

CHAPTER 14

System Procedures and Functions

About this chapter This chapter documents the system-supplied catalog stored procedures in Adaptive Server Anywhere databases, used to retrieve system information. The chapter also documents system-supplied extended procedures, including procedures for sending e-mail messages on a MAPI e-mail system.

Contents

Topic	Page
System procedure overview	752
System and catalog stored procedures	753
System extended stored procedures	761
Adaptive Server Enterprise system and catalog procedures	767

System procedure overview

Adaptive Server Anywhere includes the following kinds of system procedures:

- ◆ Catalog stored procedures, for displaying system information in tabular form.
- ◆ Extended stored procedures for MAPI e-mail support and other functions.
- ◆ Transact-SQL system and catalog procedures.

☞ For a list of these system procedures see "Adaptive Server Enterprise system and catalog procedures" on page 767.

- ◆ System functions that are implemented as stored procedures.

☞ For information see "System functions" on page 276.

This chapter documents the catalog stored procedures and the extended stored procedures for MAPI e-mail support and other external functions.


System and catalog stored procedures

System and catalog stored procedures are owned by the user ID **dbo**. Some of these procedures are for internal system use. This section documents those not intended solely for system and internal use.

sa_conn_info system procedure

Function	Reports connection property information.
Syntax	sa_conn_info ([<i>connection-id</i>])
Permissions	None.
Side effects	None
Description	<p>Returns a result set consisting of the Number, Name, Userid, DBNumber, LastReqTime, ProcessTime, Port, ReqType, CommLink, NodeAddr, LastIdle, CurrTaskSw, BlockedOn, and UncmtOps properties for each connection. If no connection-id is supplied, information for all current connections to databases on the server is returned.</p> <p>In a deadlock situation, the BlockedOn value returned by this procedure allows you to check which users are blocked, and who they are blocked on.</p>

sa_conn_properties system procedure

Function	Reports connection property information
Syntax	sa_conn_properties ([<i>connection-id</i>])
Permissions	None.
Side effects	None
See also	<p>"sa_conn_properties_by_conn system procedure" on page 754</p> <p>"sa_conn_properties_by_name system procedure" on page 754</p>
Description	<p>Returns the connection id as Number, and the PropNum, PropName, PropDescription, and Value for each available connection property.</p> <p> For a listing of available connection properties, see "System functions" on page 276. These system procedures are owned by the dbo user ID. The PUBLIC group has EXECUTE permission on these procedures.</p>

sa_conn_properties_by_conn system procedure

Function	Reports connection property information
Syntax	sa_conn_properties_by_conn ([<i>property-name</i>])
Permissions	None.
Side effects	None
See also	"sa_conn_properties system procedure" on page 753
Description	<p>This is a variant on the sa_conn_properties system procedure. It returns the connection id as Number, and the PropNum, PropName, PropDescription, and Value, but only for connection properties that match the <i>property-name</i> string. You can use wild cards in <i>property-name</i>, as the comparison uses a LIKE operator. The result set is sorted by number and property name.</p> <p>🌀 For a listing of available connection properties, see "System functions" on page 276. These system procedures are owned by the dbo user ID. The PUBLIC group has EXECUTE permission on these procedures.</p>
Example	<ul style="list-style-type: none">◆ The following statement returns the AnsiNull option setting for the current connection: <pre>call sa_conn_properties_by_conn('ansinull')</pre>◆ The following statement returns the Ansi-related option settings for the current connection: <pre>call sa_conn_properties_by_conn('ansi%')</pre>

sa_conn_properties_by_name system procedure

Function	Reports connection property information
Syntax	sa_conn_properties_by_name ([<i>connection-id</i>])
Permissions	None.
Side effects	None
See also	"sa_conn_properties system procedure" on page 753
Description	<p>This is a variant on the sa_conn_properties system procedure. It returns the connection id as Number, and the PropNum, PropName, PropDescription, and Value for each available connection property. The information is sorted by property name and number.</p> <p>🌀 For a listing of available connection properties, see "System functions" on page 276. These system procedures are owned by the dbo user ID. The PUBLIC group has EXECUTE permission on these procedures.</p>

