  - Version
  - Quantity Status
  - Expiration Date
- **edition –** is a static configuration parameter to specify the license edition.
- ***edition_type* –** specifies the edition type, and has the following possible values:
  - **null** – is the default value. When a null value is specified, no product edition is configured, and the SAP ASE server starts with a license for any edition.
  - **EE** – indicates the Enterprise edition.
  - **SE** – indicates the Small Business edition.
  - **DE** – indicates the Developer's edition.
  - **XE** – indicates the Express edition.
- **license type –** is a static configuration parameter that specifies the license type for the installation of SAP ASE, and is valid only when you specify a non-null edition.
- ***license_type_name* –** specifies the license type of a particular installation of SAP ASE. You need not specify **license type** if you are using the Developer's (DE) or Express (XE) edition. The valid, most typical values are:
  - **SRST** – server license with network seats
  - **SVST** – standby server license with network seats
  - **SRCU** – server license with concurrent user seats
  - **SVCU** – standby server license with concurrent user seats
  - **SRIA** – server license with Internet access license
  - **SVIA** – standby server license with Internet access license
  - **CP** – CPU license
  - **SF** – standby CPU license
  - **null** – default

  Note: In addition to this list, **sp_lmconfig** also accepts two-letter abbrevations for specialized and legacy license types. If the license type is not accepted, set the type to null and use the network license server options file to control the license used by this SAP ASE server.

- **smtp host, *smtp host name* –** designates the SMTP host used to send E-mails for license event notifications.
- **smtp port, *smtp port number* –** designates the SMTP port used to send Emails for license event notifications.
- **email sender, *sender email address* –** specifies the E-mail address used as the senders address on license event E-mail notifications.
- **email recipients, *email recipients* –** is a comma separated list of E-mail recipients who receive license event E-mail notifications.
- **email severity, *email severity* –** is the minimum severity of an error that causes an E-mail notification to be sent. The default is error, and the other possibilities are warning and informational.

Examples

- **Example 1 –** Displays basic license configuration information for a system:

```
1> sp_lmconfig
2> go
```

```
 Parameter Name    Config Value
 ----------------- ------------
 edition           EE
 license type      CP
 smtp host         null
 email recipients  null
 email severity    null
 smtp port         null
 email sender      null

 License Name         Version    Quantity Status     Expiry
Date         Server Name
 ---------------      ---------- -------- ---------- ------------
------- ----------
 ASE_HA               2010.03314 2        expirable  Apr 1 2010
12:00AM  cuprum
 ASE_ASM              null       0        not
used   null               null
 ASE_EJB              null       0        not
used   null               null
 ASE_EFTS             null       0        not
used   null               null
 ASE_DIRS             null       0        not
used   null               null
 ASE_XRAY             null       0        not
used   null               null
 ASE_ENCRYPTION       null       0        not
used   null               null
 ASE_CORE             2010.03314 2        expirable  Apr 1 2010
12:00AM  cuprum
 ASE_PARTITIONS       null       0        not
used   null               null
 ASE_RLAC             null       0        not
used   null               null
```

```
 ASE_MESSAGING_TIBJMS null       0        not
used   null              null
 ASE_MESSAGING_IBMMQ  null       0        not
used   null              null
 ASE_MESSAGING_EASJMS null       0        not
used   null              null

 Property Name Property Value ------------- -------------
 PE            EE
 LT            CP
 ME            null
 MC            null
 MS            null
 MM            null
 CP            0
 AS            A

(return status = 0)
```

Usage

There are additional considerations when using **sp_lmconfig**:

- When you do not specify any parameters, **sp_lmconfig** also displays the server name from the location where the license is checked out.
- If you do not specify an edition or use "null," the SAP ASE server looks for and uses whatever license edition it finds when it starts.
- The configuration options set by **sp_lmconfig** are stored in the `sylapi` properties file.

See also:

- The SAP ASE installation guide for your platform.

Permissions

The permission checks for **sp_lmconfig** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server configuration` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_lock

Reports the object names and IDs of processes that currently hold locks.

Syntax

```
sp_lock [spid1[, spid2]] | [@verbose = int]
```

Parameters

- ***spid1* –** is the SAP ASE process ID number from the `master.dbo.sysprocesses` table. Run **sp_who** to get the *spid* of the locking process.
- ***spid2* –** is another SAP ASE process ID number to check for locks.
- **@verbose = *int* –** displays a concatenated name of the table names instead of a *spid*, such as `test..testa`, following by the *spid*.

Note: *int* can be any number, as **sp_lock** only check to see whether the value of **@verbose** is null or not.

Examples

- **Example 1 –** Shows the lock status of serial processes with `spid`s 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 `spid`s 8, 9, and 10. The family with `fid` 11 has the coordinating processes with `spid` 11 and worker processes with `spid`s 12, 13, and 14:

```
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
```

- **Example 2 –** Displays information about the locks currently held by `spid 7`.

```
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
```

- **Example 3 –** First, queries the `pubs2` database about the ID of its running processes that currently hold locks (1056003762), then queries the `pubs2` database using the **@verbose** option, which returns the object name (`master..spt_values`) in addition to the process ID:

```
1> use pubs2
2> go
1> sp_lock
2> go
```

```
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 loid locktype  table_id   page row dbname class context
--- ---- ---- --------  --------   ---- --- ------ ------
----------
0     15   30 Sh_intent 1056003762    0   0 master Non Cursor Lock

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

```
1> sp_lock @verbose=0
2> go
```

```
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 loid locktype page row objectName         id       class
  context
--- ---- ---- -------- ---- --- ------------------ -------- ------
-------
0     15   30 Sh_intent   0   0  master..spt_values 1056003762 Non
Cursor Lock

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

- **Example 4 –** This example shows all locks, including partition locks, currently held by SAP ASE.

```
sp_lock
go

fid spid loid locktype           table_id    partitionid page row
dbname class context
--- ---- ---- --------           ----------  ----------- ---- ---
------ ------ -------
0     13   26 Ex_intent           420193516            0    0   0
master Non Cursor Lock
0     13   26 Ex_intent_partition  420193516    452193630    0   0
master Non Cursor Lock
0     13   26 Ex_page             420193516    452193630 4993   0
master Non Cursor Lock
0     14   28 Ex_intent           420193516            0    0   0
master Non Cursor Lock
0     14   28 Ex_intent_partition  420193516    468193687    0   0
master Non Cursor Lock
0     14   28 Ex_page             420193516    468193687 5001   0
master Non Cursor Lock
0     16   32 Sh_intent          1006623598            0    0   0
master Non Cursor Lock
```

Table lock and fine-grained lock values for `partitionid` are 0. `partitionid` is populated only for partition-level locks.

Usage

There are additional considerations when using **sp_lock**:

- **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** in versions of the Cluster Edition earlier than 15.0.3 displayed information about only the locks associated with the instance on which you issued the stored procedure. **sp_lock** on Cluster Edition version 15.0.3 and later displays information about all locks in the cluster.
- **sp_lock** output is ordered by `fid` and then `spid`.
- **sp_lock** output also displays the following lock types:
  - "Sh_row" indicates shared row locks
  - "Update_row" indicates update row locks
  - "Ex_row" indicates exclusive row locks

The **sp_lock** columns are:

- **`loid`** – The column identifies unique lock owner ID of the blocking transaction. Even `loid` values indicate that a local transaction owns the lock. Odd values indicate that an external transaction owns the lock.
- **`locktype`** – The 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. See the *Performance and Tuning Guide* for more information about lock types.
- **`class`** – The 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 SAP ASE 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**.
- **`fid`** – The column identifies the family (including the coordinating process and its worker processes) to which a lock belongs. Values for `fid` are:
  - 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.

- **`context`** **–** The 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" – 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" – the task holding the lock holds 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.
    - "Ind pg" – indicates locks on index pages (allpages-locked tables only)
    - "Inf key" – indicates an infinity key lock (for certain range queries at transaction isolation level 3 on data-only-locked tables)
    - "Range" – indicates a range lock (for range queries at transaction isolation level 3 on data-only-locked tables)

  These new values may appear in combination with "Fam dur" (which replaces "Sync pt duration") and with each other, as applicable.
- **`row`** **–** The column displays the row number for row-level locks.

See also **kill**, **select** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_lock**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_familylock* on page 335
- *sp_who* on page 736

sp_locklogin

Locks an SAP ASE account so that the user cannot log in, or displays a list of all locked accounts.

Syntax

```
sp_locklogin login | NULL | wildcard_string , "lock" | "unlock",
    [except_login_name | except_role_name]
    [, number_of_inactive_days]
```

Or:

```
sp_locklogin
```

Parameters

- **sp_locklogin –** without any parameters, displays all locked logins.
- ***loginame* –** is the name of the account to be locked or unlocked.
- ***wildcard_string* –** is any string with wildcards that identifies a set of logins.
- **NULL –** all logins, including the sa_role, are locked.
- **lock | unlock –** specifies whether to lock or unlock the account.
- ***except_login_name* –** is the name of login that is exempted from being locked.
- ***except_role_name* –** is the name of role that is exempted from being locked. For example, all logins in a role that are to be exempted.
- ***number_of_inactive_days* –** is the number of days, from 1 to 32,767, that an account has been inactive.

Examples

- **Example 1 –** Locks the login account for the user “charles”:

  ```
  sp_locklogin charles, "lock"
  ```

- **Example 2 –** Locks all logins except those with the sa_role:

  ```
  sp_locklogin NULL, "lock", sa_role
  ```

- **Example 3 –** Displays a list of all locked accounts:

  ```
  sp_locklogin
  ```

- **Example 4 –** Locks all login accounts that have not authenticated within the past 60 days:

  ```
  sp_locklogin NULL, 'lock', NULL, 60
  ```

  Note: This command has no effect if the **sp_passwordpolicy** option “**enable last login updates**” is set to “0”.

Usage

There are additional considerations when using **sp_locklogin**:

- Without any parameters, **sp_locklogin** displays all locked logins.
- The `syslogins` columns `lockdate`, `locksuid` and `lockreason` are updated at time of locking/unlocking a login.
- Conditions for using **sp_locklogin** are:
  - No wild cards are allowed for exceptions.
  - Existing functionality is undisturbed.
  - The exception specified is first matched against logins. If such a login does not exist, then the exception is checked against roles.
  - A value of NULL for a login means “all” logins.
  - You see an error if the login name or exception you specify does not exist.
  - Nothing happens if the specified “effective set” of logins to be locked is empty.
  - If the exception is NULL, the set of logins specified (through the **login** parameter) is locked.
  - High-availability Failover only – in versions of SAP ASE earlier than 15.0, **sp_locklogin** checked to see if the login to be locked or unlocked existed on a remote high-availability server by verifying that the the `suid` (server user ID) of that login existed on the server.

    In SAP ASE version 15.0, **sp_locklogin** checks both the `suid` as well as the login name.
  - You see an error if you specify any word other than **lock** or **unlock**.

See also **create login**, **alter login** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_locklogin** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any login` privilege. To unlock login account which was locked because of **maxfailedlogin**, you must be a user with `change password` privilege.
Disabled	With granular permissions disabled, you must be a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_logdevice

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 with the `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

- **Example 1 –** Creates the database `products` and puts the table `products.syslogs` on the database device `logs`:

  ```
  create database products on default = "10M", logs = "2M"
  go
  sp_logdevice products, logs
  go
  ```

- **Example 2 –** For the database `test` with log and data on the same device, places the log for `test` on the log device `logdev`:

  ```
  alter database test log on logdev
  go
  sp_logdevice test, logdev
  go
  ```

Usage

There are additional considerations when using **sp_logdevice**:

- You can only execute **sp_logdevice** in single-user mode.
- 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 this to increase the size of the log segment. If you did not use **log on**, execute **sp_logdevice**:

```
sp_extendsegment segname, devname
```

The device or segment on which you put `syslogs` is used *only* for `syslogs`. 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 **alter database** .

- Use **disk init** to format a new database device for databases or transaction logs.

See also:

- *System Administration Guide*
- **alter database**, **create database**, **dbcc**, **disk init**, **dump database**, **dump transaction**, **select** in *Reference Manual: Commands*

Permissions

The permission checks for **sp_logdevice** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner or a user with `own database` privilege.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_extendsegment* on page 330
- *sp_helpdevice* on page 402
- *sp_helplog* on page 420

sp_logging_rate

Calculates the transaction log growth rate for the specified time period

Syntax

```
sp_logging_rate {'full'|'sum', '[day,]hh:mm:ss'}[,
interval='hh:mm:ss' | clear_option='y'|'n']
```

Parameters

- **full – sp_logging_rate** provides a detailed report for each collection.
- **sum – sp_logging_rate** provides summary information, including values for the average, minimum, maximum, and the maximum rate. If you do not specify a time, **sp_logging_rate** collects information every 10 seconds.
- **day, *hh:mm:ss* –** Specifies the duration of time **sp_logging_rate** runs, using the form *date, hour:minute:second*.
- **interval = '*hh:mm:ss*' –** Period of time during which the interval runs, using the form *hour:minute:second*
- **clear_option = 'y' | 'n'' –** Determines whether to clear the monitor counters during data collection.

Examples

- **Example using sum parameter – sp_logging_rate** collects information for 1 day and 8 hours, takes a sample every 10 minutes, and prints summary information at the end of the interval:

  ```
  sp_logging_rate 'sum', '1,08:00:00', '00:10:00'
  =========================
  Total Summary Information
  =========================
  Transaction Log Growth Rate      Min GB/h        Max GB/h        Avg
  GB/h
  ---------------------------   --------------  --------------
  --------------
                                  0.000000        2.870076
  1.823028
  ```

- **Example using full parameter – sp_logging_rate** collects information for 3 minutes, takes samples every 10 seconds (the default), and prints summary information at the end of the interval:

```
sp_logging_rate 'full', '00:03:00'
Date Time                           Transaction Log Growth Rate GB/h
---------------------------         --------------------------------
Oct 22 2013  6:00:32:480AM                    0.406779


Oct 22 2013  6:00:42:483AM                    0.000000


Oct 22 2013  6:00:52:483AM                    0.000000


Oct 22 2013  6:01:02:483AM                    0.000000


Oct 22 2013  6:01:12:490AM                    0.000000


Oct 22 2013  6:01:22:500AM                    0.000000


Oct 22 2013  6:01:32:476AM                    2.341870


Oct 22 2013  6:01:42:483AM                    2.828132


Oct 22 2013  6:01:52:480AM                    2.850305


Oct 22 2013  6:02:02:483AM                    2.782750


Oct 22 2013  6:02:12:483AM                    2.853574


Oct 22 2013  6:02:22:480AM                    2.002917


Oct 22 2013  6:02:32:483AM                    2.848995


Oct 22 2013  6:02:42:483AM                    2.754143


Oct 22 2013  6:02:52:483AM                    2.854949


Oct 22 2013  6:03:02:480AM                    2.722928


Oct 22 2013  6:03:12:476AM                    2.870076


Oct 22 2013  6:03:22:480AM                    2.697094
```

```

=========================
Total Summary Information
=========================
Transaction Log Growth Rate     Min GB/h       Max GB/h       Avg 
GB/h
---------------------------   --------------  --------------  
--------------
                                 0.000000       2.870076       
1.823028
```

Usage

- You cannot run scripts or procedures that collect monitoring data (for example, sp_sysmon) while **sp_logging_rate** runs. Because **sp_logging_rate** collects and clears monitor counter as it runs, the monitoring counter information these scripts or procedures collect will not be accurate.
- **sp_logging_rate** produces unreliable results if you specify an amount of time for **interval =** '*hh:mm:ss*' that is greater than the amount of time you specify for **'day,** *hh:mm:ss*'.
- When you specify values for **interval =** '*hh:mm:ss'* and **'day,** *hh:mm:ss'*, keep in mind:
  - If the value you specify for **interval =** '*hh:mm:ss*' is greater than the value you specify for **'day,** *hh:mm:ss'*, SAP ASE issues an error message and **sp_logging_rate** produces no result set.
  - **sp_logging_rate** may produce an unreliable result if that ratio for **'day,** *hh:mm:ss'* to **interval =** '*hh:mm:ss'* is too small. For example, if you specify `day,00:10:00`, and `interval='00:04:00'`, **sp_logging_rate** collects only two values, prints an average value, with the first value as the maximum, and the second value as the minimum. A better ratio produces a more reliable result set.

Permissions

You must have system administrator privileges to execute **sp_logging_rate**.

sp_loginconfig

(Windows only) 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 @**
- **key #**

Examples

- **Example 1 –** Displays the values of all integrated security parameters:

```
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 #                -
```

- **Example 2 –** Displays the value of the **login mode** security parameter:

```
sp_loginconfig "login mode"
```

```
name                 config_item
-------------------- --------------------
login mode           standard
```

Usage

There are additional considerations when using **sp_loginconfig**:

- The values of integrated security parameters are stored in the Windows Registry. See the chapter on login security in *Configuration Guide for Windows* for instructions on changing the parameters.
- **sp_loginconfig** displays the *config_item* values that were in effect when you started the SAP ASE server. If you changed the Registry values after starting the SAP ASE server, those values are not reflected in the **sp_loginconfig** output.

See also *Configuration Guide for Windows*.

Permissions

The permission checks for **sp_loginconfig** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any login` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_revokelogin* on page 616

sp_logininfo

(Windows only) Displays all roles granted to Windows users and groups with **sp_grantlogin**.

Syntax

```
sp_logininfo ["login_name" | "group_name"]
```

Parameters

- ***login_name* –** is the network login name of the Windows user.
- ***group_name* –** is the Windows group name.

Examples

- **Example 1 –** Displays the permissions granted to the Windows user “regularjoe”:

  ```
  sp_logininfo regularjoe
  ```

  ```
  account name    mapped login name  type           privilege
  ---------------
  ------------------ --------------- --------------
  HAZE\regularjoe HAZE_regularjoe    user           'oper_role'
  ```

- **Example 2 –** Displays all permissions that were granted to Windows users and groups with **sp_grantlogin**:

  ```
  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'
  ```

Usage

There are additional considerations when using **sp_logininfo**:

- **sp_logininfo** displays all roles granted to Windows users and groups with **sp_grantlogin**.
- You can omit the domain name and domain separator (\) when specifying the Windows user name or group name.

See also **grant**, **setuser** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_logininfo** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage roles` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_displaylogin* on page 234
- *sp_grantlogin* on page 354
- *sp_revokelogin* on page 616
- *sp_role* on page 617
- *sp_who* on page 736

sp_logiosize

Changes the log I/O size used by the SAP ASE 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 the SAP ASE server's default value (two logical pages), if a memory pool that is two logical pages is available in the cache. Otherwise, the SAP ASE server sets the log I/O size to one logical page. 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 multiples of the logical page size, up to four times the amount. You must enclose the value in quotes.
- **all –** displays the log I/O size configured for all databases grouped by the cache name.

Examples

- **Example 1 –** Displays the log I/O size configured for the current database:

  ```
  sp_logiosize
  ```

  ```
  The transaction log for database 'master' will use I/O size of 2
  Kbytes.
  ```

- **Example 2 –** 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, the SAP ASE server returns an error message indicating that such a pool does not exist, and the current log I/O size does not change:

  ```
  sp_logiosize "8"
  ```

- **Example 3 –** Changes the log I/O size of the current database to the SAP ASE server's default value (one logical page size). If a memory pool the size of the logical page size does not exist in the cache used by the transaction log, the SAP ASE server uses the 2K memory pool:

  ```
  sp_logiosize "default"
  ```

- **Example 4 –** Displays the log I/O size configured for all databases:

  ```
  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
  ```

Usage

There are additional considerations when using **sp_logiosize**:

- **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, the SAP ASE server displays the log I/O size of the current database.
- When you change the log I/O size, it takes effect immediately. The SAP ASE 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.

The SAP ASE server defines the default log I/O size of a database as two logical pages, if a memory pool the size of two logical pages is available in the cache. Otherwise, the SAP ASE server sets the log I/O size to one logical page (a memory pool of one logical page is always present in any cache). For most work loads, a log I/O size of two logical pages performs much better than one of one logical page, so each cache used by a transaction log should have a memory pool the size of a logical page. See the *System Administration Guide* and the *Performance and Tuning Guide* for more information about configuring caches and memory pools.

- 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 is not used.

  For example, on a 2K server, 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 logical page size memory pool of the default cache is active, regardless of the log I/O size configured for a database. Transactions logs are read into this pool of the default cache, and all transactions that must be rolled back, or rolled forward, read data pages into the default data cache.

Permissions

Any user can execute **sp_logiosize** to display the log I/O size values.

The following permission checks for **sp_logiosize** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage data cache` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_cacheconfig* on page 90
- *sp_poolconfig* on page 582

sp_logintrigger

Sets and displays the global login trigger. This global login trigger has the same characteristics as a personal login script. It is executed before any personal login script for every user that tries to log in, including system administrators and security officers.

Syntax

```
sp_logintrigger 'global login trigger name'
```

Parameters

- ***global login trigger name* –** is the name of the global login trigger.

  If you include no parameter, **sp_logintrigger** displays the current login trigger status and name if it exists, and no rows if there is no global login trigger defined.

Examples

- **Example 1 –** Sets a global login trigger using **sp_logintrigger**:

  ```
  sp_logintrigger 'master.dbo.myproc'
  ```

- **Example 2 –** Returns an updated global login trigger:

  ```
  1> sp_logintrigger
  2> go
  Global login trigger               Status
  ---------------------------------- -------
  sybsystemprocs.dbo.myproc          Enabled
  (1 row affected)
  (return status = 0
  ```

- **Example 3 –** When a global login trigger does not exist:

```
1> sp_logintrigger
2> go
Global login trigger Status
-------------------- ------
(0 rows affected)
```

- **Example 4 –** Deletes a global login trigger specified earlier with **sp_logintrigger**:

  ```
  sp_logintrigger 'drop'
  ```

Usage

To find out if a global login trigger is defined and enabled, use the **@@logintrigger** global variable.

There is a difference between this global login and the private login script. This global login trigger is stored by name in `sysattributes`, while the private login script is stored only by object ID.

Permissions

The permission checks for **sp_logintrigger** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage security configuration` privilege to set a new login trigger. Any user can execute **sp_logintrigger** to display the current global login trigger.
Disabled	With granular permissions disabled, you must be a user with **sso_role** to set a new login trigger. Any user can execute **sp_logintrigger** to display the current global login trigger.

sp_maplogin

Maps external users to SAP ASE logins.

Syntax

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

Parameters

- ***authentication_mech* –** specifies the mechanism used for authenticating the login account.

- ***client_username* –** is an external user name. This user name can be an operating system name, a user name for an LDAP server, or anything else that the PAM library can understand. A null value indicates that any login name is valid.
- ***action* –** indicates **create login** or **drop**. When **create login** is used, the login is created as soon as the login is authenticated. **drop** is used to remove logins.
- ***login_name* –** is an SAP ASE login that already exists in `syslogins`

Examples

- **Example 1 –** Maps external user "jsmith" to SAP ASE user "guest". Once authenticated, "jsmith" gets the privileges of "guest". The audit login record shows both the *client_username* and the SAP ASE user name:

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

- **Example 2 –** Tells the SAP ASE server to create a new login for all external users authenticated with PAM, in case a login does not already exist:

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

Usage

Use **sp_maplogin** to map an external name or client name, such as "ase.open.user," defined in an LDAP directory to the SAP ASE login name of "aseopenuser." That is, the *client_username* follows the rules of a name in an LDAP server, and the *login_name* follows the SAP ASE rules for identifiers.

If you are using LDAP User Authentication and the name in the LDAP server differs from the SAP ASE login name, use **sp_maplogin** so the LDAP server uses the *client_username* for authentication, and the SAP ASE *login_name* for identity within the SAP ASE server. That is, "`isql -U` *`client_username`*`...`" has the identity of *login_name* within the SAP ASE server.

Use **sp_helpmaplogin** to determine the *client_username* and *login_name*, such as:

```
1> sp_helpmaplogin
2> go
```

```
authentication   client name     login name
----------------------------------------------
LDAP              ase.open.user   aseopenuser
```

```
C:\> isql -Uase.open.user -Pasepass
1> select @@authmech
2> go
```

```
------------------
ldap
```

Permissions

The permission checks for **sp_maplogin** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any login` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_helpmaplogin* on page 421

sp_merge_dup_inline_default

Removes existing duplicate inline default objects, converting the unique inline defaults to sharable inline default objects.

Syntax

```
sp_merge_dup_inline_default [@report_only = {yes | no}
    [, @show_progress = {yes | no}]]
```

Parameters

- **@report_only –** reports the number of unique inline defaults in the current database but performs no changes if you specify **yes**. If you specify **no**:

    - **sp_merge_dup_inline_default** removes duplicate inline defaults, and all unique inline defaults are changed to sharable inline defaults
    - Existing column definitions referencing the duplicate inline defaults are updated to reference the sharable inline defaults

  The default value for **@report_only** is **yes.**
- **@show_progress –** if set to **yes**, **sp_merge_dup_inline_default** displays hashmarks to show progress when **@report_only** is set to **no**.

  The default value for **@show_progress** is **no.**

Examples

- **Example 1 –** Runs **sp_merge_dup_inline_default** against the `pubs2` database without any options. **sp_merge_dup_inline_default** makes no changes, but displays an informational message indicating the approximate number of unique inline defaults:

```
sp_merge_dup_inline_default
```

```
=====================================================
sp_merge_dup_inline_default is used to identify duplicate inline
default objects,
subsequently to convert one of them into sharable inline default
object and remove the
rest. As the result, it will remove entries from sysobjects,
syscomments and
sysprocedures. It will also update entries in syscolumns,
syscomments and sysprocedures.

Following is the current state of your inline default objects
found out by
sp_merge_dup_inline_default and what it could potentially do to
them. By default,
sp_merge_dup_inline_default only reports the current state and
this warning message. If
you really intend to carry out the changes, please rerun this
stored procedure using

sp_merge_dup_inline_default @report_only = "NO"

Database pubs2 has about 0 unique inline defaults If you convert
them into sharable inline
defaults, the rest of total 0 duplicate defaults can be removed
from the system catalogs.
=====================================================
```

- **Example 2 –** Converts the unique inline default to shareable inline defaults:

```
sp_merge_dup_inline_default @report_only = 'NO'
```

```
Total 2 duplicate defaults are removed and 7 defaults are
converted to sharable inline
defaults. Database is modified and in single-user mode. System
```

```
Administrator (SA) must
reset it to multi-user mode with sp_dboption.
```

- **Example 3 –** Produces the following output if there are no duplicate inline defaults:

  ```
  sp_merge_dup_inline_default @report_only = 'NO'
  ```

  ```
  Database is not modified. Please try it later if duplicate inline
  defaults do exist and
  the current resource limitation is preventing this conversion
  process.
  ```

- **Example 4 –** Includes the **show_progress** parameter to indicate progress:

  ```
  sp_merge_dup_inline_default @report_only = 'NO', @show_progress =
  "YES"
  ```

  ```
  Calculating...
  Converting...
  [#                                                              ]
  [##########                                                     ]
  [################### ]
  [############################ ]
  [##################################### ]
  [############################################## ]
  [################################################### ]
  [####################################################]

  Total 2 duplicate defaults are removed and 7 defaults
  are converted to sharable inline defaults.
  Database is modified and in single-user mode.
  System Administrator (SA) must reset it to multi-user mode with
  sp_dboption
  ```

Usage

There are additional considerations when using **sp_merge_dup_inline_default**:

- You cannot run **sp_merge_dup_inline_default** on system databases.
- User databases must be in single-user mode before you run **sp_merge_dup_inline_default**.
- You may re-run **sp_merge_dup_inline_default** if the system procedure aborts.
- If **sp_merge_dup_inline_default** issues an error message stating that the SAP ASE server is out of locks:
  - Increase the value for **number of locks**, or
  - Reduce the lock promotion threshold with **sp_setpglockpromote** or **sp_setrowlockpromote**.

  Re-run **sp_merge_dup_inline_default**, and reset the values after **sp_merge_dup_inline_default** finishes.
- **sp_merge_dup_inline_default** changes only inline default objects for which the default value is a literal string constant or simple numbers (the literal string constant cannot include escaped string delimiters).

- **sp_merge_dup_inline_default** does not remove any duplicate inline default objects if their source text in `syscomments` is "`encrypted`."

Permissions

The permission checks for **sp_merge_dup_inline_default** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage database` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

sp_metrics

Backs up, drops, and flushes QP metrics—always captured in the default running group, which is group 1 in each respective database—and their statistics on queries.

Syntax

```
sp_metrics [‘backup’ backup_group_ID | ‘drop’, ‘gid’ [, ‘id’] |
    ‘flush’ | ‘help’, ‘command’]
```

Parameters

- **backup –** moves saved QP metrics from the default running group to a backup group, backs up the QP metrics from the old server into a backup group, and moves saved QP metrics from the default running group to a backup group.
- ***backup_group_ID* –** is the ID of the group the QP metrics from the old server into a backup group. To move saved QP metrics from the default running group to a backup group.
- **drop –** removes QP metrics from the system catalog. If you do not provide **‘*id*’**, **sp_metrics** drops the whole group you specified with **‘*gid*’**.
- ***gid* –** is the group ID of the QP metrics from the system catalog.
- ***id* –** is the ID of the QP metrics from the system catalog.
- **flush –** flushes all aggregated metrics in memory to the system catalog. The aggregated metrics for all statements in memory are zeroed out.
- **‘help’, ‘*command*’ –** provides usage information on **sp_metrics** commands.

Examples

- **Example 1 –** Move the QP metrics from a default group to a backup group.

  ```
  sp_metrics 'backup', '3'
  ```

- **Example 2 –** Provides information about **sp_metrics flush**:

  ```
  sp_metrics 'help', 'flush'
  ```

Usage

Access metric information using a **select** statement with **order by** against the `sysquerymetrics` view.

Use to back up the QP metrics from the old server into a backup group. To move saved QP metrics from the default running group to a backup group, to remove QP metrics from the system catalog. Flush all aggregated metrics in memory to the system catalog.

See also **select**, **set** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_metrics** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege or with `monitor qp performance` privilege (for **filter**, **show**, **help**).
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_configure* on page 167

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.

Syntax

```
sp_modify_resource_limit {name, appname}
    rangename, limittype, limitvalue, enforced, action, scope
```

Parameters

- ***name* –** is the SAP ASE 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 SAP ASE 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:
  - **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
  - **tempdb_space** – limits the number of pages from a `tempdb` database that a single session can have
- ***limit_value* –** is the maximum amount of the server resource that the login or application can use before the SAP ASE 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:
  - **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

    - **tempdb_space** – limits the number of pages from a temporary database that a single session can have.
- ***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:
    - **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

- **Example 1 –** 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:

```
sp_modify_resource_limit robin, NULL, weekends, row_count, 3000,
NULL,
    1, NULL
```

- **Example 2 –** 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:

```
sp_modify_resource_limit NULL, acctg, "at all times",
elapsed_time,
    45, 2, 2, 6
```

- **Example 3 –** This example changes the value of the resource limit that restricts elapsed time to all users of the *payroll* application during the `tu_wed_7_10` time range. The limit value for elapsed time decreases to 90 seconds (from 120 seconds). The values for time of execution, action taken, and scope remain unchanged:

```
sp_modify_resource_limit NULL, payroll, tu_wed_7_10,
elapsed_time, 90, null, null, 2
```

- **Example 4 –** This example changes the action taken by the resource limit that restricts the row count of all ad hoc queries and applications run by “joe_user” during the `saturday_night` time range. The previous value for action was 3, which aborts the transaction when a query exceeds the specified row count. The new value is to 2, which aborts the query batch. The values for limit type, time of execution, and scope remain unchanged.

```
sp_modify_resource_limit joe_user, NULL,
saturday_night, row_count, NULL, NULL, 2, NULL
```

Usage

There are additional considerations when using **sp_modify_resource_limit**:

- 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.
- SAP ASE provides resource limits to help system administrators prevent queries and transactions from monopolizing server resources. Resource limits, however, are not fully specified until they are bound to a time range.

For more information, see the *System Administration Guide*.

Permissions

The permission checks for **sp_modify_resource_limit** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage resource limit` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_add_resource_limit* on page 7
- *sp_drop_resource_limit* on page 258
- *sp_help_resource_limit* on page 368

sp_modify_time_range

Changes the start day, start time, end day, and 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, or **null** to keep the existing *starttime*:

  ```
  "HH:MM"
  ```
- ***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, or **null** to keep the existing *endtime*:

  ```
  "HH:MM"
  ```

  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

- **Example 1 –** Changes the end day of the “business_hours” time range from Friday to Saturday. Retains the existing start day, start time, and end time:

```
sp_modify_time_range business_hours, NULL, Saturday, NULL, NULL
```

- **Example 2 –** Specifies a new end day and end time for the "before_hours" time range:

  ```
  sp_modify_time_range before_hours, Monday, Saturday, NULL,
  "08:00"
  ```

Usage

There are additional considerations when using **sp_modify_time_range**:

- 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, see the *System Administration Guide*.

Permissions

The permission checks for **sp_modify_time_range** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage resource limit` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_add_resource_limit* on page 7
- *sp_add_time_range* on page 11
- *sp_drop_time_range* on page 261

sp_modifylogin

Deprecated by SAP ASE version 15.7 and later. To modify a login account in SAP ASE, use the **alter login** command. See *Reference Manual: Commands*.

sp_modifystats

Allows the system administrator, or any user with permission to execute the procedure and update statistics on the target table, to modify the density values of columns in `sysstatistics`.

Syntax

```
sp_modifystats [database].[owner].table_name,
    {"column_group" | "all"},
    modify_density,
    {range | total},
    {absolute | factor},
    "value"
    modify_default_selectivity,
    {inequality | inbetween},
    {absolute | factor},
    "value"
    modify_unique
    {range | total },
    {absolute | factor},
    "value"
```

Or:

```
sp_modifystats [database].[owner].table_name,
column_name | null,
    REMOVE_SKEW_FROM_DENSITY
    REMOVE_STICKINESS
```

Parameters

- ***table_name* –** is the name of the table to change. Specify the database name if the table is in another database, and specify the owner's name if more than one table of that name exists in the database. The default value for *owner* is the current user, and the default value for *database* is the current database.
- ***column_group* –** an ordered list of column names. To change a statistic for multiple columns (such as a density value), list the columns in the order used to create the statistic. Separate the column names with commas. For example, if your table has a density statistic on columns `a1`, `a2`, `a3`, `a4`:
    - "`a1`" modifies column `a1`.
    - "`a1,a2,a3`" modifies the column group `a1`,`a2`,`a3`,
    - You can also use a wildcard character , **%**, with the **column_group** parameter to represent a range of characters. For example, "`a1,%,a3`" modifies the groups `a1`,`a2`,`a3` and `a1`, `a4`, `a3`, and so on; "`a1,%`" modifies the groups `a1`,`a2` and `a1`,`a2`,`a3`, and so on, but not `a1`; "`a1%`" modifies the groups `a1`,`a2` and `a1`,`a2`,`a3`, and so on, as well as `a1`.
- **all –** modifies all column group for this table. Because "**all**" is a keyword, it requires quotes.
- **modify_density –** allows you to modify either the range or total density of a column or column group to the granularity specified in the *value* parameter. Range cell density represents the average number of duplicates of all values that are represented by range cells in a histogram. *value* is either the specified density value or a multiple for the current density. Must be between zero and one, inclusive, if absolute is specified.See the *Performance and Tuning Guide* for more information.Where:
    - **range** – modifies the range cell density.
    - **total** – modifies the total cell density.
    - **absolute** – ignore the current value and use the number specified by the *value* parameter.
    - **factor** – multiply the current statistical value by the *value* parameter.
- **modify_default_selectivity –** specifies the default selectivity value. Must be between zero and one, inclusive.

  Where:

    - **inequality** – indicates columns in which the predicate has an upper bound or a lower bound, but not both, and includes these range operators: > =, <=, >, <. The default value for inequality is .33

    - **inbetween** – indicates columns in which the predicate includes the upper bound and lower bound, and includes these range operators: > =, <=, >, <. The default value for inbetween is .25
    - **absolute** – ignore the current value and use the number specified by the value parameter.
    - **factor** – multiply the current statistical value by the value parameter.
- **modify_unique –** allows you to modify the **range** unique or **total** unique values of a column or column group to the granularity specified in the value parameter.
    - **range** – modifies the estimate for the number of unique values found in the range cells of the histogram. range does not include the frequency cells (that is, single-valued histogram cells). The estimate is represented as a fraction between 0.0 and 1.0, equal to:

      ```
      unique_range_values / (range_cell_rows * total rows_in_table)
      ```
    - **total** – modifies the estimate of the number of unique values for the column or column group (including the NULL value). The optimizer uses this value to estimate **group by** and **distinct** cardinality. It is represented as a fraction between 0.0 and 1.0 where the 1.0/<unique count> is stored in the catalogs.
    - **absolute** – ignore the current value and use the number specified by the value parameter.
    - **factor** – multiply the current statistical value by the value parameter.
- **REMOVE_STICKINESS –** removes the stickiness associated with the specified column. Specify **null** to remove the stickiness from all columns in the table.

  "Stickiness" occurs when the SAP ASE server retains the memory for these **update statistics** parameters:

    - **using step values**
    - **sampling**
    - **histogram_tuning_factor**
    - **hashing**
    - **no_hashing**
    - **partial_hashing**

  Once a phrase is "sticky," the SAP ASE server retains its behavior for that column on subsequent **update statistics** commands, even if you do not explicitly specify the parameters.
- ***column_name*** **–** is the name of a column in that table.
- **REMOVE_SKEW_FROM_DENSITY –** allows the system administrator to change the total density of a column to be equal to the range density, which is useful when data skew is present. Total density represents the average number of duplicates for all values, those in both frequency and range cells. Total density is used to estimate the number of matching rows for joins and for search arguments with a value that is unknown when the query is optimized. See the *Performance and Tuning Guide* for more information.

REMOVE_SKEW_FROM_DENSITY also updates the total density of any composite column statistics for which this column is the leading attribute. Most commonly, a composite index for which this column is the leading attribute would produce these composite column statistics, but they can also be produced when you issue a composite **update statistics** command.

Examples

- **Example 1 –** Changes the range density for column group `c00`, `c01` in table `tab_1` to 0.50000000:

```
sp_modifystats  "tab_1", "c00, c01", MODIFY_DENSITY, range,
absolute, "0.5"
```

- **Example 2 –** The total density for column group `c00`, `c01` in `tab_1` is multiplied by .5. That is, divided in half:

```
sp_modifystats  "tab_1", "c00,c01", MODIFY_DENSITY, total, factor,
"0.5"
```

- **Example 3 –** The total density for all the columns in table `tab_1` is multiplied by .5.

```
sp_modifystats  "tab_1", "all", MODIFY_DENSITY, total, factor,
"0.5"
```

- **Example 4 –** Total density for all column groups starting with `c12` is changed to equal the range density.

```
sp_modifystats "tab_1", "c12" REMOVE_SKEW_FROM_DENSITY
```

- **Example 5 –** Sets the default selectivity of inequality predicates with unknown constants (for example, `a1>@v1) to 0.09`.

```
sp_modifystats t10, a1, MODIFY_DEFAULT_SELECTIVITY, inequality,
absolute, "0.09"
```

- **Example 6 –** Sets the default selectivity for column a2 to use a value of 0.11 if you specify upper bound and a lower bound predicates with unknown constants (for example, `a2>@v1` and `a2<@v2`).

```
sp_modifystats t10, a2, MODIFY_DEFAULT_SELECTIVITY, inbetween,
absolute, "0.11"
```

- **Example 7 –** Modifies the range value for all columns for table t10 by a factor of 0.13.

```
sp_modifystats t10, "all", MODIFY_UNIQUE, range, factor, "0.13"
```

- **Example 8 –** Modifies the total unique value for all columns for table t10 to an absolute value of 0.14, which indicates there are (1.0 / 0.14) = 7.1428 unique values for each column in the table.

```
sp_modifystats t10, "all", MODIFY_UNIQUE, total, absolute,
"0.14"
```

Usage

There are additional considerations when using **sp_modifystats**:

- • Allows the system administrator to modify the density values of a column—or columns—in `sysstatistics`.
- Use **optdiag** to view a table's statistics. See the *Performance and Tuning Guide* for more information about table density and using **optdiag**.
- Any modification you make to the statistics with **sp_modifystats** is overwritten when you run **update statistics**. To make sure you are using the most recent statistical modifications, you should run **sp_modifystats** after you run **update statistics**.
- Because **sp_modifystats** modifies information stored in the `sysstatistics` table, you should make a backup of statistics before execute running **sp_modifystats** in a production system.
- You can use **modify_default_selectivity** only on an individual column, not a column group.
- SAP ASE uses the default selectivity for **modify_default_selectivity** when an unknown constant prevents it from using a histogram to estimate selectivity of the respective predicate. The default selectivity for a search argument using inequality is 33%. inequality search arguments include columns for which there is an upper bound predicate or a lower bound predicate, but not both, and use the >=, <=, >, < range operators. The default selectivity for search arguments that include an inbetween search arguments is 25%. inbetween search arguments include columns that have both an upper bound predicate and a lower bound predicate, or use the between operator.

See also **update statistics** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_modifystats** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `update statistics` on the object or with `manage any statistics` privilege. You must have **execute** permission on the procedure.
Disabled	With granular permissions disabled, you must be a user with with `update statistics` on the object or **sa_role**. You must have **execute** permission on the procedure.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

Tables used

`sysstatistics`

sp_modifythreshold

Modifies a threshold by associating it with a different threshold procedure, free-space level, or segment name.

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, the SAP ASE 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 SAP ASE server or on an Open Server. Thresholds cannot execute procedures on remote SAP ASE 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, the SAP ASE 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

- **Example 1 –** 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:

  ```
  sp_modifythreshold mydb, "default", 200, NULL, 175
  ```

- **Example 2 –** Modifies a threshold on the `data_seg` segment of `mydb` so that it executes the `new_proc` procedure:

  ```
  sp_modifythreshold mydb, data_seg, 250, new_proc
  ```

Usage

- You cannot use **sp_modifythreshold** to change the amount of free space or the segment name for the last-chance threshold.
- Use **sp_helpthreshold** for information about existing thresholds.
- Use **sp_dropthreshold** to drop a threshold from a segment.
- 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.

See also:

- **create procedure**, **dump transaction** in *Reference Manual: Commands*
- *System Administration Guide*.

Permissions

The permission checks for **sp_modifythreshold** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage database` privilege.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addthreshold* on page 49
- *sp_dboption* on page 193
- *sp_dropthreshold* on page 291
- *sp_helpthreshold* on page 445
- *sp_thresholdaction* on page 700

Crossing a Threshold

When a threshold is crossed, the SAP ASE server executes the associated stored procedure. The SAP ASE server uses the following search path for the threshold procedure:

- If the procedure name does not specify a database, the SAP ASE 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_**", the SAP ASE server looks in the `sybsystemprocs` database.

If the procedure is not found in either database, the SAP ASE server sends an error message to the error log.

The SAP ASE 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, the SAP ASE 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 the SAP ASE server and cannot be changed by users.

If the last-chance threshold is crossed before a transaction is logged, the SAP ASE 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 the SAP ASE 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.

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.

See also

- *sp_logdevice* on page 490

Create Threshold Procedures with sp_modifythreshold

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 granted the “sa_role”. All system roles active when the threshold procedure is created are modified in `systhresholds` as valid roles for the user writing the procedure.

The SAP ASE 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.

Execute Threshold Procedures with sp_modifythreshold

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”.

The SAP ASE 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.

Disable 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.

See also

- *sp_dboption* on page 193

sp_monitor

Displays statistics about the SAP ASE server.

Syntax

sp_monitor syntax is divided by command type for clarity, since many of the types have parameters of their own. The following code paragraph shows the syntax of the stored procedure as a whole, followed by the syntax of each command type interface.

```
sp_monitor [[connection | statement], [cpu | diskio | elapsed
time]]
    [event, [spid ]]
    [procedure, [dbname, [procname[, summary | detail]]]]
    [enable] [disable]
    [help],
    [deadlock][procstack]
```

Parameters

- **connection –** displays information on each connection. **connection** uses the following monitoring tables:

  - `monProcessSQLText`
  - `monProcessActivity`
- **statement –** displays information on each statement. **statement** uses the following monitoring tables:
  - `monProcessSQLText`
  - `monProcessStatement`
- **cpu | diskio | elapsed time –** these parameters order the output of **sp_monitor connection** or **sp_monitor statement**.
  - **cpu** – indicates the amount of CPU time consumed by each different connection or statement.
  - **diskio** – indicates the number of physical reads performed by each connection or statement.
  - **elapsed time** – indicates the sum of the CPU time and the wait times for each connection or statement.
- **event –** displays three possibilities. When you specify:
  - No option – only user tasks are displayed.
  - **sp_monitor, event, "-1"** – wait information about all tasks, both user and system, is displayed.
  - **sp_monitor, event, "spid"** – wait information pertaining to only the specified server process ID is displayed.
- ***spid* –** allows you to obtain **event** information for a specific task by entering its *spid*. You must specify the numeric value of *spid* within quotation marks.
- ***procedure* –** displays statistics about stored procedures:
  - **ProcName** – the stored procedure being monitored.
  - **DBNAME** – the database in which the stored procedure is located.
  - **NumExecs** – the approximate number of executions of this specific stored procedure.
  - **AvgCPUTime** – the average CPU time that it takes for the stored procedure to execute.
  - **AvgPhysicalReads** – the average number of disk reads performed by the stored procedure.
  - **AvgLogicalReads** – the average number of logical reads performed by the stored procedure.
  - **AvgMemUsed_KB** – the average amount of memory in KB used by the stored procedure.

  procedure uses the `monSysStatement` monitoring table.
- ***dbname* –** displays information on procedures for the specified database.
- ***procname* –** displays information on the specified procedure.

- **summary | detail –** displays either summary information, which provides an average of all instances of the procedure, or detailed information, which provides information on every instance of the stored procedure.
- **enable –** enables the new options for **sp_monitor**. It turns on the configuration parameter required to begin monitoring.
- **disable –** disables monitoring.
- **help –** displays the syntax and examples for **sp_monitor**, and also reports extensive information on using this procedure for deadlock analysis:

  ```
  sp_monitor 'help', 'deadlock'
  ```

  The **help** option also provides command-specific examples.
- **deadlock –** tells **sp_monitor** to process historical data from the `monDeadlock` table, and prints out a block of output for each instance of deadlock.
- **procstack –** examines the execution context of a task, including that of a deeply nexted stored procedure. The stack of procedures executed is extracted from the `monProcessProcedures` monitoring table.

