iAnywhere®
A SYBASE COMPANY

Adaptive Server® Anywhere Database Administration Guide

Part number: DC38123-01-0902-01
Last modified: October 2004

Contents

About This Manual

Subject	This book covers material related to running, managing, and configuring databases. It describes database connections, the database server, database files, security, backup procedures, and replication with the Replication Server, as well as administration utilities and options.
Audience	This manual is for all users of Adaptive Server Anywhere. It is to be used in conjunction with other manuals in the documentation set.

SQL Anywhere Studio documentation

This book is part of the SQL Anywhere documentation set. This section describes the books in the documentation set and how you can use them.

The SQL Anywhere Studio documentation

The SQL Anywhere Studio documentation is available in a variety of forms: in an online form that combines all books in one large help file; as separate PDF files for each book; and as printed books that you can purchase. The documentation consists of the following books:

♦ **Introducing SQL Anywhere Studio** This book provides an overview of the SQL Anywhere Studio database management and synchronization technologies. It includes tutorials to introduce you to each of the pieces that make up SQL Anywhere Studio.

♦ **What's New in SQL Anywhere Studio** This book is for users of previous versions of the software. It lists new features in this and previous releases of the product and describes upgrade procedures.

♦ **Adaptive Server Anywhere Database Administration Guide** This book covers material related to running, managing, and configuring databases and database servers.

♦ **Adaptive Server Anywhere SQL User's Guide** This book describes how to design and create databases; how to import, export, and modify data; how to retrieve data; and how to build stored procedures and triggers.

♦ **Adaptive Server Anywhere SQL Reference Manual** This book provides a complete reference for the SQL language used by Adaptive Server Anywhere. It also describes the Adaptive Server Anywhere system tables and procedures.

♦ **Adaptive Server Anywhere Programming Guide** This book describes how to build and deploy database applications using the C, C++, and Java programming languages. Users of tools such as Visual Basic and PowerBuilder can use the programming interfaces provided by those tools. It also describes the Adaptive Server Anywhere ADO.NET data provider.

♦ **Adaptive Server Anywhere SNMP Extension Agent User's Guide** This book describes how to configure the Adaptive Server Anywhere SNMP Extension Agent for use with SNMP management applications to manage Adaptive Server Anywhere databases.

♦ **Adaptive Server Anywhere Error Messages** This book provides a complete listing of Adaptive Server Anywhere error messages together with diagnostic information.

- **SQL Anywhere Studio Security Guide** This book provides information about security features in Adaptive Server Anywhere databases. Adaptive Server Anywhere 7.0 was awarded a TCSEC (Trusted Computer System Evaluation Criteria) C2 security rating from the U.S. Government. This book may be of interest to those who wish to run the current version of Adaptive Server Anywhere in a manner equivalent to the C2-certified environment.

- **MobiLink Administration Guide** This book describes how to use the MobiLink data synchronization system for mobile computing, which enables sharing of data between a single Oracle, Sybase, Microsoft or IBM database and many Adaptive Server Anywhere or UltraLite databases.

- **MobiLink Clients** This book describes how to set up and synchronize Adaptive Server Anywhere and UltraLite remote databases.

- **MobiLink Server-Initiated Synchronization User's Guide** This book describes MobiLink server-initiated synchronization, a feature of MobiLink that allows you to initiate synchronization from the consolidated database.

- **MobiLink Tutorials** This book provides several tutorials that walk you through how to set up and run MobiLink applications.

- **QAnywhere User's Guide** This manual describes MobiLink QAnywhere, a messaging platform that enables the development and deployment of messaging applications for mobile and wireless clients, as well as traditional desktop and laptop clients.

- **iAnywhere Solutions ODBC Drivers** This book describes how to set up ODBC drivers to access consolidated databases other than Adaptive Server Anywhere from the MobiLink synchronization server and from Adaptive Server Anywhere remote data access.

- **SQL Remote User's Guide** This book describes all aspects of the SQL Remote data replication system for mobile computing, which enables sharing of data between a single Adaptive Server Anywhere or Adaptive Server Enterprise database and many Adaptive Server Anywhere databases using an indirect link such as e-mail or file transfer.

- **SQL Anywhere Studio Help** This book includes the context-sensitive help for Sybase Central, Interactive SQL, and other graphical tools. It is not included in the printed documentation set.

- **UltraLite Database User's Guide** This book is intended for all UltraLite developers. It introduces the UltraLite database system and provides information common to all UltraLite programming interfaces.

- **UltraLite Interface Guides** A separate book is provided for each UltraLite programming interface. Some of these interfaces are provided as UltraLite components for rapid application development, and others are provided as static interfaces for C, C++, and Java development.

In addition to this documentation set, PowerDesigner and InfoMaker include their own online documentation.