sa_db_info system procedure

Function	Reports database property information
Syntax	sa_db_info ([<i>database-id</i>])
Permissions	None.
Side effects	None
See also	"sa_db_properties system procedure" on page 755
Description	Returns a single row containing the Number, Alias, File, ConnCount, PageSize, and LogName for the specified database.
Examples	<p>♦ The following statement returns a single row describing the current database:</p>

```
call sa_db_info
```

Sample values are as follows:

Property	Value
Number	0
Alias	asademo
File	c:\asa6\asademo.db
ConnCount	1
PageSize	1024
LogName	c:\asa6\asademo.log

sa_db_properties system procedure

Function	Reports database property information
Syntax	sa_db_properties ([<i>database-id</i>])
Permissions	None.
Side effects	None
See also	"sa_db_info system procedure" on page 755
Description	Returns the database ID number and the Number, PropNum, PropName, PropDescription, and Value, for each property returned by the sa_db_info system procedure.

sa_eng_properties system procedure


Function Reports database server property information

Syntax **sa_eng_properties**

Permissions None.

Side effects None

Description Returns the PropNum, PropName, PropDescription, and Value for each available engine property.

 For a listing of available engine properties, see "System functions" on page 276.

Example ♦ The following statement returns a set of available server properties

```
call sa_eng_properties()
```

PropNum	PropName	...
0	IdleCheck	...
1	IdleWrite	...
2	IdleChkPt	...
...

sa_table_page_usage system procedure

Function Reports information about the usage of database tables.


Syntax **sa_table_page_usage**

Permissions None.

Side effects None


See also "The Information utility" on page 82

Description The results include the same information provided by the Information utility.


 For information on the Information utility, see "The Information utility" on page 82.

sa_validate system procedure

Function To validate all tables in a database.

Syntax	sa_validate
Permissions	None
Side effects	None
Description	<p>This procedure is equivalent to using the Validation utility from outside the database.</p> <p> For information on the Validation utility, see "The Validation utility" on page 119.</p>
Example	<p>◆ Validate all tables in the current database</p> <pre>sa_validate</pre> <p>The procedure returns a single column, named msg. If all tables are valid, the column contains No errors detected.</p>

sp_login_environment system procedure

Function	To set connection options when users log in.
Syntax	sp_login_environment
Permissions	None
Side effects	None
Description	<p>At startup, sp_login_environment is the default procedure called by the LOGIN_PROCEDURE database option.</p> <p>It is recommended that you not edit this procedure. Instead, to change the login environment, set the LOGIN_PROCEDURE option to point to a different procedure.</p> <p> For more information about LOGIN_PROCEDURE, see "LOGIN_PROCEDURE option" on page 161.</p>
Example	<p>◆ Here is the text of the sp_login_environment procedure:</p> <pre>CREATE PROCEDURE dbo.sp_login_environment() BEGIN IF connection_property('CommProtocol')='TDS' THEN CALL dbo.sp_tsqldb_environment() END IF END</pre>

sp_remote_columns system procedure

Function	This procedure will produce a list of the columns on a remote table and a description of those data types.
Syntax	sp_remote_columns <i>servername</i> [, <i>tablename</i>] [, <i>owner</i>] [, <i>database</i>]
Permissions	None
Side effects	None
Description	If you are entering a CREATE EXISTING statement and you are specifying a column list, it may be helpful to get a list of the columns that are available on a remote table. sp_remote_columns will produce a list of the columns on a remote table and a description of those data types.
Standards and compatibility	<ul style="list-style-type: none">◆ SQL/92 Entry-level feature.◆ Sybase Supported by Open Client/Open Server.
Example	<ul style="list-style-type: none">◆ To get a list of the columns in the sysobjects table in the production database in an ASE named "asetest". <pre>sp_remote_columns asetest, sysobjects, null, production</pre>