Examples

- **Example 1 –** Reports information about how busy the SAP ASE server has been:

  ```
  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)
  ```
- **Example 2 –** Shows how to display information about connections:

  ```
  1> sp_monitor "connection"
  2> go
  spid    LoginName    ElapsedTime  LocksHeld  SQLText
  ----    ---------    -----------  ---------  -----------------
  12     sa          90300       2        exec get_employee_salaries
  27      sa          17700       1         exec get_employee_perks
  ```

  By default, the output by default is sorted in the descending order of the ElapsedTime.
- **Example 3 –** Identifies the connections performing the most physical reads:

  ```
  1> sp_monitor "connection","diskio"
  2> go
  ```

```
spid  LoginName  Physical_Reads  LocksHeld  SQLText
----  ---------  --------------  ---------  ---------------------
-----
12    sa        117           2        exec get_employee_salaries
27    sa         1              0          exec
get_employee_perks
```

- **Example 4 –** Displays information about each statement:

```
1> sp_monitor "statement"
2> go
spid   LoginName   ElapsedTime   SQLText
----   ---------   -----------   --------------------------
12     sa          100           exec get_employee_salaries
```

- **Example 5 –** Displays the events each task spent time waiting for and the duration of the wait, reported in descending order of wait times:

```
1> sp_monitor "event"
2> go
SPID   WaitTime    Description
------ ----------- ----------------------------------------
6      108200      hk: pause for some time
29     108200      waiting for incoming network data
10     107800      waiting while allocating new client socket
15     17100       waiting for network send to complete
14     5900        waiting for CTLIB event to complete
14     400         waiting for disk write to complete
7      200         hk: pause for some time
7      100         waiting on run queue after yield
12     100         waiting for network send to complete
```

- **Example 6 –** Displays event data for spid 14:

```
1> sp_monitor "event","14"
2> go
WaitTime    Description
----------- -----------------------------------
9000 waiting for CTLIB event to complete
600 waiting for disk write to complete
200 waiting for disk write to complete
100 waiting on run queue after yield
100 wait for buffer write to complete
```

- **Example 7 –** Provides a summary of most recently run procedures, sorted in descending order of average elapsed time. This example provides historical monitoring information rather than the current state.

```
1> sp_monitor "procedure"
2> go

Average Procedure Statistics
============================

ProcName       DBName    AvgElapsedTime   AvgCPUTime  AvgWaitTime
AvgPhysicalReads AvgLogicalReads AvgPacketsSent NumExecs
-----------------------------------------------------------------
------
```

```
neworder_remote   tpcc   1833     16    1083     26   96    0     6
neworder_local    tpcc   1394     13    1181     31   122   0     38
tc_startup        tpcc   1220     3     1157     0    3     0     59
delivery           tpcc   1000     0     800
23   49    0     2
```

Usage

Note: Before using the new parameters associated with **sp_monitor**, you must set up monitoring tables and the related stored procedures needed to enable. See *Installing Monitoring Tables* in *Performance and Tuning: Monitoring and Analyzing*.

- The SAP ASE 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 the SAP ASE 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 the SAP ASE 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 the SAP ASE server was last started, 67 of them since the last time **sp_monitor** was run.
- This list shows the monitoring tables accessed by monitoring type, as well as the configuration option and its type for each table:
  - **connection**
    - `monProcessSQLext`
      - `max SQL text monitored` – Value
      - `SQL batch capture` – Boolean
    - `monProcessActivity`
      - `wait event timing` – Boolean
      - `per object statistics active` – Boolean
  - **procstack**
    - `monProcessProcedures`
      - None – N/A
  - **statement**
    - `monProcessSQLText`
      - `max SQL text monitored` – Value

    - `SQL batch capture` – Boolean
  - `monProcessStatement`
    - `statement statistics active` – Boolean
    - `per object statistics active` – Boolean
    - `wait event timing` – Boolean
- **event**
  - `monProcessWaits`
    - `wait event timing` – Value
    - `process event waits` – Boolean
- **procedure**
  - `monSysStatement`
    - `statement statistics active` – Boolean
    - `per object statistics active` – Boolean
    - `statement pipe max messages` – Value
    - `statement pipe active` – Boolean
- **deadlock**
  - `monDeadlock`
    - `deadlock pipe max messages` – Value
    - `deadlock pipe active` – Boolean

- **sp_monitor connection** monitors connections actively executing T-SQL only, and does not report on all connections.
- You must run **sp_monitor** from the `master` database. However, if you are analyzing deadlock data archived in another database, you can run **sp_monitor deadlock** from that database.
- **sp_monitor event** no longer displays all tasks (including system tasks), when called with no options. In SAP ASE version 15.0.2 and above, the event option provides three possibilities. When:
  - No option is provided – only user tasks are displayed.
  - You specify `sp_monitor, event, "-1"`, wait information about all tasks, both user and system, is displayed.
  - You specify `sp_monitor, event, "spid"`, wait information pertaining to only the specified server process ID is displayed.
- The following table 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.

Column Heading	Equivalent Variable	Description
`last_run`		Clock time at which the **sp_monitor** procedure last ran.
`cur-rent_run`		Current clock time.
`seconds`		Number of seconds since **sp_monitor** last ran.
`cpu_busy`	***@@cpu_busy***	Number of seconds in CPU time that the SAP ASE server's CPU was doing SAP ASE work.
`io_busy`	***@@io_busy***	Number of seconds in CPU time that the SAP ASE server has spent doing input and output operations.
`idle`	***@@idle***	Number of seconds in CPU time that the SAP ASE server has been idle.
`pack-ets_re-ceived`	***@@pack_re-ceived***	Number of input packets read by the SAP ASE server.
`pack-ets_sent`	***@@pack_sent***	Number of output packets written by the SAP ASE server.
`pack-et_errors`	***@@packet_er-rors***	Number of errors detected by the SAP ASE server while reading and writing packets.
`to-tal_read`	***@@total_read***	Number of disk reads by the SAP ASE server.
`to-tal_write`	***@@total_write***	Number of disk writes by the SAP ASE server.
`total_er-rors`	***@@total_errors***	Number of errors detected by the SAP ASE server while reading and writing.
`connec-tions`	***@@connections***	Number of logins or attempted logins to the SAP ASE server.

- The first time **sp_monitor** runs after SAP ASE start-up, the number in parentheses is meaningless.
- The SAP ASE 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
  ```
- You must run **sp_monitor** when a representative workload is running on the system.

- Typically, run procedures in this sequence:
  - Run **sp_monitor enable**
  - Invoke **sp_monitor** options
  - Run **sp_monitor disable** when you have completed the monitoring
- When you are using **sp_monitor procedure**, the number of rows returned can be very large; you may want to use the **summary** option instead of the **detail** option. It may also take a while for this command to complete on an active system.

Permissions

The permission checks for **sp_monitor** are the same whether or not granular permissions is enabled:

- The database owner of `sybsystemprocs` can execute **sp_monitor** and can grant `execute` permission to other users
- The stored procedure is created with **execute as owner.** The owner is sa. The owner must have **mon_role** which user sa has by default.

For more information see *Monitoring Tables* in *Performance and Tuning: Monitoring and Analyzing*.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_who* on page 736

sp_monitorconfig

Displays cache usage statistics regarding metadata descriptors for indexes, objects, databases, and the kernel resource memory pool. **sp_monitorconfig** also reports statistics on auxiliary scan descriptors used for referential integrity queries, and usage statistics for transaction descriptors and DTX participants.

Syntax

```
sp_monitorconfig "configname"[, "result_tbl_name"][, "full"]
```

Parameters

- ***configname*** **–** is either **all**, or part of the configuration parameter name with the monitoring information that is being queried. Valid configuration parameters are listed in the “Usage” section. Specifying **all** displays descriptor help information for all indexes, objects, databases, and auxiliary scan descriptors in the server.
- **“*result_tbl_name*” –** (optional) is the name of the table you create to save the stored procedure results. If you pass a table name for *result_tabl_name* that does not already exist, **sp_monitorconfig** creates a table to hold the result set.
- **“full” –** returns a set of values for the **configname** that you specify. The values are:
  - `config_val` – reports the configured value
  - `system_val` – reports the systems default value when there’s novalue configured
  - `total_val` – reports the actual value used

Examples

- **Example 1 –** Shows all items that are open:

```
sp_monitorconfig "open"

Configuration option is not unique.
option_name                     config_value run_value
-----------------------------   ----------  ----------
number of open databases                12          12
number of open objects                 500         500
curread change w/ open cursors           1           1
open index hash spinlock ratio         100         100
number of open indexes                 500         500
open index spinlock ratio              100         100
open object spinlock ratio             100         100
number of open partitions              500         500
```

- **Example 2 –** Shows the status for all configurations:

```
sp_monitorconfig "all"
```

```
-------------
Usage information at date and time: May  6 2010  4:32PM.
Name                       Num_free Num_active Pct_act Max_Used Reuse
_cnt Instance_Name
--------------------   --------- --------- -------  -------
--------- -------------
additional network
memory  1358436  809440   37.34    825056        0           NULL
audit queue
size             100      0    0.00        0       0          NULL
disk i/o
structures            256       0    0.00        29        0
      NULL
heap memory per
user         4096       0    0.00        0        0           NULL

. . .

size of process object
he     3000       0    0.00          0        0          NULL
size of shared class
heap     6144       0    0.00          0        0          NULL
size of unilib
cache       306216     816    0.27       816       0          NULL
txn to pss
ratio             400      0    0.00         0       0          NULL
```

- **Example 3 –** Shows 61 active object metadata descriptors, with 439 free. The maximum used at a peak period since the SAP ASE server was last started is 61:

```
sp_monitorconfig "open objects"
Usage information at date and time: Apr 22 2002  2:49PM.
Name                       Num_free Num_active Pct_act Max_Used Reuse
_cnt Instance_Name
--------------------   --------- --------- -------  -------
--------- -------------
number of open
objects       439      61    12.20       61       0          NULL
```

  You can then reset the size to 550, for example, to accommodate the 439 maximum used metadata descriptors, plus space for 10 percent more:

```
sp_configure "number of open objects", 330
```

- **Example 4 –** Shows the maximum number of index metadata descriptors, which is 44:

```
sp_monitorconfig "open indexes"
Usage information at date and time: Apr 22 2002  2:49PM.
Name                       Num_free Num_active Pct_act Max_Used Reuse
_cnt Instance_Name
--------------------   --------- --------- -------  -------
--------- -------------
number of open
indexes       556     44     7.33       44       0          NULL
```

  You can reset the size to 100, the minimum acceptable value:

```
sp_configure "number of open indexes", 100
```

- **Example 5 –** Shows the number of active scan descriptors as 30, though the SAP ASE 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:

```
sp_monitorconfig "aux scan descriptors"
Usage information at date and time: Apr 22 2002  2:49PM.

Name                   Num_free Num_active Pct_act Max_Used Reuse
_cnt Instance_Name
--------------------   --------- --------- -------  -------
--------- -------------
number of aux scan
descri     170     30    15.00         32        0           NULL
```

- **Example 6 –** The SAP ASE server is configured for five open databases, all of which have been used in the current session.

```
sp_monitorconfig "number of open databases"
Name                   Num_free Num_active Pct_act Max_Used Reuse
_cnt Instance_Name
--------------------   --------- --------- -------  -------
--------- -------------
number of open
databses         0      5    100.00         5  Yes             NULL
```

However, as indicated by the `Reuse_cnt` 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.

- **Example 7 –** Only 10.2 percent of the transaction descriptors are currently being used. However, the maximum number of transaction descriptors used at a peak period since the SAP ASE server was last started is 523:

```
sp_monitorconfig "txn to pss ratio"
Usage information at date and time: Apr 22 2002  2:49PM.
Name                   Num_free Num_active Pct_act Max_Used Reuse
_cnt Instance_Name
--------------------   --------- --------- -------  -------
--------- -------------
txn to pss
ratio            784    80    10.20      523       0         NULL
```

- **Example 8 –** Using the optional parameter *result_tbl_name* to create a user table saves the **sp_monitorconfig** result to this table:

```
create table sample_table
(Name varchar(35),Config_val int, System_val int, Total_val int,
Num_free int, Num_active int, Pct_act char(6), Max_used int,
Num_Reuse int, Date varchar(30))
```

```
create table sample_table
(Name varchar(35),
Config_val int,
System_val int,
Total_val int,
Num_free int,
Num_active int,
Pct_act char(6),
Max_Used int,
Reuse_cnt int,
Date varchar(30),
Instance_Name varchar(35))
```

The name of the table created becomes the second parameter of **sp_monitorconfig**. Capture the values for **number of locks** and **number of alarms** in `sample_table`:

```
sp_monitorconfig "locks", sample_table
sp_monitorconfig "number of alarms", sample_table
```

Display the values captured in `sample_table`:

```
select * from sample_table
 Name               Config_val System_val Total_val Num_free Num_
active
Pct_act Max_used Reuse_cnt   Date                Instance_Name
------------------- ----------- ---------- --------- ---------
----------
------- -------- ---------   ------------------- ----------------
--------
 number of locks        5000       684     5000    4915        85
1.70       117         0    Aug 23 2006  6:53AM
 number of alarms            40      0        40      28        12
30.00        13         0    Aug 23 2006  6:53AM
```

The result set saved to the table accumulates until you delete or truncate the table.

Note: If `sample_table` is in another database, you must provide its fully qualified name in quotes.

- **Example 9 –** Displays the `configure_value`, `system_value`, and `run_value` columns of all the configurations:

```
sp_monitorconfig "all", null, "full"
go
Usage information at date and time: Mar 23 2004  5:15PM
Name                        Configure Value  System Value  Run Value
Num_free   Num_active  Pct_act  Max_Used  Reuse_cnt   Instance_Name
------------------------    ----------    ----------     -------
---------  ----------  -------  --------  ---------   -----------
--------
additional network memory            0       2167876     2167876
  1358436      809440    37.34    825056         0
    NULL
audit queue size                   100             0         100
     100         0    0.00        0         0               NULL
disk i/o structures                256             0         256
```

```
     256          0    0.00       29         0                NULL
heap memory per user              4096           563       4096
    4096          0    0.00        0         0                NULL
kernel resource memory             4096             0        4096
    3567        529   12.92      529        0                 NULL
max cis remote connection             0           100         100
     100          0    0.00        0         0                NULL
. . .
size of shared class heap          6144             0        6144
    6144          0    0.00        0         0                NULL
size of unilib cache            0307032       307032      306216
     816          0     .27      816         0                NULL
txn to pss ratio                     16             0          16
     400          0    0.00        0          0               NULL
```

Usage

There are additional considerations when using **sp_monitorconfig**:

- The output for **additional network memory** reports the utilization statistics for the global network memory pool regardless of whether or not memory has been added to this pool by setting additional network memory to a positive value.
- If the **max cis remote connections** configuration parameter has a `config_value`, the `system_val` reports a value of zero (0).
- If you reconfigure a resource using a value that is smaller than the original value it was given, the resource does not shrink, and the `Num_active` configuration parameter can report a number that is larger than `Total_val`. The resource shrinks and the numbers report correctly when the SAP ASE server restarts.
- **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.
- **sp_monitorconfig** monitors the following resources:
  - **additional network memory**
  - **audit queue size**
  - **heap memory per user**
  - **max cis remote connection**
  - **max memory**
  - **max number network listeners**
  - **memory per worker process**
  - **max online engines**
  - **number of alarms**
  - **number of aux scan descriptors**
  - **number of devices**

  - **number of dtx participants**
  - **number of java sockets**
  - **number of large i/o buffers**
  - **number of locks**
  - **number of mailboxes**
  - **number of messages**
  - **number of open databases**
  - **number of open indexes**
  - **number of open objects**
  - **number of open partitions**
  - **number of remote connections**
  - **number of remote logins**
  - **number of remote sites**
  - **number of sort buffers**
  - **number of user connections**
  - **number of worker processes**
  - **partition groups**
  - **permission cache entries**
  - **procedure cache size**
  - **size of global fixed heap**
  - **size of process object heap**
  - **size of shared class heap**
  - **size of unilib cache**
  - **txn to pss ratio**
- The columns in the **sp_monitorconfig** output provide the following information:
  - `Num_free` – specifies the number of available metadata or auxiliary scan descriptors not currently used.
  - `Num_active` – specifies the number of metadata or auxiliary scan descriptors installed in cache (that is, active).
  - `Pct_active` – specifies the percentage of cached or active metadata or auxiliary scan descriptors.
  - `Max_Used` – specifies the maximum number of metadata or auxiliary scan descriptors that have been in use since the server was started.
  - `Reuse_cnt` – 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_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 `Reused` 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, the SAP ASE 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 Series: Monitoring Adaptive Server with sp_sysmon*. To find the current number of indexes, objects, or databases, use **sp_countmetadata**.
- To get an accurate reading, run **sp_monitorconfig** during a normal SAP ASE 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.
- *result_tbl_name* creates a table using the following syntax. All the result information is saved in this table, which returns no standard output.

```
create table table_name(
    Name varchar(35), Num_free int,
    Num_active int, Pct_act char(6),
    Max_Used int, Reuse_cnt int,
    Date varchar(30))
```

- Some configuration parameters, such as *number of sort buffers* and *txn to pss ratio*, are dependent on the number of configured user connections, while other configuration parameters, such as *max number of network listeners*, are per engine.
- For the configuration value *permission cache entries*, the values of `Num_free`, `Num_active`, `Pct_act`, and `Max_Used` are averages of per connection values, however `Reuse_cnt` is a server-wide value.
- The output of **sp_monitorconfig** uses the number of user connections and online engines to calculate the values for the columns `num_free`, `num_active`, `pct_act` and `max_used`.
- The updates on the internal monitor counters are done without using synchronization methods because of performance reasons. For this reason, a multi-engine SAP ASE server under heavy load might report numbers in the **sp_monitorconfig** output that are not a completely accurate.
- You might see the number of active locks as greater than 0 on an idle system. These “active” locks are reserved and used internally.

Permissions

The permission checks for **sp_monitorconfig** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with mon_role or have `manage server` privileges.
Disabled	With granular permissions disabled, you must be a user with either mon_role or sa_role.

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

See also

- *sp_configure* on page 167
- *sp_countmetadata* on page 180
- *sp_helpconfig* on page 382
- *sp_helpconstraint* on page 390
- *sp_sysmon* on page 684

sp_monitor_server

Provides server-wide monitoring information.

Syntax

```
sp_monitor_server [server_name]
```

Examples

- **Example 1 –** Displays the current server monitoring information:

```
sp_monitor_server
```

```
last_run                  current_run                seconds
-------------------------- -------------------------- -----------
May 10 2010  4:23PM          May 10 2010  4:23PM          1

(1 row affected)
cpu_busy                io_busy                idle
------------------------- -------------------------
---------------------
              0(0)-0%                0(0)-0%          21(0)-0%
packets_received          packets_sent              packet_errors
------------------------- -------------------------
----------------------
0(0)                                     0(0)              0(0)
total_read          total_write         total_errors        conne
ctions
------------------- ------------------- -------------------
--------------
1743(0)                            146(0)                 0(0)
     1(0)
```

sp_object_stats

Shows lock contention, lock wait-time, and deadlock statistics for tables and indexes.

Syntax

```
sp_object_stats interval[, top_n[, dbname, objname[, rpt_option]]]
```

Parameters

- ***interval* –** specifies the time period for the sample. It must be in HH:MM:SS form, for example “00:20:00”.
- ***top_n* –** is the number of objects to report, in order of contention. The default is 10.
- ***dbname* –** is the name of the database to report on. If no database name is given, contention on objects in all databases is reported.
- ***objname* –** is the name of a table to report on. If a table name is specified, the database name must also be specified.
- ***rpt_option* –** specifies the report type:
  - **rpt_locks** reports grants, waits, deadlocks and wait times for the tables with the highest contention. **rpt_locks** is the default.
  - **rpt_objlist** reports only the names of the objects that had the highest level of lock activity.

Examples

- **Example 1 –** Reports lock statistics on the top 10 objects server-wide:

  ```
  sp_object_stats "00:20:00"
  ```

- **Example 2 –** Reports only on tables in the `pubtune` database, and lists the five tables that experienced the highest contention:

  ```
  sp_object_stats "00:20:00", 5, pubtune
  ```

- **Example 3 –** Shows only the names of the tables that had the highest locking activity, even if contention and deadlocking does not take place:

  ```
  sp_object_stats "00:15:00", @rpt_option = "rpt_objlist"
  ```

Usage

There are additional considerations when using **sp_object_stats**:

- **sp_object_stats** reports on the shared, update, and exclusive locks acquired on tables during a specified sample period. The following reports shows the `titles` tables:

  ```
  Object Name: pubtune..titles (dbid=7,
  objid=208003772,lockscheme=Datapages)


    Page Locks     SH_PAGE              UP_PAGE             EX_PAGE$

  ----------    ----------          ----------          ----------
    Grants:         94488                4052                 4828
    Waits:              532                 500                  776
    Deadlocks:            4                   0                   24
    Wait-time:     20603764 ms        14265708 ms         2831556
  ms
    Contention:        0.56%              10.98%               13.79%

   *** Consider altering pubtune..titles to Datarows locking.
  ```

- The meaning of the values are:
  - Grants – the number of times the lock was granted immediately.
  - Waits – the number of times the task needing a lock had to wait.
  - Deadlocks – the number of deadlocks that occurred.
  - Wait-times – the total number of milliseconds that all tasks spent waiting for a lock.
  - Contention – the percentage of times that a task had to wait or encountered a deadlock.
- **sp_object_stats** recommends changing the locking scheme when total contention on a table is more than 15 percent, as follows:
  - If the table uses allpages locking, it recommends changing to datapages locking.
  - If the table uses datapages locking, it recommends changing to datarows locking.
- **sp_object_stats** creates a table named `tempdb..syslkstats`. This table is not dropped when the stored procedure completes, so it can be queried by a system administrator using Transact-SQL.

- Only one user at a time should execute **sp_object_stats**. If more than one user tries to run **sp_object_stats** simultaneously, the second command may be blocked, or the results may be invalid.
- The `tempdb..syslkstats` table is dropped and re-created each time **sp_object_stats** is executed.
- The structure of `tempdb..syslkstats` is:

Column name	Datatype	Description
`dbid`	`small-int`	Database ID
`objid`	`int`	Object ID
`lockscheme`	`small-int`	Integer values 1–3: Allpages = 1, Datapages = 2, Datarows = 3
`page_type`	`small-int`	Data page = 0, or index page = 1
`stat_name`	`char(30)`	The statistics represented by this row The values in the `stat_name` column are composed of three parts: • The first part is “ex” for exclusive lock, “sh” for shared lock, or “up” for update lock. • The second part is “pg” for page locks, or “row” for row locks. • The third part is “grants” for locks granted immediately, “waits” for locks that had to wait for other locks to be released, “deadlocks” for deadlocks, and “waittime” for the time waited to acquire the lock.
`stat_value`	`float`	The number of grants, waits or deadlocks, or the total wait time

- If you specify a table name, **sp_object_stats** displays all tables by that name. If more than one user owns a table with the specified name, output for these tables displays the object ID, but not the owner name.

See also:

- **alter table** in *Reference Manual: Commands*

Permissions

The permission checks for **sp_object_stats** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege.

Setting	Description
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_objectsegment

Reports the partition name, segment name, and creation date for the specified object.

Syntax

```
sp_objectsegment object_name
```

Parameters

- ***object_name* –** name of the object. Acceptable objects are:
  - System tables
  - Views
  - User tables
  - System procedures
  - Defaults
  - Rules
  - Triggers
  - Referential constraints
  - Check constraints

- Extended types
- Functions
- Computed columns
- Partitions

Examples

- **Example 1 –** Reports information about the authors table:

  ```
  sp_objectsegment authors
  ```

  ```
  Partition_name    Data_located_on_segment     When_created
  ----------------  -----------------------      ------------
  auidind_576002052                default     Feb  9 2012 11:18AM
  ```

Permissions

Any user may run **sp_objectsegment**.

sp_opt_querystats

Returns a performance analysis for the selected query.

Syntax

```
sp_opt_querystats "query_text" | help [, "diagnostic_options" | null
    [, database_name] [, user_name]]
```

Parameters

- **"*query_text*" –** is the text of the query you are analyzing, enclosed in quotation marks.
- **help –** displays syntax and usage information for **sp_opt_querystats**.
- ***diagnostic_options* –** (Optional) the diagnostic parameters based on **set** options. See "Usage."
- **null – sp_opt_querystats** requires three parameters to specify the name of a database. If you do not require diagnostic options, enter a value of `null` for this parameter to specify a value for the *database_name* parameter.
- ***database_name* –** (optional) the name of the database in which the query is executed. Use this parameter if the query you are analyzing does not have fully qualified tables.
- ***user_name* –** (Optional) name of the user who executes the query within the database specified by the *database_name* parameter. This user must already exist in the database, and the login executing **sp_opt_querystats** must have permission to execute the **setuser** command in that database.

Examples

- **Example 1 –** Analyzes a **select** command on the `pubs2` database:

  ```
  sp_opt_querystats 'select * from pubs2.dbo.authors'
  ```

- **Example 2 –** Analyzes a **select** command on the **pubs2** database, and includes information based on enabling these **set** commands: **set showplan**, **set statistics io**, **set option show**, **set statistics plancost on**:

  ```
  sp_opt_querystats 'select * from pubs2.dbo.authors',
     'showplan,statio,option_show, plancost'
  ```

Usage

There are additional considerations when using **sp_opt_querystats**:

- You must include the **exec** command for **sp_opt_querystats** to execute the query.
- To run **sp_opt_querystats** as a different user, include the **setuser** command with the **exec immediate** command or in an out query context.
- You must include the **showdata** command for **sp_query_stats** to return the result set.
- After you issue **set quoted_identifier on**, you may surround **sp_opt_querystats** options with quotes. For example:

  ```
  sp_opt_querystats 'select "col" from "MYTABLE"', 'all','DB'
  ```
- *diagnostic_option* is one of:

diagnos-tic_option	set option	Notes
statio	**set statistics io on**	
stattime	**set statistics time on**	
showplan	**set showplan on**	
missingstats	**set option show_missing_stats long**	
resource	**set statistics resource on**	
switches	**show switches**	
option_show_long	**set option show long**	**option_show_long** and **option_show** are mutually exclusive.
option_show	**set option show on**	
showdata	**set nodata on**	**set nodata on** is not executed when you include **showdata**.
plancost	**set statistics plancost on**	Only available when you specify the **exec** or **allexec** options.

diagnostic_option	**set option**	**Notes**
exec	**set noexec on**	**set noexec on** is not executed when you include **exec**.
allrows_mix	**set plan optgoal allrows_mix**	**allrows_mix**, **allrows_oltp**, and **allrows_dss** are mutually exclusive.
allrows_oltp	**set plan optgoal allrows_oltp**	
allrows_dss	**set plan optgoal allrows_dss**	
diagmode	Returns enhanced progress information.	
all	Enables the first seven options	**all** and **allexec** cannot be combined with other parameters, and are mutually exclusive. The **allexec** option includes the **all** option.
allexec	Enables the first seven options	

- The option list must be enclosed in quotation marks if you include more than one option, or if you specify the keyword **all**.
- Running **sp_opt_querystats** without any options is the same as running it with the **all** option.

Permissions

Any user can execute **sp_opt_querystats**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	**Values**
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_optgoal

Creates a user-defined optimization goal, and defines the set of active criteria included in the goal.

This system procedure contains the functionality to make optimization goals that are run and saved into global optimization levels in the server using **sp_configure**. You can use this at the session level using the **set** command, or globally via **sp_configure**.

Syntax

```
sp_optgoal 'goal_name', action
```

Parameters

- ***goal_name* –** name of the goal you are creating. *goal_name* cannot be longer than 12 characters.
- ***action* –** action for **sp_optgoal** to perform. One of:
  - **show | null | no action** – displays the contents of the goal.
  - **save** – creates new goal or updates and existing goal.
  - **delete** – deletes the goal.

Examples

- **Example 1 –** If you set these goals for the current session:

  ```
  SET PLAN OPTLEVEL ase_current
  SET PLAN  OPTGOAL allrows_mix
  SET HASH_JOIN 1
  ```

  This command saves these settings in a goal named `goal_1`:

  ```
  sp_optgoal 'goal_1', 'save'
  ```

  Either of these allow you to use the settings for goal_1 for the current session:

  - Using the **set** command:

```
set plan optgoal goal_1
```

- Using **sp_configure**:

```
sp_configure "optimization goal", 1, "goal_1"
```

- **Example 2 –** Deletes `goal_1`:

```
sp_optgoal 'goal_1', 'delete'
```

Usage

sp_optgoal with no parameters displays a list of all user-defined optimizer goals.

Permissions

The permission checks for **sp_optgoal** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, users with sa_role and sa_serverprivs_role must have `manage opt goal` privilege to create or delete a goal. By default, sa_role and sa_serverprivs_role are granted the `manage opt goal` privilege. Once created, all users can use the goal. Any user can run **sp_optgoal 'show'**.
Disabled	With granular permissions disabled, you must be a user with sa_role to create or delete a goal. However, once created, all users can use the goal. Any user can run **sp_optgoal 'show'**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_options

Shows option values.

Syntax

```
sp_options [ [show | help
    [, option_name | category_name | null
    [, dflt | non_dflt | null [, spid] ] ] ] ]
```

Parameters

- **show –** lists the current and default values of all options, grouped according to their category. Issuing **sp_options** show with an option name specified gives you the current and default value for the individual option. You can also specify a session ID, and whether you want to view options with default settings or options with non-default settings.
- **help –** indicates that you wish to show usage information. You achieve the same result when you issue **sp_options** with no parameters.
- ***option_name* –** is the name of the option.
- ***category_name* –** is the category of the option.
- **null –** indicates the option for which you want to view the settings.
- **dflt | non_dflt | null –** indicates whether to show options with default settings or to show options with non-default settings.
- ***spid* –** specifies the session ID. Use the session ID to view other session settings.

Examples

- **Example 1 –** Shows **sp_options** usage:

```
1> sp_options
2> go
```

```
Usage:
sp_options [ [show | help
            [, <option_name>|<category_name>|null
               [, dflt | non_dflt | null
                    [, <spid>] ] ] ] ]
```

- **Example 2 –** Shows a list of all current and default options:

```
1> sp_options show
2> go
Category: Query Tuning
name                      currentsetting       defaultsetting
      scope
------------------------------------------
------------------------------
optgoal                  allrows_mix        allrows_mix        0
opttimeoutlimit           40                 10                0
```

```
merge_join                1                  1                  4
hash_join                0                  0                  4
nl_join                 1                  1                  4
distinct_sorted           1                  1                  4
distinct_sorting          1                  1                  4
distinct_hashing          1                  1                  4
group_sorted             1                  1                  4
group_hashing            1                  1                  4
group_inserting           0                  0                  4
order_sorting            1                  1                  4
append_union_all          1                  1                  4
merge_union_all          1                  1                  4
merge_union_distinct       1                  1                 4
hash_union_distinct        1                  1                 4
store_index              1                  1                  4
bushy_space_search         0                  0                 4
parallel_query            1                  1                  4
replicated_partition       0                  0                 4
ase125_primed            0                  0                  4
index_intersection         0                  0                 4
index_union              1                  1                  4
multi_table_store_ind      0                  0                 4
advanced_aggregation       0                  0                 4
opportunistic_distinct_view 1                  1                 4
repartition_degree         3                  1                 2
scan_parallel_degree       0                  1                 2
resource_granularity       10                 10                2
parallel_degree           0                  1                  2
statistics simulate        0                  0                 4
forceplan                0                  0                  7
prefetch                 1                  1                  6
metrics_capture           0                  0                  6
process_limit_action         quiet                 quiet
     2
plan replace             0                  0                  4
plan exists check         0                  0                  4
plan dump                0                  0                  4
plan load                0                  0                  4

(39 rows affected)
(return status = 0)
```

- **Example 3 –** Shows the current and default setting for an individual option:

```
1> sp_options show, "index_intersection"
2> go

name               category      currentsetting  defaultsetting
scope
-----------------------------------------------------------------
------
index_intersection Query Tuning  0              0              4

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

- **Example 4 –** Shows only the default setting for an individual option:

```
1> sp_options show, "index_intersection", dflt
2> go

name                    defaultsetting
--------------------------------------
index_intersection      0

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

- **Example 5 –** Shows the current and default settings for a category:

```
1> sp_options show, "Query Tuning"
2> go

Category: Query Tuning

name                        currentsetting  defaultsetting  scope
-----------------------------------------------------------------
optgoal                     allrows_mix     allrows_mix     0
opttimeoutlimit             10              10              0
merge_join                  1               1               4
hash_join                   0               0               4
nl_join                     1               1               4
distinct_sorted             1               1               4
distinct_sorting            1               1               4
distinct_hashing            1               1               4
group_sorted                1               1               4
group_hashing               1               1               4
group_inserting             0               0               4
order_sorting               1               1               4
append_union_all            1               1               4
merge_union_all             1               1               4
merge_union_distinct        1               1               4
hash_union_distinct         1               1               4
store_index                 1               1               4
bushy_space_search          0               0               4
parallel_query              1               1               4
replicated_partition        0               0               4
ase125_primed               0               0               4
index_intersection          0               0               4
index_union                 1               1               4
multi_table_store_ind       0               0               4
advanced_aggregation        0               0               4
opportunistic_distinct_view 1               1               4
repartition_degree          3               1               2
scan_parallel_degree        0               1               2
resource_granularity        10              10              2
parallel_degree             0               1               2
statistics simulate         0               0               4
forceplan                   0               0               7
prefetch                    1               1               6
metrics_capture             0               0               6
process_limit_action        quiet           quiet           2
plan replace                0               0               4
plan exists check            0               0               4
plan dump                    0               0               4
plan load                    0               0               4

(39 rows affected)
(return status = 0)
```

- **Example 6 –** Shows the default settings for the Query Tuning category:

```
1> sp_options show, "Query Tuning", dflt
2> go

Category: Query Tuning

name                         defaultsetting
------------------------------------------
optgoal                      allrows_mix
opttimeoutlimit              10
merge_join                   1
hash_join                    0
nl_join                      1
distinct_sorted              1
distinct_sorting             1
distinct_hashing             1
group_sorted                 1
group_hashing                1
group_inserting              0
order_sorting                1
append_union_all             1
merge_union_all              1
merge_union_distinct         1
hash_union_distinct          1
store_index                  1
bushy_space_search           0
parallel_query               1
replicated_partition         0
ase125_primed                0
index_intersection           0
index_union                  1
multi_table_store_ind        0
advanced_aggregation         0
opportunistic_distinct_view  1
repartition_degree           1
scan_parallel_degree         1
resource_granularity         10
parallel_degree              1
statistics simulate          0
forceplan                    0
prefetch                     1
metrics_capture              0
process_limit_action         quiet
plan replace                 0
plan exists check            0
plan dump                    0
plan load                    0
```

```
(39 rows affected)
(return status = 0)
```

- **Example 7 –** Shows the options that use non-default settings in the Query Tuning category:

```
1> sp_options show, "Query Tuning", non_dflt
2> go

Category: Query Tuning

name                 currentsetting   defaultsetting
-----------------------------------------------------
repartition_degree   3                1
scan_parallel_degree 0                1
parallel_degree      0                1

(3 rows affected)
(return status = 0)
```

- **Example 8 –** Shows the options in the Query Tuning category:

```
1> sp_options, show, null
2> go

Category: Query Tuning

name                         currentsetting defaultsetting scope
------------------------------------------ -------------- -----
optgoal                      allrows_mix    allrows_mix        0
opttimeoutlimit              10             10                 0
merge_join                   1              1                  4
hash_join                    0              0                  4
nl_join                      1              1                  4
distinct_sorted              1              1                  4
distinct_sorting             1              1                  4
distinct_hashing             1              1                  4
group_sorted                 1              1                  4
group_hashing                1              1                  4
group_inserting              0              0                  4
order_sorting                1              1                  4
append_union_all             1              1                  4
merge_union_all              1              1                  4
merge_union_distinct         1              1                  4
hash_union_distinct          1              1                  4
store_index                  1              1                  4
bushy_space_search           0              0                  4
parallel_query               1              1                  4
replicated_partition         0              0                  4
ase125_primed                0              0                  4
index_intersection           0              0                  4
index_union                  1              1                  4
multi_table_store_ind        0              0                  4
advanced_aggregation         0              0                  4
opportunistic_distinct_view 1               1                  4
repartition_degree           3              1                  2
scan_parallel_degree         0              1                  2
```

```
resource_granularity        10             10                  2
parallel_degree             0              1                   2
statistics simulate         0              0                   4
forceplan                   0              0                   7
prefetch                    1              1                   6
metrics_capture             0              0                   6
process_limit_action        quiet          quiet               2
plan replace                0              0                   4
plan exists check           0              0                   4
plan dump                   0              0                   4
plan load                   0              0                   4
(39 rows affected)
(return status = 0)
```

- **Example 9 –** Shows a list of the default settings for the Query Tuning category:

```
1> sp_options show, null, dflt
2> go
Category: Query Tuning

name                        defaultsetting
--------------------------- --------------
optgoal                     allrows_mix
opttimeoutlimit             10
merge_join                  1
hash_join                   0
nl_join                     1
distinct_sorted             1
distinct_sorting            1
distinct_hashing            1
group_sorted                1
group_hashing               1
group_inserting             0
order_sorting               1
append_union_all            1
merge_union_all             1
merge_union_distinct        1
hash_union_distinct         1
store_index                 1
bushy_space_search          0
parallel_query              1
replicated_partition        0
ase125_primed               0
index_intersection          0
index_union                 1
multi_table_store_ind       0
advanced_aggregation        0
opportunistic_distinct_view 1
repartition_degree          1
scan_parallel_degree        1
resource_granularity        10
parallel_degree             1
statistics simulate         0
forceplan                   0
prefetch                    1
metrics_capture             0
process_limit_action        quiet
```

```
plan replace                0
plan exists check           0
plan dump                   0
plan load                   0

(39 rows affected)
(return status = 0)
```

- **Example 10 –** Shows the options that are set to a non-default setting in the Query Tuning category:

```
1> sp_options show, null, non_dflt
2> go
```

```
Category: Query Tuning

name                 currentsetting defaultsetting
-------------------- -------------- --------------
repartition_degree   3              1
scan_parallel_degree 0              1
parallel_degree      0              1

(3 rows affected)
(return status = 0)
```

- **Example 11 –** If you enter a parameter that **sp_options** does not understand, you receive the following message:

```
1> sp_options show, "incorrect option"
2> go
```

```
Msg 19615, Level 16, State 1:
Procedure 'sp_options', Line 436:
No option or category matching 'incorrect option' is
found. Valid categories are:
category
------------
Query Tuning
(1 row affected)
(return status = 1)
```

- **Example 12 –** Shows correct usage:

```
1> sp_options help
2> go
```

```
Usage:
sp_options [ [show | help
            [, <option_name>|<category_name>|null
               [, dflt | non_dflt | null
                    [, <spid>] ] ] ] ]
```

Usage

Use **sp_options** to view settings for the following options:

- **set plan dump / load**
- **set plan exists check**
- **set forceplan**
- **set plan optgoal**
- **set [optCriteria]**
- **set plan opttimeoutlimit**
- **set plan replace**
- **set statistics simulate**
- **set metrics_capture**
- **set prefetch**
- **set parallel_degree number**
- **set process_limit_action**
- **set resource_granularity number**
- **set scan_parallel_degree number**
- **set repartition_degree number**

Permissions

Any user can execute **sp_options**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_passthru

(Component Integration Services only) 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 is 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 1024 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

- **Example 1 –** 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*, and *@rowcount* is set to 1:

  ```
  sp_passthru ORACLE, "select date from dual", @errcode output,
      @errmsg output, @rowcount output, @oradate output
  ```

Usage

- **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. See the *Component Integration Services User's Guide* for a more complete discussion of query buffer handling in passthru mode.

Permissions

Any user can execute **sp_passthru**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_autoconnect* on page 70
- *sp_remotesql* on page 603

Return Parameters and sp_passthru

The output parameters *arg1* ... *argn* becomes 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 contains. *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 succeeds. For information on implicit conversion, see *Transact-SQL Functions* in *Reference Manual:*

Building Blocks. See the *Component Integration Services Users Guide* for information on which datatype represents the datatypes from each server class when in passthru mode.

sp_password

Deprecated in SAP ASE version 15.7 and later. To add or change a password for a login account in SAP ASE, use the **create login** and **alter login** commands. See *Reference Manual: Commands*.

sp_passwordpolicy

Allows a user with `sso_role` to configure login and password policy options.

Syntax

To specify, remove, and list new password complexity options:

```
sp_passwordpolicy {“set” | “clear” | “list”}, policy_option,
option_value
```

To verify the password complexity options:

```
sp_passwordpolicy 'validate password options'
```

To generate asymmetric key pairs for network login password encryption:

```
sp_passwordpolicy "regenerate keypair"
```

To expire passwords:

```
sp_passwordpolicy "expire role passwords", "[rolename | wildcard]"
```

```
sp_passwordpolicy "expire login passwords", "[login_name |
wildcard]"
```

```
sp_passwordpolicy "expire stale role passwords", "datetime"
```

```
sp_passwordpolicy "expire stale login passwords", "datetime"
```

To display a brief description of all commands, options, and their values:

```
sp_passwordpolicy "help"
```

Parameters

- **set** – sets a value to an option. When using **set**, you must specify the *policy_option*.
- **clear** – deletes the row for the option specified in the `master.dbo.sysattributes` table. If there is no policy option specified, **clear** deletes all the option rows in the `sysattributes` table. When using **clear**, you must specify the *policy_option*.
- **list** – lists the values of the options specified. When using **list**, you must specify the *policy_option*.

- ***policy_option*, *option_value* –** string or (varchar). Is the option parameter for **set**, **clear**, and **list**, with *option_value* being the their values:
  - **allow password downgrade –** Ends the password downgrade period. During the password downgrade period, passwords are stored in `syslogins` in both old and new encodings to allow user passwords to retained if the server is downgraded, for example, to SAP ASE 15.0.2.
  - **disallow simple passwords –** Value of 1 turns this option on, and a value of 0 turns it off.
  - **enable last login updates –** Enables or disables code in SAP ASE authentication that records the timestamp when each login occurs. The parameter:
    - "**set**" – sets the value of this attribute
    - "**list**" – displays the current value of the attribute
    - "**clear**" deletes the row from `sysattributes`. Although "**clear**" deletes the row from `sysattributes`, the last setting is still effective until you restart the SAP ASE server, or when "**set**" sets the new value.
  - **expire login –** Specifies that when new logins are created or when the SSO changes login passwords, the passwords for those logins are marked as expired, thus forcing those users to change their password when they first log in.
  - **keypair regeneration period –** Indicates the regenerating period of the RSA key pair. Its option values are { **([*keypair regeneration frequency*], *datetime of first generation*) | (*keypair regeneration frequency*, [*datetime of first generation*]) }**
  - ***keypair regeneration frequency* –** Is the frequency of regeneration of an RSA key pair. The valid range of values (in hours) is from 1 to 8,760. The default value is NULL, in which case a key pair is regenerated every 24 hours. It specifies the duration's format specifier, using:
    - **'T*M'** – indicates duration in minutes, replacing the asterisk (*) with a numeric value, such as "T2M" for two minutes.
    - **'H'** – indicates duration in hours.
    - **'D'** – indicates duration in days. This is the default if you do not specify another format.
    - **'W'** – indicates duration in weeks.
    - **'M'** – indicates duration in months.
    - **'Y'** – indicates duration in years.
  - ***datetime of first generation* –** Is the date and time of when the key-pair is first generated. If you specify only the time for the value of *datetime of first generation*, RSA key pair regeneration is scheduled for that time of day in the next 24-hour period. If you:
    - Specify *datetime of first generation* – the SAP ASE server regenerates a new RSA key pair immediately if that time has elapsed; otherwise the SAP ASE server waits until that specified time.

  - Do not specify *datetime of first generation* – the SAP ASE server regenerates a new RSA key pair at a time that is obtained by adding *keypair regeneration period* to the time when the most recent RSA key pair was generated, if this calculated time is not elapsed; otherwise the SAP ASE server regenerates a new RSA key pair immediately.

  Subsequent generations of key pairs occur based on when the most recent key pair was generated and the value of *keypair regeneration period*.

  Note: You cannot simultaneously set the value of *keypair regeneration frequency* and *datetime of first generation* to NULL.

- **keypair error retry [wait | count] –** Specifies the various configurations you can set for regenerating a key pair after a failed attempt:
  - **wait** – specifies the amount of time to wait after a failure before regenerating the keypair.
  - **count** – specifies how many times you want the SAP ASE server to attempt to regenerate a key pair after a failure.
- **maximum failed logins –** Indicates the maximum number of failed logins allowed in a session before the account is locked.
- **min alpha in password –** Indicates the minimum number of alphabetic characters in a password.
- **min digits in password –** Indicates the minimum number of digits to be allowed in a password.
- **min lower char in password –** Indicates the minimum number of lower case characters allowed in a password.
- **min special char in password –** Indicates the minimum number of special characters allowed in a password.
- **min upper char in password –** Indicates the minimum number of uppercase characters allowed in a password.
- **minimum password length –** Indicates the minimum length of the password.
- **password exp warn interval –** Indicates the password expiration warning interval in days.
- **systemwide password expiration –** Indicates the system-wide password expiration in days.
- **unique keypair per session –** Specifies the configurations you can set for generating a unique key pair for every user:
  - **1** – specifies to generate a new key pair for every user connection.
  - **0** – specifies that all connections share the same RSA key pair.

  Note: If **sp_configure "net password encryption reqd"** is configured to "3", this password policy option is ignored because a unique keypair per session is not needed to secure the password.

- **“expire login passwords", "[*login_name* | *wildcard*]" –** expires login passwords, all logins or logins matching a wild card pattern. The column status in `master` database catalog `syslogins` is updated with a status bit LOGIN_EXPIRED (0x4) to indicate the password is expired.
- **"expire role passwords", "[*rolename* | *wildcard*]" –** expires the password of a role, all roles or roles matching a wild-card pattern. The column status in master database catalog syssrvroles is updated with a status bit ROLE_EXPIRED (0x4) to indicate the password is expired:
- **"expire stale login passwords", "*datetime*" –** expires login passwords have not been changed after a datetime specified. The column status in `master` database catalog `syslogins` is updated with a status bit LOGIN_EXPIRED (0x0004) to indicate that the password is expired. See *Entering Date and Time Data*” in *Reference Manual: Building Blocks* for an explanation of how `datetime` values are entered.
- **"expire stale role passwords", "*datetime*" –** expires role passwords have not been changed after a datetime specified. The column status in `master` database catalog `syssrvroles` is updated with a status bit ROLE_EXPIRED (0x4) to indicate the password is expired.
- **"regenerate keypair" –** generates the asymmetric key pairs to be used for network login password encryption.There is no catalog update for this option; the actions occur only in memory fields.
- **'validate password options' –** reports errors or inconsistencies in the password complexity option values set, including length and expiration. The result is reported in a tabular format, with each row representing a validation step, the result of the step, and the validation test performed. The result is one of Pass, Fail, or Not Applicable (NA). If any validation test fails, the return status is set to 1.

Examples

- **Example 1 –** Sets a password expiration warning interval to seven days before the password expires:

  ```
  sp_passwordpolicy 'set',
        'password exp warn interval', '7'
  ```
- **Example 2 –** Lists the option for minimum number of special characters:

  ```
  sp_passwordpolicy 'list',
        'min special char in password'
  ```
- **Example 3 –** Resets **disallow simple passwords** to the default value:

  ```
  sp_passwordpolicy 'clear', 'disallow simple passwords'
  ```
- **Example 4 –** These examples demonstrate using **validate password options**. These outputs have been reformatted for clarity, and do not resemble the output you see on your screen if you execute this procedure

  These password complexity options and their values are stored in the server:

  ```
  minimum password length:       8
  min alpha in password:         2
  ```

```
min digits in password:        2
min upper char in password:    2
min lower char in password:    2
```

To validate these options, enter:

```
sp_passwordpolicy 'validate password options'

Validation Step      Pass/Fail/NA      Validation Test
---------------      ------------
-----------------------------
min alpha in password        Fail     'min alpha in password' > =
'min                                     upper char in password +
'min                                       lower char in password'

minimum password length - 1   Pass     'minimum password length' >
= 'min                                       digits in password' +
'min special                                    char in password' +
'min alpha in                                              password'

minimum password length - 2   Pass     'minimum password length' >
= 'min                                  digits in password' + min
special                                  char in password' + 'min
upper                                   char in password' + 'min
lower                                           char in password'

maximum password length - 1    Pass    'max password length' > =
'min                                 digits in password' + 'min
                              special char in password' + 'min
                                     alpha in password'

maximum password length – 2    Pass    'max password length' > =
'min                                 digits in password' + 'min
special                                 char in password' + 'min
upper                                   char in password' + 'min
lower                                        char in password'
password exp warn interval    NA     'password exp warn interval'
< =                                      'systemwide password
expiration'

(6 rows affected)
(return status = 1)
```

There is one failure: The sum of **min upper char in password + min lower char in password** is greater than the value of **min alpha in password**, so the validation step **min alpha in password** fails.

- **Example 5 –** Sets the HouseKeeper task to automatically regenerate a key pair every two hours, starting on August 15, 2007 at 12:01 a.m.:

```
sp_passwordpolicy "set", "keypair regeneration period",
    "2H", "Aug 15 2007 12:01 AM"
```

- **Example 6 –** Sets how long the SAP ASE server should wait before trying to regenerate the key-pair after a failed attempt:

```
sp_passwordpolicy 'set', 'keypair error retry wait', '10'
```

- **Example 7 –** Sets number of times the SAP ASE server should attempt to regenerate the key-pair after a failure to 5:

  ```
  sp_passwordpolicy 'set', 'keypair error retry count', '5'
  ```

- **Example 8 –** Displays brief description about all commands, options and their values:

  ```
  sp_passwordpolicy "help"
  go
  ```

  ```
  sp_ passwordpolicy Usage: sp_passwordpolicy 'help'
  sp_ passwordpolicy Usage: sp_passwordpolicy command [, option1 [,
  option2 [, option3]]]
  sp_passwordpolicy commands:
  sp_passwordpolicy 'set',
                   {'enable last login updates' | 'disallow simple
  passwords' |
                  'min digits in password' | 'min alpha in password'
  |
                   'min special char in password' | 'min upper char
  in password' |
                  'min lower char in password' | 'password exp warn
  interval' |
                     'systemwide password expiration' | 'minimum
  password length' |
                     'maximum failed logins' | 'expire login' |
                  'allow password downgrade' | 'keypair error retry
  wait' |
                            'keypair error retry count'},
                     'value'
  sp_passwordpolicy 'set', 'keypair regeneration period',
                    {'regeneration_period' |
                     null, 'datetime' |
                     'regeneration_period', 'datetime'}
  sp_passwordpolicy 'list',
                   ['enable last login updates' | 'disallow simple
  passwords' |
                  'min digits in password' | 'min alpha in password'
  |
                   'min special char in password' | 'min upper char
  in password' |
                  'min lower char in password' | 'password exp warn
  interval' |
                     'systemwide password expiration' | 'minimum
  password length' |
                     'maximum failed logins' | 'expire login' |
                     'allow password downgrade' |
                        'keypair error retry wait' | 'keypair error
  retry count' |
                            'keypair regeneration period']
  sp_passwordpolicy 'clear',
                   {'enable last login updates' | 'disallow simple
  passwords' |
                  'min digits in password' | 'min alpha in password'
  |
                   'min special char in password' | 'min upper char
  in password' |
  ```

```
                'min lower char in password' | 'password exp warn
interval' |
                   'systemwide password expiration' | 'minimum
password length' |
                   'maximum failed logins' | 'expire login' |
                         'keypair error retry wait' |
                         'keypair error retry count' | 'keypair
regeneration period'}
sp_passwordpolicy 'expire login passwords'[, '{loginame |
wildcard}']
sp_passwordpolicy 'expire role passwords'[, '{rolename |
wildcard}']
sp_passwordpolicy 'expire stale login passwords', 'datetime'
sp_passwordpolicy 'expire stale role passwords', 'datetime'
sp_passwordpolicy 'regenerate keypair'[, 'datetime']
sp_passwordpolicy 'validate password options'
(return status = 0)
```

- **Example 9 –** Validating the following options stored in the SAP ASE server:

```
minimum password length:        8
min digits in password:         2
min special char in password:   2
min alpha in password:          6
min upper char in password:     3
min lower char in password:     3
```

```
sp_passwordpolicy 'validate password options'
```

```
Validation Step         Pass/Fail/NA   Validation Test
---------------
        ------------  -----------------------------
min alpha in password        Pass     'min alpha in password' > =
'min upper
                                   char in password' + 'min lower
                                       char in password'

minimum password length-1   Fail     'minimum password length' > =
'min
                                      digits in password' + 'min
special
                                 char in password' + 'min alpha in
password'

minimum password length-2   Fail     'minimum password length' > =
'min
                                      digits in password' + 'min
special
                                   char in password' + 'min upper
                                   char in password' + 'min lower
                                       char in password'

maximum password length-1    Pass     'max password length' > =
'min
                                      digits in password' + 'min
special
                                 char in password' + 'min alpha in
```

```
password'

maximum password length-2    Pass     'max password length' > =
'min
                                       digits in password' + 'min
                                 special char in password' + 'min
                                   upper char in password' + 'min
                                        lower char in password'

password exp warn interval   NA      'password exp warn interval'
< =                                       'systemwide password
expiration'

(6 rows affected)
(return status = 1)
```

There are two failures in step 2 and step 3.The sum of **min digits in password**, **min special char in password** and **min alpha in password** is greater than the value of **minimum password length**, so the validation step **minimum password length -1** fails. The sum of **min digits in password**, **min special char in password**, **min upper char in password** and **min lower char in password** is greater than the value of **minimum password length**, so the validation step **minimum password length -2** fails.

- **Example 10 –** Illlustrates the option **'validate password options'**. Output has been reformatted for clarity, and does not resemble the output you see on your screen when you execute this procedure.

  These password complexity options and their values are stored in the server:

```
minimum password length:       8
min alpha in password:         2
min digits in password:        2
min upper char in password:    2
min lower char in password:    2
```

```
sp_passwordpolicy 'validate password options'
```

```
Validation Step             Pass/Fail/NA   Validation Test
---------------
            ------------   --------------------------
min alpha in password      Fail         'min alpha in password' >
= 'min                                     upper char in password
+ 'min                                             lower char
in password'

minimum password length - 1  Pass        'minimum password length'
> = 'min                                      digits in password'
+ 'min special                                            char in
password' + 'min alpha in
                                            password'

minimum password length - 2  Pass           'minimum password
length' > = 'min                                       digits in
password' + min special                                      char
in password' + 'min upper
```

```
                                         char in password' + 'min
lower                                                char in password'

maximum password length - 1  Pass          'max password length' >
= 'min                                        digits in password' +
'min                                               special char in
password' + 'min                                          alpha in
password'

maximum password length – 2  Pass          'max password length' >
= 'min                                        digits in password' +
'min special                                              char in
password' + 'min upper                                       char
in password' + 'min lower
                                             char in password'

password exp warn interval   NA             'password exp warn
interval' < =                                          'systemwide
password expiration'

(6 rows affected)
(return status = 1)
```

There is one failure: the sum of **min upper char in password + min lower char in password** is greater than the value of **min alpha in password**, so the validation step **min alpha in password** fails.

Validating the following options stored in the SAP ASE server:

```
minimum password length:        8
min digits in password:         2
min special char in password:   2
min alpha in password:          6
min upper char in password:     3
min lower char in password: 3
```

```
sp_passwordpolicy 'validate password options'
```

```
Validation Step              Pass/Fail/NA   Validation Test
---------------
             ------------   --------------------------
min alpha in password       Pass        'min alpha in password' >
= 'min upper                                     char in password'
+ 'min lower                                     char in password'

minimum password length-1    Fail       'minimum password length'
> = 'min                                    digits in password' +
'min special                                    char in password'
+ 'min alpha in                                         password'

minimum password length-2    Fail       'minimum password length'
> = 'min                                     digits in password' +
'min special                                             char in
password' + 'min upper                                     char in
password' + 'min lower                                     char in
password'
```

```
maximum password length-1    Pass        'max password length' > =
'min                                         digits in password' +
'min special                                           char in
password' + 'min alpha in
                                            password'

maximum password length-2    Pass        'max password length' > =
'min                                         digits in password' +
'min                                             special char in
password' + 'min                                     upper char in
password' + 'min                                     lower char in
password'

password exp warn interval    NA          'password exp warn
interval' < =                                      'systemwide
password expiration'

(6 rows affected)
(return status = 1)
```

There are two failures in step 2 and step 3.

The sum of **min digits in password**, **min special char in password** and **min alpha in password** is greater than the value of **minimum password length**, so the validation step **minimum password length -1** fails. The sum of **min digits in password**, **min special char in password**, **min upper char in password** and **min lower char in password** is greater than the value of **minimum password length,** so the validation step **minimum password length -2** fails.

Validating the following options stored in the SAP ASE server:

```
minimum password length:      8
min digits in password:       11
min special char in password: 11
min alpha in password:        11
min upper char in password:   1
min lower char in password:   1
```

```
sp_passwordpolicy 'validate password options'
```

```
Validation Step              Pass/Fail/NA   Validation Test
---------------
             ------------   --------------------------
min alpha in password     Pass          'min alpha in password' >
= 'min                                        upper char in
password' + 'min                                  lower char
in password'

minimum password length-1  Fail         'minimum password length'
> = 'min                                    digits in password'
+ 'min                                         special char in
password' + 'min                                      alpha in
password'
```

```
minimum password length-2   Fail         'minimum password length'
> = 'min                                      digits in password' +
'min special                                               char in
password' + 'min upper                                       char in
password' + 'min lower                                       char in
password'

maximum password length-1   Fail         'max password length' > =
'min                                        digits in password' +
'min special                                               char in
password' + 'min alpha in
                                                password'

maximum password length-2   Pass         'max password length' > =
'min                                        digits in password' +
'min special                                     char in password'
+ 'min upper                                     char in password'
+ 'min lower
                                              char in password'

password exp warn interval    NA           'password exp warn
interval' < =                                        'systemwide
password expiration'

(6 rows affected)
(return status = 1)
```

There are three failures, including a serious one, a failure in a test for maximum password length, where the sum of the required password components is greater than the maximum password allowed.

Validating the following options stored in the SAP ASE server:

```
minimum password length:       8
min digits in password:        2
min special char in password:  1
min alpha in password:         4
min upper char in password:    0
min lower char in password:    0
```

```
sp_passwordpolicy 'validate password options'
```

```
Validation Step              Pass/Fail/NA   Validation Test
---------------
             ------------   --------------------------
min alpha in password     Pass           'min alpha in password' >
= 'min                                             upper char in
password' + 'min                                       lower char
in password'

minimum password length-1  Pass              'minimum password
length' > =                                       'min digits in
password' + 'min                                          special
char in password' +                                          'min
alpha in password'
```

```
minimum password length-2  Pass             'minimum password
length' > =                                        'min digits in
password' + 'min                                          special
char in password' +                                          'min
upper char in password' +
                                       'min lower char in password'

maximum password length-1  Pass           'max password length' >
= 'min                                      digits in password' +
'min                                            special char in
password' + 'min                                       'min alpha
in password'

maximum password length-2  Pass          'max password length' > =
'min                                       digits in password' +
'min                                            special char in
password' + 'min                                       upper char
in password' + 'min
                                            lower char in password'

password exp warn interval   NA             'password exp warn
interval' < =                                        'systemwide
password expiration'

(6 rows affected)
(return status = 0)
```

There are no failures with these settings. This reports all 5 rows returned, and a return status of 0.

Usage

sp_passwordpolicy information is stored in the `master.dbo.sysattributes` table.

Once the SAP ASE server has regenerated a new RSA key pair, subsequent generations use a formula of the last time when RSA key pair was generated, combined with the value you specified for *keypair regeneration frequency*.

The value of **keypair regeneration period** is stored in `master..sysattributes` under a new password policy class.

A default value of NULL for the option indicates that this row does not exist in `sysattributes` and the key pair is generated on when the SAP ASE server is restarted, and every 24 hours thereafter.

These two stored procedures do the same thing:

```
sp_passwordpolicy 'set', 'keypair regeneration period', NULL [,
    datetime of first generation]
```

```
sp_passwordpolicy 'regenerate keypair' [, datetime of first
generation]
```

These global variable use the information from **keypair regeneration period**:

- ***@@lastkpgendate*** – reflects the datetime of when the last key pair was generated.
- ***@@nextkpgendate*** – to reflect when the key pair is next generated.

Permissions

The permission checks for **sp_passwordpolicy** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage security configuration` privilege.
Disabled	With granular permissions disabled, you must be a user with **sso_role**.

Auditing

The **set** and **clear** commands in **sp_passwordpolicy** are audited through audit event 115, “Password Administration.”

A audit option “**password**” audits these actions:

- **sp_passwordpolicy 'set', '*option_name*', '*option_value*'**
- **sp_passwordpolicy 'clear', '*option_name*'**
- **sp_passwordpolicy 'expire login passwords'**
- **sp_passwordpolicy 'expire stale login passwords'**
- **sp_passwordpolicy 'regenerate keypair'**
- **sp_passwordpolicy 'expire role passwords'**
- **sp_passwordpolicy 'expire stale role passwords'**

The “**password**” audit option also audits the administration of RSA key pair regeneration period that generates the AUD_EVT_PASSWORD_ADMIN(115) auditing event.

Login Password Complexity Checks and sp_passwordpolicy

These login password complexity checks are extended to role passwords:

- **disallow simple passwords**
- **min digits in password**
- **min alpha in password**
- **min special char in password**
- **min upper char in password**
- **min lower char in password**
- **systemwide password expiration**
- **password exp warn interval**
- **minimum password length**

- **maximum failed logins**
- **expire login**

High-Availability and Password Policy Options

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

- **disallow simple passwords**
- **min digits in password**
- **min alpha in password**
- **min special char in password**
- **min upper char in password**
- **min lower char in password**
- **systemwide password expiration**
- **password exp warn interval**
- **minimum password length**
- **maximum failed login**
- **expire login**
- **keypair regeneration period**
- **keypair error retry wait**
- **keypair error retry count**

The SAP ASE server uses a “**password policy**” quorum attribute to check the inconsistency of any of those values on both the primary and secondary servers, except **keypair regeneration period**, **keypair error retry wait**, and **keypair error retry count**. A high-availability advisory check succeeds when all those value are the same on both servers, and fail when the values differ. For example:

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

```
Attribute Name   Attrib Type  Local Value  Remote Value  Advisory
--------------   -----------  -----------  -----------  ------
expire login     password po   1            0            2
maximum failed   password po   3            5            2
min alpha in pa  assword po   10           12            2
```

A value of 2 set in the `advisory` column of the output indicates that the user cannot proceed with the cluster operation unless the values on both the companions match.

The output of **sp_companion do_advisory** also indicates the inconsistency in any of the particular password policy checks on both servers.

sp_pciconfig

Manages the Java PCI Bridge. Enables or disables arguments and directives, changes configuration values, and reports configuration values.

Note: Do not use **sp_pciconfig** to change arguments or directives unless instructed to do so by SAP Product Support.

Syntax

```
sp_pciconfig {    disable { directive | argument } |
    enable { directive | argument } |
    list { list_type [, formatted ] | units | units, units_type[,
formatted ] } |
    report { directive[, formatted ] |
        directive, args[, formatted ] |
        argument[, formatted ] } |
    update { number_arg, old_value new_value } }
```

Parameters

- **disable –** disables the specified directive or argument.
- ***directive* –** is the name of any valid directive.
- ***argument* –** is the name of any valid argument.
- **enable –** enables a specified directive or argument.
- **list –** lists groups of related arguments as, for example, **sp_pciconfig "list", "directive"** or **sp_pceiconfig "list", "enabled"**. Also, lists all arguments of a specific type as, for example, **sp_pciconfig "list", "units", "switch"**.
- ***list_type* –** specifies a type of list. Values are:
  - **directives** – list of directives
  - **enabled** – list of enabled arguments
  - **disabled** – list of disabled arguments
  - **argnames** – list of argument names
- **formatted –** specifies that displayed list is to be formatted for readability.

  Note: In formatted reports, the process of improving readability may result in the truncation of wide columns. In addition, column headings may be overridden and may not match the actual table column name. Do not format reports if the output is parsed or potential data truncation is not acceptable.
- **units –** when used with **list**, generates a list of *units_type* currently in use.
- **report –** creates a report based on arguments supplied. Usually used to generate a report for an argument to see its current value and whether or not it is enabled. Can also be used to generate a report for a directive or its arguments.

- ***directive* –** specifies all arguments within a specified directive.
- **update –** modifies the numeric value of arguments where units = number. Cannot be used with arguments where units = switch.
- ***number_arg* –** is an argument of **units** = number.
- ***old_value* –** is the current value for *number_arg_name*.
- ***new_value* –** is a new value for *number_arg_name*.

Usage

Enabling and disabling a directive works like a toggle. When a directive is:

- Enabled – the SAP ASE server uses the configured value (enabled or disabled) of each argument. This is the value stored in `sybpcidb`.
- Disabled – the SAP ASE server disregards the configured value (enabled or disabled) of each argument and treats all arguments of the directive as disabled, although the base value of each argument is retained in `sybpcidb`.

Arguments can be individually enabled or disabled. Arguments for **sp_pciconfig** directives are of these types:

- switch – these arguments turn a feature on or off. For example, if the argument for logging is enabled, a log file is generated; if the argument for logging is disabled, no log file is generated.
- string – these arguments are for strings and numbers, which are treated like strings. Enabling a string argument ensures that the SAP ASE server uses the configured value. Disabling a string argument means that the SAP ASE server ignores the configured value and uses the default value. The configured and default values may be the same or different.

Table 14. Configuration Directives for sp_pciconfig

Directive	Description
PCI_BRIDGE_X_OPT	The PCI Bridge configuration parameters
PCI_BRIDGE_LOGOPT	The plug-in **diagserver** report facility
PCI_BRIDGE_INSTR	The PCI Bridge instrumentation settings

Table 15. PCI_BRIDGE_X_OPT Arguments

Argument	Units Type	Default Value	Default State	Description
pci_xopt_maxthreads	number	1056	Enabled	Maximum available PCI Bridge PLB-controlled threads.

Argument	Units Type	Default Value	Default State	Description
pci_xopt_event_scheduling	number	0	Enabled	Default PCI Bridge scheduling.
pci_xopt_failover_engine	number	-1	Enabled	Default engine to which a slot should fail over.
pci_xopt_runtime_alloc_escape	number	1	Enabled	Allow runtime escapes on memory allocation requests above PC Bridge maximum memory allocation unit.
pci_xopt_slotring_cycle	number	-1	Enabled	Disable PCI Bridge slotring washing.
pci_xopt_slotring_wash_th	number	76	Enabled	Default PCI Bridge slotring washing threshold percentage.

Table 16. PCI_BRIDGE_LOGOPT Arguments

Argument	Units Type	Default Value	Default State	Description
pci_logopt_asehi	switch	None	Disabled	PCI Bridge ASE host interface dispath logging.
pci_logopt_jst	switch	None	Disabled	PCI Bridge Job Scheduler task dispatch logging.
pci_logopt_jvm	switch	None	Disabled	PCI Bridge JVM dispatch logging.
pci_logopt_omni	switch	None	Disabled	PCI Bridge OMNI dispatch logging.
pci_logopt_pci	switch	None	Disabled	Generic PCI Bridge logging (probe [pci/pca]).
pci_logopt_runtime	switch	None	Disabled	PCI Bridge runtime dispatch logging.
pci_logopt_xml	switch	None	Disabled	PCI Bridge XML dispatch logging.

Table 17. PCI_BRIDGE_INSTR Arguments

Argument	Units Type	Default Value	Default State	Description
BRIDGE	number	1	Disabled	Forces full instrumentation (noisy).
CELL	number	1	Disabled	Forces all CELL synchronization to Report.
JAVA	number	1	Disabled	Forces all Java-related entries to Report.
JCS	number	1	Disabled	Forces all JCS entries to Report.
JDBC	number	1	Disabled	Forces all JDBC entries to Report.
JVMHOST	number	1	Disabled	Forces all ASE JVM host API entries to Report.
JVMJNI	number	1	Disabled	Forces all JVM JNI external extries to Report.
PCIS	number	1	Disabled	Forces all PCI Service code to Report.
PLB	number	1	Disabled	Forces all PLB code to Report.
SLOTRING	number	1	Disabled	Forces all “slot-ring” code to Report.
SYNC	number	1	Disabled	Forces all SYNChronization code to Report.
TPM	number	1	Disabled	Forces all TPM code to Report.
fetch_classdata	number	1	Enabled	Forces all fetch_classdata hits to Report.
pcis_service	number	2	Disabled	Forces all pcis_service hits to Freeze.

Permissions

The permission checks for **sp_pciconfig** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server configuration` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_jreconfig* on page 454

sp_placeobject

Puts future space allocations for a table or index on a particular segment.

Syntax

```
sp_placeobject segname, objname[, partitionname]
```

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***"
- ***partitionname* –** (optional) is the name of the partition, which allows you to set the segment for a specific partition.

Examples

- **Example 1 –** Places all subsequent space allocation for the table `authors` on the segment named "segment3":

  ```
  sp_placeobject segment3, authors
  ```

- **Example 2 –** Places all subsequent space allocation for the `employee` table's index named `employee_nc` on the segment named `indexes`:

  ```
  sp_placeobject indexes, 'employee.employee_nc'
  ```

- **Example 3 –** Places all subsequent space allocation for the `my_tab` table's segment called `my_seg2` in partition `part1`:

  ```
  sp_placeobject my_seg2, my_tab, part1
  ```

Usage

There are additional considerations when using **sp_placeobject**:

- does not affect the location of any existing table or index data. It affects only future space allocation. This include all existing partitions in the table/index and any new partitions added later if no segment is specified for a new partition. 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**. You can also specify a segment at the partition level as part of a partition definition. Partitions without segment specification uses the segment specified at the table/index level. If no segment is specified for the table/index level, 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.

See also **alter table**, **dbcc** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_placeobject** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the table owner or a user with `manage database` privilege.
Disabled	With granular permissions disabled, you must be the database owner, table owner, or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addsegment* on page 43
- *sp_dropsegment* on page 287
- *sp_extendsegment* on page 330
- *sp_helpindex* on page 409
- *sp_helpsegment* on page 430

sp_plan_dbccdb

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`.

Examples

- **Example 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:

  ```
  sp_plan_dbccdb master
  ```

  ```
  Recommended size for dbccdb database is 50MB (data = 48MB, log =
  2MB).
  dbccdb database already exists with size 280MB.
  Recommended values for workspace size, cache size and process
  count are:
  dbname                 scan ws  text ws  cache  comp mem  process
  count

  master                  128K     48K      640K   0K        1
  ```

- **Example 2 –** Returns configuration recommendations for creating a dbccdb database suitable for checking all databases in the server. The output includes Compresssion Memory Requirement, which has a non-zero value only for archive databases using any compressed device. No dbccdb database existed at the time this command was run:

  ```
  sp_plan_dbccdb
  ```

  ```
  Recommended size for dbccdb database is 50MB (data = 48MB, log =
  2MB).
  dbccdb database already exists with size 280MB.
  Recommended values for workspace size, cache size and process
  count are:
  dbname                 scan ws  text ws  cache  comp mem  process
  count

  master                  128K     48K      640K   0K        1
  tempdb                  656K     176K     1280K  0K        2
  model                   64K      48K      640K   0K        1
  sybsystemdb             64K      48K      640K   0K        1
  sybsystemprocs          1488K    384K     640K   0K        1
  sybsecurity             272K     80K      1280K  0K        2
  adb                     80K      64K      1920K  12M       3
  ```

- **Example 3 –** Returns configuration recommendations for creating a dbccdb database suitable for checking pubs2:

  ```
  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
```

Usage

There are additional considerations when using **sp_plan_dbccdb**:

- **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**.

See also **dbcc** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_plan_dbccdb** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, any user may execute the procedure.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *Chapter 4, dbcc Stored Procedures* on page 785
- *sp_dbcc_evaluatedb* on page 795

sp_poolconfig

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_pool K"], instance instance_name]
```

To change a pool's wash size:

```
sp_poolconfig cache_name, "affected_poolK", "wash=size[P|K|M|G]"
```

To change a pool's asynchronous prefetch percentage:

```
sp_poolconfig cache_name, "affected_poolK",
    "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 with the specified I/O size. The minimum size of a pool is 256 logical server pages. For a 2K logical page size server, the minimum size is 256K. 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 multiples of the logical page size, up to four times the amount.
- ***affected_pool* –** is the size of I/O performed in the memory pool where the memory is to be deallocated, or the pools attributes such as **'wash size'** and **'prefetch limit'** are to be

modified. If *affected_pool* is not specified, the memory is taken from the lowest logical page size memory pool.

- ***instance_name*** **–** (Cluster Edition) is the name of the instance with the buffer pool you are adjusting.
- **wash=*size*** **–** Changes the wash size (the point in the cache at which the SAP ASE 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. Valid values are 0–100. Setting the prefetch limit to 0 disables asynchronous prefetching in a pool.

Examples

- **Example 1 –** Creates a 16K pool in the data cache `pub_cache` with 10MB of space. All space is taken from the default 2K memory pool:

```
sp_poolconfig pub_cache, "10M", "16K"
```

- **Example 2 –** Creates 16MB of space to the 32K pool from the 64K pool of `pub_cache`:

```
sp_poolconfig pub_cache, "16M", "32K", "64K"
```

- **Example 3 –** Reports the current configuration of `pub_cache`:

```
sp_poolconfig "pub_cache"
```

- **Example 4 –** Removes the 16K memory pool from `pub_cache`, placing all of the memory assigned to it in the 2K pool:

```
sp_poolconfig pub_cache, "0K", "16K"
```

- **Example 5 –** Changes the wash size of the 2K pool in `pubs_cache` to 508K:

```
sp_poolconfig pub_cache, "2K", "wash=508K"
```

- **Example 6 –** Changes the asynchronous prefetch limit for the 2K pool to 15 percent:

```
sp_poolconfig pub_cache, "2K", "local async prefetch limit=15"
```

- **Example 7 –** (Cluster environment) Creates a a 16KB buffer pool of size 25MB in the default data cache on instance blade1:

```
sp_poolconfig 'default data cache', '25M', '16K', 'instance
blade1'
```

- **Example 8 –** (Cluster environment) Displays the buffer pool configuration in the default data cache on instance blade1:

```
sp_poolconfig 'default data cache', 'instance blade1'
```

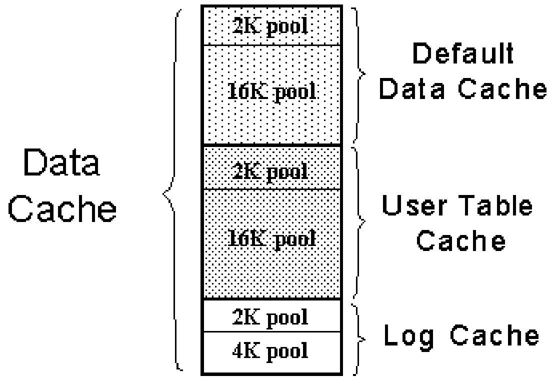
- **Example 9 –** (Cluster environment) Displays the buffer pool configuration for named cache `c_log` on all instances in the cluster:

```
sp_poolconfig c_log
```

Usage

- When you create a data cache with **sp_cacheconfig**, all space is allocated to the logical page size 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, the SAP ASE server performs I/O in logical page size 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 for a 2K logical page size server.
- The combination of cache name and I/O size must be unique. In other words, you can specify only one pool of a given I/O size in a particular data cache in **sp_poolconfig** commands.
- 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.
- The following figure shows a data cache on a server that uses 2K logical pages 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: Data Cache with Default and User-Defined Caches



- You can create pools with I/O sizes up to 16K in the default data cache for a 2K page size server.
- The minimum size of a memory pool is 256 logical pages (for example, a 2K logical page size server, the minimum size is 512K). You cannot reduce the size of any memory pool in any cache to less than 256 pages 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 is also deleted or be changed to the desired size when the server is restarted.
- You can create memory pools while the SAP ASE server is active; no restart is needed for them to take effect. However, the SAP ASE 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, the SAP ASE 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 the SAP ASE server, all pools are created at the configured size.
- Some **dbcc** commands and **drop table** perform only logical page size I/O. **dbcc checkstorage** can perform large I/O, and **dbcc checkdb** performs large I/O on tables and logical page size I/O on indexes.
- Most SAP ASE servers perform best with I/O configured for transactions logs that is twice the logical page size. The SAP ASE server uses the default I/O size of twice the logical page size if the default cache or a cache with a transaction log bound to it is configured with a memory pool twice the logical page size. Otherwise, it uses the logical page size 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, the SAP ASE server uses the logical page size memory pools.

Permissions

The permission checks for **sp_poolconfig** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage data cache` privilege to reconfigure memory pools. Any user can execute **sp_poolconfig** to retrieve information about memory pools.
Disabled	With granular permissions disabled, you must be a user with **sa_role** to reconfigure memory pools. Any user can execute **sp_poolconfig** to retrieve information about memory pools.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_cacheconfig* on page 90
- *sp_helpcache* on page 379
- *sp_logiosize* on page 499
- *sp_unbindcache* on page 710
- *sp_unbindcache_all* on page 713

Wash Percentage and sp_poolconfig

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, the SAP ASE 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 the SAP ASE 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. Run **sp_sysmon** to look for recommendations, or contact Sybase Technical Support for more information about choosing an effective wash size.

Local Asynchronous Prefetch Percentage and sp_poolconfig

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.

See the *Performance and Tuning Guide* for information on the performance impact of changes to the asynchronous prefetch limit.

sp_post_xpload

Checks and rebuilds indexes after a cross-platform **load database** where the endian types are different.

Syntax

```
sp_post_xpload [force]
```

Parameters

- **force –** when specified, uses **reindex_opt_force** for **dbcc reindex** in **sp_post_xpload**.

Examples

- **Example 1 –** Once the database is loaded from another platform, rebuilds its indexes by executing:

  ```
  sp_post_xpload
  ```

Usage

- The following indexes are rebuilt on all user tables in the database:
  - Nonclustered index on an APL table
  - Clustered index on a DOL table
  - Nonclustered index on a DOL table
- Indexes on system tables are not processed with **sp_post_xpload** only. System table indexes are rebuilt when **online database** is executed.
- You can also rebuild indexes using **drop index** and **create index**.
- Run **sp_post_xload** only when the database is loaded across platforms with different endian types.
- Where the index status is suspect, reset the index by executing **sp_post_xpload, drop index**, or **create index**.

- Stored procedures are recompiled from the SQL text in `syscomments` at the first execution after the **load database**. Use **dbcc upgrade_object** to upgrade objects if you do not have permission to recompile from text.

See also **dump database**, **load database** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_post_xload** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `load database` privilege or `own database` privilege on the database.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

Handling Suspect Partitions in Cross-Platform Dump and Load Operations

During the first **online database** command, after you execute **load database** across two platforms with different endian types, the hash partition is marked suspect.

Any global clustered index on a round-robin partition, which has an internally generated partition condition with a `unichar` or `univarchar` partition key, is marked suspect.

After the database is online, use **sp_post_xpload** to fix the suspect partitions and indexes.

sp_primarykey

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

- **Example 1 –** Defines the `au_id` field as the primary key of the table `authors`:

  ```
  sp_primarykey authors, au_id
  ```

- **Example 2 –** Defines the combination of the fields `lastname` and `firstname` as the primary key of the table `employees`:

  ```
  sp_primarykey employees, lastname, firstname
  ```

Usage

There are additional considerations when using **sp_primarykey**:

- 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.

See also **alter table**, **create table**, **create trigger** in *Reference Manual: Commands*.

Permissions

You must be the table owner to execute **sp_primarykey**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_commonkey* on page 157
- *sp_dropkey* on page 275
- *sp_foreignkey* on page 349
- *sp_helpjoins* on page 415
- *sp_helpkey* on page 417

sp_processmail

(Windows only) Reads, processes, sends, and deletes messages in the SAP ASE 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 SAP ASE 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 SAP ASE 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

- **Example 1 –** Processes all unread messages in the SAP ASE 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:

```
sp_processmail @subject="SQL REPORT", @originator="janet",
@dbuser="sa",
    @dbname="salesdb", @filetype="res", @separator=";"
```

- **Example 2 –** Processes all unread messages in the SAP ASE 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:

```
sp_processmail @dbuser="sa"
```

Usage

There are additional considerations when using **sp_processmail**:

- **sp_processmail** reads, processes, sends, and deletes messages in the SAP ASE 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 SAP ASE queries. It reads messages from the SAP ASE 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 SAP ASE 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 SAP ASE message inbox.
- **sp_processmail** does not process attachments to incoming mail. The query must be in the body of the incoming message.

See also **isql** in the *Utility Guide*.

Permissions

You must have `execute` permission to run **sp_processmail**. The permission can be granted to other users by the database owner of `sybsystemprocs`.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *xp_deletemail* on page 773
- *xp_findnextmsg* on page 775
- *xp_readmail* on page 777
- *xp_sendmail* on page 779
- *xp_startmail* on page 782

sp_procxmode

Displays or changes the execution modes associated with stored procedures.

Syntax

```
sp_procxmode [procname [, tranmode]]
```

Parameters

- ***procname* –** is the name of the stored procedure with the transaction mode you are examining or changing.
- ***tranmode* –** is the new execution mode for the stored procedure. Values are **"chained"**, **"unchained"**, and **"anymode"**, **for transaction modes**, and **'[No] Dynamic Ownership Chain'**.

Examples

- **Example 1 –** Displays the transaction mode for all stored procedures in the current database:

  ```
  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
  ```
- **Example 2 –** Displays the transaction mode of the stored procedure `byroyalty`:

  ```
  sp_procxmode byroyalty
  ```

  ```
  procedure name                  transaction mode
  ------------------------------  ----------------
  byroyalty                       Unchained
  ```
- **Example 3 –** Changes the transaction mode for the stored procedure `byroyalty` in the `pubs2` database from “unchained” to “chained”:

  ```
  sp_procxmode byroyalty, "chained"
  ```

Usage

There are additional considerations when using **sp_procxmode**:

- 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, the SAP ASE 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, the SAP ASE server returns a warning message, but the current transaction, if any, is not affected.
- Executing **sp_procxmode procname, 'Dynamic Ownership Chain'** makes sure that any **Dynamic SQL (execute immediate)** statements within the stored procedure get their permissions checked against the procedure creator.
- Executing **sp_procxmode procname, 'No Dynamic Ownership Chain'** (the default behaviour if omitted) makes sure that any **Dynamic SQL (execute immediate)** statements within the stored procedure get their permissions checked against the procedure executor.

See also:

- **begin transaction**, **commit**, **save transaction**, **set** in *Reference Manual: Commands*

Permissions

The permission checks for **sp_procxmode** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the owner of the procedure or a user with `manage database` privilege. Any user can execute **sp_procxmode** to for its own procedure.

Setting	Description
Disabled	With granular permissions disabled, you must be the database owner, the owner of the procedure, or a user with **sa_role**. Any user can execute **sp_procxmode** to display the transaction mode.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_querysmobj

(Only when the TSM is licensed at your site) Queries the Tivoli Storage Manager (TSM) for a list of the SAP ASE backup objects.

Syntax

```
sp_querysmobj "syb_tsm", "output_file", "server_name"
    {, "database_name", "object_name", "dump_type",
    "until_time", "bs_name"}
```

Parameters

- **syb_tsm –** is the keyword that invokes the libsyb_tsm.so module that enables communication with TSM.
- ***output_file* –** is the file to which Backup Server writes the list of TSM backup objects.
- ***server_name* –** is the name of the SAP ASE server associated with the TSM backup objects.
- ***database_name* –** is the name of the database associated with the TSM backup objects. An asterisk (*) indicates all databases.

- ***object_name*** **–** is the name of theTSM backup object as provided in the **dump database** or **dump transaction** command. If this parameter is omitted, all backup objects are queried. An asterisk (*) indicates all backup objects.
- ***dump_type*** **–** is the backup object type to be queried. Valid values are:
    - **DB** – database backup objects created by the **dump database** command.
    - **XACT** – database backup objects created by the **dump transaction** command.
    - ***** – all database backup objects. This is the default.
- **until_time –** is the date timestamp. All backup objects matching the criteria entered in **sp_querysmobj** before the specified time are queried. If you omit this parameter, all backup objects matching the specified criteria are queried.
- ***bs_name*** **–** is the name of the remote Backup Server. If *bs_name* is omitted, the default, SYB_BACKUP, is used.

Examples

- **Example 1 –** Queries all TSM backup objects for the SAP ASE "demo_svr1" and writes the list to `/tmp/qtsm/5_1.out`.

  ```
  sp_querysmobj "syb_tsm", "/tmp/qtsm/5_1.out", "demo_srv1"
  ```
- **Example 2 –** Queries all TSM backup objects for the SAP ASE "demo_svr1" and the database `pubs2` and writes the list to `/tmp/qtsm/5_2.out`.

  ```
  sp_querysmobj "syb_tsm", "/tmp/qtsm/5_2.out", "demo_srv1",
  "pubs2"
  ```
- **Example 3 –** Queries all TSM database backup objects for the SAP ASE "demo_svr1" and the database `pubs2` and writes the list to `/tmp/qtsm/5_3.out`.

  ```
  sp_querysmobj "syb_tsm", "/tmp/qtsm/5_3.out", "demo_srv1",
  "pubs2", "*", "DB"
  ```

Usage

See also *Using Backup Server with IBM Tivoli Storage Manager*.

Permissions

The permission checks for **sp_querysmobj** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must have `dump any database` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_deletesmobj* on page 215

sp_recompile

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

- **Example 1 –** Recompiles each trigger and stored procedure that uses the table `titles` the next time the trigger or stored procedure is run:

  ```
  sp_recompile titles
  ```

Usage

There are additional considerations when using **sp_recompile**:

- Compilation involves the optimizer creating a query plan that is stored in procedure cache from the normalized query tree stored in `sysprocedures`. This occurs whenever a procedure or trigger is executed and no free plan for it is found in procedure cache. As you add indexes or make other changes to your database that affect its statistics, these query plans may lose efficiency. By recompiling the stored procedures and triggers that act on a table, you can optimize the queries for maximum efficiency.

  Note: Do not run **sp_recompile** when executing **create index** or **update statistics**. These commands results in minor schema changes, which then automatically recompile stored procedures and triggers that reference the target table on next execution.
- **sp_recompile** looks for *objname* only in the current database. Running it causes triggers and stored procedures that reference *objname* to recompile the next time they are executed.
- You cannot use **sp_recompile** on system tables.
- In SAP ASE versions 12.5 and earlier, **sp_recompile** could influence adhoc queries that you execute. The SAP ASE server would return a schema change error (error number 540), and abort the adhoc query. **sp_recompile** no longer affects such adhoc queries, and you no longer see error 540.

See also:

- **create index**, **update statistics** in *Reference Manual: Commands*

Permissions

Any user can execute **sp_recompile**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_refit_admin

(Cluster environments only) Provides an interface to perform various disk refit-related actions, such as showing the current status of the disk refit process, resetting the state of the **disk refit** process, skipping the disk refit process for an instance, and so on.

Syntax

```
sp_refit_admin [‘help’] | 'status' | ['reset’ | 'skiperfit' [, 
instance_name]]
    |[ 'removedevice’, device_name]
```

Parameters

- **help –** displays information on **sp_refit_admin** syntax and usage.
- **status –** displays the current status of the disk refit process. It lists all the instances and their private devices for which disk refit is still pending. If no such device exists, it prints a message saying so.
- **reset –** resets the state of the disk refit process. It takes an optional parameter *instance_name*.

  If *instance_name* is not supplied, this parameter resets the disk refit process back to the beginning of Phase One, so that subsequent **disk refit** command starts the disk refit process from Phase One and refits all the regular shareable devices, as well as private devices of the instance.

  If *instance_name* is supplied, this parameter resets the disk refit process back to the beginning of Phase Two for that instance, so that a subsequent **disk refit** command on that instance starts the disk refit process from Phase Two for that instance, and refits only the private devices of that instance.
- **skiprefit –** skips running Phase Two of the disk refit process for one or all instances in the cluster without dropping the device. This parameter is meaningful only after the completion of Phase One of the disk refit process. It takes **instance_name** as an optional parameter.
- **removedevice –** removes a device from the disk refit process. This parameter requires the name of the device that is to be removed, as the input parameter *device_name* or *instance_name*.

Examples

- **Example 1 –** Resets the state of the disk refit process to the start of Phase One:

  ```
  sp_refit_admit 'reset'
  ```

  After executing **reset**, the user must run Phase One and Phase Two of the disk refit process.

- **Example 2 –** Resets the state of the disk refit process on the instance named 'cluster1_instance1' to the start of Phase Two for the instance:

  ```
  sp_refit_admin 'reset', 'cluster1_instance1'
  ```

  This interface removes `sysdatabases` entry for all the databases created on the private devices owned by 'cluster1_instance1', and the `sysusages` entries corresponding to the private devices owned by 'cluster1_instance1'. After executing, you must run Phase Two of **disk refit** on 'cluster1_instance1'.

- **Example 3 –** Skips the disk refit process of all the refit-pending private devices of instance `'cluster1_instance1'`:

  ```
  sp_refit_admin 'skiprefit', 'cluster1_instance1'
  ```

  This example removes the `sysdatabases` entry for all the databases that use any of the refit-pending private devices owned by `'cluster1_instance1'`, and removes all the entries in `sysusages` for all the deleted databases.

  To skip the disk refit process on all the refit-pending private devices of all the instances in the cluster, enter:

  ```
  sp_refit_admin 'skiprefit'
  ```

- **Example 4 –** To remove the device "device1' from the disk refit process:

  ```
  sp_refit_admin 'removedevice', 'device1'
  ```

  This action removes the `sysdatabases` entry for all databases created on `'device1'`, and all the sysusages entries corresponding to `'device1'`. It also removes `'device1'` from `sysdevices`.

Usage

There are additional considerations when using **sp_refit_admin**:

- You must follow the instructions in *Troubleshooting* in the *Clusters Users Guide* after executing **skiprefit**, to ensure the consistency of the system tables before resuming normal operation.
- Use **removedevice** only during the disk refit process, to remove the device from the refit process. Do not use it in place of **sp_dropdevice**
- You can use **sp_refit_admin** even when the instance is started with the `-m` option and trace flag 3608 ON.

For information on problems encountered with **disk refit**, see the *Troubleshooting and Error Guide*

Permissions

The permission checks for **sp_refit_admin** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage disk` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

sp_remoteoption

Displays or changes remote login options.

Syntax

```
sp_remoteoption [remoteserver[, loginname
    [, remotename[, optname[, optvalue]]]]]
```

Parameters

- ***remoteserver*** **–** is the name of the server that executes 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."
- ***loginname*** **–** is the login name that identifies the local login for the ***remoteserver***, ***loginname***, ***remotename*** combination.
- ***remotename*** **–** is the remote user name that identifies the remote login for the ***remoteserver***, ***loginname***, ***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. The SAP ASE 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

- **Example 1 –** Displays a list of the remote login options:

  ```
  sp_remoteoption
  ```

  ```
  Settable remote login options.
  remotelogin_option
  ------------------------
  trusted
  ```
- **Example 2 –** Defines the remote login from the remote server GATEWAY to be **trusted**; that is, the password is not checked:

  ```
  sp_remoteoption GATEWAY, churchy, pogo, trusted, true
  ```

- **Example 3 –** Defines the remote login "pogo" from the remote server GATEWAY as a login that is not trusted; that is, the password is checked:

  ```
  sp_remoteoption GATEWAY, churchy, pogo, trusted, false
  ```

- **Example 4 –** Defines all logins from GATEWAY that map to login "albert" on the local server to be trusted:

  ```
  sp_remoteoption GATEWAY, albert, NULL, trusted, true
  ```

Usage

There are additional considerations when using **sp_remoteoption**:

- 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.

  See the *System Administration Guide* for more information about setting up servers for remote procedure calls and for using "unified login."

See also **isql** in the *Utility Guide*.

Permissions

The permission checks for **sp_remoteoption** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any remote login` privilege.
Disabled	With granular permissions disabled, you must be a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addremotelogin* on page 40
- *sp_dropremotelogin* on page 281
- *sp_helpremotelogin* on page 424

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.

Syntax

```
sp_remotesql server, query [, query2, ... , query254]
```

Parameters

- ***server* –** is the name of a remote server defined with **sp_addserver**.
- ***query* –** is a query buffer a with maximum length of 255 characters.
- ***query2 … 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

- **Example 1 –** Passes the query buffer to FREDS_SERVER, which interprets **select @@version** and returns the result to the client. The SAP ASE server does not interpret the result:

  ```
  sp_remotesql FREDS_SERVER, "select @@version"
  ```

- **Example 2 –** Uses **sp_remotesql** in a stored procedure. This example and Example 1 return the same information to the client:

  ```
  create procedure freds_version
  as
  exec sp_remotesql FREDS_SERVER, "select @@version"
  go
  exec freds_version
  go
  ```

- **Example 3 –** 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***.