Documentation formats

SQL Anywhere Studio provides documentation in the following formats:

- **Online documentation** The online documentation contains the complete SQL Anywhere Studio documentation, including both the books and the context-sensitive help for SQL Anywhere tools. The online documentation is updated with each maintenance release of the product, and is the most complete and up-to-date source of documentation.

  To access the online documentation on Windows operating systems, choose Start ➤ Programs ➤ SQL Anywhere 9 ➤ Online Books. You can navigate the online documentation using the HTML Help table of contents, index, and search facility in the left pane, as well as using the links and menus in the right pane.

  To access the online documentation on UNIX operating systems, see the HTML documentation under your SQL Anywhere installation.

- **PDF books** The SQL Anywhere books are provided as a set of PDF files, viewable with Adobe Acrobat Reader.

  The PDF books are accessible from the online books, or from the Windows Start menu.

- **Printed books** The complete set of books is available from Sybase sales or from eShop, the Sybase online store at *http://eshop.sybase.com/eshop/documentation*.

Documentation conventions

This section lists the typographic and graphical conventions used in this documentation.

Syntax conventions

The following conventions are used in the SQL syntax descriptions:

- **Keywords** All SQL keywords appear in upper case, like the words ALTER TABLE in the following example:

  ALTER TABLE [*owner*.]*table-name*

- **Placeholders** Items that must be replaced with appropriate identifiers or expressions are shown like the words *owner* and *table-name* in the following example:

  ALTER TABLE [*owner*.]*table-name*

- **Repeating items** Lists of repeating items are shown with an element of the list followed by an ellipsis (three dots), like *column-constraint* in the following example:

  ADD *column-definition* [*column-constraint*, . . .]

  One or more list elements are allowed. In this example, if more than one is specified, they must be separated by commas.

- **Optional portions** Optional portions of a statement are enclosed by square brackets.

  RELEASE SAVEPOINT [*savepoint-name*]

  These square brackets indicate that the *savepoint-name* is optional. The square brackets should not be typed.

- **Options** When none or only one of a list of items can be chosen, vertical bars separate the items and the list is enclosed in square brackets.

  [**ASC** | **DESC**]

  For example, you can choose one of ASC, DESC, or neither. The square brackets should not be typed.

- **Alternatives** When precisely one of the options must be chosen, the alternatives are enclosed in curly braces and a bar is used to separate the options.

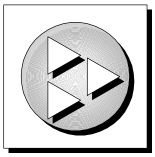
  [**QUOTES** { **ON** | **OFF** }]

  If the QUOTES option is used, one of ON or OFF must be provided. The brackets and braces should not be typed.

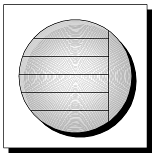
Graphic icons

The following icons are used in this documentation.

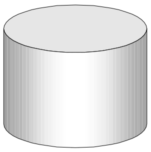
- ♦ A client application.



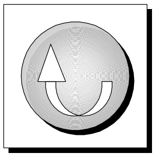
- ♦ A database server, such as Sybase Adaptive Server Anywhere.



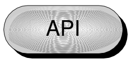
- ♦ A database. In some high-level diagrams, the icon may be used to represent both the database and the database server that manages it.



- ♦ Replication or synchronization middleware. These assist in sharing data among databases. Examples are the MobiLink Synchronization Server and the SQL Remote Message Agent.



- ♦ A programming interface.



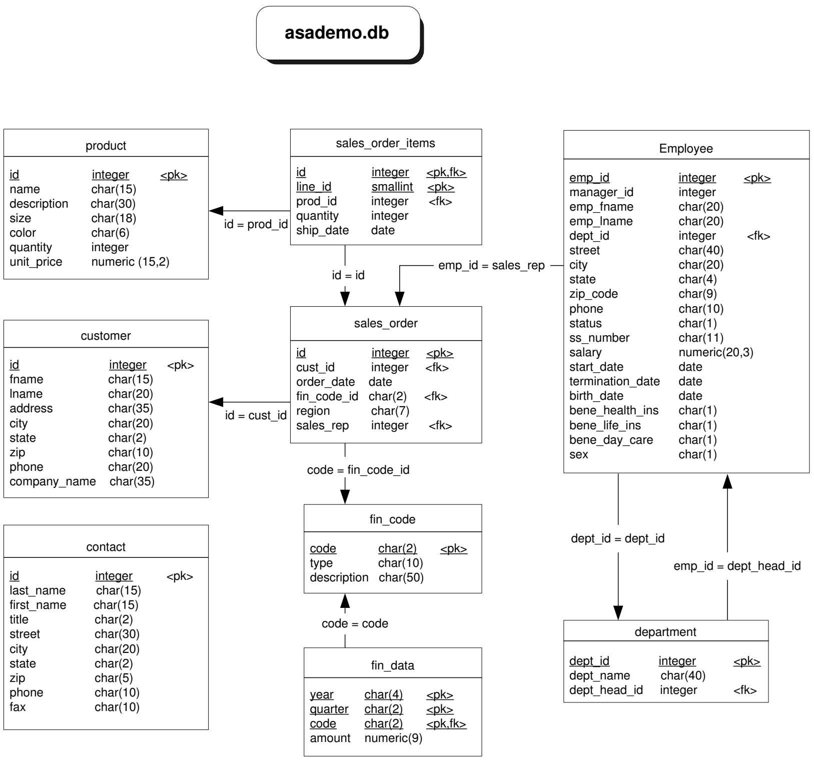
The Adaptive Server Anywhere sample database