sp_remote_tables system procedure

Function	This procedure returns a list of the tables on a server.
Syntax	sp_remote_tables <i>servername</i> [, <i>tablename</i>] [, <i>owner</i>] [, <i>database</i>]
Permissions	None
Side effects	None
Description	<p>It may be helpful when you are configuring your ASA to get a list of the remote tables available on a particular server. This procedure returns a list of the tables on a server.</p> <p>If a tablename, owner, or database name is given, the list of tables will be limited to only those that match.</p>
Standards and compatibility	<ul style="list-style-type: none">◆ SQL/92 Entry-level feature.◆ Sybase Supported by Open Client/Open Server.
Examples	<ul style="list-style-type: none">◆ To get a list of all of the Microsoft Excel worksheets available from an ODBC datasource named 'excel': <pre>sp_remote_tables excel</pre>

- ◆ To get a list of all of the tables in the 'production' database in an ASE named "asetest", owned by 'fred':

```
sp_remote_tables asetest, null, fred, production
```

sp_servercaps system procedure


Function	To display information about a remote server's capabilities.
Syntax	sp_servercaps <i>servername</i>
Permissions	None
Side effects	None
Description	This procedure will display information about a remote server's capabilities. Adaptive Server Anywhere will use this capability information to determine how much of a SQL statement can be passed of to a remote server. The system tables which contain server capabilities are not populated until after Adaptive Server Anywhere first connects to the remote server. This information comes from syscapability and syscapabilityname. The servername specified must be the same servername used in the CREATE SERVER statement.
Standards and compatibility	<ul style="list-style-type: none"> ◆ SQL/92 Entry-level feature. ◆ Sybase Supported by Open Client/Open Server.
Example	<ul style="list-style-type: none"> ◆ To display information about the remote server testasa issue the following stored procedure: <pre>sp_servercaps testasa</pre>

sp_tsql_environment system procedure

Function	To set connection options when users connect from jConnect or Open Client applications.
Syntax	sp_tsql_environment
Permissions	None
Side effects	None
See also	"sp_login_environment system procedure" on page 757
Description	At startup, sp_login_environment is the default procedure called by the LOGIN_PROCEDURE database option. If the connection uses the TDS communicatino protocol (that is, if it is an Open Client or jConnect connection), then sp_login_environment in turn calls sp_tsql_environment.

This procedure sets database options so that they are compatible with default Sybase Adaptive Server Enterprise behavior.

If you wish to change the default behavior, it is recommended that you create new procedures and alter your LOGIN_PROCEDURE option to point to these new procedures.

 For more information about LOGIN_PROCEDURE, see "LOGIN_PROCEDURE option" on page 161.

Example

- ◆ Here is the text of the sp_tsql_environment procedure:

```
create procedure dbo.sp_tsql_environment()
begin
    if db_property('IQStore')='OFF' then
        -- ASA datastore
        set temporary option AUTOMATIC_TIMESTAMP='ON'
    end if;
    set temporary option ANSINULL='OFF';
    set temporary option TSQL_VARIABLES='ON';
    set temporary option ANSI_BLANKS='ON';
    set temporary option TSQL_HEX_CONSTANT='ON';
    set temporary option CHAINED='OFF';
    set temporary option QUOTED_IDENTIFIER='OFF';
    set temporary option ALLOW_NULLS_BY_DEFAULT='OFF';
    set temporary option
CONTINUE_AFTER_RAISERROR='ON';
    set temporary option FLOAT_AS_DOUBLE='ON';
    set temporary option ISOLATION_LEVEL='1';
    set temporary option DATE_FORMAT='YYYY-MM-DD';
    set temporary option TIMESTAMP_FORMAT='YYYY-MM-DD
HH:NN:SS.SSS';
    set temporary option TIME_FORMAT='HH:NN:SS.SSS';
    set temporary option DATE_ORDER='MDY';
    set temporary option ESCAPE_CHARACTER='OFF'
end
```

System extended stored procedures

A set of system extended procedures are included in Adaptive Server Anywhere databases. These procedures are owned by the **dbo** user ID.