  ```
  sp_remotesql DCO_SERVER,
  "insert into remote_table
  (numbercol,intcol, floatcol,datecol )",
  "values (109.26,75, 100E5,'10-AUG-85')"
  select @@error
  ```

- **Example 4 –** Illustrates the use of local variables as parameters to **sp_remotesql**:

  ```
  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
  ```

Usage

There are additional considerations when using **sp_remotesql**:

- **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 get 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, the SAP ASE 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.

See also **connect to...disconnect** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_remotesql**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addserver* on page 46
- *sp_autoconnect* on page 70
- *sp_passthru* on page 557

sp_rename

Changes the name of a user-created object or user-defined datatype in the current database.

Syntax

```
sp_rename objname, newname [,“index” | “column” | “partition”]
```

Parameters

- ***objname* –** is the original name of the user-created object (table, view, column, partition, 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.
- **"index" –** specifies that the object you are renaming is an index, not a column. This argument allows you to rename an index that has the same name as a column, without dropping and re-creating the index.
- **"column" –** specifies that the object you are renaming is a column, not an index. This argument is part of the same option as the **index** argument.
- **"partition" –** specifies that the object you are renaming is a partition when the table-partition name conflicts with a column or index name.

Examples

- **Example 1 –** Renames the `titles` table to `books`:

  ```
  sp_rename titles, books
  ```

- **Example 2 –** Renames the `title` column in the `books` table to `bookname`:

  ```
  sp_rename "books.title", bookname
  ```

- **Example 3 –** Renames the `titleind` index in the `books` table to `titleindex`:

  ```
  sp_rename "books.titleind", titleindex
  ```

- **Example 4 –** Renames the user-defined datatype `tid` to `bookid`:

  ```
  sp_rename tid, bookid
  ```

- **Example 5 –** Renames the `title_id` index in the `titles` table to `isbn`:

  ```
  sp_rename "titles.title_id", isbn, "index"
  ```

- **Example 6 –** Renames the table index `my_tab.ind1.i_part1` to `i_part1_rename`:

  ```
  sp_rename "my_tab.ind1.i_part1", i_part1_rename
  ```

- **Example 7 –** Renames the index partition `my_tab.ind1.ind1_928003306` to `ind1_928003306_rename` using **"partition"** to avoid conflicts between the table-partition name and index name:

  ```
  sp_rename "my_tab.ind1.ind1_928003306", ind1_928003306_rename,
  "partition"
  ```

Usage

There are additional considerations when using **sp_rename**:

- **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 name in *newname*. See Examples 2, 3, and 5.
- If a column and an index have the same name, use the **[,"*index*" | "*column*"]** argument, which specifies whether to rename the index or the column. In the following sample, assume that both an index and a column named **idx** exist:

  ```
  sp_rename "t.idx", new_idx, "column"
  -------------
  Column name has been changed. (Return status = 0)
  sp_rename "t.idx", new_idx, "index"
  -------------
  Index name has been changed. (Return status = 0)
  ```
- If you change the name of a an object or column name referenced by a view, you see a warning message, such as:

  ```
  Changing an object or column name could break
  existing stored procedures, cached statements or
  other compiled objects.
  ```
- **sp_engine** can run in sessions using chained transaction mode if there are no open transactions.
- You cannot change the names of system objects and system datatypes.

Permissions

You must be the object owner to execute **sp_rename**. Permission checks do not differ based on the granular permissions settings.

Use the **setuser** command to assume another database user's identity to rename objects owned by other users.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_depends* on page 217
- *sp_rename* on page 605

sp_rename_qpgroup

Renames an abstract plan group.

Syntax

```
sp_rename_qpgroup old_name, new_name
```

Parameters

- ***old_name* –** is the current name of the abstract plan group.
- ***new_name* –** is the new name for the group. The specified *new_name* cannot be the name of an existing abstract plan group in the database.

Examples

- **Example 1 –** Changes the name of the group from `dev_plans` to `prod_plans`:

  ```
  sp_rename_qpgroup dev_plans, prod_plans
  ```

Usage

There are additional considerations when using **sp_rename_qpgroup**:

- Use **sp_rename_qpgroup** to rename an abstract plan group. You cannot use the name of an existing plan group for the new name.
- **sp_rename_qpgroup** does not affect the contents of the renamed group. IDs of existing abstract plans are not changed.
- You cannot rename the default abstract plan groups, `ap_stdin` and `ap_stdout`.
- **sp_rename_qpgroup** cannot be run in a transaction.

Permissions

The permission checks for **sp_rename_qpgroup** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_help_qpgroup* on page 370

sp_renamedb

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

- **Example 1 –** Renames the `accounting` database to `financial`:

  ```
  sp_renamedb accounting, financial
  ```

- **Example 2 –** Renames the database named `work`, which is a Transact-SQL reserved word, to `workdb`. This example shows how **sp_dboption** is used to place the `work` database in single-user mode before renaming it and restore it to multi-user mode afterward:

  ```
  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
  ```

Usage

There are additional considerations when using **sp_renamedb**:

- **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(frgndbid)
  from sysreferences
  where frgndbid 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(pmrydbid)
  from sysreferences
  where pmrydbid 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 with a name that 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**.

See also **create database** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_renamedb** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `own database` privilege on the database.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_changedbowner* on page 102
- *sp_dboption* on page 193
- *sp_depends* on page 217
- *sp_helpdb* on page 394
- *sp_rename* on page 605

sp_reportstats

Reports statistics on system usage.

Syntax

```
sp_reportstats [loginame]
```

Parameters

- ***loginame* –** is the login name of the user to show accounting totals for.

Examples

- **Example 1 –** Displays a report of current accounting totals for all SAP ASE users:

  ```
  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
  ```

- **Example 2 –** Displays a report of current accounting totals for user “kathy”:

```
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
```

Usage

There are additional considerations when using **sp_reportstats**:

- **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.
- The units reported for "CPU" are SAP ASE clock ticks.
- The "probe" user exists for the two-phase commit probe process, which uses a challenge-and-response mechanism to access the SAP ASE server.

Permissions

The permission checks for **sp_reportstats** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_clearstats* on page 129
- *sp_configure* on page 167

sp_restore_system_role

Restores the system defined role or database owner to the default role privilege configuration.

Syntax

```
sp_restore_system_role [role_name [, all_dbs]]
```

Parameters

- ***role_name*** **–** One of sa_role, sso_role, oper_role, replication_role, keycustodian_role, sa_serverprivs_role, and dbo. A usage message is displayed if no parameter is specified.
- **all_dbs –** Restores the database owner or the role to the default role privilege configuration in all online databases. If **all_dbs** is not specified, only perform the change in the current database.

Examples

- **Example 1 –** Restore **sso_role** to the default role privilege configuration in all databases.

  ```
  sp_restore_system_role sso_role, all_dbs
  ```

- **Example 2 –** Restores **sa_role** to the default role privilege configuration in `db1` only.

  ```
  use db1
  ```

  ```
  sp_restore_system_role sa_role
  ```

- **Example 3 –** Restore dbo to the default privilege configuration in master.

  ```
  use master
  ```

  ```
  sp_restore_system_role dbo
  ```

Usage

There are additional considerations when using **sp_restore_system_role**:

- **sp_restore_system_role** restores a system-defined role, user-defined role sa_serverprivs_role, or database owner to the default role privilege configuration. The allowed system-defined roles include: **sa_role**, **sso_role**, **oper_role**, **replication_role**, and **keycustodian_role**. For the list of privileges granted to the above roles or database owner in the default role privilege configuration, see *Using Granular Permissions* in the *Security Administration Guide*.
- When you specify **all_dbs**, the restoration operation does not apply to `sybsecurity` database. You need to manually restore privileges of the role or database owner in `sybsecurity` if needed.

Permissions

The permission checks for **sp_restore_system_role** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server permissions` privileges to restore **sa_role**, and a user with `manage security permissions` to restore other roles or the database owner. To use **all_dbs** option, you also need to have `use any database` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role** to restore **sa_role**, and a user with **sso_role** to restore other roles and the database owner.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	81
Audit option	If **dbcc** auditing is on
Command or access audited	Execution of a **dbcc** command
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – **upgd_grantrev_sysrole_perms** • *Previous value* – NULL • *Current value* – NULL • *Other information* – parameter list • *Proxy information* – Original login name, if **set proxy** in effect

sp_revokelogin

(Windows only) Revokes SAP ASE roles and default permissions from Windows 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 user.
- ***group_name* –** is the Windows group name.

Examples

- **Example 1 –** Revokes all permissions from the Windows user named “jeanluc”:

  ```
  sp_revokelogin jeanluc
  ```

- **Example 2 –** Revokes all roles from the Windows Administrators group:

  ```
  sp_revokelogin Administrators
  ```

Usage

Use **sp_revokelogin** only when the SAP ASE server is running in Integrated Security mode or Mixed mode, when the connection is Named Pipes. If the SAP ASE 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 the SAP ASE server over a trusted connection.

See also **grant**, **revoke**, **setuser** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_revokelogin** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage roles` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_droplogin* on page 278
- *sp_dropuser* on page 293
- *sp_logininfo* on page 497

sp_role

This system procedure is deprecated by SAP ASE 15.7 and higher. To grant or revoke roles, use the **greant role** or **revoke role** commands.

See also

- *sp_activeroles* on page 4
- *sp_displayroles* on page 240

sp_securityprofile

Lists the attributes or bindings associated with a login profile.

Syntax

```
sp_securityprofile 'attributes','login profile',
    {wildcard | login_profile_name | 'default'}
```

```
sp_securityprofile 'bindings', 'login profile'
    [, {wildcard | login_profile_name | 'default'}
    [, 'login' ,{wildcard | login_name}]]
```

```
sp_securityprofile ‘help’
```

Parameters

- **attributes –** specifies to list attributes associated with a login profile.
- **login profile –** specifies to obtain information about login profiles.
- **bindings –** when **login** is specified, list binding of login accounts. When **login profile** is specified, list bindings of login profiles.
- **login –** specifies to obtain information about login accounts.
- ***wildcard* | *login_profile_name* | default –** specifies the login profile in which to obtain information. Options include a specific a name of a login profile, the default login profile, or wildcard characters can be used identify login profiles.
- ***wildcard* | *login_name* –** specifies to use a specific login account name or allows the use of wildcard characters to identify login accounts.
- **help –** displays usage.

Examples

- **Example 1 –** Lists all attributes of the default login profile.

  ```
  sp_securityprofile 'attributes', 'login profile',
      'default'
  ```

  ```
  Name                            Value
  ------------                    ----------------
  login profile                   def_login_profile
  default                         yes
  default database                master
  default language                NULL
  login script                    NULL
  auto activated roles            emp_role
  auto activated roles            def_role
  manually activated roles        special_role
  authenticate with               ANY
  track lastlogin                 TRUE
  stale period                    180D
  ```

- **Example 2 –** Displays all the attributes associated with all login profiles.

  ```
  sp_securityprofile 'attributes', 'login profile', '%'
  ```

  ```
  Name                            Value
  ----------                      ----------------
  login profile                   def_login_profile
  default                         yes
  default database                master
  default language                NULL
  login script                    NULL
  auto activated roles            emp_role
  auto activated roles            def_role
  authenticate with               ANY
  track lastlogin                 TRUE
  stale period                    180D
  ```

```
Name                           Value
-----------                    ----------------
login profile                  eng_login_profile
default
default database               work
login script                   engr_script
auto activated roles           emp_role
auto activated roles           def_role
auto activated roles           engr_role
authenticate with              LDAP

Name                           Value
------------                   ----------------
login profile                  mgr_login_profile
default
default database               work
login script                   mgr_script
auto activated roles           emp_role
auto activated roles           def_role
auto activated roles           mgr_role
manually activated roles       activate_emp_role
authenticate with              LDAP

Name                           Value
-------------                  ----------------
login profile                  sa_login_profile
manually activated roles       admin_role
default
```

- **Example 3 –** Displays all login accounts associated with a specific login profile.

```
sp_securityprofile 'bindings', 'login profile',
    'engr_login_profile'
```

```
Login name          Login profile name
-----------         ----------------
anderson            eng_login_profile
gupta               eng_login_profile
lchang              eng_login_profile
tsato               eng_login_profile
```

- **Example 4 –** Displays the login profile for the login account named `sa`.

```
sp_securityprofile 'bindings', 'login profile', null,
    'login', 'sa'
```

```
Login name          Login profile name
-----------         ----------------
sa                  sa_login_profile
```

Usage

Precedence rules are followed for attributes no set in profiles.

See also:

- **create login profile**, **alter login profile** in *Reference Manual: Commands*
- *Applying Login Profile and Password Policy Attributes* in the *Security Administration Guide*
- **sp_displaylogin**

Permissions

The permission checks for **sp_securityprofile** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any login profile` privilege.
Disabled	With granular permissions disabled, you must be a user with **sso_role** to see attributes and bindings of all login profiles. For a non-privileged login account: • You can only see the attributes of a login profile associated with the login (either directly or the default login profile). • You cannot see the bindings of a login profile with login accounts.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_sendmsg

(UNIX only) Sends a message to a User Datagram Protocol (UDP) port.

Syntax

```
sp_sendmsg ip_address, port_number, message
```

Parameters

- ***ip_address* –** is the IP address of the machine where the UDP application is running.
- ***port_number* –** is the port number of the UDP port.
- ***message* –** is the message to send, up to 4096 characters in length.

Examples

- **Example 1 –** Sends the message "Hello World" to IP address 120.10.20.5 using port 3456:

  ```
  sp_sendmsg "120.10.20.5", 3456, "Hello World"
  ```

Usage

There are additional considerations when using **sp_sendmsg**:

- To enable the use of UDP messaging, a system security officer must set the configuration parameter **allow sendmsg** to 1.
- No security checks are performed with **sp_sendmsg**. Be very cautious when using **sp_sendmsg** to send sensitive information across the network. By enabling this functionality, the user accepts any security problems that result from its use.
- This sample C program listens on a port that you specify and echoes the messages it receives. For example, to receive the **sp_sendmsg** calls for Example 1, use:

  ```
  updmon 3456
  #include <stdlib.h>
  #include <stdio.h>
  #include <sys/types.h>
  #include <sys/socket.h>
  #include <netinet/in.h>
  #include <arpa/inet.h>
  #include <unistd.h>
  #include <fcntl.h>

  main(argc, argv)
  int argc; char *argv[];
  {
          struct sockaddr_in sadr;
          int portnum,sck,dummy,msglen;
          char msg[256];
  ```

```
        if (argc < 2) {
                printf(“Usage: udpmon <udp portnum>\n”);
                exit(1);
        }

        if ((portnum=atoi(argv[1])) < 1) {
                printf("Invalid udp portnum\n");
                exit(1);
        }

        if ((sck=socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP)) < 0) {
                printf("Couldn't create socket\n");
                exit(1);
        }

        sadr.sin_family = AF_INET;
        sadr.sin_addr.s_addr = inet_addr("0.0.0.0");
        sadr.sin_port = portnum;

        if (bind(sck,&sadr,sizeof(sadr)) < 0) {
                printf("Couldn't bind requested udp port\n");
                exit(1);
        }

        for (;;)
        {
                if((msglen=recvfrom(sck,msg,sizeof(msg),
0,NULL,&dummy)) < 0)
                        printf("Couldn't recvfrom() from udp port
\n");
                printf("%.*s\n", msglen, msg);
        }
}
```

See also:

- **syb_sendmsg** in *Reference Manual: Building Blocks*

Permissions

Any user can execute **sp_sendmsg**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**

Information	Values
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_serveroption

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. The following table lists the option names.
  - **mutual authentication –** Sets mutual authentication for all connections to the remote server using Kerberos authentication.
  - **external engine auto start –** Specifies that EJB Server starts up each time the SAP ASE server starts up. The default is **true**; starting the SAP ASE server also starts up EJB Server.
  - **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.
  - **net password encryption reqd –** The SAP ASE server allows the use of asymmetric encryption to securely transmit passwords from client to server using the RSA public key encryption algorithm. The SAP ASE server generates the asymmetric key pair and sends the public key to clients that use the new login protocol. The client encrypts the user's login password with the public key before sending it to the server. The server, decrypts the password with the private key to begin the authentication of the client connecting.

    Configures the SAP ASE server to require clients to use this protocol. Set the SAP ASE configuration parameter **net password encryption reqd** to require all username- and password-based authentication requests to use RSA asymmetric encryption. The valid values for **net password encryption reqd** are:

  - 0 – Allows the client to choose the encryption algorithm used for login passwords on the network, including no password encryption. This is the default value for this configuration parameter and provides functionality most similar to earlier releases. This allows the choice of network password encryption to be established by the client application.
  - 1 – Restricts clients to use either RSA or Sybase proprietary encryption algorithms to encrypt login passwords on the network. This provides an incrementally restrictive setting that allows older clients to connect with the Sybase proprietary algorithm and new clients to connect with the stronger RSA algorithm. A client that attempts to connect without using password encryption fails.
  - 2 – Restricts clients to use only the RSA encryption algorithms to encrypt login passwords on the network. This provides strong RSA encryption of passwords and requires use of newer clients. A client that attempts to connect without using the RSA encryption fails.

- **allow password downgrade –**
- **readonly –** (Component Integration Services only) Specifies that access to the server named is read only.
- **security mechanism –** This option specifies the security mechanism for the remote server. Enables Kerberos authentication for connections to the remote server when your login is authenticated using the Kerberos mechanism.
- **server cost –** (Component Integration Services only) Specifies the cost of a single exchange under the user's control, on a per-server basis. See *Understanding Component Integration Services* in *Understanding CIS* for more information.
- **server logins –** (Component Integration Services only) To fully support remote logins, Client-Library provides connection properties that enable CIS to request a server connection. This connection is recognized at the receiving server as a server connection (as opposed to an ordinary client connection), allowing the remote server to validate the connection through the use of `sysremotelogins` as if the connection were made by a site handler.

  When enabled, Omni connects to the specified server using the CS_LOGIN_TYPE connection property, with type set to LREMUSER. Also, if the remote server is an SAP ASE server, the CS_LOGIN_REMOTE_SERVER property is set to the value of the local server name, and remote passwords are set using **ct_remote_pwd()**.
- **server principal –** Sets the server principal name for a remote server.
- **negotiated logins –** (Component Integration Services only) This option is necessary if CIS connections to XP server or Backup Server are required.

  When enabled, Omni connects to the specified server using the CS_SEC_CHALLENGE property, and establishes a callback handler that can respond appropriately to login challenges from XP Server and Backup 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 **false**.
    - **use message confidentiality –** Sets message confidentiality for all connections to the remote server using Kerberos authentication.
    - **use message integrity –** Sets message integrity for all connections to the remote server using Kerberos authentication.
    - **cis hafailover –** (Component Integration Services only) If enabled, instructs Open Client to use automatic failover when connections fail. In this case, CIS connection failures automatically failover to the server specified in directory services (such as the `interface` file and ldap server) as the failover server.

  The SAP ASE 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) for all options except the **security mechanism** option.

  For the **security mechanism** option, specify the name of the security mechanism. To see the names of the security mechanisms available on a server, execute:

  ```
  select * from syssecmechs
  ```

Examples

- **Example 1 –** Displays a list of the server options:

  ```
  sp_serveroption
  ```

  ```
  Settable server options.
  ------------------------
  cis hafailover
  enable login redirection
  external engine auto start
  incompatible sort order
  mutual authentication
  negotiated logins
  net password encryption
  readonly
  relocated joins
  security mechanism
  server cost
  server logins
  server principal
  timeouts
  use message confidentiality
  use message integrity
  ```
- **Example 2 –** Tells the server not to time out inactive physical connections with the remote server GATEWAY:

  ```
  sp_serveroption GATEWAY, "timeouts", false
  ```
- **Example 3 –** Specifies that when connecting to the remote server GATEWAY, GATEWAY sends back an encryption key to encrypt the password to send to it:

```
sp_serveroption GATEWAY, "net password encryption", true
```

- **Example 4 –** Specifies that the EJB Server SYB_EJB starts up each time the SAP ASE server starts up:

  ```
  sp_serveroption SYB_EJB, "external engine auto start", true
  ```

- **Example 5 –** Specifies Kerberos authentication for connections to remote server S2.

  ```
  sp_serveroption S2, "security mechanism", csfkrb5
  ```

- **Example 6 –** Specifies mutual authentication for all connections to the remote server using Kerberos authentication.

  ```
  sp_serveroption TEST3, "mutual authentication", true
  ```

- **Example 7 –** Disables automatic startup, where SYB_EJB is the logical name of the EJB Server:

  ```
  sp_serveroption 'SYB_EJB', 'external engine auto start', 'false'
  ```

  To enable automatic startup, enter:

  ```
  sp_serveroption 'SYB_EJB', 'external engine auto start', 'true'
  ```

  See the *EJB Server User's Guide* for more information about using **external engine auto start**.

Usage

There are additional considerations when using **sp_serveroption**:

- 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 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**.
- The **security mechanism**, **mutual authentication**, **use message confidentiality**, and **use message integrity** options apply to Kerberos logins only.

See also:

- See the *System Administration Guide* for more information on server options.
- **isql** in the *Utility Guide*

Permissions

The permission checks for **sp_serveroption** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege. For a shared-disk cluster, you must be a user with `manage server` and `manage cluster` privileges. Any user can execute **sp_serveroption** with no parameters to display a list of options.
Disabled	With granular permissions disabled, you must be a user with **sa_role** to set the **timeouts** option. You must be a user with **sso_role** to set: • **net password encryption** • **security mechanism** • **mutual authentication** • **use message confidentiality** • **use message integrity** Any user can execute **sp_serveroption** with no parameters to display a list of options.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_helpserver* on page 434
- *sp_password* on page 559

sp_set_qplan

Changes the text of the abstract plan of an existing plan without changing the associated query.

Syntax

```
sp_set_qplan id, plan
```

Parameters

- ***id*** **–** is the ID of the abstract plan.
- ***plan*** **–** is a new abstract plan.

Examples

- **Example 1 –** Changes the text of the abstract plan:

  ```
  sp_set_qplan 563789159,
          "( g_join (scan t1) (scan t2))"
  ```

Usage

There are additional considerations when using **sp_set_qplan**:

- Use **sp_set_qplan** to change the abstract plan of an existing plan. You can specify a maximum of 255 characters for a plan. If the abstract plan is longer than 255 characters, drop the old plan with **sp_drop_qplan**, then use **create plan** to create a new plan for the query.
- When you change a plan with **sp_set_qplan**, plans are not checked for valid abstract plan syntax and the plan is not checked for compatibility with the SQL text. Immediately check all plans modified with **sp_set_qplan** for correctness by running the query for the specified ID.
- To find the ID of a plan, use **sp_help_qpgroup**, **sp_help_qplan** or **sp_find_qplan**. Plan IDs are also returned by **create plan** and are included in **showplan** output.

See also **create plan** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_set_qplan** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage abstract plans` privilege. Any user can execute **sp_set_qplan** to change the text of a plan for which they own.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**. Any user can execute **sp_set_qplan** to change the text of a plan for which they own.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_drop_qpgroup* on page 255
- *sp_drop_qplan* on page 256
- *sp_find_qplan* on page 337
- *sp_help_qplan* on page 373

sp_setlangalias

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

- **Example 1 –** Assigns the alias name "français" for the official language name "french":

```
sp_setlangalias french, français
```

Usage

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.

See also **set** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_setlangalias** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addlanguage* on page 32
- *sp_droplanguage* on page 276
- *sp_helplanguage* on page 419

sp_setpglockpromote

Sets or changes the lock promotion thresholds for a database, for a table, or for the SAP ASE 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

- **Example 1 –** Sets the server-wide lock promotion LWM to 200, the HWM to 300, and the PCT to 50:

  ```
  sp_setpglockpromote "server", NULL, 200, 300, 50
  ```

- **Example 2 –** Sets lock promotion thresholds for the `master` database:

  ```
  sp_setpglockpromote "database", master, 1000, 1100, 45
  ```

- **Example 3 –** Sets lock promotion thresholds for the `titles` table in the `pubs2` database. This command must be issued from the `pubs2` database:

```
sp_setpglockpromote "table", "pubs2..titles", 500, 700, 10
```

- **Example 4 –** Changes the HWM threshold to 1600 for the `master` database. The thresholds were previously set with **sp_setpglockpromote**. This command must be issued from the `master` database:

```
sp_setpglockpromote "database", master, @new_hwm=1600
```

Usage

There are additional considerations when using **sp_setpglockpromote**:

- You can display database-level lock promotions using **sp_helpdb *dbname*** and table-level locks using **sp_helpdb *tablename***.
- **sp_setpglockpromote** configures the lock promotion values for a table, for a database, or for the SAP ASE server.

  The SAP ASE 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 the SAP ASE 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, the SAP ASE server attempts to escalate to a table lock. When the number of locks on a table is below the LWM, the SAP ASE 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, the SAP ASE 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 the SAP ASE 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. The SAP ASE 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**.

Permissions

The permission checks for **sp_setpglockpromote** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage lock promotion threshold` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_configure* on page 167
- *sp_dropglockpromote* on page 270
- *sp_help* on page 358
- *sp_helpdb* on page 394

sp_setpglockpromote_ptn

Sets partition-lock promotion thresholds at the server, database, and table level.

Syntax

The syntax for setting the partition lock promotion threshold at the server level is:

```
sp_setpglockpromote_ptn "server", null, new_lwm, new_hwm, new_pct
```

The syntax for setting the partition lock promotion threshold at the database or table level is:

```
sp_setpglockpromote_ptn "database | table", objname, 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. These are Transact-SQL keywords and therefore, require quotes.
- ***objname* –** is either the name of the partition, table, or database for which you are setting the lock promotion thresholds, or null, if you are setting server-wide values. If you are setting partition-wide values, use the format *table_name.partition_name* for the *objname*.
- ***new_lwm* –** specifies a minimum number of page locks that must be acquired before SAP ASE acquires a partition lock.
- ***new_hwm* –** specifies a maximum number of page locks allowed on the object before SAP ASE attempts to escalate to a partition lock.
- ***new_pct* –** specifies the percentage of locked pages (based on the table size) above which SAP ASE attempts to acquire a partition lock when the number of locks is between the *new_hwm* and *new_lwm* lock promotions.

Examples

- **Example 1 –** Sets the server-wide partition lock promotion threshold values LWM to 200, the HWM to 300, and the PCT to 50:

  ```
  sp_setpglockpromote_ptn "server", NULL, 200, 300, 50
  ```

- **Example 2 –** Sets partition lock promotion thresholds for the master database:

  ```
  sp_setpglockpromote_ptn "database", master, 1000, 1100, 45
  ```

- **Example 3 –** Sets partition lock promotion thresholds for the titles table in the pubs2 database. This command must be issued from the pubs2 database:

  ```
  sp_setpglockpromote_ptn "table", "pubs2..titles", 500, 700, 10
  ```

Permissions

Any user can execute **sp_setpglockpromote_ptn**.

sp_setpsexe

Sets custom execution attributes for a session while the session is active.

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 `spids`.
- ***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

- **Example 1 –** This example sets the priority of the process with an ID of 1 to HIGH:

```
sp_setpsexe 1, "priority", "HIGH"
```

Usage

There are additional considerations when using **sp_setpsexe**:

- 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.

Permissions

The permission checks for **sp_setpsexe** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any execution class` privilege. Any user can execute **sp_setpsexe** to lower the priority of a process owned by that user.
Disabled	With granular permissions disabled, you must be a user with **sa_role.** Any user can execute **sp_setpsexe** to lower the priority of a process owned by that user.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addexeclass* on page 23
- *sp_bindexeclass* on page 83
- *sp_dropexeclass* on page 266
- *sp_showexeclass* on page 652

sp_setrowlockpromote

Sets or changes row-lock promotion thresholds for a datarows-locked table, for all datarows-locked tables in a database, or for all datarows-locked tables on a server.

Syntax

```
sp_setrowlockpromote "server", NULL, new_lwm, new_hwm, new_pct
```

```
sp_setrowlockpromote {"database" | "table"}, objname, new_lwm,
    new_hwm, new_pct
```

Parameters

- **server –** sets server-wide values for the row lock promotion thresholds.
- **"database" | "table" –** specifies whether to set the row-lock promotion thresholds for a database or table.
- ***objname* –** is either the name of the table or database for which you are setting the row-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 high watermark (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

- **Example 1 –** Sets row lock promotion values for all datarows-locked tables in the `engdb` database:

  ```
  sp_setrowlockpromote "database", engdb, 400, 400,95
  ```

- **Example 2 –** Sets row lock promotion values for the `sales` table:

  ```
  sp_setrowlockpromote "table", sales, 250, 250, 100
  ```

Usage

There are additional considerations when using **sp_setrowlockpromote**:

- You can display database-level lock promotions using **sp_helpdb** ***dbname*** and table-level locks using **sp_helpdb** ***tablename***.
- **sp_setrowlockpromote** sets or changes row-lock promotion thresholds for a table, a database, or the SAP ASE server.

  The SAP ASE server acquires row locks on a datarows-locked table until the number of locks exceeds the lock promotion threshold. If the SAP ASE server is successful in acquiring a table lock, the row locks are released.

  When the number of row locks on a table exceeds the HWM, the SAP ASE server attempts to escalate to a table lock. When the number of row locks on a table is below the LWM, the SAP ASE server does not attempt to escalate to a table lock. When the number of row locks on a table is between the HWM and LWM, and the number of row locks exceeds the PCT threshold as a percentage of the number of rows in a table, the SAP ASE server attempts to escalate to a table lock.
- Lock promotion is always two-tiered, that is, row locks are promoted to table locks. The SAP ASE server does not promote from row locks to page locks.
- Lock promotion thresholds for a table override the database or server-wide settings. Lock promotion thresholds for a database override the server-wide settings.
- To change the lock promotion thresholds for a database, you must be using the `master` database. To change the lock promotion thresholds for a table in a database, you must be using the database where the table resides.
- Server-wide row lock promotion thresholds can also be set with **sp_configure**. When you use **sp_setrowlockpromote** to change the values server-wide, it changes the configuration

parameters, and saves the configuration file. When you first install SAP ASE, the server-wide row lock promotion thresholds set by the configuration parameters are:

- **row lock promotion HWM –** 200
- **row lock promotion LWM –** 200
- **row lock promotion PCT –** 100

See the *System Administration Guide* for more information.

- The system procedure **sp_sysmon** reports on row lock promotions.
- Database-level row lock promotion thresholds are stored in the `master..sysattributes` table. If you dump a database, and load it only another server, you must set the row lock promotion thresholds on the new server. Object-level row lock promotion thresholds are stored in the `sysattributes` table in the user database, and are included in the dump.

Permissions

The permission checks for **sp_setrowlockpromote** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage lock promotion threshold` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_configure* on page 167
- *sp_droprowlockpromote* on page 283
- *sp_helpdb* on page 394
- *sp_sysmon* on page 684

sp_setrowlockpromote_ptn

Sets partition-lock promotion thresholds at the server, database, and table level.

Syntax

The syntax for setting the partition lock promotion threshold at the server level is:

```
sp_setrowlockpromote_ptn "server", null, new_lwm, new_hwm, new_pct
```

The syntax for setting the partition lock promotion threshold at the database or table level is:

```
sp_setrowlockpromote_ptn "database | table", objname, 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. These are Transact-SQL keywords and therefore, require quotes.
- ***objname* –** is either the name of the partition, table, or database for which you are setting the lock promotion thresholds, or null, if you are setting server-wide values. If you are setting partition-wide values, use the format *table_name.partition_name* for the *objname*.
- ***new_lwm* –** specifies a minimum number of row locks that must be acquired before SAP ASE acquires a partition lock.
- ***new_hwm* –** specifies a maximum number of row locks allowed on the object before SAP ASE attempts to escalate to a partition lock.
- ***new_pct* –** specifies the percentage of locked rows (based on the table size) above which SAP ASE attempts to acquire a partition lock when the number of locks is between the *new_hwm* and *new_lwm* lock promotions.

Examples

- **Example 1 –** Sets the server-wide partition lock promotion threshold values LWM to 200, the HWM to 300, and the PCT to 50:

  ```
  sp_setrowlockpromote_ptn "server", NULL, 200, 300, 50
  ```

- **Example 2 –** Sets partition lock promotion thresholds for the master database:

  ```
  sp_setrowlockpromote_ptn "database", master, 1000, 1100, 45
  ```

- **Example 3 –** Sets partition lock promotion thresholds for the titles table in the pubs2 database. This command must be issued from the pubs2 database:

  ```
  sp_setrowlockpromote_ptn "table", "pubs2..titles", 500, 700, 10
  ```

Permissions

Any user can execute **sp_setrowlockpromote_ptn**.

sp_setsuspect_granularity

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

- **Example 1 –** Displays the recovery fault isolation mode for the current database:

  ```
  sp_setsuspect_granularity
  ```

  ```
  DB Name  Cur. Suspect Gran.  Cfg. Suspect Gran.  Online mode
  -------  ------------------  ------------------  -----------
  pubs2    database           database           read/write
  ```

- **Example 2 –** Displays the current and configured recovery fault isolation mode for the `pubs2` database:

  ```
  sp_setsuspect_granularity pubs2
  ```

- **Example 3 –** The next time recovery runs in the `pubs2` database, if any corrupt pages are detected, only the suspect pages are taken offline and the rest of the database is brought online:

```
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’.
```

- **Example 4 –** The next time recovery runs in the `pubs2` database, if any corrupt pages are detected, only the suspect pages are taken offline and the rest of the database is brought online in read only mode:

  ```
  sp_setsuspect_granularity pubs2, "page", "read_only"
  ```

- **Example 5 –** The next time recovery runs in the `pubs2` database, if any corrupt data is detected, the entire database is marked suspect and taken offline:

  ```
  sp_setsuspect_granularity pubs2, "database"
  ```

Usage

There are additional considerations when using **sp_setsuspect_granularity**:

- **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. See the *System Administration Guide* for more information.
- 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.
- The Cluster Edition allows only the **database** option with **sp_setsuspect_granularity**.
- You must be in the `master` database to set the recovery fault isolation mode.
- Data marked suspect due to corruption persists across SAP ASE 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**, the SAP ASE 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 is 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.

See also **dump database**, **dump transaction**, **load database** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_setsuspect_granularity** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `own database` privilege on the specified database to set the recovery fault isolation mode. Any user can execute **sp_setsuspect_granularity** to display settings.
Disabled	With granular permissions disabled, you must be a user with **sa_role** to set the recovery fault isolation mode. Any user can execute **sp_setsuspect_granularity** to display settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_dboption* on page 193
- *sp_forceonline_db* on page 343
- *sp_forceonline_page* on page 347
- *sp_listsuspect_db* on page 475
- *sp_listsuspect_page* on page 478
- *sp_setsuspect_threshold* on page 643

sp_setsuspect_threshold

Displays or sets the maximum number of suspect pages that the SAP ASE 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 datapages that recovery allows before marking the entire database suspect. The default is 20 pages. The minimum is 0.

Examples

- **Example 1 –** Sets the maximum number of suspect pages to 5. If there are more than 5 suspect pages, recovery marks the entire database suspect:

  ```
  sp_setsuspect_threshold pubs2, 5
  ```

- **Example 2 –** Displays the current and configured settings for the suspect escalation threshold for the `pubs2` database:

  ```
  sp_setsuspect_threshold pubs2
  ```

- **Example 3 –** Displays the current and configured settings for the recovery fault isolation threshold for the current user database:

  ```
  sp_setsuspect_threshold
  ```

Usage

There are additional considerations when using **sp_setsuspect_threshold**:

- 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, the SAP ASE 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 is in effect the next time recovery is executed in the database.

Permissions

The permission checks for **sp_setsuspect_threshold** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `own database` privilege on the database to set the escalation threshold. Any user can execute **sp_setsuspect_threshold** to display settings.
Disabled	With granular permissions disabled, you must be a user with **sa_role** to set the the escalation threshold. Any user can execute **sp_setsuspect_threshold** to display settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_forceonline_db* on page 343
- *sp_forceonline_page* on page 347
- *sp_listsuspect_db* on page 475
- *sp_listsuspect_page* on page 478
- *sp_setsuspect_granularity* on page 640

sp_setup_table_transfer

Run once in each database containing the tables marked for incremental transfer to create the `spt_TableTransfer` table in this database.

Syntax

```
sp_setup_table_transfer
```

Usage

Although it is optional, you should run **sp_setup_table_transfer** before you transfer a table. If you do not run **sp_setup_table_transfer**, the SAP ASE server automatically creates `spt_TableTransfer` when a table is marked for incremental transfer or when you perform the first transfer.

Permissions

The permission checks for **sp_setup_table_transfer** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage database` privilege.
Disabled	With granular permissions disabled, you must be the daabase owner or a user with **sa_role**.

sp_shmdumpconfig

Specifies the dump condition of a shared memory dump, and displays current settings. You must enable the **dump on conditions** configuration parameter to perform shared memory dumps.

Syntax

```
sp_shmdumpconfig "action", type, value, max_dumps, dump_dir,
    dump_file, option1, option2, option3, option4, option5
```

Parameters

- **“action” –** action requested. One of:
  - **add** – adds the specified shared memory dump conditions.
  - **drop** – drops the specified shared memory dump conditions.
  - **update** – changes the settings for an existing memory dump condition.
  - **reset** – resets the dump count for a shared memory dump condition.
  - **display** – dispalys the current shared memory dump conditions.
  - **config** – one of:
    - **include errorlog** – determines if the errorlog file is included in the dump file:
      - 0 – do not inlude the error log in the dump file.
      - 1 – include the errorlog in the dump file.
    - **merge files** – determines if the dump files are merged after a parallel dump:

    - 0 – do not merge dump files.
    - 1 – merge the dump files.
- ***type, value* –** valid values are:

    - **error** – generates a dump file for the specified server error number (for example, error numbers 1105 or 813).
    - **signal** – generates a dump file when the specified operating system signal occurs (for example, signals 11 or 10).
    - **severity** – generates a dump file when an error occurs with a severity equal to or greater than the specified severity. See *Diagnosing System Problems* in the *System Administration Guide Volume 1* for more information about error severity levels.
    - **module** – generates a dump file for a range of server error numbers. The range is delimited by multiples of 100, for example 800 or 1200.
    - **defaults**
    - **timeslice** – generates a dump file when a process receives a timeslice error.
    - **panic** – generates a dump file when a server panic occurs. A server panic terminates the SAP ASE server after perfoming the shared memory dump.
    - **message** – generates a dump file when a specified error log message occurs. Contact Sybase Technical Support to optain specific error message numbers.
    - **dbcc** – sets up a configuration with defaults and omissions as requested. Upon the next occurrence of the problem, issue **dbcc memdump** at the **isql** prompt to create a memory dump.
- **max_dumps –** maximum number of dumps generated for a dump condition. The dump count is reset each time you restart the server. You can also reset the dump count with the **reset** *action* parameter.
- ***dump_dir* –** is the directory in which the SAP ASE server creates the dump file. The “sybase” user must have **read** and **write** permission in this directory.

  You should set the *dump_dir* to a known, consistent location. Make sure there is sufficient space in this directory to hold the required number of dump files. Remove a *dump_dir* setting by performing an **update** action with two double quotes ("") as the *dump_dir* value:

  ```
  sp_shmdumpconfig 'update', signal, 11, null, null, ""
  ```
- ***dump_file* –** is the file name for the dump. If you do not supply a file name, the SAP ASE server creates a name that is guaranteed to be unique. If you provide a file name, all files for the affected conditions use this name, and existing files are overwritten.
- ***option1, . . . , option5* –** determine whether areas of SAP ASE memory are included in the dump file (by default, the procedure cache is included). One of:

    - **include_page** – include all pages from data caches.
    - **omit_page** – omit all pages from data caches.
    - **default_page** – use the default value when including data cache pages.
    - **include_proc** – include all pages from the procedure cache.
    - **omit_proc** – omit all pages from the procedure cache.

  - **default_proc** – use the default values for the procedure cache.
  - **include_unused** – include unused pages.
  - **omit_unused** – omit unused pages.
  - **default_unused** – use the default value for unused pages.

  Values for these options override the system-wide default settings. Specify **default_cache**, **default_proc**, or **default_unused** to inherit the appropriate value from the system-wide default settings.

  Note: Unless you are instructed otherwise by SAP Product Support, you should iinclude the procedure cache in all shared memory dumps.

- **halt –** determines if the SAP ASE server halts the engine while writing the dump file. One of:

  - **no_halt** – no engines halted during the dump. Use this option if you do not want to use shared memory dumps (for example, because the downtime is unacceptable or because the event triggering the shared memory dump is based on a synchronization problem, and you need to see what other engines are doing).

    Memory dumps made with the **no_halt** option may contain a “fuzzy” image and the dump file likely contains corrupted lock tables, run queues, and so on.
  - **default_halt**
  - **halt**

Examples

- **Example 1 –** Requests a one-time memory dump on signal 11:

  ```
  sp_shmdumpconfig "add", signal, 11,1,"dump_dir"
  ```

- **Example 2 –** Requests a memory dump on the occurrence of a 605 error:

  ```
  sp_shmdumpconfig ’add’, error, 605, null, null, null,
      include_page
  ```

  The equivalent on Windows is a `STATUS_ACCESS_VIOLATION (0xc0000005)` message:

  ```
  declare @sig int
  select @sig=hextoint("0xc0000005")
  exec sp_shmdumpconfig 'add', signal, @sig,1,"dump_dir"
  ```

- **Example 3 –** Requests a memory dump for the 8xx range of errors:

  ```
  sp_shmdumpconfig ’add’, module, 800
  ```

- **Example 4 –** Removes a previously defined dump_file by performing an **update** action with two double quotes ("") as the *dump_file* value:

  ```
  sp_shmdumpconfig 'update', signal, 11, null, null, null, ""
  ```

Usage

The **sp_shmdumpconfig** stored procedure uses positional parameters. When setting a parameter that falls to the right of parameters you do not want to set, specify null values for the unset parameters.

Permissions

The permission checks for **sp_shmdumpconfig** differ based on your granular permissions settings. If **action** is equal to **add**, **update**, **reset**:

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server configuration` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

sp_show_options

Prints all the server options that have been set in the current session.

Syntax

```
sp_show_options
```

Examples

- **Example 1 –** Displays the output from **sp_show_options**:

```
create procedure sp_show_options
as
select a.number, a.name
        from master.dbo.spt_values a, master.dbo.spt_values c
        where   c.type = "P"
        and a.type='N'
        and c.low <= datalength(@@options)
        and a.number = c.number
        and convert(tinyint,substring(@@options, c.low, 1)) &
c.high != 0
    return (0)
go
```

```
1> sp_show_options
2> go
```

```
 number      name
 ----------- ----------------------------
           7 arithabort
           8 numeric_truncation
          13 control
```

```
            40 prefetch
            41 triggers
            42 replication
            43 replication force_dll
            48 transactional_rpc
            58 remote_indexes
            62 statement_cache
            64 proc_return_status
            65 proc_output_params

(12 rows affected)
(return status = 0)
```

Usage

@@options the array of bits corresponding to server options. For every option, "low" is the byte number in **@@options**, and "high" is the bit within that byte corresponding to the option. If the bit is set, print name of that option.

Permissions

Any user can execute **sp_show_options**. Permission checks do not differ based on the granular permissions settings.

sp_showcontrolinfo

Displays information about thread pool assignments, bound client applications, logins, and stored procedures.

Considerations for process mode

When you configure the SAP ASE server for process mode, **sp_showcontrolinfo** displays information about engine group assignments, bound client applications, logins, and stored procedures.

Syntax

```
sp_showcontrolinfo [object_type, object_name, spid ]
```

Parameters

- ***object_type* –** one of:
  - **AP** for application
  - **LG** for login
  - **PR** for stored procedure
  - **EG** for thread pool (threaded mode) or engine group (process mode)

- **SV** for service task
- **PS** for process
- **DF** for user-defined default execution class

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** or **DF** 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 SAP ASE process ID. Specify a `spid` only if you specify **PS** as the *object_type*. If you do not specify a `spid` (or specify a `spid` of **null**), **sp_showcontrolinfo** displays information for all `spid`s. Use **sp_who** to see `spid`s.

Examples

- **Example 1 –** Shows all user-assigned execution class-to-object bindings:

```
sp_showcontrolinfo
```

- **Example 2 –** Displays the execution class of the **isql** application:

```
sp_showcontrolinfo 'AP', 'isql'
```

- **Example 3 –** Displays the execution class for all processes assigned to thread pools:

```
sp_showcontrolinfo 'PS'
```

- **Example 4 –** Displays the execution class for `spid` 7:

```
sp_showcontrolinfo 'PS', null, 7
```

Usage

There are additional considerations when using **sp_showcontrolinfo**:

- When used with no parameters, **sp_showcontrolinfo** displays information about all user-assigned thread pool 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, thread pool compositions, and session-level attribute bindings. See *Distributing Engine Resources* in the *Performance and Tuning Series: Basics*.
- When run in process mode, **sp_showcontrolinfo** replaces `thread_pool` with the `engine_group` and `engine` columns.
- 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.
  - **DF** – *object_name* and **spid** should be null, and **sp_showcontrolinfo** shows information about the user-defined default execution class.
- 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.

See also **isql** in the *Utility Guide*.

Permissions

Any user can execute **sp_showcontrolinfo**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrain-fo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addexeclass* on page 23
- *sp_bindexeclass* on page 83
- *sp_clearpsexe* on page 128
- *sp_dropengine* on page 264
- *sp_dropexeclass* on page 266
- *sp_showexeclass* on page 652
- *sp_showpsexe* on page 662
- *sp_unbindexeclass* on page 716

- *sp_who* on page 736

sp_showexeclass

Displays the execution class attributes and the thread pool name associated with the specified execution class.

Considerations for process mode

In process mode, **sp_showexeclass** 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

- **Example 1 –** Displays the priority and thread pool for all execution classes:

  ```
  sp_showexeclass
  ```

  ```
  classname            priority      threadpool
  ------------------   ---------     ---------------
  EC1                       HIGH    syb_default_pool
  EC2                     MEDIUM    syb_default_pool
  EC3                          LOW     syb_default_pool
  ```

- **Example 2 –** Displays the attribute values of execution class `EC1`:

  ```
  sp_showexeclass 'EC1'
  ```

  ```
  classname            priority      threadpool
  ------------------   ---------     ---------------
  EC1                          HIGH      syb_default_pool
  ```

Usage

If *execlassname* is NULL or absent, **sp_showexeclass** displays the priority and thread pool attribute values for all execution classes, including the attribute values of the system-defined classes `EC1`, `EC2`, and `EC3`.

Permissions

Any user can execute **sp_showexeclass**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addexeclass* on page 23
- *sp_bindexeclass* on page 83
- *sp_dropexeclass* on page 266
- *sp_showcontrolinfo* on page 649
- *sp_unbindexeclass* on page 716

sp_showoptstats

Similar in function to the **optdiag** standalone utility in an XML document but in a system procedure format, **sp_showoptstats** extracts and displays statistics and histograms for various data objects from system tables such as `systabstats` and `sysstatistics`.

Syntax

```
sp_showoptstats
[[database_name.[owner].]{table_name|prs_name}],[column_name], [h]
```

Parameters

- ***database_name* –** is the name of the database for which **sp_showoptstats** displays statistics and histograms. *dbname* has these restrictions:
  - Cross-database execution is not supported
  - You must currently be in the specified database to execute **sp_showoptstats**.

  - If you do not specify a database, **sp_showoptstats** displays statistics and histograms about the current database
- ***owner* –** is the name of the table owner. If owner name is not specified, the current user or dbo is used.
- ***table_name* –** is the name of the table for which **sp_showoptstats** displays statistics and histograms. *table_name* has these restrictions:
  - If you do not specify a table, **sp_showoptstats** displays statistics and histograms about all tables in the current database. However, to reduce the length of output, **sp_showoptstats** does not display column statistics and histograms are at database level.
  - *table_name* must exist in the current database.
- ***prs_name* –** the name of the precomputed result set for which you are displaying statistics.
  - No parameter – includes statistical information for all precomputed result sets in the current database.
  - A precomputed result set – includes statistical information for precomputed result sets.
- ***column_name* –** is the name of the column for which the SAP ASE server displays statistics and histograms. If you do not specify a column, the SAP ASE server displays the statistics for all columns and all indexes on the table you specify. If you specify a *column_name*, **sp_showoptstats** displays statistics and histograms for only this column
- **h –** displays help information about the procedure.

Examples

- **Example 1 –** Displays statistics for all user tables in the `pubs2` database:

```
1> use pubs2
2> go
1> sp_showoptstats 'pubs2..publishers'
2> go
```

- **Example 2 –** Displays statistics and histograms for the `publishers` table in the `pubs2` database, in XML format:

```
1> sp_showoptstats publishers
```

```
 <?xml version="1.0" encoding="UTF-8"?>
<optStats>
  <procVersion>sp_showoptstats/1.1/AnyPlatform/AnyOS/Tues April 3
14:21:21 2012</procVersion>
  <serverVersion>SAP ASE/15.7.0/EBF 20161 SMP ESD#02 Prelim#2/P/
x86_64/Enterprise Linux/ase157x/3087/64-bit/FBO/Tue May 15
05:35:01 2012</serverVersion>
    <serverName></serverName>
    <specifiedDatabase>pubs2</specifiedDatabase>
    <specifiedTableOwner></specifiedTableOwner>
    <specifiedTable>publishers</specifiedTable>
    <specifiedCol></specifiedCol>
    <tables>
        <tableOwner>dbo</tableOwner>
        <tableName>publishers</tableName>
        <clusteredIndStats>
            <indName>pubind</indName>
            <colList>"pub_id"</colList>
            <stats>
                <pgCnt>1</pgCnt>
                <emptyPgCnt>0</emptyPgCnt>
                <rowCnt>3.0000000000000000</rowCnt>
                <fwdRowCnt>0.0000000000000000</fwdRowCnt>
                <delRowCnt>0.0000000000000000</delRowCnt>
                <CRCnt>1.0000000000000000</CRCnt>
                <oamAllocPgCnt>2</oamAllocPgCnt>
                <firstExtLeafPgs>0</firstExtLeafPgs>
                <dataRowSz>39.3333333333333357</dataRowSz>
                <indHeight>1</indHeight>
                <joinDegree>0.0000000000000000</joinDegree>
                <unusedPgCnt>14</unusedPgCnt>
                <oamPgCnt>1</oamPgCnt>
                <derivedStats>
                  <clusterRatio>0.0000000000000000</clusterRatio>
                    <spaceUtil>0.0072162426614481</spaceUtil>
                  <IOEfficiency>0.5000000000000000</IOEfficiency>
                </derivedStats>
            </stats>
        </clusteredIndStats>
        <colStats>
            <colName>pub_id</colName>
            <lastUpdate>May 15 2012  4:44:40:136PM</lastUpdate>
            <cellDensity>0.3333333333333333</cellDensity>
            <totalDensity>0.3333333333333333</totalDensity>
            <select>default used (0.33)</select>
            <inBetSel>default used (0.25)</inBetSel>
            <rangeVal>0.3333333333333333</rangeVal>
            <totalVal>0.3333333333333333</totalVal>
            <avgColWidth>default used (4.00)</avgColWidth>
            <statsVer>4</statsVer>
            <histogram>
                <colName>pub_id</colName>
                <dataType>char(4)</dataType>
                <requestedStepCnt>20</requestedStepCnt>
                <actualStepCnt>6</actualStepCnt>
                <samplingPct>0</samplingPct>
                <TuningFact>20</TuningFact>
                <steps>
                    <step>1</step>
                    <weight>0.00000000</weight>
                    <equation>&lt;</equation>
                    <value>"0736"</value>
                </steps>
                <steps>
                    <step>2</step>
                    <weight>0.33333334</weight>
                    <equation>=</equation>
                    <value>"0736"</value>
                </steps>
                <steps>
```

```
                    <step>3</step>
                    <weight>0.00000000</weight>
                    <equation>&lt;</equation>
                    <value>"0877"</value>
                </steps>
                <steps>
                    <step>4</step>
                    <weight>0.33333334</weight>
                    <equation>=</equation>
                    <value>"0877"</value>
                </steps>
                <steps>
                    <step>5</step>
                    <weight>0.00000000</weight>
                    <equation>&lt;</equation>
                    <value>"1389"</value>
                </steps>
                <steps>
                    <step>6</step>
                    <weight>0.33333334</weight>
                    <equation>=</equation>
                    <value>"1389"</value>
                </steps>
            </histogram>
        </colStats>
        <noStatsCol>city,pub_name,state
        </noStatsCol>
    </tables>
 </optStats>
```

- **Example 3 –** Shows the syntax of the procedure:

```
1> sp_showoptstats a,b,h
2> go
```

```
Usage: sp_showoptstats  [[database.[owner].]table], [column],
[option]
(return status = 0)
```

- **Example 4 –** Shows output for the `prs1` precomputed result set:

```
sp_showoptstats prs1
------------------------------------------------------------------------
 <?xml version="1.0" encoding="UTF-8"?>
 <optStats>
    <procVersion>sp_showoptstats/1.1/AnyPlatform/AnyOS/
                 Tues April 3 14:21:21 2012</procVersion>
    <serverVersion>Adaptive Server Enterprise/15.7.1/EBFXXXXX SMP
     ''/P/x86_64/Enterprise Linux/asecarina/ENG/64-bit/DEBUG/Mon
     Jul 9 00:16:37 2012</serverVersion>
    <serverName></serverName>
    <specifiedDatabase>prsdb</specifiedDatabase>
    <specifiedTableOwner></specifiedTableOwner>
    <specifiedTable>prs1</specifiedTable>
    <specifiedCol></specifiedCol>
    <tables>
        <tableOwner>dbo</tableOwner>
        <tableName>prs1</tableName>
```

```
       <tableType>precomputed result set</tableType>
       <tableStats>
. . .