Many of the examples throughout the documentation use the Adaptive Server Anywhere sample database.

The sample database is held in a file named *asademo.db*, and is located in your SQL Anywhere directory.

The sample database represents a small company. It contains internal information about the company (employees, departments, and finances) as well as product information and sales information (sales orders, customers, and contacts). All information in the database is fictional.

The following figure shows the tables in the sample database and how they relate to each other.



product		
id	integer	<pk>
name	char(15)	
description	char(30)	
size	char(18)	
color	char(6)	
quantity	integer	
unit_price	numeric (15,2)	

sales_order_items		
id	integer	<pk,fk>
line_id	smallint	<pk>
prod_id	integer	<fk>
quantity	integer	
ship_date	date	

Employee		
emp_id	integer	<pk>
manager_id	integer	
emp_fname	char(20)	
emp_lname	char(20)	
dept_id	integer	<fk>
street	char(40)	
city	char(20)	
state	char(4)	
zip_code	char(9)	
phone	char(10)	
status	char(1)	
ss_number	char(11)	
salary	numeric(20,3)	
start_date	date	
termination_date	date	
birth_date	date	
bene_health_ins	char(1)	
bene_life_ins	char(1)	
bene_day_care	char(1)	
sex	char(1)	

customer		
id	integer	<pk>
fname	char(15)	
lname	char(20)	
address	char(35)	
city	char(20)	
state	char(2)	
zip	char(10)	
phone	char(20)	
company_name	char(35)	

sales_order		
id	integer	<pk>
cust_id	integer	<fk>
order_date	date	
fin_code_id	char(2)	<fk>
region	char(7)	
sales_rep	integer	<fk>

contact		
id	integer	<pk>
last_name	char(15)	
first_name	char(15)	
title	char(2)	
street	char(30)	
city	char(20)	
state	char(2)	
zip	char(5)	
phone	char(10)	
fax	char(10)	

fin_code		
code	char(2)	<pk>
type	char(10)	
description	char(50)	

fin_data		
year	char(4)	<pk>
quarter	char(2)	<pk>
code	char(2)	<pk,fk>
amount	numeric(9)	

department		
dept_id	integer	<pk>
dept_name	char(40)	
dept_head_id	integer	<fk>

Finding out more and providing feedback

We would like to receive your opinions, suggestions, and feedback on this documentation.

You can provide feedback on this documentation and on the software through newsgroups set up to discuss SQL Anywhere technologies. These newsgroups can be found on the *forums.sybase.com* news server.

The newsgroups include the following:

- sybase.public.sqlanywhere.general
- sybase.public.sqlanywhere.linux
- sybase.public.sqlanywhere.mobilink
- sybase.public.sqlanywhere.product_futures_discussion
- sybase.public.sqlanywhere.replication
- sybase.public.sqlanywhere.ultralite
- ianywhere.public.sqlanywhere.qanywhere

Newsgroup disclaimer

iAnywhere Solutions has no obligation to provide solutions, information or ideas on its newsgroups, nor is iAnywhere Solutions obliged to provide anything other than a systems operator to monitor the service and insure its operation and availability.

iAnywhere Solutions Technical Advisors as well as other staff assist on the newsgroup service when they have time available. They offer their help on a volunteer basis and may not be available on a regular basis to provide solutions and information. Their ability to help is based on their workload.

You can e-mail comments and suggestions to the SQL Anywhere documentation team at iasdoc@ianywhere.com. Although we do not undertake to reply to e-mails at that address, you can be sure we will read your suggestions with interest.

Part I

Starting and Connecting to Your Database

This part describes how to start the Adaptive Server Anywhere database server, and how to connect to your database from a client application.

CHAPTER 1

Running the Database Server

About this chapter

This chapter describes how to start and stop the Adaptive Server Anywhere database server, and the options available to you on startup under different operating systems.

Contents

Introduction

Adaptive Server Anywhere provides two versions of the database server:

♦ **The personal database server** This executable does not support client/server communications across a network. Although the personal database server is provided for single-user, same-machine use—for example, as an embedded database server—it is also useful for development work.