The following sections describe each of the stored procedures.

MAPI system extended stored procedures

Adaptive Server Anywhere includes three system procedures for sending electronic mail using Microsoft's Messaging API standard (MAPI). These system procedures are implemented as extended stored procedures: each procedure calls a function in an external DLL.

In order to use the MAPI stored procedures, a MAPI e-mail system must be accessible from the database server machine.

The MAPI stored procedures are:

- ◆ **xp_startmail** Starts a mail session in a specified mail account by logging on the MAPI message system
- ◆ **xp_sendmail** Sends a mail message to specified users
- ◆ **xp_stopmail** Closes the mail session

The following procedure notifies a set of people that a backup has been completed.

```
CREATE PROCEDURE notify_backup()
BEGIN
    CALL xp_startmail( mail_user='ServerAccount',
                      mail_password='ServerPassword'
                      );
    CALL xp_sendmail( recipient='IS Group',
                      subject='Backup',
                      "message"='Backup completed'
                      );
    CALL xp_stopmail( )
END
```

The MAPI system procedures are discussed in the following sections.

xp_startmail system procedure

Function To start an e-mail session.

Syntax `[[variable =] CALL] xp_startmail (`
 ... `[mail_user = mail-login-name]`

```
... [, mail_password = mail-password ]  
... )
```

Usage Anywhere.

Authorization None.

Description xp_startmail is a system stored procedure that starts an e-mail session. It is implemented as a user-defined function.

The *mail-login-name* and *mail-password* values are strings containing the MAPI login name and password to be used in the mail session.

If you are using Microsoft Exchange, the *mail_login_name* argument is an Exchange profile name, and you should not include a password in the procedure call.

Return codes The xp_startmail system procedure issues one of the following return codes:

Return code	Meaning
0	Success
2	Failure

xp_sendmail system procedure

Function To send an e-mail message.

Syntax [[*variable* =] **CALL**] **xp_sendmail** (
... [, **recipient** = *mail-address*]
... [, **cc_recipient** = *mail-address*]
... [, **bcc_recipient** = *mail-address*]
... [, **"message"** = *message-body*]
... [, **include_file** = *file-name*]
...)

Usage Anywhere.

Authorization Must have executed **xp_startmail** to start an e-mail session.

Description xp_sendmail is a system stored procedure that sends an e-mail message once a session has been started to :name_startmail:ename.. It is implemented as a user-defined function.

The argument values are strings. The *message* parameter name requires double quotes around it, because MESSAGE is a keyword.

Return codes The xp_sendmail system procedure issues one of the following return codes:

Return code	Meaning
0	Success
5	Failure (general)
11	Ambiguous recipient
12	Attachment not found
13	Disk full
14	Insufficient memory
15	Invalid session
16	Text too large
17	Too many files
18	Too many recipients
19	Unknown recipient

Example

The following call sends a message to the user ID **Sales Group** containing the file *prices.doc* as a mail attachment:

```
CALL xp_sendmail(recipient='Sales Group',
  subject='New Pricing',
  include_file = 'C:\\DOCS\\PRICES.DOC'
)
```

xp_stopmail system procedure

Function To close an e-mail session.

Syntax [*variable* =] [**CALL**] **xp_stopmail** ()

Usage Anywhere.

Authorization None.

Description xp_stopmail is a system stored procedure that ends an e-mail session. It is implemented as a user-defined function.

xp_stopmail returns an integer. The return value is zero if the mail session is successfully closed, and non-zero otherwise.