       </noStatsCol>
   </tables>
</optStats>
```

Usage

There are additional considerations when using **sp_showoptstats**:

- You cannot execute **sp_showoptstats** across databases.
- **sp_showoptstats** does not include the system tables unless you explicitly specify them.
- Nonprintable and `univarchar` characters appear in hexidecimal format.
- **sp_showoptstats** displays both global and partition-level statistics.
- When the output is larger than the value you set for ***@@textsize***, the SAP ASE server returns a message to increase the ***@@textsize*** setting so that it can display the large output.
- Parameter values that include a period (.) require double quotation marks.
- You can issue **sp_showoptstats** against system tables.
- **sp_showoptstats** does not return statistical information if you specify only the database and owner.

The DTD file for the XML output of **sp_showoptstats** is:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT optStats (procVersion, serverVersion, serverName?,
specifiedDatabase?, specifiedTableOwner?, specifiedTable?,
specifiedCol?, tables*)>
<!ELEMENT procVersion (#PCDATA)>
<!ELEMENT serverVersion (#PCDATA)>
<!ELEMENT serverName (#PCDATA)>
<!ELEMENT specifiedDatabase (#PCDATA)>
<!ELEMENT specifiedTableOwner (#PCDATA)>
<!ELEMENT specifiedTable (#PCDATA)>
<!ELEMENT specifiedCol (#PCDATA)>

<!ELEMENT tables (tableOwner, tableName, partitionCnt?, (tableStats|
clusteredIndStats|indStats|partitionStats|
partitionClusteredIndStats|partitionIndStats)*,  (colStats|
colPartitionStats)*, noStatsCol?)>

<!ELEMENT tableOwner (#PCDATA) >
<!ELEMENT tableName (#PCDATA) >

<!ELEMENT tableStats (tableName, stats)>

<!ELEMENT clusteredIndStats (indName, colList, stats)>

<!ELEMENT indName (#PCDATA) >
<!ELEMENT colList (#PCDATA) >

<!ELEMENT partitionStats (partition*, stats*)>
```

```
<!ELEMENT partition (#PCDATA) >

<!ELEMENT partitionIndStats (indName, partition, colList, stats)>

<!ELEMENT partitionClusteredIndStats (indName, partition, colList,
stats)>

<!ELEMENT stats (pgCnt?, leafCnt?, (emptyPgCnt|emptyLeafCnt)?,
CRCnt?, indCRCnt?, indPgCRCnt?, (dataRowCRCnt|leafRowCRCnt)?,
rowCnt?, fwdRowCnt?, delRowCnt?, indPgCRCnt?, CRCnt?,
oamAllocPgCnt?, (firstExtDataPgs|firstExtLeafPgs)?, (dataRowSz|
leafRowSz)?, indHeight?, dataPages?, joinDegree?, unusedPgCnt?,
oamPgCnt?, derivedStats?) >

<!ELEMENT pgCnt (#PCDATA) >
<!ELEMENT leafCnt (#PCDATA) >
<!ELEMENT CRCnt (#PCDATA) >
<!ELEMENT indCRCnt (#PCDATA) >
<!ELEMENT dataRowCRCnt (#PCDATA) >
<!ELEMENT leafRowCRCnt (#PCDATA) >
<!ELEMENT emptyPgCnt (#PCDATA) >
<!ELEMENT emptyLeafCnt (#PCDATA) >
<!ELEMENT rowCnt (#PCDATA) >
<!ELEMENT fwdRowCnt (#PCDATA) >
<!ELEMENT delRowCnt (#PCDATA) >
<!ELEMENT oamAllocPgCnt (#PCDATA) >
<!ELEMENT firstExtDataPgs (#PCDATA) >
<!ELEMENT firstExtLeafPgs (#PCDATA) >
<!ELEMENT dataRowSz (#PCDATA) >
<!ELEMENT leafRowSz (#PCDATA) >
<!ELEMENT indHeight (#PCDATA) >
<!ELEMENT dataPages (#PCDATA) >
<!ELEMENT joinDegree (#PCDATA) >
<!ELEMENT unusedPgCnt (#PCDATA) >
<!ELEMENT oamPgCnt (#PCDATA) >
<!ELEMENT rowClusterRatio (#PCDATA) >

<!ELEMENT derivedStats (clusterRatio, indClusterRatio?,
(dataClusterRatio|rowClusterRatio)?, spaceUtil?, IOEfficiency?) >

<!ELEMENT clusterRatio (#PCDATA) >
<!ELEMENT indClusterRatio (#PCDATA) >
<!ELEMENT dataClusterRatio (#PCDATA) >
<!ELEMENT spaceUtil (#PCDATA) >
<!ELEMENT IOEfficiency (#PCDATA) >

<!ELEMENT indStats (indName, colList, stats?) >

<!ELEMENT colStats ((colName|colGroup)?, lastUpdate?, cellDensity?,
totalDensity?, select?, inBetSel?, rangeVal?, totalVal?,
avgColWidth?,
statsVer? statsSamDen?, statsSamU?, histogram?) >

<!ELEMENT colGroup (#PCDATA) >
<!ELEMENT lastUpdate (#PCDATA) >
```

```
<!ELEMENT cellDensity (#PCDATA) >
<!ELEMENT totalDensity (#PCDATA) >
<!ELEMENT selectivity (#PCDATA) >
<!ELEMENT inBetweenSelectivity (#PCDATA) >
<!ELEMENT rangeVal (#PCDATA) >
<!ELEMENT totalVal (#PCDATA) >
<!ELEMENT avgColWidth (#PCDATA) >
<!ELEMENT statsVer (#PCDATA) >
<!ELEMENT statsSamDen (#PCDATA) >
<!ELEMENT statsSamU (#PCDATA) >

<!ELEMENT colPartitionStats (ptnName,(colName|colGroup)?,
lastUpdate?, cellDensity?, totalDensity?, select?, inBetSel?,
rangeVal?, totalVal?, avgColWidth?, statsVer? statsSamDen?,
statsSamU?, histogram?) >

<!ELEMENT ptnName (#PCDATA) >

<!ELEMENT histogram (colName, dataType, requestedStepCnt,
actualStepCnt, samplingPct?, TuningFact?, statsOutRan?,
statsHashLow?, statsHashHigh?, statsSamSt?, statsStepSt?,
statsHtSt?, statsPHashSt?, statsHashSt?, statsNoHashSt?, steps*) >

<!ELEMENT colName (#PCDATA) >
<!ELEMENT dataType (#PCDATA) >
<!ELEMENT requestedStepCnt (#PCDATA) >
<!ELEMENT actualStepCnt (#PCDATA) >
<!ELEMENT samplingPct (#PCDATA) >
<!ELEMENT TuningFact (#PCDATA) >
<!ELEMENT statsOutRan (#PCDATA) >
<!ELEMENT statsHashLow (#PCDATA) >
<!ELEMENT statsHashHigh (#PCDATA) >
<!ELEMENT statsSamSt (#PCDATA) >
<!ELEMENT statsStepSt (#PCDATA) >
<!ELEMENT statsHtSt (#PCDATA) >
<!ELEMENT statsPHashSt (#PCDATA) >
<!ELEMENT statsHashSt (#PCDATA) >
<!ELEMENT statsNoHashSt (#PCDATA) >

<!ELEMENT steps (step, weight, equation, value) >

<!ELEMENT step (#PCDATA) >
<!ELEMENT weight (#PCDATA) >
<!ELEMENT equation (#PCDATA) >
<!ELEMENT value (#PCDATA) >

<!ELEMENT noStatsCol (#PCDATA) >
```

See also:

- *Statistics Tables and Displaying Statistics with optdiag* in *Performance and Tuning Series: Improving Performance with Statistical Analysis*; **optdiag** reference page in the *Utility Guide*.
- **optdiag** in the *Utility Guide*

Permissions

Any user can execute **sp_showoptstats**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_showplan

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

- **Example 1 –** 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:

```
declare @batch int
declare @context int
declare @statement int
exec sp_showplan 99, @batch output, @context output, @statement
output
```

- **Example 2 –** Displays the **showplan** output for the current statement running in the user session with a *spid* value of 99:

```
sp_showplan 99, null, null, null
```

Usage

There are additional considerations when using **sp_showplan**:

- **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 SAP ASE Monitor™ Server.
- **sp_showplan** can run in sessions using chained transactions after you use **sp_procxmode** to change the transaction mode to **anymode**.
- 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.

Permissions

The permission checks for **sp_showplan** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `monitor qp performance` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_who* on page 736

sp_showpsexe

Displays execution class, current priority, and thread pool affinity for all client sessions running on the SAP ASE server.

Considerations for process mode

sp_showpsexe displays engine information instead of task affinity.

Syntax

```
sp_showpsexe [spid]
```

Parameters

- ***spid*** **–** is the SAP ASE 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

- **Example 1 –** Displays execution class, current priority, and affinity for all current client sessions:

  ```
  sp_showpsexe
  ```

  ```
  spid   appl_name   login_name   exec_class  current_priority    t
  ask_affinity
  ----   ----------  ----------   ----------
   ----------------   ---------------
  ```

```
   5   NULL        NULL         NULL        LOW                s
yb_default_pool
   6   NULL        NULL         NULL        MEDIUM             s
yb_default_pool
   7   NULL        NULL         NULL        LOW
              syb_default_pool
  26   isql        sa           EC2         MEDIUM             s
yb_default_pool
```

- **Example 2 –** Displays the application name, login name, current priority, and engine affinity of the process with `spid` 5:

```
sp_showpsexe 5
```

```
spid   appl_name   login_name   exec_class  current_priority    t
ask_affinity
----   ----------  ----------   ----------
 ----------------   ---------------
  5    NULL         NULL         NULL       LOW                s
yb_default_pool
```

Usage

There are additional considerations when using **sp_showpsexe**:

- **sp_showpsexe** displays execution class, current priority, and affinity for all sessions (objects with an *spid*). See *Distributing Engine Resources* in *Performance and Tuning Series: Basics*.
- If the *spid* is NULL or absent, **sp_showpsexe** reports on all sessions currently running on the SAP ASE 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.*

Permissions

Any user can execute **sp_showpsexe**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addengine* on page 21
- *sp_addexeclass* on page 23
- *sp_bindexeclass* on page 83
- *sp_clearpsexe* on page 128
- *sp_dropengine* on page 264
- *sp_dropexeclass* on page 266
- *sp_showcontrolinfo* on page 649
- *sp_showexeclass* on page 652
- *sp_unbindexeclass* on page 716

sp_spaceusage

Reports the space usage for a table, index, or transaction log and estimates the amount of fragmentation for tables and indexes in a database. The estimates are computed using an average row-length for data and index rows, and the number of rows in a table. You can archive the space usage and fragmentation data for future reporting and trends analysis.
sp_spaceusage supports a number of actions, including **help**, **display**, **archive** and **report**, to indicate the current SAP ASE space usage.

Syntax

The "**help**" action syntax:

```
sp_spaceusage   'help'[, 'all']
```

```
sp_spaceusage 'help' [, {'display' |  'display summary'
        | 'report' | 'report summary' | 'archive'}
        [, {'table' | 'index' | 'tranlog'}]]
```

The "**display**" action syntax:

```
sp_spaceusage 'display summary [using unit= {KB | MB | GB |
PAGES} ]',
    {'table' | 'index'}, name
    [,where_clause [,order_by[,command ] ] ]
```

```
sp_spaceusage 'display [using unit= {KB | MB | GB | PAGES} ]',
    {'table' | 'index'}, name
    [,select_list
    [,where_clause [,order_by[,command] ] ] ]
```

```
sp_spaceusage 'display [using unit={KB | MB | GB | PAGES} ]',
    'tranlog' [, name[,select_list[,where_clause [,order_by]]]]
```

The “**archive**” action syntax:

```
sp_spaceusage 'archive [ using_clause ]',
    {'table' | 'index'}, name[,where_clause[,command] ]
```

```
sp_spaceusage 'archive [ using_clause ]',
    'tranlog' [,name[,where_clause] ]
```

The “**report**” action syntax:

```
sp_spaceusage 'report summary [ using_clause ]',
    {'table' | 'index'}, name
    [,where_clause [,order_by[,from_date [,to_date]]]]
```

```
sp_spaceusage 'report [ using_clause ]',
    {'table' | 'index'}, name
    [,select_list[,where_clause [,order_by[,from_date [,to_date]]]]]
```

```
sp_spaceusage 'report [ using_clause ]',
    'tranlog' [, name
    [,select_list[,where_clause [,order_by
    [,from_date [,to_date]]]]]]
using_clause = USING using_item [, using_item ...]
```

```
using_item = { unit={ KB | MB | GB | PAGES }
    | dbname=database_name | prefix=string }
```

Parameters

- **help –** displays the entire **sp_spaceusage** syntax. **help *action*** displays the syntax for individual actions supported..
- **display –** displays current space usage information for the specified objects.
- **display summary –** displays a summary of current space usage information for the specified objects.
- **archive –** archives the space usage report to a table. If the archive table does not already exist, **sp_spaceusage** creates one. New data is appended to existing data. You can specify a prefix for the archive table name and the database in which the archive table resides with the *using* clause.
- **report –** reports the space usage information for the specified objecs from previously archived data. The output is same as the **display** action. Include the optional **using** clause to specify the archive table.
- **report summary –** reports a summary of space usage information for the specified objecs from previously archived data. The output is same as the **display summary** action. Include the optional **using** clause to specify the archive table.

- ***using_item* –** specifies the unit, archive database name, and prefix string for the archive table. You can use a *unit* size of kilobytes (KB), megabytes (MB), gigabytes (GB), and pages. By default *unit* size is KB, the current database is the archive database, and no prefix string is assumed.
- ***name* –** is the name of the entity. Depending on the entity type, you can include multipart names such as *owner_name.table_name*, or *owner_name.table_name.index_name*. For the entity type **tranlog**, the name must be **syslogs** or NULL. Pattern specifiers are allowed for each part of a multipart name to support reporting on multiple objects in one pass.
- ***select_list* –** is the comma-separated list of columns to select in the output columns for the **display** and **report** actions. Use `*` to include all columns in the output. Columns can be renamed using the **alias=*name*** notation.
- ***where_clause* –** is the filter to apply to the result set. Use with the **display**, **report**, or **archive** actions to selectively filter unnecessary data.
- ***order_by* –** returns query results in the specified columns in sorted order.
- ***command* –** command run on the entity selected (table, column, or so on) prior to gathering the space usage information for qualifying objects. The following commands are supported: **update statistics**, **update table statistics**, and **update index statistics**.
- ***from_date* –** specifies beginning of the time range you are interested in.
- ***to_date* –** specifies end of the time range you are interested in.

Examples

- **Example 1 –** Displays a brief description, syntax, and usage information for the display action:

```
sp_spaceusage 'help', 'display'
```

```
Display the space usage information for an entity in the current
database.

Usage:
sp_spaceusage 'display', {'table'|'index'}, <name>
              [,<select_list> [,<where_clause> [,<order_by>
[,<command>]]]]

sp_spaceusage 'display summary', {'table'|'index'}, <name>
              [,<where_clause> [,<order_by> [,<command>]]]

sp_spaceusage 'display', 'tranlog' [,{'syslogs'|NULL}
              [,<select_list> [,<where_clause> [,<order_by>]]]]

For more information, use:
sp_spaceusage 'help', 'display', 'table'
sp_spaceusage 'help', 'display', 'index'
sp_spaceusage 'help', 'display', 'tranlog'
```

- **Example 2 –** Displays a summary of the space usage on the `titles` table:

```
sp_spaceusage 'display summary', 'table', 'titles'
```

```
All the page counts in the result set are in the unit 'KB'.
OwnerName  TableName  Type   UsedPages  RsvdPages  ExpRsvdPages
```

```
PctBloatRsvdPages
---------  ---------  ----   ---------  --------   ------------
-----------------
dbo       titles     DATA   6.0       30.0      16.0         87.50
dbo       titles     INDEX  8.0       64.0      32.0         50.00
```

- **Example 3 –** Displays the space usage information for the `titles` table:

```
sp_spaceusage 'display', 'table', 'titles'
```

```
All the page counts in the result set are in the unit 'KB'.
OwnerName   TableName   IndId   NumRows   UsedPages   RsvdPages
  ExtentUtil     ExpRsvdPages    PctBloatUsePages     PctBloatRsv
dPages
-----------  -----------  ------  -----------  ----------   -----
---------
  ----------     ------------    ----------------     -----------
------
dbo          titles       0       18.0         6.0          30.0
  20.00          16.0           0.0                 87.50
dbo          titles       1       NULL         4.0          32.0
  12.50          16.0           0.00                100.00
dbo          titles       2       NULL         4.0          32.0
  12.50          16.0           0.00                100.00
```

- **Example 4 –** Displays the space usage information, in megabytes, for all indexes on the `titles` table with names that start with `title`:

```
sp_spaceusage 'display using unit-MB', 'index', 'titles.title%'
```

```
All the page counts in the result set are in the unit 'MB'.
OwnerName  TableName   IndId  IndexName   UsedPages      RsvdPages
  ExtentUtil   ExpRsvdPages     PctBloatUsedPages   PctBloatRsvdP
ages
--------   --------    -----  ----------  ----------     ---------
  ----------   ------------     -----------------   -------------
---
dbo       titles       0    titles     .005859375    .029296875
  20.00        .015625          0.00              87.50
dbo        titles       1      titleidind  .00390625      .
03125
  12.50        .015625          0.00              100.00
dbo        titles       2      titleind    .00390625     .03125
  12.50        .015625          0.00              100.00
(1 row affected)
(return status = 0)
```

- **Example 5 –** Displays a summary of the space usage for all index names starting with *title* in the `titles` table:

```
sp_spaceusage 'display summary', 'index', 'titles.title%'
```

```
All the page counts in the result set are in the unit of 'KB'.
OwnerName TableName  IndexName   IndId  UsedPages  RsvdPages  Exp
RsvdPages
    PctBloatRsvdPages
--------  ---------  ---------   -----
---------  ----------  ------------
    -----------------
```

```
dbo       titles     titles      0     6.0         30.0      16.0
    46.67
dbo       titles     titleidind  1     4.0         32.0      16.0
    50.00

dbo       titles     titleind    2     4.0         32.0      16.0
    50.00
```

- **Example 6 –** Displays a summary of the space usage for all indexes starting with *title* in the `titles` table where the value of `PctBloatRsvdPages` is less than 50:

```
sp_spaceusage 'display summary', 'index', 'titles.title%',
    'where PctBloatRsvdPages < 50'
```

```
All the page counts in the result set are in the unit 'KB'.
OwnerName  TableName  IndexName  IndId  UsedPages  RsvdPages  Exp
RsvdPages
     PctBloatRsvdPages
---------  ---------  ---------  -----  ---------  ---------  ---
-------
     -----------------
dbo        titles     titles     0      6.0        30.0        16.0
     46.67
```

- **Example 7 –** Displays a summary of the space usage for all indexes in the `titles` table in descending order of `PctBloatRsvdPages` where the value of `PctBloatRsvdPages` is greater than 30:

```
1> sp_spaceusage 'display summary', 'index', 'titles.title%',
   'where PctBloatRsvdPages > 30', 'order by PctBloatRsvdPages
desc'
```

```
All the page counts in the result set are in the unit 'KB'.
OwnerName  TableName  IndexName  IndId  UsedPages  RsvdPages  Exp
RsvdPages
     PctBloatRsvdPages
---------  ---------  ---------  -----  ---------  ---------  ---
-------
     -----------------
dbo        titles     titleidind  1     4.0        32.0       16.0
     50.00
dbo        titles     titleind    2     4.0        32.0       16.0
     50.00
dbo        titles     titles      0     6.0        30.0       16.0
     46.67
```

- **Example 8 –** Runs **update table statistics** on the `authors` table and summarizes its space usage information in the unit *pages*:

```
sp_spaceusage 'display summary using unit=pages', 'table',
'authors', null, null, null,
 'update table statistics'
```

```
All the page counts in the result set are in the unit 'pages'.
OwnerName TableName Type UsedPages RsvdPages ExpRsvdPages
PctBloatRsvdPages
--------- --------- ---- --------- --------- ------------
-----------------
```

```
dbo       authors  DATA    2.0        16.0        8.0        100.00
dbo        authors  INDEX   4.0          32.0         16.0
  50.00
```

- **Example 9 –** Displays the space usage information for the transaction log of the current database (`pubs2`):

```
sp_spaceusage 'display', 'tranlog'
```

```
TableName TotalPages UsedPages CLRPagesFreePages PctUsedPages
PctFreePages
--------- ----------- --------- ----------------- -------------
-----------
syslogs   4096.0      18.0
0.0  1482.0             0.43         36.18
```

- **Example 10 –** Archives the space usage information for the `authors` table in the currrent database into the default table (`spaceusage_object` for tables and indexes):

```
sp_spaceusage 'archive', 'table', 'authors'
```

```
Data was successfully archived into table
'pubs2.dbo.spaceusage_object'.
```

- **Example 11 –** Archives the space usage information for the `authors` table into the default table (`spaceusage_object` for tables and indexes) in the `pubs3` database:

```
sp_spaceusage 'archive using dbname = pubs3', 'table', 'authors'
```

```
Data was successfully archived into table
'pubs3.dbo.spaceusage_object'.
```

- **Example 12 –** Runs **update table statistics** on the `authors` table and archives its space usage information into a table in the current database with the prefix `monday_` (for this example, `monday_spaceusage_object`):

```
1> sp_spaceusage 'archive using dbname = pubs2, prefix=monday_',
  'table','authors', null, 'update table statistics'
```

- **Example 13 –** Archives the space usage information for the transaction log of the current database into the default table (`spaceusage_tranlog` for transaction logs) in the `pubs3` database:

```
sp_spaceusage 'archive using dbname=pubs3', 'tranlog'
```

```
Data was successfully archived into table
'pubs3.dbo.spaceusage_tranlog'.
```

- **Example 14 –** Reports in detail the last archived space usage information for the `authors` table from the default table (`spaceusage_object` for table or index) in the current database:

```
sp_spaceusage 'report', 'table', 'authors'
```

```
All the page counts in the result set are in the unit 'KB'.
All the data in the result set are dated 'Jun 15 2013 11:50PM'.
OwnerName  TableName  IndId  NumRows  UsedPages  RsvdPages  Exten
tUtil
   ExpRsvdPages    PctBloatUsedPages    PctBloatRsvdPages
```

```
---------  ---------  -----  -------  ---------  ---------  -----
-----
   ------------    -----------------    -----------------
dbo        authors    0      23.0     4.0        32.0       12.50
   16.0            0.00                 100.00
dbo        authors    1      NULL     4.0        32.0       12.50
   16.0            0.00                 100.00
dbo        authors    2      NULL     4.0        32.0       12.50
   16.0            0.00                 100.00
(1 row affected)(return status = 0)
```

- **Example 15 –** Reports in summary the last archived space usage information for the `authors` table from the default table in the `pubs3` database:

```
sp_spaceusage 'report summary using dbname=pubs3', 'table',
'authors'
```

```
All the page counts in the result set are in the unit 'KB'.
All the data in the result set are dated 'Jan 17 2013 11:29AM'.
OwnerName TableName Type UsedPages RsvdPages ExpRsvdPages
PctBloatRsvdPages
--------- --------- ---- ---------- -------- ------------
-----------------
dbo       authors   DATA   4.0    32.0      16.0         100.00
dbo       authors   INDEX  8.0    64.0      32.0         50.00
```

- **Example 16 –** Reports a summary from the `monday_spaceusage_object` table in the current database the last archived space usage information (in megabytes) for the `authors` table:

```
sp_spaceusage 'report summary using prefix=monday_, unit=MB',
'table',
    'authors'
```

```
All the page counts in the result set are in the unit 'MB'.
All the data in the result set are dated 'Jan 17 2013 11:29AM'.
OwnerName TableName Type UsedPages RsvdPages ExpRsvdPages
PctBloatRsvdPages
--------- --------- ---- ---------- -------- ------------
-----------------
dbo       authors   DATA   .00390625  .03125   .015625   100.00
dbo       authors   INDEX   .0078125   .0625    .03125   50.00
```

- **Example 17 –** Reports the space usage information from the default table in the current database for all the indexes on the `authors` table archived on Jun 9, 2013 or later:

```
sp_spaceusage 'report', 'index', 'authors.%', null, null, null,
'Jun 9 2013'
```

```
All the page counts in the result set are in the unit 'KB'.
ArchiveDateTime      OwnerName  TableName   IndId  IndexName  Use
dPages
  RsvdPages  ExtentUtil  ExpRsvdPages  PctBloatUsedPages   PctBlo
atRsvdPages
---------------      ---------  ---------   -----  ---------  ---
-------
  ---------  ---------   -----------
 ----------------    --------------
```

```
Jun  9 2013 12:06AM   dbo      authors     0    authors   4.0
  32.0       12.50       16.0         0.00  100.00
Jun 10 2013 12:05AM  dbo        authors      0     authors    4.0
  32.0       12.50       16.0         0.00  100.00
Jun 11 2013 11:35PM  dbo        authors      0     authors    4.0
  32.0       12.50       16.0         0.00 100.00
Jun  9 2013 12:06AM  dbo        authors      1     auidind    4.0
  32.0       12.50       16.0         0.00  100.00
Jun 10 2013 12:05AM  dbo        authors      1     auidind    4.0
  32.0       12.50       16.0         0.00  100.00
Jun 11 2013 11:35PM  dbo        authors      1     auidind    4.0
  32.0       12.50       16.0         0.00  100.
Jun  9 2013 12:06AM  dbo        authors      2     aunmind    4.0
  32.0       12.50       16.0         0.00  100.00
Jun 10 2013 12:05AM  dbo        authors      2     aunmind    4.0
  32.0       12.50       16.0         0.00  100.00
Jun 11 2013 11:35PM  dbo        authors      2     aunmind    4.0
  32.0       12.50        16.0        0.00  100.00
(1 row affected)
(return status = 0)
```

- **Example 18 –** Reports the space usage information for the `authors` table from the default table in the current database archived between Jun 10 2013 and Jun 15 2013:

```
sp_spaceusage 'report', 'table', 'authors', null, null, null, 'Jun
10 2013', 'Jun 15 2013'
```

```
All the page counts in the result set are in the unit 'KB'.
ArchiveDateTime      OwnerName TableName  IndId  NumRows     Used
Pages
 RsvdPages  ExtentUtil  ExpRsvdPages  PctBloatUsedPages  PctBloat
RsvdPages
---------------      --------- ---------  -----  -------
  ---------
 ---------  ----------   ------------   -----------------   -----
---------
Jun 10 2013 12:05AM  dbo       authors    0      23.0        4.0
 32.0       12.50     16.0         0.00                100.00
Jun 11 2013 11:35PM  dbo       authors    0      23.0        4.0
 32.0       12.50     16.0         0.00                100.00
Jun 13 2013 11:46PM  dbo       authors    0      23.0        4.0
 32.0       12.50     16.0         0.00                100.00
Jun 14 2013 11:46PM  dbo       authors    0      23.0        4.0
 32.0       12.50     16.0         0.00                100.00
Jun 14 2013 11:46PM  dbo       authors    0      23.0        4.0
 32.0       12.50     16.0         0.00                100.00
Jun 10 2013 12:05AM  dbo       authors    1      NULL        4.0
 32.0       12.50     16.0         0.00                100.00
Jun 11 2013 11:35PM  dbo       authors    1      NULL        4.0
 32.0       12.50     16.0         0.00                100.00
Jun 13 2013 11:46PM  dbo       authors    1      NULL        4.0
 32.0       12.50     16.0         0.00                100.00
Jun 14 2013 11:46PM  dbo       authors    1      NULL        4.0
 32.0       12.50     16.0         0.00                100.00
Jun 14 2013 11:46PM  dbo       authors    1      NULL        4.0
 32.0       12.50     16.0         0.00                100.00
Jun 10 2013 12:05AM  dbo       authors    2      NULL        4.0
```

```
 32.0       12.50     16.0          0.00                100.00
Jun 11 2013 11:35PM  dbo       authors    2      NULL        4.0
 32.0       12.50     16.0          0.00                100.00
Jun 13 2013 11:46PM  dbo       authors    2      NULL        4.0
 32.0       12.50     16.0          0.00                100.00
Jun 14 2013 11:46PM  dbo       authors    2      NULL        4.0
 32.0       12.50     16.0          0.00                100.00
Jun 14 2013 11:46PM  dbo       authors    2      NULL        4.0
 32.0       12.50     16.0          0.00                100.00

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

Usage

- **sp_spaceusage** provides space usage information for tables, indexes, and the transaction log of the current database.
- The database in which you are archiving the space usage data must have **sp_dboption ... select into** enabled.
- The archive tables are created if they do not already exist at the time of archiving, otherwise the results are appended to the current table. Because of this, any user running **sp_spaceusage** must have **create table** permission in the archive database.
- While archiving or reporting data, only tables owned by the user running **sp_spaceusage** are considered for the archive table. Tables with the same name but owned by another user are ignored. By default, the results are archived to or reported from the `spaceusage_object` table for tables or indexes and `spaceusage_tranlog` for the transaction log. .
- You can use the *from_date* and *to_date* arguments only for the **report** action when reporting from archived data. The SAP ASE server uses only the data in the archive table that falls within the specified time-range when generating the report. If you do not include a *from_date* or a NULL, the SAP ASE server uses all archived data prior to the *to_date*. If you do not include a *to_date* or NULL, the SAP ASE server uses the current date as the value for *to_date*. If you do not include either the *from_date* or *to_date*, the SAP ASE server uses the most recent data in the archive table to generate the report.
- **sp_spaceusage** results are estimated based on statistical data. These estimates are only as good the statistics provided. You can run **update statistics** to improve the accuracy of the results.

Permissions

Any user can execute **sp_spaceusage** to view space usage. However, you may not be able to view certain information about tables that you do not have persmissions to view.

Interpreting PctBloatUsedPages and PctBloatReservedPages Values

The `PctBloatUsedPages` and `PctBloatReservedPages` columns give an estimate of how many more pages than the minimum the table is using and reserving, respectively. These values indicate how beneficial it may be for you to run **reorg rebuild** on the table.

`ExtentUtil` is the ratio of the number of pages that are actually being used against the number of pages that are reserved for the object. Values closer to 100 indicate that most of the pages in the extents reserved for the object are currently used. The synopsis of the measurements are:

`PctBloatUsedPages` Value	`PctBloatReservedPages` Value	Interpretation
Close to 0, low value	Close to 0, low value	Indicates the table is well compacted, and all allocated pages and allocation units are used completely. `ExtentUtil` should be close to 1.0
Close to 0, low value	Not close to 0, high value	Indicates the used pages are well compacted, but the table's extents are under-utilized, and there is a large degree of interpage fragmentation, possibly due to large-scale deletions or empty pages. The `unusedpgcnt` in `systabstats` is probably also high. The high value of `PctBloatReservedPages` suggests that `ExtentUtil` to is probably much less than 1.0. You can probably resolve most issues by running **reorg rebuild**.
Not close to 0, high value	Close to 0, low value	Indicates a large degree of intrapage fragmentation, but a smaller degree of inter-page fragmentation. Because the extent utilization is probably high, `ExtentUtil` value should be close to 1.0. Running **reorg compact** may help resolve these issues.
Not close to 0, high value	Not close to 0, high value	A high value for `PctBloatUsedPages` indicates a large degree of intrapage fragmentation, where data rows in used pages are not fully compacted (the used pages contain most of the free space). Because interpage and intrapage fragmentation may cause he high value of `PctBloatReservedPages`, the value of **Extent_Util** may still be less than 1.0. Running **reorg compact** and **reorg rebuild** may resolve these issues.

Output Columns from sp_spaceusage

The set of columns that appear in the **sp_spaceusage** output depend on the action and entity type.

By default, only a standard set of columns are displayed. However, you can include others with the *select_list* parameter, and you can view them all with the `*` wildcard in the **select** list.The following two lists provide the set of all output column names and their description for the entity types `table`, `index` and `tranlog`, respectively. Column names in the *select_list*, *where_clause*, *orderby_clause* parameters must belong to the set listed in these tables.

Output columns for table or index entity types are:

- **`ArchiveDateTime`** – Timestamp of the data
- **`ServerName`** – Server name
- **`MaxPageSize`** – Logical page size, in *@@maxpagesize*
- **`DBName`** – Object's database name
- **`OwnerName`** – Object's owner name
- **`TableName`** – Table name
- **`Id`** – ID of the table
- **`IndId`** – ID of the index
- **`IndexName`** – Index name
- **`PtnId`** – ID of the partition
- **`PtnName`** – Partition name
- **`DataPtnID`** – ID of the data partition with data that the index covers
- **`RowSize`** –
  - For tables – Average size of the data row.
  - For indexes – Average size of a leaf row for nonclustered and clustered indexes (data-only-locked tables)
- **`RowCount_ts`** – Number of rows in the partition as per the systabstats table
- **`NumFwdRows`** – Number of forwarded rows in the partition
- **`NumDelRows`** – Number of deleted rows in the partition
- **`NumlRows`** – Number of rows in the partition
- **`PctFwdRows`** – Percentage of rows that were forwarded in the partition
- **`NonLeafRowSize`** – Average non-leaf row size in the partition
- **`FF`** – Fill factor in the partition
- **`MRPP`** – Maximum number of rows per page in the partition
- **`ERS`** – Expected row size in the partition
- **`RPG`** – Reserve page gap in the partition
- **`IndexHeight`** – Height of the index tree in the partition

- **`OAMAPageCount`** – Number of OAM and AP pages (in pages)
- **`Extent0PageCount`** – Number of pages in the extent 0 (in pages)
- **`Status`** – `status` from `sysindexes` table
- **`Sysstat`** – `sysstat` from `sysobjects` table
- **`Sysstat2`** – `sysstat2` from `sysobjects` table
- **`LockScheme`** – Lock scheme of the table
- **`NumVarCols`** – Number of variable columns the table has
- **`HasAPLCI`** – Indicates whether the table has an APL CI
- **`SpUtil`** – Space utilization derived statistic for the partition
- **`DPCR`** – Data page cluster ratio derived statistic for the partition
- **`DRCR`** – Data row cluster ratio derived statistic for the partition
- **`IPCR`** – Index page cluster ratio derived statistic for the partition
- **`LGIO`** – Large IO efficiency derived statistic for the partition
- **`ExtentUtil`** – Extent utilization for the partition
- **`EmptyPages`** – Number of empty pages in the partition (in units)
- **`DataPages`** – Number of data pages in the partition (in units)
- **`UsedPages`** – Number of used pages in the partition (in units)
- **`RsvdPages`** – Number of pages reserved in the partition (in units)
- **`LeafPages`** – Number of leaf pages in the partition (in units)
- **`ExpDataPages`** – Expected number of data pages in the partition had the data been compact (in units)
- **`ExpUsedPages`** – Expected number of used pages in the partition had the data been compact (in units)
- **`ExpRsvdPages`** – Expected number of reserved pages in the partition had the data been compact (in units)
- **`ExpLeafPages`** – Expected height of the index tree in the partition had the data been compact
- **`PctBloatUsedPages`** – Percentage bloat in the used pages in the partition
- **`PctBloatRsvdPages`** – Percentage bloat in the reserved pages in the partition
- **`PctBloatLeafPages`** – Percentage bloat in the leaf pages in the partition
- **`PctEmptyPages`** – Percentage of data pages that were empty in the partition

The parameters available for the `tranlog` entity type are:

Column	Description
ArchiveDateTime	The timestamp of the data
ServerName	The server name

Column	Description
MaxPageSize	The logical page size set in *@@maxpagesize*
DBName	The object's database name
OwnerName	The object's owner name
TableName	The name of the transaction log, for example, `syslogs`
Id	ID of the `syslogs` table
IsMLD	Is it "mixed log and data" transaction log?
IsLogFull	Is the transaction log full?
LCTPages	The "last chance threshold" value of the log (in units)
TotalPages	Total number of log pages (in units)
UsedPages	The number of log pages already used (in units)
CLRPages	The number of log pages reserved for rollbacks (in units)
FreePages	The number of log pages that has not been used yet (in units)
PctUsedPages	The percentage of log pages that are in use already
PctFreePages	The percentage of log pages that are free

sp_spaceused

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

- **Example 1 –** 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:

```
sp_spaceused titles
```

```
name       rowtotal   reserved   data    index_size   unused
---------- ---------  ---------- ------- ----------   ----------
titles     18         46 KB      6 KB    4 KB         36 KB
```

- **Example 2 –** In addition to information on the `titles` table, prints information for each index on the table:

```
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
```

- **Example 3 –** 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:

```
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
```

- **Example 4 –** Prints a summary of space used in the current database:

```
sp_spaceused
```

```
database_name   database_size
--------------- ---------------------------
master          5 MB

reserved    data         index_size  unused
---------   ---------    ----------- --------
2176 KB     1374 KB      72 KB       730 KB
```

- **Example 5 –** Reports on the amount of space reserved and the amount of space available for the transaction log:

```
sp_spaceused syslogs
```

```
name        rowtotal    reserved   data    index_size   unused
---------- ---------   ---------  ------- ----------   ----------
syslogs    Not avail. 32 KB      32 KB   0 KB         0 KB
```

Usage

There are additional considerations when using **sp_spaceused**:

- **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.

See also **create index**, **create table**, **drop index**, **drop table** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_spaceused**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**

Information	Values
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_helpindex* on page 409
- *sp_statistics* on page 763

sp_ssladmin

Adds, deletes, or displays a list of server certificates for the SAP ASE server.

Syntax

```
sp_ssladmin {[addcert, certificate_path [, password | NULL]]
    [dropcert, certificate_path]
    [lscert]
    [help]}
    [lsciphers]
    [setciphers,
    {"FIPS" | "Strong" | "Weak" | "All" |
quoted_list_of_ciphersuites}]
```

Parameters

- **addcert –** adds a certificate for the local server in the certificates file.
- ***certificate_path* –** specifies the absolute path to the certificates file on the local server.
- ***password* –** the password that is used to encrypt the private key when adding a new server certificate to the certificates file.
- **NULL –** used to require an attended atart-up of the SAP ASE server by requesting the password during start-up from the command line.
- **dropcert –** deletes the certficate from the certificate file.
- **lscert –** lists the certificates in the certificate file.
- **help –** displays online help for **sp_ssladmin**.
- **lsciphers –** displays the values for any set cipher suite preferences.
- **setciphers, {"FIPS" | "Strong" | "Weak" | "All" | *quoted_list_of_ciphersuites*} –** sets a specific cipher suite preference. Select one of these options:

- **"FIPS"** – is the set of encryptions, hash, and key exchange algorithms that are FIPS-compliant. The algorithms included in this list are AES, 3DES, DES, and SHA1.
- **"Strong"** – is the set of encryption algorithms using keys longer than 64 bits.
- **"Weak"** – is the set of encryption algorithms from the set of all supported cipher suites that are not included in the strong set.
- **"All"** – is the set of default cipher suites.
- *quoted_list_of_ciphersuites* – specifies a set of cipher suites as a comma-separated list, ordered by preference. Use quotes (" ") to mark the beginning and end of the list. The quoted list can include any of the predefined sets as well as individual cipher suite names. Unknown cipher suite names cause an error to be reported, and no changes are made to preferences. See Chapter 19, "Confidentiality of Data," in the System Administration Guide for the list of cipher suites included in the defined sets.

Examples

- **Example 1 –** Adds an entry for the local server, Server1.crt, in the certificates file in the absolute path to `/sybase/ASE-12_5/certificates` (`x:\sybase\ASE-12_5\certificates` on Windows). The private key is encrypted with the password "mypassword". The password should be the one specified when you created the private key:

```
sp_ssladmin addcert, "/sybase/ASE-12_5/certificates/
Server1.crt",
    "mypassword"
```

- **Example 2 –** Deletes the certificate, Server1.crt from the certificates file located in `/sybase/ASE-12_5/certificates` (`x:\sybase\ASE-12_5\certificates` on Windows):

```
sp_ssladmin dropcert , "/sybase/ASE-12_5/certificates/
Server1.crt"
```

- **Example 3 –** Lists of all server certificates on the local server:

```
sp_ssladmin lscert
go
```

```
certificate_path
----------------------------------------
/sybase/ASE-12_5/certificates/Server1.crt
```

- **Example 4 –** On initial startup, before any cipher suite preferences have been set, no preferences are shown by **sp_ssladmin lscipher**.