Return codes The xp_stopmail system procedure issues one of the following return codes:

Return code	Meaning
0	Success
3	Failure

Other system extended stored procedures

The other system extended stored procedures included are:

- ◆ **xp_cmdshell** Executes a system command.
- ◆ **xp_msver** Returns a string containing version information.
- ◆ **xp_sprintf** Builds a string from a format string and a set of input strings.
- ◆ **xp_scanf** Extracts substrings from an input string and a format string.

The following sections provide more detail on each of these procedures.

xp_cmdshell system procedure

Function	To carry out an operating system command from a procedure.
Syntax	[<i>variable</i> = CALL] xp_cmdshell (<i>string</i>)
Usage	Anywhere.
Authorization	None.
Description	xp_cmdshell is a system stored procedure that executes a system command and then returns control to the calling environment.
Example	<p>The following statement lists the files in the current directory in the file <i>c:\temp.txt</i></p> <pre>xp_cmdshell('dir > c:\\temp.txt')</pre>

xp_msver system function

Function	To retrieve version and name information about the database server.
Syntax	xp_msver (<i>string</i>) The string must be one of the following, enclosed in string delimiters.

Argument	Description
ProductName	The name of the product (Sybase Adaptive Server Anywhere)
ProductVersion	The version number, followed by the build number. The format is as follows: 6.5.02 (1200)
CompanyName	Returns the following string: Sybase Inc.
FileDescription	Returns the name of the product, followed by the name of the operating system.
LegalCopyright	Returns a copyright string for the software
LegalTrademarks	Returns trademark information for the software

Returns String containing information appropriate to the argument.

Usage Anywhere.

Authorization None.

Description xp_msver returns product, company, version, and other information.

Example ♦ The following statement requests the version and operating system description:

```
select xp_msver( 'ProductVersion') Version
       xp_msver('FileDescription') Description
```

Sample output is as follows:

Version	Description
6.5.02 (1438)	Sybase Adaptive Server Anywhere Windows NT

xp_sprintf system procedure

Function To build up a string from a format string and a set of input strings.

Syntax [*variable* = **CALL**] **xp_sprintf** (*out-string*,
... *format-string*
... [*input-string*, ...])

Usage Anywhere.

Authorization None.

Description `xp_sprintf` is a system stored procedure that builds up a string from a format string and a set of input strings. The format-string can contain up to fifty string placeholders (`%s`). These placeholders are filled in by the *input-string* arguments.

All arguments must be strings of less than 254 characters.

Example The following statements put the string *Hello World!* into the variable **mystring**.

```
CREATE VARIABLE mystring CHAR(254) ;
xp_sprintf( mystring, 'Hello %s', 'World!' )
```

xp_scanf system procedure

Function To extract substrings from an input string and a format string.

Syntax [*variable* = **CALL**] **xp_scanf** (*in-string*,
 ... *format-string*
 ... [*output-string*, ...])

Usage Anywhere.

Authorization None.

Description `xp_scanf` is a system stored procedure that extracts substrings from an input string and a format string. The format-string can contain up to fifty string placeholders (`%s`). The values of these placeholders are placed in the *output-string* variables.

All arguments must be strings of less than 254 characters.

Example ♦ The following statements put the string *World!* into the variable **mystring**.

```
CREATE VARIABLE mystring CHAR(254) ;
xp_scanf( 'Hello World!', 'Hello %s', mystring )
```


Adaptive Server Enterprise system and catalog procedures

Adaptive Server Enterprise provides system and catalog procedures to carry out many administrative functions and to obtain system information. Anywhere has implemented support for some of these procedures.

System procedures are built-in stored procedures used for getting reports from and updating system tables. Catalog stored procedures retrieve information from the system tables in tabular form.

Adaptive Server Enterprise system procedures

The following list describes the Adaptive Server Enterprise system procedures that are provided in Adaptive Server Anywhere.