```
1> sp_ssladmin lscipher
2> go
```

```
 Cipher Suite Name  Preference
-----------------   ----------
(0 rows affected)
(return status = 0)
```

This example specifies the set of cipher suites that use FIPS algorithms:

```
1> sp_ssladmin setcipher, 'FIPS'
2> go
```

A preference of 0 (zero) **sp_ssladmin** output indicates a cipher suite is not used by the SAP ASE server. The other, non-zero numbers, indicate the preference order that the SAP ASE server uses the algorithm during the SSL handshake. The client side of the SSL handshake chooses one of these cipher suites that matches its list of accepted cipher suites.

- **Example 5 –** Uses a quoted list of cipher suites to set preferences in the SAP ASE server:

```
1> sp_ssladmin setcipher, 'TLS_RSA_WITH_AES_128_CBC_SHA,
TLS_RSA_WITH_AES_256_CBC_SHA'
2> go
```

Usage

- The SAP ASE listener must present to the client a certificate. The common name in the certificate must match the common name used by the client in the interfaces file. If they do not match, the server authentication and login fail.
- When NULL is specified as the password, **dataserver** must be started with a `-y` flag. This flag prompts the administrator for the private-key password at the command line.
- The use of NULL as the password is intended to protect passwords during the intitial configuration of SSL, before the SSL encrypted session begins.

  After restarting the SAP ASE server with an SSL connection established, use **sp_ssladmin** again, this time using the actual password. The password is then encrypted and stored by the SAP ASE server. Any subsequent starts of the SAP ASE server from the command line would use the encrypted password; you do not have to specify the password on the command line during start up.
- You can specify "localhost" as the *hostname* in the `interfaces` file (`sql.ini` on Windows) to prevent clients from connecting remotely. Only a local connection can be established, and the password is never transmitted over a network connection.

See also *Confidentiality of Data* in the *System Administration Guide*.

Permissions

The permission checks for **sp_ssladmin** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage security configuration` privilege.
Disabled	With granular permissions disabled, you must be a user with **sso_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

Using lsciphers and setciphers to Set Cipher Suites

The **lsciphers** and **setciphers** options allow you to restrict the set of cipher suites that the SAP ASE server uses, giving control to the system security officer over the kinds of encryption algorithms that may be used by client connections to the server or outbound connections from the SAP ASE server. By default, the SAP ASE server uses an internally defined set of preferences for cipher suites.

sp_ssladmin setciphers sets cipher suite preferences to the given ordered list. This restricts the available SSL cipher suites to the specified set of **"FIPS"**, **"Strong"**, **"Weak"**, **"All"**, or a quoted list of cipher suites. This takes effect on the next listener started, and requires that you restart the SAP ASE server to ensure that all listeners use the new settings.

You can display any cipher suite preferences that have been set using **sp_ssladmin lsciphers**. If no preferences have been set, **sp_ssladmin lsciphers** returns 0 rows to indicate no preferences are set and the SAP ASE server uses its default (internal) preferences.

See *Confidentiality of Data* in the *System Administration Guide* for more information.

sp_syntax

Displays the syntax of Transact-SQL statements, system procedures, utilities, and other routines for the SAP ASE 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

- **Example 1 –** Displays all **sp_syntax** modules available on your server:

  ```
  sp_syntax
  ```

  ```
  sp_syntax provides syntax help for Sybase products.
  These modules are installed on this Server:

  Module
  --------------------
  Transact-SQL
  UNIX Utility
  System Procedure

  Usage: sp_syntax command [, module [, language]]
  ```

- **Example 2 –** 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:

  ```
  sp_syntax "disk"
  ```

Usage

The text for **sp_syntax** is in the database `sybsyntax`. Load **sp_syntax** and the `sybsyntax` database onto the SAP ASE server with the installation script described in 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.

Permissions

Any user can execute **sp_syntax**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

Tables used

`sybsyntax..sybsyntax`

See also

- *sp_helpdb* on page 394

sp_sysmon

Displays performance information.

Syntax

```
sp_sysmon {'begin_sample' | {{'end_sample' | 'interval'}[,
section]}}
    [, applmon=
        {'appl_only' | 'appl_and_login' | 'no_appl' | 'noclear' |
'clear'}
    [, filter={cache_wizard_filter_value | 'noclear' | 'clear'}
    [, dumpcounters={'y' | 'n' | 'noclear' | 'clear' | NULL}
    [, option={'noclear' | 'clear' | NULL}]]]]
```

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**. The values and corresponding names of the report sections are:
  - **appmgmt** – Application Management
  - **cache wizard** – Cache Wizard
  - **dcache** – Data Cache Management
  - **diskio** – Disk I/O Management
  - **esp** – ESP Management
  - **housekeeper** – Housekeeper Task Activity
  - **indexmgmt** – Index Management
  - **kernel** – Kernel Utilization
  - **locks** – Lock Management
  - **memory** – Memory Management
  - **mdcache** – Metadata Cache Management
  - **monaccess** – Monitor Access to Executing SQL
  - **netio** – Network I/O Management
  - **parallel** – Parallel Query Management
  - **pcache** – Procedure Cache Management
  - **recovery** – Recovery Management
  - **repagent** – RepAgent
  - **taskmgmt** – Task Management
  - **xactmgmt** – Transaction Management
  - **xactsum** – Transaction Profile
  - **wpm** – Worker Process Management

  You can also obtain most of the information available through **sp_sysmon mdcache** report using **sp_monitorconfig**.
- ***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 and the information they report are:
  - **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*.
- **noclear | clear –** specifies whether to clear or not clear monitor counters:

  - **clear** – explicity clears the monitor counters.
  - **noclear** – **sp_sysmon** does not clear the monitor counters. The primarily purpose of the **noclear** parameter is to provide backward compatibility (earlier versions of **sp_sysmon** cleared monitor counters by default).

  Note: You can use the **noclear** parameter only when you specify a sample interval in **sp_sysmon**. You cannot use **noclear** if you specify **begin_sample** or **end_sample**.

  By default, **sp_sysmon** does not clear the monitor counters that are used as source data for the report. If other applications or instances of the **sp_sysmon** report are running when this is done, clearing the counters may cause the data that they report to be invalid.
- ***filter*** **–** is a `varchar` datatype that allows you to specify a pattern for the cache(s) included in the report.

  For example, if it is specified as default data cache, the report only contains information about the default data cache. If it is specified as `emp%`, the output includes information on all caches with a name matching this pattern.If no value is given the output contains all the caches with the default data cache appearing first, followed by the other caches in alphabetical order.
- **'cache wizard' –** aids in the monitoring and configuring of data caches for optimal performance.
- ***dumpcounters*** **–**

  returns the contents of the `master..sysmonitors` table (which contains the names and values of all monitor counters) as a result set, after returning the requested report sections.
- ***option*** **–** allow you to specify the **clear** or **noclear** parameters if you used all the other **sp_sysmon** parameters to specify alternative **sp_sysmon** behaviors. Valid values are **clear** and **noclear**.

Examples

- **Example 1 –** Prints monitor information after 10 minutes:

```
sp_sysmon "00:10:00"
```

- **Example 2 –** Prints only the “Disk Management” section of the **sp_sysmon** report after 5 minutes:

```
sp_sysmon "00:05:00", diskio
```

- **Example 3 –** Starts the sample, executes procedures and a query, ends the sample, and prints only the “Data Cache” section of the report:

```
sp_sysmon begin_sample
go
execute proc1
go
execute proc2
go
select sum(total_sales) from titles
```

```
go
sp_sysmon end_sample, dcache
go
```

- **Example 4 –** Prints the full report and includes application and login detail for each login:

```
sp_sysmon "00:05:00", @applmon = appl_and_login
```

- **Example 5 –** Report usage without clearing the counters:

```
sp_sysmon "00:01:00", kernel, noclear
```

  You can also use:

```
sp_sysmon "00:01:00", noclear
```

- **Example 6 –** Prints a report using the cache wizard:

```
sp_sysmon '00:00:30', 'cache wizard'
```

```
=================================================================
============
Cache Wizard
=================================================================
============

------------------
default data cache
------------------Run Size        :   100.00 Mb   Usage
%               :      2.86
LR/sec           :    41.10      PR/
sec               :     22.57   Hit%:  45.09
Cache Partitions:     4         Spinlock Contention%:      0.00
Buffer Pool Information
-----------------------------------------------------------------
------------



IO Size Wash Size  Run Size    APF%   LR/sec   PR/sec   Hit%   APF-
Eff% Usage%
------- ---------- ----------- ------ -------- -------- ------
-------- ------
4 Kb     3276 Kb    16.00 Mb  10.00     0.47     0.13  71.43    n/
a     0.20
2 Kb    17200 Kb    84.00 Mb  10.00    40.63    22.43  44.79    n/
a     3.37

(1 row affected)

Object Statistics

-----------------------------------------------------------------
------------
Object                        LR/sec  PR/sec  Hit%   Obj_Cached%
Cache_Occp%
------------------------------ ------- ------- ------
----------- -----------
empdb.dbo.t1                     0.57    0.30  47.06       56.25
        0.02
empdb.dbo.t2                     0.30    0.30   0.00       56.25
```

```
        0.02
empdb.dbo.t3                      0.30    0.30   0.00       56.25
        0.02
empdb.dbo.t4                      0.30    0.30   0.00       56.25
        0.02
empdb.dbo.t5                      0.30    0.30   0.00       56.25
        0.02
empdb.dbo.t6                      0.30    0.30   0.00       56.25
        0.02
empdb.dbo.t8                      0.30    0.30   0.00       56.25
        0.02
empdb.dbo.t7                      0.57    0.20  64.71       62.50
        0.02
tempdb.dbo.tempcachedobjstats     3.63    0.00
100.00       50.00        0.01
tempdb.dbo.tempobjstats           0.47    0.00
100.00       25.00        0.00

Object                          Obj Size    Size in Cache
------------------------------ ----------- -------------
empdb.dbo.t1                         32 Kb       18 Kb
empdb.dbo.t2                         32 Kb       18 Kb
empdb.dbo.t3                         32 Kb       18 Kb
empdb.dbo.t4                         32 Kb       18 Kb
empdb.dbo.t5                         32 Kb       18 Kb
empdb.dbo.t6                         32 Kb       18 Kb
empdb.dbo.t8                         32 Kb       18 Kb
empdb.dbo.t7                         32 Kb       20 Kb
tempdb.dbo.tempcachedobjstats        16 Kb        8 Kb
tempdb.dbo.tempobjstats              16 Kb        4 Kb

---------
company_cache
---------
Run Size        :     1.00 Mb   Usage%              :      0.39
LR/sec          :     0.07      PR/
sec              :      0.07   Hit%:   0.00
Cache Partitions:        1      Spinlock Contention%:      0.00

Buffer Pool Information
-----------------------------------------------------------------
------------
IO Size Wash Size  Run Size    APF%   LR/sec   PR/sec   Hit%  APF-
Eff% Usage%
------- ---------- ----------- ------ -------- -------- ------
-------- ------
2 Kb      204 Kb      1.00 Mb  10.00    0.07    0.07    0.00    n/
a     0.39
Object Statistics
-----------------------------------------------------------------
------------

Object              LR/sec  PR/sec  Hit%   Obj_Cached% Cache_Occp%
-------------------- ------- ------- ------ -----------
-----------
empdb.dbo.history     0.07    0.07   0.00      25.00      0.39
```

```
Object               Obj Size    Size in Cache
-------------------- ----------- -------------
empdb.dbo.history      16 Kb      4 Kb

-------------
companydb_cache
-------------
Run Size        :     5.00 Mb   Usage%              :    100.00
LR/sec          :   380.97      PR/
sec              :     56.67   Hit%:  85.13
Cache Partitions:        1      Spinlock Contention%:      0.00

Buffer Pool Information
-----------------------------------------------------------------
------------
IO Size Wash Size  Run Size    APF%   LR/sec   PR/sec   Hit%   APF-
Eff% Usage%
------- ---------- ----------- ------ -------- -------- ------
-------- ------
2 Kb     1024 Kb    5.00
Mb  10.00    380.97    56.67  85.13  98.42   100.00


Object Statistics
-----------------------------------------------------------------
------------

Object                        LR/sec  PR/sec  Hit%   Obj_Cached%
Cache_Occp%
----------------------------- ------- ------- ------ -----------
-----------
company_db.dbo.emp_projects     41.07   22.80  44.48       19.64
       9.45
company_db.dbo.dept_det         93.03   20.67  77.79       99.08
      54.53
company_db.dbo.emp_perf        116.70    2.63  97.74       97.77
      34.18
company_db.dbo.dept_locs         0.43    0.17  61.54       50.00
       0.16

Object                          Obj Size    Size in Cache
----------------------------- ----------- -------------
company_db.dbo.emp_projects       2464 Kb      484 Kb
company_db.dbo.dept_det           2818 Kb     2792 Kb
company_db.dbo.emp_perf           1790 Kb     1750 Kb
company_db.dbo.dept_locs            16 Kb        8 Kb

TUNING RECOMMENDATIONS
-----------------------------------------------------------------
---
Usage% for 'default data cache' is low (< 5%)
Usage% for 4k buffer pool in cache:default data cache is low (< 5%)
Usage% for 2k buffer pool in cache:default data cache is low (< 5%)

Usage% for 'company_cache' is low (< 5%)
```

```
Usage% for 2k buffer pool in cache:company_cache is low (< 5%)
Consider adding a large I/O pool for 'companydb_cache'
```

Usage

There are additional considerations when using **sp_sysmon**:

- **sp_sysmon** displays information about SAP ASE server performance. It sets internal counters to 0, 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 the *Performance and Tuning Guide* for more information.
- To print only a single section of the report, use the valid values for **sp_sysmon** ***applmon***.
- 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.
- During the sample interval, results are stored in signed integer values. Especially on systems with many CPUs and high activity, these counters can overflow. If you see negative results in your **sp_sysmon** output, reduce your sample time.

See also *Performance and Tuning Series: Monitoring Adaptive Server with sp_sysmon*.

Permissions

You must be a user with `execute` permission to run **sp_sysmon**. The permission can be granted to other users by the database owner of `sybsystemprocs`.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_tab_suspectptn

Lists tables with suspect partitions. A range-partitioned table on character-based partition keys can become suspect after a sort-order change, and hash-partitioned tables can become suspect after a cross-platform dump load.

Syntax

```
sp_tab_suspectptn [table_name]
```

Parameters

- ***table_name* –** is the name of the table containing suspect partitions.

Usage

If you:

- Provide a table name – the SAP ASE server checks only the table named by *table_name*.
- Do not provide a table name – the SAP ASE server checks all the tables in the current database.

Permissions

Any user can execute **sp_tab_suspectptn**. Permission checks do not differ based on the granular permissions settings.

See also

- *sp_indsuspect* on page 453

sp_tempdb

sp_tempdb allows users to:

- Create and manage temporary database groups.
- Bind users or applications to the `default` or other temporary database group or to a specific local temporary database.
- Manage bindings to local temporary databases and temporary database groups.

These bindings are stored in the `sysattributes` table in `master` database.

sp_tempdb provides the binding interface for maintaining bindings in `sysattributes` that are related to the multiple temporary database.

Syntax

```
sp_tempdb [
    [ { “create” | “drop” } , “groupname” ] |
    [ { “add” | “remove” } , “tempdbname”, “groupname” ] |
    [ { “bind”, “objtype”, “objname”, “bindtype”, “bindobj”
        [, “scope”, “hardness” ] } |
        { “unbind”, “objtype”, “objname” [, “scope” ]
“instance_name”} ] |
    [ “unbindall_db”, “tempdbname” ] |
    [ show [, "all" | "gr" | "db" | "login" | "app" [, “name” ] ] |
    [ who, “dbname”]
    [ help ] ]
```

Parameters

- **create –** creates the `default` temporary database group.
- **drop –** drops a temporary database group.
- ***groupname* –** is the name of the temporary database group.
- **add –** adds temporary databases to the `default` temporary database group.
- **remove –** removes temporary databases from the `default` temporary database group.
- ***tempdbname* –** is the name of the temporary database you are adding or removing. For the Cluster Edition, *tempdbname* must be a local user temporary database.
- **bind –** binds logins and applications to temporary databases or the `default` temporary database group.
- **unbind –** unbinds logins and applications to temporary databases or the `default` temporary database group.
- ***objtype* –** is the object type. Valid values are:
  - **login_name** (or **LG**)
  - **application_name** (or **AP**)

  Values are not case-sensitive.
- ***objname* –** is the name of the object you bind or unbind.
- ***bindtype* –** is the bind type. Valid values are:
  - **group** (or **GR**)
  - **database** (or **DB**)

  Values are not case-sensitive.
- ***bindobj* –** is the name of the object being bound, and is either a group or a database depending on the *bindtype*.
- ***scope* –** NULL.
- ***instance_name* –** *in cluster environments* – is the name of the instance owning the local temporary database that is to be unbound. This option is for the Cluster Edition only.

- ***hardness* –** hardness – is **hard**, **soft**, or NULL. The default is **soft**. When you set the value of *hardness* to **hard**, a failure to assign a temporary database according to the binding results in a failure of the login.

  When you set the value to **soft**, such a failure results in the assignment of a temporary database from the default group or a local system temporary database.
- **unbindall_db –** removes all login and application bindings for a given temporary database. It does not remove any database to group memberships. The *tempdbname* variable is required with this option.

  Existing assignments to active sessions are not affected by this operation.
- **show –** displays information stored in the `sysattributes` table about the existing groups, group members, login and application bindings, and active sessions that are assigned to a given database. The values are:

  - **all** or no argument – displays the `default` temporary database group, all database-to-group memberships, and all login and application bindings.
  - **gr** – displays the `default` temporary database group. **sp_tempdb show** displays all temporary databases bound to the `default` temporary database group whether you specify "default" for the *name* option or not.
  - **db** – displays all databases and temporary databases to group memberships. If you provide *name*, then only the database to group memberships for the database *name* are printed.
  - **login** – displays all login bindings where login is not NULL. If you provide *name*, then only the bindings for the login *name* are printed.
  - **app** – displays all bindings where the application is not NULL. If you provide *name*, then the bindings for the application *name* are printed.

  Note: `tempdb` is always part of the `default` database group.
- **who –** displays all active sessions assigned to the given temporary database. When using the **who** parameter, you must use:

  - *dbname* – is the name of a temporary database. If you provide a nontemporary database name for *dbname*, **sp_tempdb who** executes, but does not report any active sessions bound to it.

    If **system_view** is set to **cluster**, all active sessions of the cluster are examined. If **system_view** is set to **instance**, sessions that are active on the current instance are examined

    This command may be executed from any instance in the cluster.
- **help –** displays usage information. Executing **sp_tempdb** without specifying a command is the same as executing **sp_tempdb "help"**.

Examples

- **Example 1 –** Adds `mytempdb1` to the `default` group:

```
sp_tempdb "add", "mytempdb1", "default"
```

- **Example 2 –** Removes `mytempdb1` from the `default` group:

  ```
  sp_tempdb "remove", "mytempdb1", "default"
  ```

- **Example 3 –** Binds login “sa” to the `default` group:

  ```
  sp_tempdb "bind", "lg", "sa", "GR", "default"
  ```

  The value for **objtype** in this example is **login_name**. You can substitute **login_name** with **lg** or **LG**.

  The value for **bindtype** in this example is **group**. You can substitute **group** with **gr** or **GR**.

- **Example 4 –** Changes the previous binding of login “sa” from the `default` group to `mytempdb1`:

  ```
  sp_tempdb "bind", "lg", "sa", "DB", "mytempdb1"
  ```

  The value for **bindtype** in this example is **database**. You can substitute **database** with **db** or **DB**.

- **Example 5 –** Binds **isql** to `mytempdb1`:

  ```
  sp_tempdb "bind", "ap", "isql", "DB", "mytempdb1"
  ```

  The value for **objtype** in this example is **application_name**. You can substitute **application_name** with **ap** or **AP**.

- **Example 6 –** Changes the previous binding of **isql** from `mytempdb1` to the `default` group:

  ```
  sp_tempdb "bind", "ap", "isql", "GR", "default"
  ```

- **Example 7 –** Removes the bindings of login “sa” and application “isql”.

  ```
  sp_tempdb "unbind", "lg", "sa"
  ```

  ```
  sp_tempdb "unbind", "ap", "isql"
  ```

- **Example 8 –** Removes all login and application bindings for the `mytempdb1` database:

  ```
  sp_tempdb "unbindall_db", "mytempdb1"
  ```

- **Example 9 –** Demonstrates the **sp_temp show** command. A selection of the different variations is chosen, and abbreviated sample output is displayed.

  ```
  sp_tempdb show
  ```

  ```
  Temporary Database Groups
  -------------------------------
  default

  Database                        GroupName
  ------------------------------- ----------------
  tempdb                          default
  mytempdb                        default
  mytempdb1                       default
  ```

```
mytempdb2                       default
mytempdb3                       default

Login   Application  Group    Database    Hardness
------- ------------- -------- ----------- --------
NULL   isql          default  NULL        SOFT
sa      NULL          NULL     mytempdb3   HARD
```

- **Example 10 –** Displays the `default` temporary database group:

```
sp_tempdb show, "gr"
```

```
Temporary Database Groups
-------------------------------
default
```

- **Example 11 –** Displays all the temporary database group names that are bound to the `default` group:

```
sp_tempdb show, "gr", "default"Member Databases
-------------------------------
tempdb
mytempdb
mytempdb1
mytempdb2
mytempdb3
```

- **Example 12 –** Displays all the databases-to-group memberships:

```
sp_tempdb show, "db"
```

```
Database              Group
--------------------- ----------------
tempdb                default
mytempdb              default
mytempdb1             default
mytempdb2             default
mytempdb3             default
```

- **Example 13 –** Displays all the databases-to-group memberships for the `mytempdb1` database.

```
sp_tempdb show, "db", "mytempdb1"
```

```
Database              Group
--------------------- ----------------
mytempdb1             default
```

- **Example 14 –** Displays all the login bindings where login is not NULL:

```
sp_tempdb show, "login"
```

```
Login   Application   Group   Database    Hardness
------- ------------- ------- ----------- --------

sa      NULL          NULL    mytempdb3   HARD
```

- **Example 15 –** Displays all active sessions that are assigned to the system `tempdb`:

```
sp_tempdb who, "tempdb"
```

```
spid   loginame
------ ------------------------------
2      NULL
3      NULL
4      NULL
5      NULL
6      NULL
7      NULL
8      NULL
```

- **Example 16 –** Displays all active sessions that are assigned to the `mytempdb3` user-created temporary database:

```
sp_tempdb who, "mytempdb3"
```

```
spid   loginame
------ ------------------------------
17      sa
```

- **Example 17 –** Displays usage information:

```
sp_tempdb help
```

```
Usage:
sp_tempdb 'help'
sp_tempdb 'create', <groupname>
sp_tempdb 'drop', <groupname>
sp_tempdb 'add', <tempdbname>, <groupname>
sp_tempdb 'remove', <tempdbname>, <groupname>
sp_tempdb 'bind', <objtype>, <objname>, <bindtype>, <bindobj>,
<scope>,
          <hardness>
sp_tempdb 'unbind', <objtype>, <objname>, <scope>
sp_tempdb 'unbindall_db', <tempdbname>
sp_tempdb 'show', <command>, <name>
sp_tempdb 'who', <dbname>

<objtype> = ['LG' ('login_name') | 'AP' ('application_name')];
<bindtype> =['GR' ('group') | 'DB' ('database')]
<hardness> = ['hard' | 'soft']
<command> = ['all' | 'gr' | 'db' | 'login' | 'app']
```

- **Example 18 –** Displays all temporary databases and the names of the groups to which the temporary databases belong:

```
create temporary database mytempdb
-------------
CREATE DATABASE: allocating 1536 logical pages (3.0 megabytes) on
disk 'master'.|

create temporary database mytempdb1
----------
CREATE DATABASE: allocating 1536 logical pages (3.0 megabytes) on
disk 'master'.

sp_tempdb 'add', mytempdb,'default'
------------
(return status = 0
sp_tempdb show, db
-------------
Database Group
--------
tempdb default
mytempdb default
mytempdb1
(3 rows affected)
(return status = 0)
```

Usage

There are additional considerations when using **sp_tempdb**:

- To display the distribution of users across all temporary databases, use both options, **show** and **who**:
  - To obtain the names of all temporary databases, execute

    ```
    sp_tempdb 'show'
    ```
  - Pass each temporary database name to

    ```
    sp_tempdb 'who', tempdbname
    ```

  In SAP ASE versions 15.0 and above, you can obtain the same output by executing **sp_who**.
- When using the **sp_tempdb create** stored procedure, the *groupname* variable:
  - Must be a valid identifier
  - Cannot already exist

  The `default` group is the system-generated group, of which `tempdb` is always a member. This `default` group is present if you:
  - Upgrade using the SAP ASE server containing this feature, or
  - Create a new master device.

  If the `default` group is not present, you can create it by using:

  ```
  sp_tempdb create, "default"
  ```

  An error message displays if you attempt to create a `default` group that already exists.
- To add a temporary database to the `default` temporary database group, both the temporary database and the group name must already exist. When you use **sp_tempdb add** to add a *tempdbname* to a set of databases that are members of the `default` temporary database group, *tempdbname* becomes available for round-robin assignment from within that group.

  Note: **sp_tempdb add** fails if *tempdbname* is not already part of the global list of available temporary databases in the SAP ASE server.

  User-created temporary databases need not belong to the `default` temporary database group. The system `tempdb` is implicitly a member of the `default` group.

If you try to add a temporary database to the `default` temporary database group when it is already a part of that group, you get an error message, and no changes take place in `sysattributes`.

Permissions

The permission checks for **sp_tempdb** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage server` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_tempdb_markdrop

(Cluster Edition) Places a local system temporary database in the drop state.

Syntax

```
sp_tempdb_markdrop database_name [, {'mark' | 'unmark'}]
```

Parameters

- ***database_name*** **–** is the name of the local system temporary database you are dropping

- **mark –** marks the specified database for dropping.
- **unmark –** clears the mark from the database.

Examples

- **Example 1 –** Marks a local system temporary database named “old_cluster_tempdb1” to be dropped:

  ```
  sp_tempdb_markdrop 'old_cluster_tempdb1', 'mark'
  ```

- **Example 2 –** Removes the mark from the local system temporary database “old_cluster_tempdb1”:

  ```
  sp_tempdb_markdrop 'old_cluster_tempdb1, 'unmark'
  ```

Usage

To delete the last local temporary database:

1. Use **sp_tempdb_markdrop** to place the local system temporary database in the drop state.
2. Shut down and restart the instance that owns the last local temporary database.

   Note: After you mark the local system temporary database to be dropped, the owner instance restarts if there are no other active instances. This instance does not use the marked local system temporary database when it starts.
3. Use **drop database** to delete the last local system temporary database.

Permissions

The permission checks for **sp_tempdb_markup** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with the `own database` privilegeon the specified database or the `manage cluster` privilege.
Disabled	With granular permissions disabled, you must be the database owner or a user with **sa_role**.

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. SAP does not provide this procedure.

Syntax

When a threshold is crossed, the SAP ASE 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 logical pages.
- ***@status* –** is 1 for the last-chance threshold; 0 for all other thresholds.

Examples

- **Example 1 –** Creates a threshold procedure for the last-chance threshold that dumps the transaction log to a tape device:

  ```
  create procedure sp_thresholdaction
          @dbname varchar(30),
          @segmentname varchar(30),
          @space_left int,
          @status int
  as
          dump transaction @dbname to tapedump1
  ```

Usage

There are additional considerations when using **sp_thresholdaction**:

- **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, the SAP ASE server searches for the **sp_thresholdaction** procedure in the database where the threshold event occurs. If it does not exist in that database, the SAP ASE server searches for it in `sybsystemprocs`. If it

does not exist in `sybsystemprocs`, it searches `master`. If the SAP ASE 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.

For 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, then use **dump transaction** to truncate the transaction log.
- 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, then use the **kill** command to kill them.

See also **create procedure**, **dump transaction** in *Reference Manual: Commands*.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addthreshold* on page 49
- *sp_dboption* on page 193
- *sp_dropthreshold* on page 291
- *sp_helpsegment* on page 430
- *sp_helpthreshold* on page 445
- *sp_modifythreshold* on page 520
- *sp_who* on page 736

sp_tran_dumpable_status

If you cannot make a transaction dump on a database, **sp_tran_dumpable_status** displays the reasons the dump is not possible.

Syntax

```
sp_tran_dumpable_status [database_name]
```

Parameters

- ***database_name* –** name of the database you are researching.

Examples

- **Example 1 –** Describes the reasons you cannot currently make a transaction dump on `sybsystemprocs`:

```
sp_tran_dumpable_status sybsystemprocs
```

```
bit               description
-----------        ----------------------------
          2        Log is not on its own device
          8        Trunc log on ckpt is set
         32        Dump tran with truncate_only
         64        Database is new or upgraded
```

Usage

This system procedure simply calls the **tran_dumpable_status** built-in function.

Permissions

Any user can execute **tran_dumpable_status**. Permission checks do not differ based on the granular permissions settings.

sp_transactions

Reports information about active transactions.

Syntax

```
sp_tranactions ["xid", xid_value] |
    ["state", {"heuristic_commit" | "heuristic_abort"
    | "prepared" | "indoubt"} [, "xactname"]] |
    ["gtrid", gtrid_value]
```

Parameters

- ***xid_value* –** is a transaction name from the `xactname` column of `master.dbo.systransactions`.
- ***gtrid_value* –** is the global transaction ID name for a transaction coordinated by the SAP ASE server.

Examples

- **Example 1 –** Displays general information about all active transactions:

```
sp_transactions
```

```
xactkey                        type   coordinator
starttime         state
   connection dbid spid loid failover srvname  namelen  xactname
------------------------------ ----
  ----------- ---------------  ---------
   ---------- ---- ---- ---- -------- --------
-------  -------------------
0x00000b1700040000dd6821390001 Local    None        Jun 1 1999
3:47PM Begun
   Attached       1   1    2 Resident Tx       NULL     17
$user_transaction
0x00000b1700040000dd6821390001 Remote   ASTC        Jun 1 1999
3:47PM Begun
   NA             0   8    0 Resident Tx        caserv2 108

00000b1700040000dd6821390001-
```

```
aa01f04ebb9a-00000b1700040000dd6821390001-aa0
1f04ebb9a-caserv1-caserv1-0002
```

- **Example 2 –** Displays detailed information for the specified transaction:

```
sp_transactions "xid", "00000b1700040000dd6821390001-
aa01f04ebb9a-00000b1700040000dd6821390001-aa01f04ebb9a-caserv1-
caserv1-0002"
```

```
xactkey                        type        coordinator
starttime        state
   connection dbid spid loid  failover  srvname  namelen  xactname
      commit_node            parent_node
gtrid
------------------------------ ----------- -----------
----------       --------     ---------- ------ ------
---------  -------- --------- ---------- -------
      -------------   -------------  -------------
0x00000b2500080000dd6821960001 External    ASTC       Jun 1 1999
3:47PM  Begun
   Attached      1    8  139  Resident  Tx       NULL         108

00000b1700040000dd6821390001-
aa01f04ebb9a-00000b1700040000dd6821390001-aa0
1f04ebb9a-caserv1-caserv1-0002

caserv1            caserv1
00000b1700040000dd6821390001-aa01f04ebb9a
```

- **Example 3 –** Displays general information about transactions that are in the "prepared" state:

```
sp_transactions "state", "prepared"
```

- **Example 4 –** Displays only the transaction names of transactions that are in the "prepared" state:

```
sp_transactions "state", "prepared", "xactname"
```

- **Example 5 –** Displays status information for transactions having the specified global transaction ID:

```
sp_transactions "gtrid", "00000b1700040000dd6821390001-
aa01f04ebb9a"
```

```
xactkey                        type    coordinator
starttime          state
   connection dbid spid loid failover srvname  namelen   xactname
commit_node
parent_node
------------------------------ ------- ----------
----------------- -----------
    ---------- ------ ------ ----------- ------------
--------------- -------
-------------
-------------
0x00000b1700040000dd6821390001 Local   None        Jun 1 1999
3:47PM  Begun
   Attached     1     1    2  Resident Tx      NULL       17  $us
```

```
er_transaction
caserv1
caserv1
```

Usage

There are additional considerations when using **sp_transactions**:

- **sp_transactions** translates data from the `systransactions` table to display information about active transactions. `systransactions` itself comprises data in the **syscoordinations** table, as well as in-memory information about active transactions.
- **sp_transactions** with no keywords displays information about all active transactions.
- **sp_transactions** with the **xid** keyword displays the `gtrid`, `commit_node`, and `parent_node` columns only for the specified transaction.
- **sp_transactions** with the **state** keyword displays information only for the active transactions in the specified state.

  sp_transactions with both **xid** and **xactname** displays only the transaction names for transactions in the specified state.
- **sp_transactions** with the **gtrid** keyword displays information only for the transactions with the specified global transaction ID.
- **sp_transactions** replaces the **sp_xa_scan_xact** procedure provided with XA-Library and XA-Server products.

The columns for **sp_transactions** output are:

Column	Description
`xactkey`	The column shows the internal transaction key that the SAP ASE server uses to identify the transaction.
`type`	The column indicates the type of transaction: • “Local” means that the transaction was explicitly started on the local SAP ASE server with a **begin transaction** statement. • “Remote” indicates a transaction executing on a remote SAP ASE server. • “External” means that the transaction has an external coordinator associated with it. For example, transactions coordinated by a remote SAP ASE server, MSDTC, or an X/Open XA transaction manager are flagged as “External.” • “Dtx_State” is a special state for distributed transactions coordinated by the SAP ASE server. It indicates that a transaction on the local server was either committed or aborted, but the SAP ASE server has been unable to resolve a branch of that transaction on a remote participant. This may happen in cases where the SAP ASE server loses contact with a server it is coordinating.

Column	Description
`coordinator`	The column indicates the method or protocol used to manage a distributed transaction. The values for coordinator are: • None – transaction is not a distributed transaction and does not require a coordinating protocol. • ASTC – transaction is coordinated using the SAP ASE transaction coordination services. • XA – transaction is coordinated by the X/Open XA-compliant transaction manager via the SAP ASE XA-Library interface. Such transaction managers include Encina, CICS, and Tuxedo. • DTC – transaction is coordinated by MSDTC. • SYB2PC – transaction is coordinated using Sybase two-phase commit protocol.
`starttime`	The column indicates the time that the transaction started.

Column	Description
`state`	The column indicates the state of the transaction at the time **sp_transactions** ran: • Begun – transaction has begun but no updates have been performed. • Done Command – transaction completed an update command. • Done – X/Open XA transaction has finished modifying data. • Prepared • Transaction has successfully prepared. • In Command – transaction is currently modifying data. • In Abort Cmd – execution of the current command in the transaction has been aborted. • Committed – transaction has successfully committed, and the commit log record has been written. • In Post Commit – transaction has successfully committed, but is currently deallocating transaction resources. • In Abort Tran – transaction is being aborted. This may happen either as a result of an explicit command, or because of a system failure. • In Abort Savept – transaction is being rolled back to a savepoint. • Begun-Detached – transaction has begun, but there is no thread currently attached to it. • Done Cmd-Detached – transaction has finished modifying data, and no thread is currently attached to it. • Done-Detached – transaction modifies no more data, and no thread is currently attached to it. • Prepared-Detached – transaction has successfully prepared, and no thread is currently attached to it. • Heur Committed – transaction has been heuristically committed using the **dbcc complete_xact** command. • Heur Rolledback – transaction has been heuristically rolled back using the **dbcc complete_xact** command.
`connec-tion`	The column indicates whether or not the transaction is currently associated with a thread: • “Attached” indicates that the transaction has an associated thread of control. • “Detached” indicates that there is no thread currently associated with the transaction. Some external transaction managers, such as CICS and TUXEDO, use the X/Open XA “suspend” and “join” semantics to associate different threads with the same transaction.
`dbid`	The column indicates the database ID of the database in which transaction started.
`spid`	The column indicates the server process ID associated with the transaction. If the transaction is “Detached,” the “spid” value is 0.

Column	Description
`loid`	The column indicates the unique lock owner ID from `master.dbo.systransactions`.
`failover`	The column indicates the failover state for the transaction: • "Resident Tx" indicates that the transaction started and is executing on the same server. "Resident Tx" is displayed under normal operating conditions, and on systems that do not utilize SAP ASE high availability features. • "Failed-over Tx" is displayed after there has been a failover to a secondary companion server. "Failed-over Tx" means that a transaction originally started on a primary server and reached the prepared state, but was automatically migrated to the secondary companion server (for example, as a result of a system failure on the primary server). The migration of a prepared transaction occurs transparently to an external coordinating service. • "Tx by Failover-Conn" indicates that there was an attempt to start the transaction on a designated server, but the transaction was instead started on the secondary companion server. This occurs when the original server has experienced a failover condition.
`srvname`	The column indicates the name of the remote server on which the transaction is executing. This column is only meaningful for remote transactions. For local and external transactions, `srvname` is null.
`namelen`	The column indicates the total length of the *xactname* value. *xactname* is the transaction name. For local transactions, the transaction name may be defined as part of the **begin transaction** command. External transaction managers supply unique transaction names in a variety of formats. For example, X/Open XA-compliant transaction managers supply a transaction ID (*xid*) consisting of a global transaction identifier and a branch qualifier, both of which are stored in *xactname*.
`gtrid`	For transactions coordinated by the SAP ASE server, the column displays the global transaction ID. Transaction branches that are part of the same distributed transaction share the same `gtrid`. You can use a specific `gtrid` with the **sp_transactions gtrid** keyword to determine the state of other transaction branches in the same distributed transaction. **sp_transactions** cannot display the `gtrid` for transactions that have an external coordinator. For transactions coordinated by an X/Open XA-compliant transaction manager, MSDTC, or SYB2PC, the `gtrid` column shows the full transaction name supplied by the external coordinator.

Column	Description
`commit_node`	For transactions coordinated by the SAP ASE server, the column indicates the server that executes the outermost block of the distributed transaction. This outermost block ultimately determines the commit status of all subordinate transactions. For transactions not coordinated by the SAP ASE server, `commit_node` displays one of these values: • *server_name* – commit or parent node is an SAP ASE server with the specified *server_name*. • XATM – commit or parent node is an X/Open XA-compliant transaction manager. • MSDTCTM – ommit or parent node is MSDTC. • SYB2PCTM – transaction is coordinated using SYB2PC protocol.
`parent_node`	For transactions coordinated by the SAP ASE server, the column indicates the server that is coordinating the external transaction on the local server. For transactions not coordinated by the SAP ASE server, `parent_node` displays the same values as those displayed by `commit_node`. **Note:** The values for `commit_node` and `parent_node` can be different, depending on the levels of hierarchy in the distributed transaction.

See also *Using Adaptive Server Distributed Transaction Management Features*.

Permissions

Any user can execute **sp_transactions**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_lock* on page 483
- *sp_who* on page 736

sp_unbindcache

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 with an index, `text` object, or `image` object that 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

- **Example 1 –** Unbinds the `titles` table from the cache to which it is bound:

  ```
  sp_unbindcache pubs2, titles
  ```

- **Example 2 –** Unbinds the `titleidind` index from the from the cache to which it is bound:

  ```
  sp_unbindcache pubs2, titles, titleidind
  ```

- **Example 3 –** Unbinds the `text` or `image` object for the `au_pix` table from the cache to which it is bound:

```
sp_unbindcache pubs2, au_pix, "text only"
```

- **Example 4 –** Unbinds the transaction log, `syslogs`, from its cache:

  ```
  sp_unbindcache pubs2, syslogs
  ```

Usage

There are additional considerations when using **sp_unbindcache**:

- 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.
- The SAP ASE server issues error number 857 if you attempt to use **sp_unbindcache** to unbind a database that is in use.
- Cache unbindings take effect immediately and do not require a restart of the server, except with the system `tempdb`.
- Although you can still use **sp_unbindcache** on a system `tempdb`, the binding of the system `tempdb` is now non-dynamic. Until you restart the server:
  - The changes do not take effect
  - **sp_helpcache** reports a status of “P” for pending, unless you have explicitly bound the system `tempdb` to the default data cache, in which case the status as “V” for valid, because by default the system `tempdb` is already bound to the default datacache.
- 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.
- These 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**.

See also *Performance and Tuning Guide*.

Permissions

The permission checks for **sp_unbindcache** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage data cache` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_bindcache* on page 76
- *sp_dboption* on page 193
- *sp_help* on page 358
- *sp_helpdb* on page 394
- *sp_helpcache* on page 379
- *sp_helpindex* on page 409
- *sp_unbindcache_all* on page 713

sp_unbindcache_all

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

- **Example 1 –** Unbinds all databases, tables, indexes, `text` objects and `image` objects that are bound to `pub_cache`:

  ```
  sp_unbindcache_all pub_cache
  ```

Usage

There are additional considerations when using **sp_unbindcache_all**:

- 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**.
- You cannot use **sp_unbindcache_all** if the system `tempdb` is bound to `pub_cache`. If you do, you get an error message, and **sp_unbindcache_all** rejects the unbind for all objects.

  Use **sp_unbindcache** to unbind the system `tempdb` first.
- See **sp_unbindcache** for more information about unbinding caches.

Permissions

The permission checks for **sp_unbindcache_all** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage data cache` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_bindcache* on page 76
- *sp_helpcache* on page 379
- *sp_unbindcache* on page 710

sp_unbindefault

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, with a default that 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

- **Example 1 –** Unbinds the default from the `startdate` column of the `employees` table:

  ```
  sp_unbindefault "employees.startdate"
  ```

- **Example 2 –** Unbinds the default from the user-defined datatype named `ssn` and all columns of that type:

  ```
  sp_unbindefault ssn
  ```

- **Example 3 –** Unbinds defaults from the user-defined datatype `ssn`, but does not affect existing columns of that type:

  ```
  sp_unbindefault ssn, futureonly
  ```

Usage

There are additional considerations when using **sp_unbindefault**:

- 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.

See also **create default**, **drop default** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_unbindefault** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the object owner or the user datatype owner.
Disabled	With granular permissions disabled, you must be the object owner.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**

Information	Values
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

Information	Values
Event	67
Audit option	**unbind**
Command or access audited	**sp_unbindefault**
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – NULL • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_bindefault* on page 80
- *sp_helptext* on page 438

sp_unbindexeclass

Removes the execution class attribute previously associated with an client application, login, stored procedure, or default execution class 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 you remove the association to the execution class. If the *object_type* is **DF**, *object_name* should be null.

- ***object_type* –** identifies the type of *object_name* as **AP**, **LG**, **PR** , or **DF** for application, login, stored procedure, or default execution class.
- ***scope* –** is the application name or login name for which the unbinding applies for an application or login. It is the stored procedure owner name (user name) for stored procedures. It is null for object type `DF`.

Examples

- **Example 1 –** 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`:

  ```
  sp_unbindexeclass 'sa', 'lg', 'isql'
  ```

Usage

There are additional considerations when using **sp_unbindexeclass**:

- The parameters must match an existing entry in the `sysattributes` system table.
- If you specify a null value for scope, the SAP ASE 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.
- When unbinding a stored procedure from an execution class, you must use the name of the stored procedure owner (user name) for the `scope` parameter.
- When unbinding a stored procedure from a user-defined default execution class, all tasks running with user-defined default execution class attributions run with attributes of system-defined default execution class `EC2`.
- 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 fails 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_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. If there is no use-defined default execution class, the object binds to class `EC2`.

See also **isql** in the *Utility Guide*.

Permissions

The permission checks for **sp_unbindexeclass** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be a user with `manage any execution class` privilege.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addexeclass* on page 23
- *sp_bindexeclass* on page 83
- *sp_dropexeclass* on page 266
- *sp_showexeclass* on page 652

sp_unbindmsg

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

- **Example 1 –** Unbinds a user-defined message from the constraint `positive_balance`:

  ```
  sp_unbindmsg positive_balance
  ```

Usage

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**.

Permissions

You must be the constraint owner to execute **sp_unbindmsg**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

Information	Values
Event	69
Audit option	**unbind**
Command or access audited	**sp_unbindmsg**

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – NULL • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_addmessage* on page 35
- *sp_bindmsg* on page 86
- *sp_getmessage* on page 353

sp_unbindrule

Unbinds a rule from a column or from a user-defined datatype.

Syntax

```
sp_unbindrule objname [, futureonly [, “accessrule” | “all”]]
```

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.
- **accessrule –** indicates that you are unbinding the access rule bound to *objname*.
- **all –** specifies that you are unbinding all rules bound to *objname*.

Examples

- **Example 1 –** Unbinds the rule from the `startdate` column of the `employees` table:

  ```
  sp_unbindrule "employees.startdate"
  ```

- **Example 2 –** Unbinds the rule from the user-defined datatype named `def_ssn` and all columns of that type:

  ```
  sp_unbindrule def_ssn
  ```

- **Example 3 –** The user-defined datatype `ssn` no longer has a rule, but existing `ssn` columns are unaffected:

  ```
  sp_unbindrule ssn, futureonly
  ```

- **Example 4 –** You can use the **all** parameter to unbind both accesss rules and domain rules. For example, to unbind all the access rules and domain rules on the `publishers` table:

  ```
  sp_unbindrule publishers, null, "all"
  ```

  To unbind the access rule from a user-defined datatype for subsequent uses of this datatype, issue:

  ```
  sp_unbindrule def_ssn, futureonly, "accessrule"
  ```

  To unbind both access rules and domain rules for subsequent uses of this datatype, issue:

  ```
  sp_unbindrule def_ssn, futureonly, "all"
  ```

- **Example 5 –** This access rule is bound to the `publishers` table:

  ```
  sp_bindrule empl_access, "publishers.pub_id"
  ```

  To unbind this rule, issue:

  ```
  sp_unbindrule "empl_access", NULL, "accessrule"
  ```

Usage

There are additional considerations when using **sp_unbindrule**:

- 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.

See also **create rule**, **drop rule** in *Reference Manual: Commands*.

Permissions

You must be the table owner or datatype owner to execute **sp_unbindrule**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_bindrule* on page 88
- *sp_helptext* on page 438

sp_version

Returns the version information of the installation scripts (`installmaster`, `installdbccdb`, and so on) that was last run and whether it was successful.

Syntax

```
sp_version [script_file, [all]]
```

Parameters

- ***script_file* –** is the name of the installation script (the default value is NULL).
- **all –** reports details about the installation scripts, such as the date it was run and the time it took to run.

Examples

- **Example 1 –** Returns the script name, version, and status of all installation scripts that have been run:

```
sp_version
```

```
Script          Version
Status
-----------      ------------------------------------------------
----------------------
installmaster    15.0/EBF XXXXX/B/Sun_svr4/OS 5.8/asemain/1/32-
bit/OPT/Thu
```

```
Sep 23 22:12:12 2004
Complete
installmaster    15.0/EBF XXXXX/B/Sun_svr4/OS 5.8/asemain/1/32-
bit/OPT/Thu
Sep 23 22:12:12 2004
Complete
installmodel    15.0/EBF XXXXX/B/Sun_svr4/OS 5.8/asemain/1861/32-
bit/OPT/Mon Sep 27 23:40:02 2004
Complete
```

- **Example 2 –** Returns information about the `installmaster` installation script:

```
sp_version        installmaster
```

```
-----------      -------------------------------------------------
---------
installmaster    15.0/EBF XXXXX/B/Sun_svr4/OS 5.8/asemain/1/32-
bit/OPT/Thu Sep 23 22:12:12 2004
Complete
```

- **Example 3 –** Returns script file name, date, time, version, and status for all the installation scripts run:

```
sp_version  null, 'all'
```

```
Script
Version           Status
Start/End Date
-----------------------------------------------------------------
----
installdbccdb     15.0/EBF XXXXX/B/Sun_svr4/OS 5.8/asemain/
1861/32-
bit/OPT/Mon Sep 27 23:40:02 2004
Complete [Started=Sep 29 2004  4:41PM]-[Completed=Sep 29 2004
4:42PM]
installmaster
15.0/EBF XXXXX/B/Sun_svr4/OS 5.8/asemain/1/32-bit/OPT/Thu Sep 23
22:12:
12 2004
Complete [Started=Sep 29 2004  3:49PM]-[Completed=Sep 29 2004
3:58PM]
installmodel
15.0/EBF XXXXX/B/Sun_svr4/OS 5.8/asemain/1861/32-bit/OPT/Mon Sep
27 23:
40:02 2004
Complete [Started=Sep 29 2004  4:51PM]-[Completed=Sep 29 2004
4:51PM]
```

- **Example 4 –** Returns script file name, version, and status of installation of all the install scripts having names like *install%*:

```
sp_version 'install%'
```

```
Script
Version           Status
----------        ------------------------------------------------
--------
installdbccdb
15.0/EBF XXXXX/B/Sun_svr4/OS 5.8/asemain/1861/32-bit/OPT/Mon Sep
```

```
27
23:40:02 2004     Complete
installmaster
15.0/EBF XXXXX/B/Sun_svr4/OS 5.8/asemain/1/32-bit/OPT/Thu Sep 23
22:12:
12 2004           Complete
installmodel
15.0/EBF XXXXX/B/Sun_svr4/OS 5.8/asemain/1861/32-bit/OPT/Mon Sep
27 23:
40:02 2004         Complete
```

- **Example 5 –** Returns all detailed information about installation scripts matching the wildcard "install%":

  ```
  sp_version 'install%', 'all'
  ```

  ```
  Script
  Version           Status
  Start/End Date
  -----------------------------------------------------------------
  ----
  installmaster
  15.0/EBF XXXXX/B/Sun_svr4/OS 5.8/asemain/1/32-bit/OPT/Thu Sep 23
  22:12:
  12 2004
  Complete [Started=Sep 29 2004  3:49PM]-[Completed=Sep 29 2004
  3:58PM]
  ```

- **Example 6 –** Returns all detailed information about the `installmaster` installation script:

  ```
  sp_version 'installmaster', 'all'
  ```

  ```
  Script
  Version           Status
  Start/End Date
  -----------------------------------------------------------------
  ----
  installmaster
  15.0/EBF XXXXX/B/Sun_svr4/OS 5.8/asemain/1/32-bit/OPT/Thu Sep 23
  22:12:
  12 2004
  Complete [Started=Sep 29 2004  3:49PM]-[Completed=Sep 29 2004
  3:58PM]
  ```

Usage

sp_version allows you to determine the current version of the scripts (`installmaster`, `installdbccdb`, and so on) installed on the SAP ASE server, and whether they ran successfully or not, and the time they took to complete

Permissions

Any user can execute **sp_version**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

sp_volchanged

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 SAP ASE 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 with contents that 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

- **Example 1 –** The operator changes the tape, then issues:

  ```
  sp_volchanged 8, "/dev/nrmt4", RETRY
  ```

  This 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' }
  ```

Permissions

Any user can execute **sp_volchanged**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure

Information	Values
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

When Backup Server Detects a Problem

There are additional considerations when using **sp_volchanged**.

If the Backup Server detects a problem with the currently mounted volume, it requests a volume change:

- (UNIX) 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 SAP ASE server that can communicate with the Backup Server performing the dump or load. The operator does not have to log into the SAP ASE server on which the dump or load originated.

See also:

- **dump database**, **dump transaction**, **load database**, **load transaction** in *Reference Manual: Commands*
- **isql** in the *Utility Guide*

Changing Tape Volumes on UNIX

The Backup Server requests a volume change when the tape capacity has been reached. The operator mounts another tape and executes **sp_volchanged**.

The following table illustrates this process.

Table 18. Changing Tape Volumes on a UNIX System

Sequence	Operator, Using isql	SAP ASE Server	Backup Server
1	Issues the **dump database** command		
2		Sends dump request to Backup Server	

Sequence	Operator, Using isql	SAP ASE Server	Backup Server
3			**1.** Receives dump request message from the SAP ASE server **2.** Sends message for tape mounting to operator **3.** Waits for operator's reply
4	**1.** Receives volume change request from Backup Server **2.** Mounts tapes **3.** Executes **sp_volchanged**		
5			**1.** Checks tapes **2.** If tapes are okay, begins dump **3.** When tape is full, sends volume change request to operator
6	**1.** Receives volume change request from Backup Server **2.** Mounts tapes **3.** Executes **sp_volchanged**		
7			**1.** Continues dump **2.** When dump is complete, sends messages to operator and the SAP ASE server
8	**1.** Receives message that dump is complete **2.** Removes and labels tapes	**1.** Receives message that dump is complete **2.** Releases locks **3.** Completes the **dump database** command	

sp_webservices

Creates and manages the proxy tables used in the SAP ASE Web Services Engine.

Syntax

To create a proxy table:

```
sp_webservices 'add', 'wsdl_uri' [, sds_name]
    [, 'method_name=proxy_table
     [,method_name=proxy_table ]* ' ]
```

To display usage information for **sp_webservices**:

```
sp_webservices help [, ’option’]
```

To list the proxy tables mapped to a WSDL file:

```
sp_webservices 'list' [, 'wsdl_uri'] [, sds_name]
```

To modify timeout setting:

```
sp_webservices 'modify', 'wsdl_uri', 'timeout=time'
```

To remove proxy tables mapped to a WSDL file:

```
sp_webservices 'remove', 'wsdl_uri' [, sds_name]
```

Options for User-Defined Web Services:

- To create a database alias for user-defined Web services:
  ```
  sp_webservices 'addalias' alias_name , database_name
  ```
- To deploy a user-defined Web service:
  ```
  sp_webservices 'deploy', ['all' | 'service_name']
  ```
- To drop a database alias in user-defined Web services:
  ```
  sp_webservices 'dropalias' alias_name
  ```
- To list the proxy tables mapped to a WSDL file in user-defined Web services:
  ```
  sp_webservices 'listudws' [, 'service_name']
  ```
- To list a database alias or aliases for a user-defined Web service.
  ```
  sp_webservices 'listalias'
  ```
- To undeploy a user-defined Web service:
  ```
  sp_webservices 'undeploy', ['all' | 'service_name']
  ```

Parameters

- **'add', '*wsdl_uri*' [, *sds_name*] [, '*method_name*=*proxy_table*[, *method_name*=*proxy_table*]* '] –** is used to create a proxy table for a Web method specified by a WSDL file. When the **add** option is used successfully, the **list** option is invoked automatically to describe the schema of the new proxy table:

    - *wsdl_uri* – is the location for the WSDL file to be mapped to the new proxy table. If this parameter is specified, Web Services ensures that the URI exists in the `syswsdl` table.
    - *sds_name* – is the name specified for the ASE Web Services Engine in the `interfaces` or `sql.ini` file. The default value is **ws**. If no entry exists in the `sysattributes` table, an error results.
    - *method_name* – is the name of the Web method to be mapped to a proxy table. The *method_name* specified must be the name of a Web method specified in the associated WSDL file.
    - *proxy_table* – is the name of proxy table to which the Web method specified in *method_name* is mapped.
- **'addalias' *alias_name* , *database_name* –** is used to create an alias representing a database name in user-defined Web services, where:

    - *alias_name* – (required) is the alias for the specified database.
    - *database_name* – (required) is the name of the database for which the alias is specified.

  An alias provides greater control in specifying the portion of the URL representing the database name. Used with the **userpath** option of the **create service** command, an alias provides complete control over the URL used to access a user-defined Web service.
- **'deploy', ['all' | '*service_name*'] –** is used to deploy a user-defined Web service, making it accessible to the ASE Web Services Engine through HTTP or HTTPS, where:

    - **all** – specifies that all user-defined Web services are to be deployed for the current database.
    - *service_name* – is the name of the user-defined Web service to be deployed.

  The **deploy** and **undeploy** options are used to control when user-defined Web services are available. The system role `webservices_role` privilege is required for this option.

  If the **all** parameter is specified, the ASE Web Services Engine deletes its internal cache of user-defined Web services and rereads all metadata about user-defined Web services from SAP ASE.

  You cannot drop or rename a user-defined Web service that is currently deployed.
- **'dropalias' *alias_name* –** is used to drop an alias representing a database name, where *alias_name* is the alias to be dropped.

  You cannot drop an alias if it is being referenced by a deployed user-defined Web service. To drop the alias, undeploy the user-defined Web service that references the alias first.
- **help[, '*option*'] –** provides instructions and examples illustrating how to use the **sp_webservices** stored procedure. The valid values for **'*option*'** are **add**, **list**, **remove**, and **modify**.

If you do not specify a value for *option*, the **help** option prints a brief syntax description for the **add**, **addalias**, **deploy**, **dropalias**, **list**, **listalias**, **listudws**, **modify**, **remove**, and **undeploy** options.

- **'list' [, '*wsdl_uri*'] [, *sds_name*] –** lists Web methods described in a WSDL file, where:

  - *wsdl_uri* – is the URI for the mapped WSDL file. If you do not specify a value for *wsdl_uri*, the **list** option displays information about all Web methods that have been mapped to proxy tables.
  - *sds_name* – is the name of the SDS server specified for the ASE Web Services Engine in the `interfaces` or `sql.ini` file. The default value is **ws**. If no entry exists in the `sysattributes` table, an error results.

  If you specify neither the *wsdl_uri* nor the *sds_name* parameter, all entries in the `sysattributes` table are listed, ordered by `wsdlid`.

  If the Web methods described in the WSDL file:

  - Have already been mapped to proxy tables – the **list** option prints information about each proxy table.
  - Have **not** already been mapped to proxy tables – the **list** option prints SQL that can be used to create proxy tables.
- **'listalias' –** is used to list all aliases in user-defined Web services.
- **'listudws' [, '*service_name*'] –** is used to list user-defined Web services for the current database, where *service_name* is the name of the user-defined Web service to be listed.

  If you do not specify the *service_name* parameter, all user-defined Web services are listed.
- **'modify', '*wsdl_uri*', 'timeout=*time*' –** is used to modify the attribute information for a WSDL file, where:

  - *wsdl_uri* – is the URI of the WSDL file for which attribute information is to be changed.
  - *time* – is the interval in seconds during which a Web method must respond before the operation is aborted.
- **'remove', '*wsdl_uri*' [, *sds_name*] –** is used to remove a proxy table mapping for a Web method, where:

  - *wsdl_uri* – is the URI of the WSDL file for which the proxy table is to be removed.
  - *sds_name* – is the name of the SDS server specified for the ASE Web Services Engine in the `interfaces` or `sql.ini` file. The default value is **ws**.

  Note: An error results if no entry exists in the `sysattributes` table.
- **'undeploy', ['all' | '*service_name*'] –** is used to make a user-defined Web service inaccessible to the SAP ASE Web Services Engine through HTTP or HTTPS, where:

  - **all** – specifies that all user-defined Web services are to be undeployed for the current database.

- *service_name* – is the name of the user-defined Web service to be undeployed.

Use the **deploy** and **undeploy** options to control when user-defined Web services are available. The system role `webservices_role` privilege is required for this option.

Examples

- **Example 1 –** Invokes an RPC/encoded Web method to display the exchange rate between two currencies.

  1. Use the **add** option of **sp_webservices** to map Web methods to proxy tables:

     ```
     1> sp_webservices 'add', 'http://www.xmethods.net/sd/2001/
     CurrencyExchangeService.wsdl'
     2> go
     ```

     The **getRate** Web method is mapped to a proxy table of the same name.
  2. Invoke the Web method by selecting from the proxy table:

     ```
     1> select * from getRate where _country1 ='usa' and _country2 =
     'india'
     2> go
     ```

     The results returned for the previous **select** show the exchange rate for the specified parameters:

     ```
     Result          _country1       _country2
     43.000000       usa             india
     (1 row affected)
     ```

- **Example 2 –** Invokes a Web method to display stock information within an XML document.

  1. Use the **add** option of **sp_webservices** to map Web methods to proxy tables:

     ```
     1> sp_webservices "add" , "http://www.webservicex.net/
     stockquote.asmx?WSDL"
     2> go
     ```

     The **GetQuote** Web method is mapped to a proxy table of the same name.
  2. Invoke the Web method by selecting the `outxml` column of the `GetQuote` proxy table:

     ```
     1> select outxml from GetQuote where _inxml = '<?xml
     version="1.0"
     encoding="utf-8"?>
     2>      <GetQuote xmlns="http://www.webserviceX.NET/">
     3>        <symbol>SY</symbol>
     4>      </GetQuote>'
     5> go
     ```

     The results for the previous **select** display quote information within an XML document:

     ```
      outxml

     <?xml version="1.0" encoding="UTF-8" ?><GetQuoteResponse
     ```

```
xmlns="http://
www.webserviceX.NET/"><GetQuoteResult><StockQuotes><Stock>
<Symbol>SY</Symbol><Last>21.48</Last><Date>7/21/2005</
Date><Time>4:01pm
</Time><Change>+1.72</Change><Open>20.00</Open><High>21.60</
High>
<Low>19.91</Low><Volume>2420100</Volume><MktCap>1.927B</
MktCap>
<PreviousClose>19.76</PreviousClose><PercentageChange>+8.70%
</PercentageChange><AnnRange>12.75 - 20.44</
AnnRange><Earns>0.706</Earns>
<P-E>27.99</P-E><Name>SYBASE INC</Name></Stock></StockQuotes>
</GetQuoteResult></GetQuoteResponse>

(1 row affected)
```

- **Example 3 –** Invokes the **GetQuote** Web method, mapped to a proxy table in the previous example, through a view to display stock information.

  1. Create a table to hold symbols representing stocks to use this Web service:

     ```
     1> create table stocksymbol(symbol varchar(100))
     2> go
     ```

  2. Insert data into the `stocksymbol` table:

     ```
     1> insert stocksymbol values("SY")
     2> insert stocksymbol values("ORCL")
     3> go
     ```

  3. Create a view that invokes the **GetQuote** Web method:

     ```
     1> CREATE VIEW getstockvw as
     2> select Symbol = xmlextract('//Stock/Symbol/text()',outxml
     returns varchar(5)),
     3>   Name = xmlextract('//Stock/Name/text()',outxml returns
     varchar(20)),
     4>   Time = xmlextract('//Stock/Time/text()',outxml returns
     varchar(10)),
     5>   Date = xmlextract('//Stock/Date/text()',outxml returns
     date),
     6>   High = xmlextract('//Stock/High/text()',outxml returns
     decimal(15,2)),
     7>   Low = xmlextract('//Stock/Low/text()',outxml returns
     decimal(15,2))
     8> FROM GetQuote ,stocksymbol
     9> WHERE _inxml = '<GetQuote xmlns="http://
     www.webserviceX.NET/"><symbol>'+symbol+'</symbol></GetQuote>'
     10> go
     ```

  4. Select from the `getstockvw` view to view output from the **GetQuotes** method:

     ```
     1> select * from getstockvw
     2> go
     ```

     The results for the previous **select** display quote information for the parameters specified by the view definition:

     ```
     Symbol   Name             Time      Date           High     Low
     -------  -------------    -------   ------------   ------   ---
     ---
     ```

```
SY       SYBASE INC      4:01pm    Jul 21 2005    21.60    19.91
ORCL     ORACLE CORP     4:00pm    Jul 21 2005    14.05    13.54
MSFT     MICROSOFT CP    4:00pm    Jul 21 2005    26.48    26.19

(3 rows affected)
```

- **Example 4 –** Shows an audit table entry for the following command entered in the `pubs2` database by the user "bob":

  ```
  sp_webservices 'deploy', 'all'
  ```

  The corresponding audit table entry lists `110`, `bob`, and `pubs2` as values in the `event`, `loginname`, and `dbname` columns, respectively. The `extrainfo` column contains the following:

  ```
  webservices_role; deploy_all; ; ; ; ; bob/ase;
  ```

- **Example 5 –** Shows an audit table entry for the following command entered in the `pubs2` database by the user "bob":

  ```
  sp_webservices 'deploy', 'rawservice'
  ```

  The corresponding audit table entry lists `110`, `bob`, and `pubs2` as values in the `event`, `loginname`, and `dbname` columns, respectively. The `extrainfo` column contains the following:

  ```
  webservices_role; deploy; ; ; ; ; bob/ase;
  ```

- **Example 6 –** Shows an audit table entry for the following command entered in the `pubs2` database by the user "bob":

  ```
  sp_webservices 'undeploy', 'all'
  ```

  The corresponding audit table entry lists `111`, `bob`, and `pubs2` as values in the `event`, `loginname`, and `dbname` columns, respectively. The `extrainfo` column contains the following:

  ```
  webservices_role; undeploy_all; ; ; ; ; bob/ase;
  ```

- **Example 7 –** Shows an audit table entry for the following command entered in the `pubs2` database by the user "bob":

  ```
  sp_webservices 'undeploy', 'rawservice'
  ```

  The corresponding audit table entry lists `111`, `bob`, and `pubs2` as values in the `event`, `loginname`, and `dbname` columns, respectively. The `extrainfo` column contains the following:

  ```
  webservices_role; deploy; ; ; ; ; bob/ase;
  ```

  For a full description of `sysaudits` table columns, see the *System Administration Guide*.

Usage

If you not specify *method_name* and *proxy_table* values for a Web method, the proxy table generated for the Web method is, by default, the name of the Web method specified in the

WSDL file. If there is already a proxy table with the name of this Web method, a new proxy table is generated with a name like:

```
method_nameN
```

Where:

- *method_name* – is the default proxy table name
- *N* – is a digit from 1 to 9 denoting each successive mapping of the Web method. There can be as many as 99 duplicate proxy tables.

If you do specify *method_name* and *proxy_table* values for a Web method, the name of the proxy table must be new. If there is already a proxy table with the name specified in *proxy_table*, an error results, and none of the Web methods specified in the **add** option are mapped to proxy tables.

The output from the **add** option lists the methods that have been successfully mapped to proxy tables as well as those that have not been mapped. The name of a proxy table for an unmapped Web method is indicated as NULL in the output from the **add** option.

Note: The columns used for input and output vary for proxy tables generated for RPC/encoded Web methods and document/literal Web methods. A proxy table representing an RPC/encoded Web method contains a column for each input and output parameter. A proxy table representing a document/literal Web method contains two columns, `_inxml` and `outxml`.

See also:

- **create service** in *Reference Manual: Commands*
- *Web Services User's Guide*

Permissions

You must be a user with **webservices_role** (for **deploy** and undeploy) to execute **sp_webservices**. Permission checks do not differ based on the granular permissions settings.

The system role `webservices_role` is required to use the **deploy** and **undeploy** options for **sp_webservices**. To execute a user-defined Web service, a valid login and permissions to execute the corresponding stored procedure are required.

To create, drop, and execute user-defined Web services, you need the same privileges as are necessary to create, drop, and execute stored procedures in SAP ASE. See the *System Administration Guide* for details on how to set the proper privileges using the **grant** and **revoke** commands.

Auditing

- Audit event number 110 corresponds to the **deploy** option of **sp_webservices**.
- Audit event number 111 corresponds to the **undeploy** option of **sp_webservices**.

User-defined Web services are modeled as stored procedures within SAP ASE. In manipulating user-defined Web services, SAP ASE generates the following events using the existing auditing coverage for stored procedures:

- The creation of a user-defined Web service – Event 11 named "Create Procedure" is generated
- The dropping of a user-defined Web service – Event 28 named "Drop Procedure" is generated
- The execution of a user-defined Web service – Event 38 named "Execution of Stored Procedure" is generated

For detailed information on existing auditing functionality, see the *System Administration Guide*.

In addition to existing auditing functionality, SAP ASE provides two audit events for the **deploy** and **undeploy** options of **sp_webservices**.

Audit records are stored in the `sysaudits` system table. You can enable auditing for Web services with the following command:

```
sp_audit "security", "all", "all", "on"
```

sp_who

Reports information about all current SAP ASE users and processes or about a particular user or process. Includes the `thread_pool` column, which describes the thread pool the SAP ASE server uses to execute a task.

Considerations for process mode

sp_who does not include the `threadpool` column.

Syntax

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

Parameters

- ***loginame*** **–** is the SAP ASE 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 (the SAP ASE server expects a `char` type).

Examples

- **Example 1 –** Reports on the processes running on the SAP ASE server. Although no user processes other than **sp_who** are running, the server still shows activity. During idle cycles, the housekeeper wash task moves dirty buffers into the buffer wash region, the housekeeper chores task performs other maintenance tasks. The housekeeper garbage collection task , which cleans up data that was logically deleted and resets the rows so that tables have space again, operates at the priority level of the ordinary user.

```
sp_who
```

```
fid spid  status   loginame   origname   hostname          blk_s
pid  dbname
   tempdbname   cmd              block_xloid    threadpool
--- ----  --------- --------- ---------  ----------------   -----
---  -----------
   ----------- ----------------- -----------    -----------------
  0    2   sleeping      NULL      NULL              NULL
  0       master
         tempdb     DEADLOCK TUNE           0    syb_default_pool
  0    3   sleeping      NULL      NULL              NULL
  0       master
   tempdb           ASTC HANDLER            0    syb_default_pool
  0    4   sleeping      NULL      NULL              NULL
  0       master
   tempdb       CHECKPOINT SLEEP            0    syb_default_pool
  0    5   sleeping      NULL      NULL              NULL
  0       master
   tempdb                HK WASH            0    syb_default_pool
  0    6   sleeping      NULL      NULL              NULL
  0       master
   tempdb                  HK GC            0    syb_default_pool
  0    7   sleeping      NULL      NULL              NULL
  0       master
   tempdb              HK CHORES            0    syb_default_pool
  0    8   sleeping      NULL      NULL              NULL
  0       master
   tempdb           PORT MANAGER            0    syb_default_pool
  0    9   sleeping      NULL      NULL              NULL
  0       master
   tempdb        NETWORK HANDLER            0    syb_default_pool
  0   10   sleeping      NULL      NULL              NULL
  0       master
   tempdb      LICENSE HEARTBEAT            0    syb_default_pool
  0   13   sleeping      NULL      NULL              NULL
  0       master
   tempdb        NETWORK HANDLER            0    syb_default_pool
  0   14   sleeping      NULL      NULL              NULL
  0       master
   tempdb        NETWORK HANDLER            0    syb_default_pool
  0   17   sleeping      NULL      NULL              NULL
  0       master
   tempdb        NETWORK HANDLER            0    syb_default_pool
  0   20   sleeping      NULL      NULL              NULL
  0       master
   tempdb        NETWORK HANDLER            0    syb_default_pool
  0   26   running         sa        sa  tiger.sybase.com
  0       master
   tempdb                  INSERT           0    syb_default_pool
```

- **Example 2 –** Reports on the processes running on the SAP ASE server. Process 11 (a **select into** on a table) is blocked by process 8 (a **begin transaction** followed by an **insert** on the same table). For process 8, the current *loginame* is "robert", but the original *loginame* is "sa". Login "sa" executed a **set proxy** command to impersonate the user "robert":

```
sp_who
```

```
fid spid  status      loginame   origname   hostname          blk
_spid  dbname
   tempdbname  cmd               block_xloid    threadpool
--- ----  ---------   ---------  ---------  ---------------   ---
-----  -----------
   ----------- ----------------- -----------    -----------------
  0    1 recv sleep        bird       bird            jazzy
         0       master
        tempdb  AWAITING COMMAND           0     syb_default_pool
  0    2   sleeping        NULL       NULL
    0       master
        tempdb  NETWORK HANDLER            0     syb_default_pool
  0    3   sleeping        NULL       NULL
    0       master
        tempdb  MIRROR HANDLER             0     syb_default_pool
  0    4   sleeping        NULL       NULL
    0       master
        tempdb  AUDIT PROCESS              0     syb_default_pool
  0    5   sleeping        NULL       NULL
    0       master
        tempdb  CHECKPOINT SLEEP           0     syb_default_pool
  0    6 recv sleep        rose                 rose
petal         0       master
        tempdb  AWAITING COMMAND           0     syb_default_pool
  0    7   sleeping        NULL       NULL             actor
    0  sybsystemdb
        tempdb  ASTC HANDLER               0     syb_default_pool
  0    8    running      robert          sa             helos
    0       master
        tempdb  SELECT                     0     syb_default_pool
  0    9  send
sleep      daisy     daisy            chain        0        pubs2
        tempdb  SELECT                     0     syb_default_pool
  0   10 alarm
sleep       lily      lily             pond        0       master
        tempdb  WAITFOR                    0     syb_default_pool
  0   11  lock
sleep      viola     viola            cello        8        pubs2
        tempdb  INSERT                     0     syb_default_pool
```

- **Example 3 –** Reports on the processes being run by the user "joe":

```
sp_who joe
```

```
fid spid  status    loginame   origname   hostname           blk_s
pid  dbname
   tempdbname   cmd              block_xloid    threadpool
--- ----  --------- --------- ---------  ----------------   -----
---  -----------
   ----------- ---------------- -----------    ----------------
  0   28  recv
sleep      joe       joe  tiger.sybase.com         0        pubs2
        tempdb        SELECT          0         syb_default_pool
```

- **Example 4 –** Reports what the SAP ASE server process number 17 is doing:

```
sp_who "17"
```

```
fid spid  status   loginame   origname   hostname          blk_s
pid  dbname
   tempdbname   cmd              block_xloid    threadpool
--- ----  --------- --------- ---------  ----------------   -----
---  -----------
   ----------- ---------------- -----------    ----------------
  0   17   sleeping      NULL      NULL             NULL
  0        pubs2
         tempdb     NETWORK
HANDLER            0       syb_default_pool
```

- **Example 5 –** Reports on a system-induced rollback, either of a transaction or a command:

```
sp_who
```

```
fid spid  status   loginame   origname   hostname          blk_s
pid  dbname
   tempdbname   cmd              block_xloid    threadpool
--- ----  --------- --------- ---------  ----------------   -----
---  -----------
   ----------- ---------------- -----------    ----------------
  0   28    running       joe       joe  tiger.sybase.com
  0        pubs2
       tempdb       rollback            0      syb_default_pool
```

Usage

There are additional considerations when using **sp_who**:

- **sp_who** reports information about a specified user or the SAP ASE server process.
- Without parameters, **sp_who** reports which users are running what processes in all databases.
- The columns returned by **sp_who** are:
  - **`fid`** – Identifies the family (including the coordinating process and its worker processes) to which a lock belongs. For more information, see **sp_familylock**.
  - **`spid`** – Identifies the process number. A system administrator can use this number with the Transact-SQL **kill** command to stop the process.
  - **`status`** – Indicates whether the process is running or sleeping.
  - **`loginame`** – The login or alias of the user who started the process. For all system processes, `loginame` is NULL.

  - **`origname`** – If the `loginame` is an alias, `origname` shows the real login name. If not, `origname` shows the same information as `loginame`.
  - **`hostname`** – The name of the server on which the database resides.
  - **`blk_spid`** – 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.
  - **`dbname`** – Indicates the name of the database on which the process is running.
  - **`tempdb`** – Temporary database assigned to the session.
  - **`cmd`** – Identifies the command or process currently being executed. Evaluation of a conditional statement, such as an **if** or **while** loop, returns `cond`.
  - **`block_xloid`** – Identifies the unique lock owner ID of a blocking transaction.
  - **`threadpool`** – Thread pool the task uses.
- Running **sp_who** on a single-engine server shows the **sp_who** process currently running and all other processes that are runnable or in one of the sleep states. In multiengine servers, there can be a "running" process for each engine.
- If you enable mirrored disks or remote procedure calls, the mirror handler and the site handler also appear in the report from **sp_who**.

See also **kill** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_who**. Permission checks do not differ based on the granular permissions settings.

Auditing

Values in `event` and `extrainfo` columns from the `sysaudits` table are:

Information	Values
Event	38
Audit option	**exec_procedure**
Command or access audited	Execution of a procedure
Information in `extrainfo`	• *Roles* – Current active roles • *Keywords or options* – NULL • *Previous value* – NULL • *Current value* – NULL • *Other information* – All input parameters • *Proxy information* – Original login name, if **set proxy** in effect

See also

- *sp_familylock* on page 335
- *sp_lock* on page 483

sp_xact_loginfo

sp_xact_loginfo provides the span of oldest active transaction in terms of percentage of total log space.

Syntax

```
sp_xact_loginfo dbid [, vcharparam1] [, vcharparam2]
[, intparam1] [, intparam2]['span_pct] [, startpage]
[, xact_spid] [, starttime] [, firstlog_page] [, stp_page]
[, stp_pages] [, stp_blocking] [, canfree_without_abort_pct]
[, dump_in_progress] [, activexact] [, errorcode]
```

Parameters

- **dbid –** is the database ID.
- **vcharparam1 –** varchar parameter indicating the mode. If **oldestactive**, the output parameter values are indicative of oldest active transaction. If **xactspanbyspid**, then output parameter values reflect values of active transaction for given spid.
- **vcharparam2 –** reserved for future use. Provide NULL as a value.
- **intparam1 –** integer parameter1 (SPID if *vcharparam1* = **xactspanbyspid**)
- **intparam2 –** integer parameter2
- **span_pct –** a value from 0 to 100. Indicates the span of transaction in percentage of total log space based on value of *vcharparam1*(output parameter).
- **startpage –** page number that is the start of the active transaction in the log based on value of vcharparam1. This page will hold the begin transaction log record of the active transaction.
- **xact_spid –** server process ID of the client having the active transaction based on *vcharparam1*.
- **starttime –** start time of active transaction based on *vcharparam1*.
- **firstlog_page –** server process ID of the client having active transaction based on *vcharparam1*.
- **stp_page –** secondary truncation point logical page number in the log. Returns -1 if replication is not active.
- **stp_pages –** returns the total number of log pages between the secondary truncation point and the oldest active transaction. Returns 0 if:
  - Replication is not active
  - There is no active transaction in the log

  - There is no secondary truncation point before oldest active transaction
- **stp_blocking –** value of 0 or 1. 1 indicates that the secondary truncation checkpoint will block some portion for truncation beyond oldest active transaction span. Meaning that secondary truncation point is in between the start of the log and the start of oldest active transaction and replication agent must catch up. 0 indicates that aborting the oldest active transaction will free transaction log space without the secondary checkpoint blocking the abort.
- **canfree_without_abort_pct –** value from 0 to 100. Indicates the difference between **startlogpagenum** and **startxactpagenum** in terms of percentage of total log space. This portion can be truncated with the dump transaction command without aborting the oldest active transaction.
- **dump_in_progress –** returns1 if the dump transaction command is in progress, returns 0 if no dump command is in progress. Values of output parameters **firstlog_page** and **canfree_without_abort_pct** are not reliable. (output parameter).
- **activexact –** Boolean flag indicating that there are active transactions in the log.
- **errorcode –** Values are:

  - 0 there are no errors
  - 1 insufficient permission to execute
  - 2 error in opening dbtable. This could be due to various reasons including the dbid or database name given does not exist.
  - 3 cannot start xls session for log scan.
  - 4 there are no open transaction in the log against this database

Note: For a Mixed Log Data (MLD) database, this procedure returns values equivalent to 0 in output parameters. This procedure is not supported or meant to be used for MLD databases.

sp_xmlschema

Creates and maintains the `spt_xmlcatalog` user table in the SAP ASE database. `spt_xmlcatalog` stores schema definitions that the **xmlvalidate** function uses to validate XML documents

Syntax

See *XML Services* for syntax, examples, and usage information for **sp_xmlschema**.

CHAPTER 2 Catalog Stored Procedures

Catalog stored procedures retrieve information from the system tables in tabular form. Created by **installmaster** at installation, they are located in the `sybsystemprocs` database and are owned by the system administrator.

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 procedures report a return status. For example, this means that the procedure executed successfully. The examples in this book do not include the return status:

```
return status = 0
```

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. Doing so may produce unexpected results.

In many cases, it is more convenient to supply parameters to the catalog stored procedures in this form than to supply all the parameters:

```
@parametername = value
```

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_column** 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, the SAP ASE server ignores the extra parameters.

Pattern Matching

SAP ASE supports wildcards _, %, [], and [^]. For maximum interoperability, use only the % and _ wildcard characters as defined by ANSI SQL Standards.

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 such as status bits into human-readable format.

The catalog stored procedures **sp_column_privileges** and **sp_table_privileges** create and then drop temporary tables.

ODBC Datatypes

These two tables list the datatype code numbers and matching datatype names returned by **sp_columns** and **sp_column_privileges** in the `data_type` column. The source for the description is the Open Database Connectivity (ODBC) Application Programming Interface (API).

Table 19. Code Numbers for ODBC Datatypes

Datatype	Code #
`char`	1
`decimal`	3
`double precision`	8
`float`	6
`integer`	4
`numeric`	2
`real`	7

Datatype	Code #
`smallint`	5
`varchar`	12
`wchar`	-8
`wvarchar`	-9
`wlongvarchar`	-10

Table 20. Code Numbers for Extended Datatypes

Datatype	Code #
`bigint`	-5
`binary` (bit datatype)	-2
`bit`	-7
`date`	9
`java.lang.Object`	1111
`long univarchar`	-10
`long varbinary`	-4
`long varchar`	-1
`time`	10
`timestamp`	11
`tinyint`	-6
`unichar`	-8
`univarchar`	-9
`varbinary` (bit-varying datatype)	-3

sp_column_privileges

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 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 with permissions that 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

- **Example 1 –** This example displays information about the `discounts` table:

```
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
```

Usage

The results set for **sp_column_privileges** is:

Column	Data-type	Description
`table_qualifier`	`varchar(32)`	The name of the database in which the table specified for the *table_name* parameter is stored.
`table_owner`	`varchar(32)`	The table owner. If no value was specified for the *table_owner* parameter, this value is the current owner or the Database Owner.
`table_name`	`varchar(32)`	The name specified for the *table_name* parameter. This value cannot be NULL.
`column_name`	`varchar(32)`	The specified column name. If no column name was specified in the statement, the results include all columns in the specified table.
`grantor`	`varchar(32)`	The name of the database user who has granted permissions on `column_name` to `grantee`. This value cannot be NULL.
`grantee`	`varchar(32)`	The name of the database user who was granted permissions on `column_name` by `grantor`. This value cannot be 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 only for referential 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.

Permissions

Any user can execute **sp_column_privileges**.

sp_columns

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 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

- **Example 1 –** Displays information about all columns in the `publishers` table that begin with "p":

  ```
  sp_columns "publishers", null, null, "p%"
  ```

  ```
  table_qualifier  table_owner  table_name  column_name data_type t
  ype_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
  ```

- **Example 2 –** Displays information about all columns beginning with "st" in tables that begin with "s":

  ```
  sp_columns "s%", null, null, "st%"
  ```

Usage

The results set for **sp_columns** is:

Column	Datatype	Description
`table_qualifier`	`varchar(32)`	The name of the database in which the table specified for the *table_name* parameter is stored.
`table_owner`	`varchar(32)`	The table owner. If no value was specified for the *table_owner* parameter, this value is the current owner or the Database Owner.
`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 datatypes.
`nullable`	`smallint`	The value 1 means NULL is possible; 0 means NOT NULL.
`remarks`	`varchar(254)`	
`ss_data_type`	`smallint`	An SAP ASE datatype.
`colid`	`tinyint`	A column appended to the results set.
`column_def`	`varchar(255)`	NULL.

Column	Datatype	Description
`sql_data_type`	`smallint`	An SAP ASE datatype.
`sql_datetime_sub`	`smallint`	NULL.
`char_octet_length`	`int`	The value of `char_octet_length` is the same as the value for the `precision` column if the datatype for `char_octet_length` is: • `binary` • `char` • `image` • `nchar` • `nvarchar` • `sysname` • `text` • `timestamp` • `varbinary` • `varchar` Otherwise, the value of `char_octet_length` is 0.
`ordinal_position`	`int`	The ordinal position of the column in the table. The first column in the table is 1.
`is_nullable`	`varchar(3)`	Describes whether the column or parameter allows NULL as a value. From syscolumns.

sp_columns reports the `type_name` as float, and `data_type` as 6 for columns defined as `double precision`. The SAP ASE `double precision` datatype is a float implementation supports the range of values as specified in the ODBC specifications.

Permissions

Any user can execute **sp_columns**.

sp_databases

Returns a list of databases in the SAP ASE server.

Syntax

```
sp_databases
```

Examples

- **Example 1 –** Returns a list of databases in the server:

```
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
```

Usage

The results set for **sp_databases** is:

Column	Datatype	Description
`database_name`	`char(32)`	NOT NULL database name.
`database_size`	`bigint`	Size of database, in kilobytes.
`remarks`	`varchar(254)`	SAP ASE always returns NULL.

Permissions

Any user can execute **sp_databases**.

sp_datatype_info

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.

Usage

The results set for **sp_datatype_info** is:

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.
`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.

Permissions

Any user can execute **sp_datatype_info**.

sp_fkeys

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 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 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**.

Usage

The results set for **sp_fkeys** is:

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.

Column	Datatype	Description
`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 multicolumn 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.

There are additional considerations when using **sp_fkeys**:

- **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.
- 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 **sp_commonkey**, **sp_foreignkey**, or **sp_primarykey**.

See also **alter table**, **create table** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_fkeys**.

See also

- *sp_commonkey* on page 157
- *sp_foreignkey* on page 349
- *sp_primarykey* on page 589

sp_pkeys

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 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.

Usage

The results set for **sp_pkeys** is:

Column	Data-type	Description
`table_qualifier`	`varchar(32)`	The database name. This field can be NULL.
`table_owner`	`varchar(32)`	The table owner. If no value was specified for the *table_owner* parameter, this value is the current owner or the Database Owner.
`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 **sp_primarykey**.

The term *primary key* refers to a logical primary key for a table. The SAP ASE 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**.

See also **alter table**, **create table** in *Reference Manual: Commands*.

Permissions

Any user can execute **sp_pkeys**.

See also

- *sp_primarykey* on page 589
- *sp_statistics* on page 763

sp_server_info

Returns a list of SAP ASE attribute names and current values.

Syntax

```
sp_server_info [attribute_id]
```

Parameters

- ***attribute_id* –** is the integer ID of the server attribute.

Examples

- **Example 1 –** Returns a list of server attributes for attribute ID 12:

```
sp_server_info 12
```

```
attribute_id attribute_name           attribute_value
------------ ------------------------ --------------------------
          12 MAX_OWNER_NAME_LENGTH    0
```

- **Example 2 –** Returns the list of server attributes, described by the mandatory rows, and their values:

```
sp_server_info
```

Usage

The results set for **sp_server_info** is:

Column	Datatype	Description
`attribute_id`	`int`	NOT NULL.
`attribute_name`	`varchar(60)`	NOT NULL.
`attribute_value`	`varchar(255)`	

The mandatory rows in the results set returned by **sp_server_info** are:

ID	Server Attribute Name	Description	Value
1	DBMS_NAME	Name of the DBMS.	`SQL SERVER`
2	DBMS_VER	The value used by the ODBC driver to determine version compatibility. **Note:** Do not change the value of `DBMS_VER`. This is not the same as **@@version**, and the value must match the value used by the ODBC driver.	Actual value
6	DBE_NAME	Unused	
10	OWNER_TERM	SAP ASE server's term for a table owner (the second part of a three-part name).	`owner`
11	TABLE_TERM	SAP ASE server's term for a table (the third part of a three-part name).	`table`

ID	Server Attribute Name	Description	Value
12	MAX_OWN-ER_NAME_LENGTH	Maximum length of the name for a table owner (the second part of a three-part name).	`30`
13	TABLE_LENGTH	The maximum number of characters for a table name.	`30`
14	MAX_QUAL_LENGTH	Maximum length of the name for a table qualifier (the first part of a three-part table name).	`30`
15	COLUMN_LENGTH	The maximum number of characters for a column 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`
18	COLLATION_SEQ	The assumed ordering of the character set for this server.	
19	SAVEPOINT_SUPPORT	Does the underlying DBMS support named save-points?	`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`
22	ACCESSIBLE_TABLES	In **sp_tables**, 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`
100	USERID_LENGTH	The maximum number of characters for a user name.	`30`
101	QUALIFIER_TERM	SAP ASE server's term for a table qualifier (the first part of a three-part name).	`database`
102	NAMED_TRANSACTIONS	Does the underlying DBMS support named transactions?	`Y`
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`
104	ACCESSIBLE_SPROC	In **sp_stored_procedures**, does the gateway return only stored procedures that are executable by the current user?	`Y`

ID	Server Attribute Name	Description	Value
105	MAX_INDEX_COLS	Maximum number of columns in an index for the DBMS.	`32`
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_COL-UMN_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?	`Y`
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`

Permissions

Any user can execute **sp_server_info**.

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 `timestamp` columns, with values that 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 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 with values that uniquely identify any row in the table, or "V" to return information about `timestamp` columns, with values that are generated by the SAP ASE server each time a row is inserted or updated.

Examples

- **Example 1 –** Returns the optimal set of columns for `systypes`:

```
sp_special_columns systypes
```

```
scope  column_name  data_type  type_name  precision  length  scal
e
------ ------------ ---------- ---------- ---------- ------- ----
--
   0   name          12        varchar          30       30   NULL
```

- **Example 2 –** Returns the optimal set from the from the `authors` table with values that uniquely identify any row in the table:

```
sp_special_columns @table_name=authors, @col_type=R
```

```
scope  column_name  data_type  type_name  precision  length  scal
e
------ ------------ ---------- ---------- ---------- ------- ----
--
   0   au_id         12        varchar      11          11   NULL
```

Usage

The results set for **sp_special_columns** is:

Column	Datatype	Description
`scope`	`int`	NOT NULL. Actual scope of the row ID. The SAP ASE 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.
`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.

Permissions

Any user can execute **sp_special_columns**.

sp_sproc_columns

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 no owner is specified, **sp_sproc_columns** returns all columns.
- ***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.

Usage

The results set for **sp_sproc_columns** is:

Column	Datatype	Description
procedure_qualifier	varchar(30)	Procedure qualifier name. Can be NULL.
procedure_owner	varchar(30)	Procedure owner name. Always returns a value.
procedure_name	varchar(41)	Procedure name. Always returns a value.

Column	Datatype	Description
`column_name`	`varchar(30)`	Column name for each column of the *table_name* returned. Always returns a value.
`column_type`	`smallint`	
`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`	The 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)`	The description of the procedure column. NULL.
`ss_data_type`	`tinyint`	An SAP ASE datatype.
`colid`	`tinyint`	The column ID from `syscolumns`.
`column_def`	`varchar(255)`	NULL.
`sql_data_type`	`smallint`	An SAP ASE datatype.
`sql_datetime_sub`	`smallint`	NULL.

Column	Datatype	Description
`char_octet_length`	`int`	The value of `char_octet_length` is the same as the value for the `precision` column if the datatype for `char_octet_length` is: • `binary` • `char` • `image` • `nchar` • `nvarchar` • `sysname` • `text` • `timestamp` • `varbinary` • `varchar` Otherwise, the value of `char_octet_length` is 0.
`ordinal_position`	`int`	The ordinal position of the parameter in the parameter list. The first parameter in the list is 1, and return values have an ordinal.
`is_nullable`	`varchar(3)`	Describes whether the column or parameter allows NULL as a value. From `syscolumns`.
`mode`	`varchar(20)`	The parameter mode information stored in `syscolumns` that contains: • *For SQL procedures* – in, out, or "return value". • *For SQLJ procedures (Java)* – in, out, inout, or "return value".

sp_sproc_columns reports the `type_name` as float, and `data_type` as 6 for parameters defined as `double precision`. The SAP ASE `double precision` datatype is a float implementation supports the range of values as specified in the ODBC specifications.

Permissions

Any user can execute **sp_sproc_columns**.

sp_statistics

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 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

- **Example 1 –** Shows the list of indexes for `publishers`:

```
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
```

Usage

The results set for **sp_statistics** is:

Column	Data-type	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.

Permissions

Any user can execute **sp_statistics**.

sp_stored_procedures

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.

Usage

The results set for **sp_stored_procedures** is:

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.

Column	Data-type	Description
`remarks`	`var-char(254)`	NULL.

sp_stored_procedures returns information about stored procedures in the current database only.

sp_stored_procedures can return the name of stored procedures for which the current user does not have execute permission. However, 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.

Permissions

Any user can execute **sp_stored_procedures**.

See also

- *sp_server_info* on page 756

sp_table_privileges

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 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.

Usage

The results set for **sp_table_privileges** is:

Column	Data-type	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.
`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 rows containing data. • 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.

Permissions

Any user can execute **sp_table_privileges**.

sp_tables

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

- **Example 1 –** This example returns information about all tables in the current database of the type TABLE and VIEW and excludes information about system tables:

  ```
  sp_tables @table_type = "'TABLE', 'VIEW'"
  ```

Usage

The results set for **sp_tables** is:

Column	Datatype	Description
`table_qualifier`	`varchar(30)`	The database name. This field can be NULL.
`table_owner`	`varchar(30)`	

Column	Datatype	Description
`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

- The SAP ASE 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.

Permissions

Any user can execute **sp_tables**.