While these procedures perform the same functions as they do in Adaptive Server Enterprise and pre-Version 12 Adaptive Server IQ, they are not identical. If you have preexisting scripts that use these procedures, you may want to examine the procedures. To see the text of a stored procedure, you can open it in Sybase Central or, in Interactive SQL, run the following command.

```
sp_helptext procedure_name
```

You may need to reset the width of your Interactive SQL output to see the full text, by selecting Command ► Options and entering a new Limit Display Columns value.

System procedure	Description
sp_addgroup <i>group-name</i>	Adds a group to a database
sp_addlogin <i>userid</i> , <i>password</i> [, <i>defdb</i> [, <i>deflanguage</i> [, <i>fullname</i>]]]	Adds a new user account to a database
sp_addmessage <i>message-num</i> , <i>message_text</i> [, <i>language</i>]	Adds a user-defined message to SYSUSERMESSAGES, for use by stored procedure PRINT and RAISERROR calls
sp_addtype <i>typename</i> , <i>data-type</i> , [, "identity" <i>nulltype</i>]	Creates a user-defined data type
sp_adduser <i>login_name</i> [, <i>name_in_db</i> [, <i>grpname</i>]]	Adds a new user to a database
sp_changegroup <i>new-group-name</i> , <i>userid</i>	Changes a user's group or adds a user to a group
sp_dboption [<i>dbname</i> ,	Displays or changes a database option

System procedure	Description
<i>optname</i> , { <i>true</i> <i>false</i> }]	
sp_dropgroup <i>group-name</i>	Drops a group from a database
sp_droplogin <i>userid</i>	Drops a user from a database
sp_dropmessage <i>message-number</i> [, <i>language</i>]	Drops a user-defined message
sp_droptype <i>typename</i>	Drops a user-defined data type
sp_dropuser <i>userid</i>	Drops a user from a database
sp_getmessage <i>message-num</i> , @ <i>msg-var</i> output [, <i>language</i>]	Retrieves a stored message string from SYSMESSAGES or SYSUSERMESSAGES, for PRINT and RAISERROR statements.
sp_helptext <i>object-name</i>	Displays the text of a system procedure, trigger, or view
sp_password <i>caller_passwd</i> , <i>new_passwd</i> [, <i>userid</i>]	Adds or changes a password for a user ID

Adaptive Server Enterprise catalog procedures

Adaptive Server Anywhere implements all the Adaptive Server Enterprise catalog procedures with the exception of the **sp_column_privileges** procedure. The implemented catalog procedures are described in the following table.

Catalog procedure	Description
sp_column_privileges	Unsupported
sp_columns <i>table-name</i> [, <i>table-owner</i>] [, <i>table-qualifier</i>] [, <i>column-name</i>]	Returns the data types of the specified column
sp_fkeys <i>pktable_name</i> [, <i>pktable-owner</i>] [, <i>pktable-qualifier</i>] [, <i>fktable-name</i>] [, <i>fktable-owner</i>] [, <i>fktable-qualifier</i>]	Returns foreign key information about the specified table
sp_pkeys <i>table-name</i> [, <i>table-owner</i>] [, <i>table-qualifier</i>]	Returns primary key information about the specified table
sp_special_columns <i>table_name</i> [, <i>table-owner</i>] [, <i>table-qualifier</i>] [, <i>col-type</i>]	Returns the optimal set of columns that uniquely identify a row in the specified table
sp_sproc_columns <i>proc-name</i> [, <i>proc-owner</i>] [, <i>proc-qualifier</i>] [, <i>column-name</i>]	Returns information about a stored procedure's input and return parameters
sp_stored_procedures [<i>sp-name</i>] [, <i>sp-owner</i>] [, <i>sp-qualifier</i>]	Returns information about one or more stored procedures
sp_tables <i>table-name</i> [, <i>table-owner</i>] [, <i>table-qualifier</i>] [, <i>table-type</i>]	Returns a list of objects that can appear in a FROM clause for the specified table