Tables used

`master.dbo.sysattributes`, `master.dbo.sysloginroles`, `master.dbo.syssrvroles`, `sysroles`

See also

- *sp_server_info* on page 756

CHAPTER 3 System Extended Stored Procedures

System extended stored procedures (ESPs) are supplied by SAP.

- ESPs are created by **installmaster** during the installation process. They are located in the `sybsystemprocs` database and owned by the system administrator. They can be run from any database.
- Permissions are set in the `sybsystemprocs` database.

  Users with the **sa_role** have default execution permissions on the system ESPs. These System Administrators can grant execution permissions to other users.
- You can get the names of the DLLs associated with the system ESPs by running **sp_helpextendedproc** in the `sybsystemprocs` database.
- 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. The SAP ASE server starts XP Server the first time an ESP is invoked. XP Server continues to run until you shut down the SAP ASE server.

xp_cmdshell

Executes a native operating system command on the host system running the SAP ASE server.

Syntax

```
xp_cmdshell command[, no_output] [return_status | no_wait]
```

Parameters

- ***command*** **–** is the operating system command string; maximum length is 8192 bytes.
- **no_output –** if specified, suppresses any output from the command.
- **return_status –** if specified, returns the completion status of the operating system command specified in the **command** parameter. If you do not use this parameter, the returned value is either 0 for success, or or 1 for failure, respectively.
- **no_wait –** if specified, the **xp_cmdshell** operation immediately returns to the caller and the specified command executes as a background process. You see no output, and the returned result reflects only the success or failure of starting the command as a background process, not the success or failure of the process itself.

Examples

- **Example 1 –** (On Windows) Silently copies the file named **log** on the C drive to a file named `log.0102` on the A drive:

  ```
  xp_cmdshell 'copy C:\log A:\log.0102', no_output
  ```
- **Example 2 –** (On UNIX) Executes the operating system's **ls** command and returns the list directory contents as a row of data:

  ```
  xp_cmdshell 'ls'
  ```

Usage

There are additional considerations when using **xp_cmdshell**:

- **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 the SAP ASE 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 the SAP ASE server. This account may have fewer restrictions than the user's own account.
- 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
  ```
- To find out if **xp_cmdshell** was successful in spawning an external command XP Server, enter the following, where *command* is the name of the command you ran with **xp_cmdshell**:

  ```
  exec @ret = xp_cmdshell command
  ```

  If **xp_cmdshell** was successful, **exec @ret = xp_cmdshell *command*** returns a value of 0. If **xp_cmdshell** failed, **exec @ret = xp_cmdshell *command*** returns a value of 1.
- To find out if the command you ran using **xp_cmdshell** was itself successful, enter the following, where *command* is the name of the command you ran with **xp_cmdshell**:

  ```
  exec @ret = xp_cmdshell command, return_status
  ```

  exec @ret = xp_cmdshell *command*, return_status causes **xp_cmdshell** to return the actual exit status code of the command. If a failure occurrs and XP Server cannot run the command, **xp_cmdshell** returns a value of 1. If the command runs successfully, **xp_cmdshell** returns a value of 0.

If the command was successful, **exec @ret = xp_cmdshell *command*** returns a value of 0. If the command failed, **exec @ret = xp_cmdshell *command*** returns a value of 1.

Note: Both **exec @ret = xp_cmdshell *command*** and **exec @ret = xp_cmdshell *command*, return_status** are backward-compatible. Old stored procedures that do not use the **return_status** parameter treat **exec @ret = xp_cmdshell *command*, return_status** as if it were **exec @ret = xp_cmdshell *command***.

Also, the **no_output** parameter can still be used in combination with **return_status**, in any order.

- You must use the `cmdstr` column name when you create a proxy table with the **xp_cmdshell** remote procedure:

```
create existing table xpoutput
(
       cmdstr varchar(255) null
)
external procedure at "THIS...xp_cmdshell"

select cmdstr from xpoutput where cmdstr = "date"
```

  If you do not use **cmdstr**, you see an error message.

  See *Remote Procedures as Proxy Tables* in the *Component Integration Services User's Guide* for more information about results returned from the proxy table.

See also *System Administration Guide*.

Permissions

By default, only a system administrator can execute **xp_cmdshell**. A system administrator can grant execute permission to other users.

xp_deletemail

(Windows only) Deletes a message from the SAP ASE message inbox.

Syntax

```
xp_deletemail [msg_id]
```

Parameters

- ***msg_id* –** is the message identifier of the mail message to be deleted.

Examples

- **Example 1 –** Deletes from the SAP ASE message inbox the message with the message identifier specified in the `cur_msg_id` variable:

```
1> declare @cur_msg_id binary(255)
2> exec xp_deletemail @msg_id = @cur_msg_id
```

- **Example 2 –** Deletes the first message from the SAP ASE message inbox:

```
xp_deletemail
```

Usage

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.

Permissions

By default, only a system administrator can execute **xp_deletemail**. A system administrator can grant this permission to other users.

See also

- *xp_findnextmsg* on page 775

xp_enumgroups

(Windows only) Displays groups for a specified Windows domain.

Syntax

```
xp_enumgroups [domain_name]
```

Parameters

- ***domain_name* –** is the Windows domain for which you are listing user groups.

Examples

- **Example 1 –** Lists all user groups on the Windows computer running XP Server:

```
xp_enumgroups
```

- **Example 2 –** Lists all user groups in the PCS domain:

```
xp_enumgroups 'PCS'
```

Usage

There are additional considerations when using **xp_enumgroups**:

- **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.

Permissions

By default, only a system administrator can execute **xp_enumgroups**. A system administrator can grant this permission to other users.

xp_findnextmsg

(Windows only) Retrieves the next message identifier from the SAP ASE 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 = {true | false} –** if this parameter is set to:
  - **true** (default) – **xp_findnextmsg** considers only unread messages.
  - **false** – **xp_findnextmsg** considers all messages, both read and unread, when retrieving the next message identifier.

Examples

- **Example 1 –** 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***:

  ```
  xp_findnextmsg @msg_id = @out_msg_id output
  ```

- **Example 2 –** 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:

```
xp_findnextmsg @msg_id = @out_msg_id output, NULL, @unread_only =
false
```

Usage

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**.

Permissions

By default, only a system administrator can execute **xp_findnextmsg**. A system administrator can grant this permission to other users.

See also

- *xp_deletemail* on page 773
- *xp_readmail* on page 777

xp_logevent

(Windows only) Provides for logging a user-defined event in the Windows Event Log from within the SAP ASE 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

- **Example 1 –** An informational event, number 55555, is logged in the Windows Event Log. The text of the description in the event detail window is “Email message deleted”:

  ```
  xp_logevent 55555, 'Email message deleted.'
  ```

- **Example 2 –** An error event, number 66666, is logged in the Windows Event Log. The text of the description in the event detail window is “DLL not found”:

  ```
  xp_logevent 66666, 'DLL not found.', 'error'
  ```

Usage

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 the SAP ASE server
Category	User

Permissions

Only a system administrator can execute **xp_logevent**.

xp_readmail

(Windows only) Reads a message from the SAP ASE 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 **suppress_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

- **Example 1 – 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:

```
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
```

- **Example 2 –** 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:

```
declare @msgid binary(255)
declare @mess varchar(255)
declare @msg_length char(255)
declare @len int
declare @skip int
```

```
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
```

Usage

xp_readmail reads a message from the SAP ASE message inbox.

To get the message identifier of the next message in the message inbox, use **xp_findnextmsg**.

Permissions

By default, only a system administrator can execute **xp_readmail**. A system administrator can grant this permission to other users.

See also

- *xp_findnextmsg* on page 775

xp_sendmail

(Windows only) 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}]
    [, no_output = {true | false}]
    [, width]
    [, separator]
    [, dbuser]
    [, dbname]
    [, type]
    [, include_query = {true | false}]
```

Parameters

- ***recipient* –** is the e-mail 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.
- ***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. For the complete list of options that are ignored when you use message, see the "Usage" section.
- ***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 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 the SAP ASE server messages, including the count of rows affected message, along with the query results. If set to **false**, does not send the SAP ASE 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

- **Example 1 – xp_sendmail** sends a text message on the backup status of an SAP ASE server to "sally" and "ramon" with a copy to the "admin" group:

```
xp_sendmail @recipient = "sally;ramon",
@subject = "Adaptive Server Backup Status",
@message = "Adaptive Server Backup for SERVER2 is complete.",
@copy_recipient="admin"
```

- **Example 2 –** 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:

```
xp_sendmail "peter",
@query = "select * from authors",
@attachname = "au_list.res",
@attach_result= true
```

Usage

These 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**.

Permissions

By default, only a system administrator can execute **xp_sendmail**. A system administrator can grant this permission to other users.

xp_startmail

(Windows only) Starts an SAP ASE mail session.

Syntax

```
xp_startmail [mail_user ] [, mail_password]
```

Parameters

- ***mail_user*** **–** is a mail profile name used by the SAP ASE server to log into the Windows mail system. If *mail_user* is not used, **xp_startmail** uses the mail user name that was used to set up the SAP ASE mail account.
- ***mail_password*** **–** is the mail password used by the SAP ASE server to log into the Windows mail system. If *mail_password* is not used, **xp_startmail** uses the mail password that was used to set up SAP ASE mail account.

Examples

- **Example 1 –** Starts an SAP ASE mail session using the mail user name and password for Sybmail's user account:

  ```
  xp_startmail
  ```

- **Example 2 –** Starts an SAP ASE mail session with "mailuser" as the profile name and the password associated with that profile name:

  ```
  xp_startmail "mailuser", "tre55uu"
  ```

Usage

Additional considerations when using **xp_startmail**:

- **xp_startmail** does not start an SAP ASE mail session if one is already running.
- An SAP ASE mail session must be started, either by an explicit call to **xp_startmail** or by configuring the SAP ASE server to start an SAP ASE mail session automatically at start-up, before any Sybmail-related system ESPs or the **sp_processmail** stored procedure can be executed. See **start mail session** in the *System Administration Guide* for information about initiating an SAP ASE mail session automatically at start-up.
- When the Windows **automail** session is not on, you must use the *mail_user* and *mail_password* parameters with **xp_startmail**.
- 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
  ```

Permissions

By default, only a system administrator can execute **xp_startmail**. A system administrator can grant this permission to other users.

See also

- *sp_displaylogin* on page 234
- *sp_processmail* on page 590

xp_stopmail

(Windows only) Stops an SAP ASE mail session.

Syntax

```
xp_stopmail
```

Examples

- **Example 1 –** Stops an SAP ASE mail session:

  ```
  xp_stopmail
  ```

Usage

You cannot execute SAP ASE mail-related system ESPs and the **sp_processmail** stored procedure after a mail session has been terminated with **xp_stopmail**.

Permissions

By default, only a system administrator can execute **xp_stopmail**. A system administrator can grant this permission to other users.

CHAPTER 4 dbcc Stored Procedures

dbcc stored procedures access the tables only in the `dbccdb` database or in the alternate database, `dbccalt`.

See the *System Administration Guide* for details on setting up `dbccdb` or `dbccalt`. See *dbccdb Tables* in *Reference Manual: Tables* for information on the tables used in these databases.

For details on the **dbcc** system procedure **sp_plan_dbccdb**, see **sp_plan_dbccdb**. See the *System Administration Guide* for more information on this system procedure and the **dbcc** stored procedures.

The permission checks for **dbcc** stored procedures differ based on your granular permissions settings. See the *Security Administration Guide* for more information on granular permissions.

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

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.

Examples

- **Example 1 –** Changes the size of the `scan_ws_000001` workspace on `dbccdb` to 30MB:

  ```
  sp_dbcc_alterws dbccdb, scan_ws_000001, "30M"
  ```

  ```
  Workspace scan_ws_000001 has been altered successfully to size
  30MB
  ```

Usage

There are additional considerations when using **sp_dbcc_alterws**:

- **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**.
- To delete a workspace, in `dbccdb` issue:

  ```
  drop table workspace_name
  ```

See also:

- **dbcc** in *Reference Manual: Commands*
- See the *System Administration Guide* for more information on the `scan` and `text` workspaces, and the `dbccalt` database.

Permissions

The permission checks for **sp_dbcc_alterws** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`).
Disabled	With granular permissions disabled, you must be the database owner of the specified database, or a user with **sa_role**.

See also

- *sp_dbcc_createws* on page 789
- *sp_dbcc_evaluatedb* on page 795
- *sp_plan_dbccdb* on page 579
- *sp_helpdb* on page 394

sp_dbcc_configreport

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

- **Example 1 –** 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:

```
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
```

Usage

There are additional considerations when using **sp_dbcc_configreport**:

- **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**.

See also **dbcc** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_dbcc_configreport** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`), or have the `report checkstorage` privilege on the specified database.
Disabled	With granular permissions disabled, any valid user for the database name specified can run **sp_dbcc_configreport**.

See also

- *sp_dbcc_alterws* on page 786
- *sp_dbcc_fullreport* on page 803
- *sp_dbcc_statisticsreport* on page 810
- *sp_dbcc_summaryreport* on page 813
- *sp_dbcc_updateconfig* on page 816

sp_dbcc_createws

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_wsnnnnnn` for the `scan` workspace and `text_wsnnnnnn` 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.

Examples

- **Example 1 –** Creates a 10MB `scan` workspace named `scan_wspubs2` on the `scanseg` segment in `dbccdb`:

  ```
  sp_dbcc_createws dbccdb, scanseg, scan_wspubs2, scan, "10M"
  ```

- **Example 2 –** Creates a 14MB `scan` workspace named `text_ws000001` on the `textseg` segment in `dbccdb`:

  ```
  sp_dbcc_createws dbccdb, textseg, text, "14M"
  ```

Usage

There are additional considerations when using **sp_dbcc_createws**:

- **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.
- When you create a workspace, make sure to add a 5 percent overhead on the space needed on the device because of large page allocation scheme used when creating the workspace.
- 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.
- To delete a workspace, in `dbccdb` issue:

  ```
  drop table workspace_name
  ```

See also:

- **dbcc** in *Reference Manual: Commands*
- See the *System Administration Guide* for more information on the `scan` and `text` workspaces, and the `dbccalt` database.

Permissions

The permission checks for **sp_dbcc_createws** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`).
Disabled	With granular permissions disabled, you must be the database owner of `dbccdb` (or `dbccalt`), or have **sa_role** to run **sp_dbcc_createws**.

See also

- *sp_addsegment* on page 43
- *sp_dbcc_alterws* on page 786
- *sp_dbcc_evaluatedb* on page 795
- *sp_dbcc_updateconfig* on page 816
- *sp_plan_dbccdb* on page 579
- *sp_helpsegment* on page 430

sp_dbcc_deletedb

Deletes from `dbccdb` all the information related to the specified target database.

Syntax

```
sp_dbcc_deletedb [dbname | dbid]
```

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*, the SAP ASE server deletes data from all databases in `dbccdb..dbcc_config`. If the target database is `dbccdb,` and `dbccalt` exists, the SAP ASE server deletes the data from `dbccalt`.
- ***dbid* –** specifies the database ID number of the target database for which you want the configuration information deleted.

Examples

- **Example 1 –** Deletes all information for the database named `engdb` from `dbccdb`:

  ```
  sp_dbcc_deletedb "engdb"
  ```

  ```
  All information for database engdb has been deleted from dbccdb.
  ```

Usage

There are additional considerations when using **sp_dbcc_deletedb**:

- **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**.
- Using the *dbid* option is the only way to delete the contents of the `dbccdb` database for a database that has already been dropped.

See also:

- **dbcc** in *Reference Manual: Commands*
- *System Administration Guide* for information about the `dbccalt` database.

Permissions

The permission checks for **sp_dbcc_deletedb** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`), or have the `manage checkstorage` privilege on the specified database.
Disabled	With granular permissions disabled, you must be the database owner of the specified database or have **sa_role** to run **sp_dbcc_deletedb**.

See also

- *sp_dbcc_deletehistory* on page 792
- *sp_dbcc_evaluatedb* on page 795
- *sp_plan_dbccdb* on page 579

sp_dbcc_deletehistory

Deletes the results of **dbcc checkstorage** operations performed on the target database before the specified date and time.

Note: **sp_dbcc_deletehistory** does not free any space associated with the deleted historical data, as workspaces are pre-allocated and of a fixed size.

Syntax

```
sp_dbcc_deletehistory [cutoffdate [, dbname | dbid]]
```

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.
- ***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`.
- ***dbid* –** specifies the database ID number of the target database for which you want the history information deleted.

Examples

- **Example 1 –** Deletes results of all operations performed on the database `pubs2` on or before March 4, 1997:

```
sp_dbcc_deletehistory "03/04/1997", "pubs2"
```

Usage

There are additional considerations when using **sp_dbcc_deletehistory**:

- **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.
- Use the *dbid* option to delete the historical data of the `dbccdb` database for a database that has already been dropped.
- Using the *dbid* option is the only way to delete the historical data of the `dbccdb` database for a database that has already been dropped.
- To see the dates when **dbcc checkstorage** was run so that you can choose the value for *cutoffdate*, run **sp_dbcc_summaryreport**.

See also:

- **dbcc** in *Reference Manual: Commands*
- *System Administration Guide* for information on the `dbccalt` database.

Permissions

The permission checks for **sp_dbcc_deletehistory** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`), or have the `manage checkstorage` privilege on the specified database.
Disabled	With granular permissions disabled, you must be the database owner of the specified database or a user with **sa_role** to run **sp_dbcc_deletehistory** on a specific database. Only a user with **sa_role** can run **sp_dbcc_deletehistory** without specifying a database name.

See also

- *Specifying the Date* on page 785
- *sp_dbcc_deletedb* on page 791
- *sp_dbcc_evaluatedb* on page 795
- *sp_dbcc_summaryreport* on page 813
- *sp_plan_dbccdb* on page 579

sp_dbcc_differentialreport

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

- **Example 1 –** 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:

  ```
  sp_dbcc_differentialreport master, sysprocedures,
      checkstorage, "05/01/97", "05/04/97"
  ```

Usage

There are additional considerations when using **sp_dbcc_differentialreport**:

- **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.

See also **dbcc** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_dbcc_differentialreport** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`), or have the `report checkstorage` privilege on the specified database.
Disabled	With granular permissions disabled, any valid user for the specified database can run **sp_dbcc_differentialreport**.

See also

- *sp_dbcc_fullreport* on page 803
- *sp_dbcc_statisticsreport* on page 810
- *sp_dbcc_summaryreport* on page 813
- *sp_dbcc_updateconfig* on page 816

sp_dbcc_evaluatedb

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 you do not specify *dbname*, **sp_dbcc_evaluatedb** compares all databases listed in the **dbcc_config** table.

Examples

- **Example 1 –** Recomputes configuration information for the current database, `sybsystemprocs`, and suggests new values for some parameters:

```
1> sp_dbcc_evaluatedb
2> go
```

```
Recommended values for workspace size, cache size and process
count are:

Database name : one_G
                                      current          suggested
scan workspace size :                    750M                16M
text workspace size :                      2K                48K
cache size          :                  10240K              1280K
process count       :                       3                  2
compression mem size:                   2048K                12M

Each of the reported quantities is reported in a scaled unit
according to
G if size > 10G
M if 10M < size <=10 G
K otherwise
```

Usage

There are additional considerations when using **sp_dbcc_evaluatedb**:

- When there is an archive database with a compressed data or log device, the output includes a line with the recomendation of the compression memory size.
- **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**.

See also **dbcc** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_dbcc_evaluatedb** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`), or have the `manage checkstorage` privilege on the specified database.
Disabled	With granular permissions disabled, you must be the database owner of the specified database, a user with **sa_role**. Only a system administrator can run **sp_dbcc_evaluatedb** without specifying a database name.

See also

- *sp_dbcc_updateconfig* on page 816
- *sp_plan_dbccdb* on page 579

sp_dbcc_exclusions

Allows the user to create and manage persistent exclusion lists for use by **checkverify** and **sp_dbcc_faultreport**.

Syntax

```
sp_dbcc_exclusions dbname, op, type, exclusion_list
```

Parameters

- ***dbname –*** is the name of the database for which the exclusions apply, or **null** if it applies to all databases.
- ***op –*** is the operation you want to perform. Valid values are:
  - **add** – registers new exclusions (duplicates are ignored).
  - **drop** – drops the specified exclusions if they were previously registered
  - **listall** – lists the recorded exclusions for all databases.
- ***type –*** is the type of item to be excluded. Accepted values are `faults`, `tables`, *combo*, or null (when *op* is either null or **listall**). Type, `varchar`.
- ***exlusion_list –*** is a comma-separated list of faults, tables, table and fault entries, or nulls. Type, `varchar`.

Examples

- **Example 1 –** Excludes the tables `syslogs` and `syscomments` from **sp_dbcc_faultreport** processing on all databases:

```
sp_dbcc_exclusions null, 'add', 'tables', 'syslogs,
    syscomments'
```

- **Example 2 –** Excludes fault type 100036 from processing of the database `my_db`:

  ```
  sp_dbcc_exclusions my_db, 'add', 'faults', '100036'
  ```

- **Example 3 –** Adds the following to the exclusion list corresponding to `my_db`: fault type 100002 pertaining to table `mytable` and fault type 100035 pertaining to `syslogs`:

  ```
  sp_dbcc_exclusions my_db, 'add', 'combo', ‘mytable:100002,
  syslogs:100035’
  ```

- **Example 4 –** Removes fault type 100036 from the exclusion list corresponding to `my_db`:

  ```
  sp_dbcc_exclusions my_db, 'drop', 'faults', '100036'
  ```

- **Example 5 –** Displays the exclusion list corresponding to `my_db`:

  ```
  sp_dbcc_exclusions my_db
  ```

- **Example 6 –** Displays the recorded exclusions for all databases:

  ```
  sp_dbcc_exclusions null, 'listall'
  ```

Usage

There are additional considerations when using **sp_dbcc_exclusions**:

- *dbname* must be null when *listall* is specified. If *op* is null, **sp_dbcc_exclusions** lists the recorded exclusions for the specified database.
- Only a system administrator or the Database Owner can run **sp_dbcc_exclusions** with a *dbname* parameter that is not null.
- If the *dbname* and *op* parameters are null, the user must either be a system administrator or own at least one of the databases for which exclusions have been recorded.
- If the *dbname* parameter is null and the *op* parameter is *listall*, the user must either be a system administrator or own at least one of the databases for which exclusions have been recorded. If the user is not a system administrator, only the recorded exclusions for databases owned by the user are reported.

Permissions

The permission checks for **sp_dbcc_exclusions** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`), or have the `manage checkstorage` privilege on the specified database.
Disabled	With granular permissions disabled, you must be a user with **sa_role**.

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.The report lists the tables and indexes in order.

Syntax

```
sp_dbcc_faultreport [report_type [, dbname [, objectname
    [, date [, hard_only [, exclusion_mode[, exclusion_faults
    [, exclusion_tables [, exclusion_combo
    [, display_recommendations [, opid [,fault_type_in]]]]]]]]]]]
```

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 exact date and time that the **dbcc checkstorage** operation finished. You can find this value in `dbcc_operation_log.finish`. You can create the value by combining the date from `start time` and the hours and minutes from `end time` in the **sp_dbcc_summaryreport** output. If you do not specify *date*, the SAP ASE server uses the date of the most recent operation.

  When you specify the *date* parameter, be certain that the time you enter is later than the date of the operation. **sp_dbcc_faultreport** cannot report faults that occur later than the time you enter in this parameter.

  Note: To focus on the *date* parameter, use "null" for all other parameters. If you omit a parameter entirely, **sp_dbc_faultreport** cannot generate a correct report.
- ***hard_only*** **–** enables the reporting of hard faults when you specify 1. Valid values are 0 or 1, and the default is 0.
- ***display_recommendations*** **–** enables reporting the recommendations generated by **sp_dbcc_recommendations**, and the parameters *exclusion_mode*, *exclusion_faults*, *exclusion_tables*, *display_recommendations*, and *exclusion_combo* refer to exclusion support and are optional.
- ***exclusion_mode*** **–** is a `varchar` and is on by default. To disable this, you must provide an "ignore" each time the **sp_dbcc_faultreport** is run. Use either of the following:

    - **ignore** – ignores the persistent exclusion list and uses the temporary exclusion list, if one is provided (type, `varchar`).
    - **extend** – applies the temporary exclusion list as well as the persistent exclusion list (type, `varchar`).
- ***exclusion_faults* –** is a comma-separated list of fault types to be excluded from reporting (type, `varchar`).
- ***exclusion_tables* –** is a comma-separated list of tables to be excluded from reporting (type is `varchar`).
- ***exclusion_combo* –** is a comma-separated list of fault/table combinations to be excluded from reporting (type is `varchar`).
- ***opid* –** enables fault reporting for a specific—instead of latest—operation ID for a specific date. No operation ID is specified by default.
- ***fault_type_in* –** enables fault reporting for a specific fault type. The default is NULL.

Examples

- **Example 1 –** 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:

```
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
```

- **Example 2 –** 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:

```
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:
0x00003747000037880000374600000005000648B803EF0001000103FE0080000
F
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)
.
.
.
```

- **Example 3 –** Generates a short report of faults from all tables on all databases, for an operation finished at a date and time found as an End Time, from the output of **sp_dbcc_summaryreport**. It is important that you use accurate end times in the *date* parameter; for instance, if you enter:

  ```
  7/25/2000 9:58
  ```

  instead of

  ```
  7/25/2000 9:58:0:190
  ```

  the report generates faults only up to 9:58, not after it. You could use 9:59 if you do not want to enter the exact time the operation ends:

  ```
  sp_dbcc_faultreport "short", NULL, NULL,
      "07/25/00 9:59"
  ```

  In this case, the report generates faults up to 9:59.

- **Example 4 –** Generates a short form report only for hard faults reported by the latest **checkstorage** run for a database called `mydb`:

  ```
  sp_dbcc_faultreport short, mydb, @hard_only = 1
  ```

- **Example 5 –** Adds recommended fixes to the fault report of database `my_db`:

  ```
  sp_dbcc_faultreport @dbname = my_db,
      @display_recommendations = 1
  ```

- **Example 6 –** Generates a fault report that does not contain fix recommendations:

  ```
  sp_dbcc_faultreport @dbname = my_db
  ```

- **Example 7 –** Runs **sp_dbcc_faultreport** on database `my_db` with the persistent exclusion list disabled:

  ```
  sp_dbcc_faultreport @dbname = 'my_db', @exclusion_mode = 'ignore'
  ```

- **Example 8 –** Runs **sp_dbcc_faultreport** on database `my_db` with the persistent exclusion list enabled and extended to exclude from processing fault type 100036:

  ```
  sp_dbcc_faultreport @dbname = 'my_db', @exclusion_mode =
  'extend',
      @exclusion_faults = '100036'
  ```

- **Example 9 –** Runs **sp_dbcc_faultreport** on database `my_db` with the persistent exclusion list enabled and extended to exclude from processing and the table `tab`:

  ```
  sp_dbcc_faultreport @dbname = 'my_db', @exclusion_mode = 'extend',
      @exclusion_tables = 'tab'
  ```

- **Example 10 –** Runs **sp_dbcc_faultreport** on database `my_db` with the persistent exclusion list disabled and an enabled temporary exclusion list that excludes from processing the table `tab` and fault type 100036:

```
sp_dbcc_faultreport @dbname = 'my_db', @exclusion_mode =
'ignore',
    @exclusion_faults = '100036', @exclusion_tables = 'tab'
```

- **Example 11 –** Runs **sp_dbcc_faultreport** on database `my_db` with the persistent exclusion list disabled and an enabled temporary exclusion list that excludes from processing fault type '100002' pertaining to the table `mytable` and fault type 100035 pertaining to the table `tab`:

```
sp_dbcc_faultreport @dbname = 'my_db', @exclusion_mode =
'ignore',
    @exclusion_combo ='mytable:100002, tab:100035'
```

- **Example 12 –** Generates a long form report for the 100029 faults reported by the latest **checkstorage** run for the `mydb` database (100029 is the fault type for page header errors):

```
sp_dbcc_faultreport long, mydb, @fault_type_in = 100029
```

- **Example 13 –** Generates a short form report for faults reported by the **checkstorage** run with operation ID 5 for the `mydb` database:

```
sp_dbcc_faultreport short, mydb, @opid = 5
```

Usage

There are additional considerations when using **sp_dbcc_faultreport**:

- **sp_dbcc_faultreport** generates a report that shows all faults for the specified object in the target database.
- **sp_dbcc_faultreport** issues numerous error message number 10028 If you use:
  - **sp_placeobject** to make an object that has existing allocations put new allocations on a new segment.
  - **sp_dropsegment** to remove a segment from a fragment that contains allocations of an object assigned to that segment.

    Error message number 100028 is an informational message rather than an indication of a serious error. If you prefer not to receive such messages, you can create your own reporting procedure that does not report this (or any other) error. One way to do this is to add the following to the very beginning of the standard **sp_dbcc_faultreport** stored procedure in the `installdbccdb` script:

    ```
    print "removing 100028 errors from dbcc_faults table"
    delete dbcc_faults where type_code = 100028
    ```
- If **sp_dbcc_faultreport** returns a number for *object_name*, it means the object was dropped after the **dbcc checkstorage** operation completed.

See also:

- **dbcc** in *Reference Manual: Commands*
- `type_code` in the *System Administration Guide* for information on the fault ID and on the fault status.

Permissions

The permission checks for **sp_dbcc_faultreport** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`), or have the `report checkstorage` privilege on the specified database.
Disabled	With granular permissions disabled, any valid user for the database name specified can run **sp_dbcc_faultreport**.

See also

- *sp_dbcc_fullreport* on page 803
- *sp_dbcc_statisticsreport* on page 810
- *sp_dbcc_summaryreport* on page 813
- *sp_dbcc_updateconfig* on page 816

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*.

Syntax

```
sp_dbcc_fullreport [dbname [, objectname [, date]]]
```

Parameters

- ***dbname*** **–** specifies the name of the database. For example, `master..sysdatabases`. 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

- **Example 1 –** 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:

```
sp_dbcc_fullreport master, sysprocedures
```

Usage

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

See also **dbcc** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_dbcc_fullreport** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`), or have the `report checkstorage` privilege on the specified database.
Disabled	With granular permissions disabled, any valid user for the database name specified can run sp_dbcc_fullreport.

See also

- *sp_dbcc_configreport* on page 787
- *sp_dbcc_faultreport* on page 799
- *sp_dbcc_statisticsreport* on page 810
- *sp_dbcc_summaryreport* on page 813
- *sp_dbcc_updateconfig* on page 816

sp_dbcc_help_fault

Provides a description of the specified fault type and the recommended fix.

Syntax

```
sp_dbcc_help_fault [fault_type]
```

Parameters

- ***fault_type* –** is the fault type for which a description and recommended fix should be reported. This parameter is type `int`. If *fault_type* is not provided, **sp_dbcc_help_fault** reports on all fault types.

Examples

- **Example 1 –** To view a description of fault type 100038, and its recommended fix, enter:

  ```
  sp_dbcc_help_fault 100038
  ```

- **Example 2 –** To view a description of all fault types and their recommended fixes, enter:

  ```
  sp_dbcc_help_fault
  ```

Usage

sp_dbcc_help_fault provides a description of the specified fault type and the recommended fix.

Permissions

The permission checks for **sp_dbcc_help_fault** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`).
Disabled	With granular permissions disabled, any user can run **sp_dbcc_help_fault**.

sp_dbcc_patch_finishtime

Facilitates reporting on aborted **checkverify** and **checkstorage** operations.

Syntax

```
sp_dbcc_patch_finishtime dbname, opid [,optype [,seq [,finishtime]]]
```

Parameters

- ***dbname* –** is the name of the database **checkstorage** or **checkverify** was operating on when it aborted. This parameter's type is `varchar`.
- ***opid* –** is the operation ID corresponding to the aborted operation. This parameter's type is `smallint`.
- ***optype* –** is the type of operation you are investigating. Accepted values are either 'checkstorage' or 'checkverify'. This parameter's type is `varchar`.
- ***seq* –** is the **checkverify** sequence number (not used for **checkstorage** but required for **checkverify**). This parameter's type is `smallint`.
- ***finishtime* –** is a `datetime` value representing the time the checkstorage or **checkverify** operation aborted. The default value is the current time.

Examples

- **Example 1 –** Enables reporting on **checkstorage** and **checkverify** for database `my_db` when the following errors occur:

  ```
  dbcc checkstorage (my_db)

  Checking my_db: Logical pagesize is 2048 bytes
  00:00000:00014:2003/01/20 11:50:05.01 server  Error: 9960,
  Severity: 20, State: 1
  A non-recoverable error has occurred in the CHECKSTORAGE
  operation. The
  operation has been aborted.

  Msg 9970, Level 20, State 1:
  Line 2:
  DBCC cannot update the finish time in dbcc_operation_log table for
  this
  operation(opid = '1') of database 'my_db'. This can be patched by
  executing
  sp_dbcc_patch_finishtime.
  ```

- **Example 2 –** Enables reporting on **checkstorage** and **checkverify** for database `my_db` when the following errors occur:

  ```
  dbcc checkstorage (my_db)

  Checking my_db: Logical pagesize is 2048 bytes
  00:00000:00014:2003/01/20 11:50:05.01 server  Error: 9960,
  Severity: 20, State: 1
  A non-recoverable error has occurred in the CHECKSTORAGE
  operation. The
  operation has been aborted.

  Msg 9970, Level 20, State 1:
  Line 2:
  DBCC cannot update the finish time in dbcc_operation_log table for
  this
  operation(opid = '1') of database 'my_db'. This can be patched by
  executing
  sp_dbcc_patch_finishtime.
  ```

  Execute **sp_dbcc_patch_finishtime** with the information included in the error message:

  ```
  sp_dbcc_patch_finishtime my_db, 1
  ```

Usage

When a **checkstorage** or **checkverify** operation aborts, it prints a message that contains the operation's ID and the name of the database that was being examined when the operation aborted. An aborted **checkverify** operation also provides a sequence number in the message. The message instructs the user to run **sp_dbcc_patch_finishtime**, and provides the *dbname*, *opid*, and if it was a **checkverify** operation, the sequence number, *seq*. After executing **sp_dbcc_patch_finishtime**, you can create fault reports on the aborted operation.

Permissions

The permission checks for **sp_dbcc_patch_finishtime** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`), or have the `manage checkstorage` privilege on the specified database.
Disabled	With granular permissions disabled, you must be the database owner of the specified database or a user with **sa_role**.

sp_dbcc_recommendations

Analyzes faults reported by the **checkstorage** operation corresponding to the specified operation ID, or date, and generates a list of recommended corrective actions for the specified object in the target database.

Syntax

```
sp_dbcc_recommendations dbname [,"date"[, opid [, "objectname"]]]
```

Parameters

- ***dbname* –** is the name of the database for which recommendations are generated. Type is `varchar`, and this parameter is required.
- ***date* –** is a `datetime` value representing the date and time the **dbcc checkstorage** operation (for which the reported faults are analyzed) finished. If you do not specify *date* or *opid*, the SAP ASE server uses the date of the most recent operation. If you specify both *date* and *opid*, the SAP ASE server ignores the date. *date* is optional.
- ***opid* –** is the operation ID of the **checkstorage** operation, for which the reported faults are analyzed. If an *opid* or *date* is not specified, the SAP ASE server uses the date of the most recent operation. If both *date* and *opid* are specified, the SAP ASE server ignores the *date*. The type for this parameter is `int`.
- ***objectname* –** is the name of the object for which **sp_dbcc_recommendations** generates the recommendations. If an *objectname* is not specified, recommendations for all objects in the database are generated. The type for this parameter is `varchar`.

Examples

- **Example 1 –** Generates a list of recommended fixes for the object `t1`, in database `my_db`, based on the faults reported by the **checkstorage** operation corresponding to operation id 2:

```
sp_dbcc_recommendations my_db, null, 2, ‘t1’
```

- **Example 2 –** Generates a list of recommended fixes for all objects in database `my_db`, based on the faults reported by the `checkstorage` operation that finished on Sep 15 2002 at 7:10:18:463PM:

  ```
  sp_dbcc_recommendations my_db, ‘Sep 15 2002 7:10:18:463PM’
  ```
- **Example 3 –** Generates a list of recommended fixes for all objects in database `my_db`, based on the most recent checkstorage operation:

  ```
  sp_dbcc_recommendations my_db
  ```

Usage

sp_dbcc_recomendations analyzes faults reported by the **checkstorage** operation corresponding to the specified operation ID, or date, and generates a list of recommended corrective actions for the specified object in the target database

Permissions

The permission checks for **sp_dbcc_recommendations** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`), or `report checkstorage` privilege on the specified database.
Disabled	With granular permissions disabled, any valid user of the specified database can run **sp_dbcc_recommendations**.

sp_dbcc_runcheck

Runs **dbcc checkstorage** on the specified database, 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_sunmmaryreport**.

Examples

- **Example 1 –** Checks the database `engdb` and generates a summary report on the information found:

  ```
  sp_dbcc_runcheck "engdb"
  ```

- **Example 2 –** Checks the database `pubs2` and generates a full report:

  ```
  sp_dbcc_runcheck "pubs2", sp_dbcc_fullreport
  ```

Usage

There are additional considerations when using **sp_dbcc_runcheck**:

- **sp_dbcc_runcheck** runs **dbcc checkstorage** on the specified database.
- If **sp_dbcc_runcheck** discovers any errors while running **dbcc checkstorage**, it then automatically runs **dbcc checkverify** in order to confirm or dismiss soft faults from **checkstorage** before running **sp_dbcc_summaryreport**.
- 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**. See the *System Administration Guide* for a brief description and examples of all the report-generating stored procedures provided with `dbccdb`.
- 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 the SAP ASE server.

See also **dbcc** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_dbcc_runcheck** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must have the `dbcc checkstorage` and `dbcc checkvverify` privileges on the specified database.
Disabled	With granular permissions disabled, you must be the database owner of the specified database or a user with **sa_role**.

See also

- *sp_dbcc_summaryreport* on page 813

sp_dbcc_statisticsreport

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*, the SAP ASE 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*, the SAP ASE server uses the date of the most recent operation.

Examples

- **Example 1 –** Generates a statistics report on the `sysobjects` table in the `sybsystemprocs` database:

```
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
```

Usage

There are additional considerations when using **sp_dbcc_statisticsreport**:

- **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 5000–5024.

  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.

See also:

- **dbcc** in *Reference Manual: Commands*
- **dbcc_counters** in *Reference Manual: Tables*

Permissions

The permission checks for **sp_dbcc_statisticsreport** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`), or have the `report checkstorage` privilege on the specified database.
Disabled	With granular permissions disabled, any valid user for the database name specified can run **sp_dbcc_statisticsreport**

See also

- *sp_dbcc_fullreport* on page 803
- *sp_dbcc_summaryreport* on page 813
- *sp_dbcc_updateconfig* on page 816

sp_dbcc_summaryreport

Generates a summary report on the specified database.

Syntax

```
sp_dbcc_summaryreport [dbname [, date [, op_name [,
    display_recommendations]]]]
```

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.
- ***opname* –** specifies the operation. *opname* may be either **checkstorage**, which is the default, or **checkverify**, or both. If *opname* is not specified, reports are generated for all operations.
- ***display_recommendations* –** enables reporting the recommendations generated by **sp_dbcc_recommendations**

Examples

- **Example 1 –** Generates a summary report on the `sybsystemprocs` database, providing information on all **dbcc checkstorage** and **dbcc checkverify** operations performed:

  ```
  sp_dbcc_summaryreport
  ```

  ```
  DBCC Operation : checkstorage

  Database Name      Start time          End Time    Operation ID
     Hard Faults Soft Faults Text Columns Abort Count     User Name
  ------------------ -------------------- ----------- ------------
      ----------- ----------- ------------
  -----------    ------------------
  sybsystemprocs     05/11/1999 14:53:11  14:53:32:163           1
                0           0           0           0     sa
  sybsystemprocs     05/11/1999 14:55:06  14:55:29:200           2
                0           0           0           0     sa
  sybsystemprocs    05/11/1999 14:56:10  14:56:27:750           3
  ```

```
              0           0            0           0     sa

DBCC Operation : checkverify

Database Name      Start time           End Time    Operation ID
    Hard Faults    Soft Faults     User Name
------------------ -------------------- ----------- ------------
    -------------- --------------- ---------------------
sybsystemprocs     05/11/1999 14:55:29  14:55:29:310           2
                 0               0 sa
```

- **Example 2 –** Generates a summary report on the user database `testdb`, providing information on all **dbcc checkstorage** operations performed. **dbcc checkstorage** was the only operation run on this database, so no **dbcc checkverify** information appears on the report:

```
sp_dbcc_summaryreport "testdb"
```

```
DBCC Operation : checkstorage

Database Name   Start time           End Time    Operation ID
    Hard Faults Soft Faults Text Columns Abort Count  User Name
--------------- -------------------- ------------ ------------
    ----------- ----------- ------------
-----------  ----------------
testdb         05/11/1999 14:55:29  14:55:49:903  1
     0           0            0           0           sa
testdb         05/11/1999 14:55:50  14:56:9:546   2
     0           0            0           0           sa
 testdb         05/11/1999 14:56:28  14:56:40:666  3
     0           0            0           0           sa
```

- **Example 3 –** Generates a summary report on the `sybsystemprocs` database, providing information on all **dbcc checkverify** operations performed. Because **dbcc checkverify** was the specified operation, no **dbcc checkstorage** information appears on the report:

```
1> sp_dbcc_summaryreport null, null, "checkverify"
2> go
```

```
DBCC Operation : checkverify

 Database Name  Start time          End Time      Operation ID  Run
Srl      Table Name   Table Id    Hard Faults   Soft Faults  User
Name
 -------------- -------------------- -------------
------------ -------
    ----------- ----------- -----------    ------------
----------------  testdb        08/31/2004
11:06:11  11:6:11:370            3         1
    NULL          NULL                 0            0  sa

(1 row affected)
```

- **Example 4 –** Generates a summary report on the `sybsystemprocs` database, providing information on all **dbcc checkstorage** operations performed. Because **dbcc**

checkstorage was the specified operation, no **dbcc checkverify** information appears on the report:

```
sp_dbcc_summaryreport sybsystemprocs, null, "checkstorage"
```

```
DBCC Operation : checkstorage

Database Name   Start time           End Time    Operation ID
    Hard Faults Soft Faults Text Columns Abort Count  User Name
--------------- -------------------- ------------ ------------
    ----------- ----------- ------------ -----------  ---------
sybsystemprocs   05/11/1999 14:53:11  14:53:32:163  1
    0           0           0            0            sa
sybsystemprocs   05/11/1999 14:55:06  14:55:29:200  2
    0           0           0            0            sa
sybsystemprocs   05/11/1999 14:56:10  14:56:27:750  3
    0           0           0            0            sa
```

- **Example 5 –** Adds recommended fixes to the summary report of database `my_db`:

```
sp_dbcc_summaryreport @dbname = my_db, @display_recommendations =
1
```

Usage

There are additional considerations when using **sp_dbcc_summaryreport**:

- **sp_dbcc_summaryreport** generates a summary report of **checkstorage** or **checkverify** operations, or both, 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. See the *System Administration Guide* for more information.
- The "Text Columns" 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**.

See also **dbcc** in *Reference Manual: Commands*.

Permissions

The permission checks for **sp_dbcc_summaryreport** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt)`, or have the `report checkstorage` privilege on the specified database.
Disabled	With granular permissions disabled, any valid user for the database name specified can run **sp_dbcc_summaryreport**.

See also

- *sp_dbcc_fullreport* on page 803
- *sp_dbcc_statisticsreport* on page 810
- *sp_dbcc_updateconfig* on page 816

sp_dbcc_updateconfig

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. To configure the default values, enter a null *dbname* parameter.
- ***type* –** specifies the type name from the `dbcc_types` table.
- ***str1* –** specifies the first configuration value for the specified *type* to be updated in the `dbcc_config` table.
- ***str2* –** specifies the second configuration value for the specified *type* that you want to update in the `dbcc_config` table.

Examples

- **Example 1 –** 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:

  ```
  sp_dbcc_updateconfig pubs2, "max worker processes", "4"
  ```

- **Example 2 –** This sets the **max worker processes** to 2:

  ```
  sp_dbcc_updateconfig null, 'max worker processes', '2'
  ```

- **Example 3 –** Updates `dbcc_config` with the size of the **dbcc** named cache "pubs2_cache". The new size is 10K:

  ```
  sp_dbcc_updateconfig pubs2, "dbcc named cache", pubs2_cache, "10K"
  ```

- **Example 4 –** 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:

  ```
  sp_dbcc_updateconfig pubs2, "scan workspace", scan_pubs2
  ```

- **Example 5 –** 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:

  ```
  sp_dbcc_updateconfig pubs2, "text workspace", text_pubs2
  ```

- **Example 6 –** Updates `dbcc_config` with the OAM count threshold value for the `pubs2` database. The new value is 5:

  ```
  sp_dbcc_updateconfig pubs2, "OAM count threshold", "5"
  ```

- **Example 7 –** Updates `dbcc_config` with the I/O error abort value for the `pubs2` database. The new value is 3:

  ```
  sp_dbcc_updateconfig pubs2, "IO error abort", "3"
  ```

- **Example 8 –** Updates `dbcc_config` with the linkage error abort value for the `pubs2` database. The new value is 8:

  ```
  sp_dbcc_updateconfig pubs2, "linkage error abort", "8"
  ```

- **Example 9 –** Enables **automatic workspace expansion** for the database `my_db`:

  ```
  sp_dbcc_updateconfig my_db, "enable automatic workspace
  expansion", "1"
  ```

Usage

There are additional considerations when using **sp_dbcc_updateconfig**:

- **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.
- The **OAM count threshold** parameter represents the percentage by which the actual row count can vary from the row count (as reported by the OAM pages) before **dbcc checkstorage** raises error 100025, `row count error`. Generally, you can leave **OAM count threshold** at the default value of 2%.

- The valid type names to use for *type* and the expected value for *str1* or *str2* are:

Table 21. 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.
automatic workspace expansion	Allows **checkstorage** to automatically expands the workspace if adequate space is available on the respective segments. The default value of 1 enables automatic workspace expansion, and the value of 0 disables it.

See also:

- **dbcc** in *Reference Manual: Commands*
- *System Administration Guide* for more information on the *type* names and values.

Permissions

The permission checks for **sp_dbcc_updateconfig** differ based on your granular permissions settings.

Setting	Description
Enabled	With granular permissions enabled, you must be the database owner of `dbccdb` (or `dbccalt`), or have the `manage checkstorage` privilege on the specified database.
Disabled	With granular permissions disabled, you must be the database owner of the specifed database or a user with **sa_role**.

See also

- *sp_dbcc_alterws* on page 786
- *sp_dbcc_evaluatedb* on page 795
- *sp_plan_dbccdb* on page 579