  On Windows operating systems, except Windows CE, the name of the personal server executable is *dbeng9.exe*. Windows CE does not support the personal server. On UNIX operating systems its name is *dbeng9*.

♦ **The network database server** Intended for multi-user use, this executable supports client/server communications across a network.

  On Windows operating systems, including Windows CE, the name of the network server executable is *dbsrv9.exe*. On Novell NetWare the name is *dbsrv9.nlm* and on UNIX operating systems it is *dbsrv9*.

  It is recommended that you run the network database server on Windows NT/2000/XP or Windows 2003, rather than Windows 95/98/Me, to ensure better performance and reliability.

Server differences

The request-processing engine is identical in both servers. Each one supports exactly the same SQL, and exactly the same database features. The main differences include:

♦ **Network protocol support** Only the network server supports communications across a network.

♦ **Number of connections** The personal server has a limit of ten simultaneous connections. The limit for the network server depends on your license.

♦ **Number of CPUs** With per-seat licensing, the network database server uses all CPUs available on the machine (the default). With CPU-based licensing, the network database server uses only the number of processors you are licensed for. In addition, the personal database server and runtime database server are both limited to a single processor.

♦ **Default number of internal threads** You can configure the number of requests the server can process at one time using the -gn option. The personal and network database servers have a default of 20 threads, except on Windows CE, where the default is 3 threads.

  ☞ For more information about database server options, see "The database server" on page 116.

- **Startup defaults** To reflect their use as a personal server and a network server for many users, the startup defaults are slightly different for each.

First steps

You can start a personal server running on a single database very simply. For example, you can start both a personal server and a database called *test.db* by typing the following command in the directory where *test.db* is located:

```
dbeng9 test
```

Where to specify commands

You can specify commands in several ways, depending on your operating system. For example, you can:

- type the command at a command prompt.
- place the command in a shortcut or desktop icon.
- run the command in a batch file.
- include the command as a StartLine (START) connection parameter in a connection string.

  ☞ For more information, see "StartLine connection parameter [START]" on page 204.

There are slight variations in how you specify the basic command from platform to platform. These are described in the following section.

You can also start a personal server using a database file name in a connection string.

☞ For more information, see "Connecting to an embedded database" on page 47.

Start the database server

The way you start the database server varies slightly depending on the operating system you use. This section describes how to specify commands for the simple case of running a single database with default settings on each supported operating system.

Notes

- Except where otherwise noted, these commands start the personal server (**dbeng9**). To start a network server, simply replace **dbeng9** with **dbsrv9**.
- If the database file is in the starting directory for the command, you do not need to specify *path*.
- If you do not specify a file extension in *database-file*, the extension *.db* is assumed.

❖ **To start the personal database server using default options (Windows except Windows CE)**

1. Open a command prompt.

2. Type the following command:

   ```
   start dbeng9 path\database-file
   ```

   If you omit the database file, the Server Startup Options dialog appears, where you can locate a database file using a Browse button.

☞ For information about starting a database server on Windows CE, see "Starting the database server on Windows CE" on page 9.

❖ **To start the personal database server using default options (UNIX)**

1. Open a command prompt.

2. Type the following command:

   ```
   dbeng9 path/database-file
   ```

There is no personal server for Novell NetWare, just a network server. The following command starts the network server on NetWare.

❖ **To start the database server using default options (NetWare)**

1. The database server for NetWare is a NetWare Loadable Module (NLM) (*dbsrv9.nlm*). A NLM is a program that you can run on your NetWare server. Load a database server on your NetWare server as follows:

   ```
   load dbsrv9.nlm path\database-file
   ```

   The database file must be on a NetWare volume. A typical filename is of the form *DB:\database\sales.db*.

   You can load the server from a client machine using the Novell remote console utility. See your Novell documentation for details.

   You can put the command into your Novell *autoexec.ncf* file so that Adaptive Server Anywhere loads automatically each time you start the NetWare server.

What else is there to it?

Although you can start a personal server in the simple way described in the previous section, there are many other aspects to running a database server in a production environment. For example,

- You can choose from many **options** to specify such features as how much memory to use as cache, how many CPUs to use (on multi-processor machines running a network database server), and the network protocols to use (network server only). The options are one of the major ways of tuning Adaptive Server Anywhere behavior and performance.
- You can run the server as a Windows **service**. This allows it to continue running even when you log off the machine.
- You can start the personal server from an application and shut it down when the application has finished with it. This is typical when using the database server as an **embedded database**.

The remainder of this chapter describes these options in more detail.

Starting the server

The general form for the server command is as follows:

executable [*server-options*] [*database-file* [*database-options*], . . .]

If you supply no options and no database file, then on Windows operating systems a dialog appears, allowing you to use a Browse button to locate your database file.

The elements of the database server command include the following:

- ♦ **Executable** This can be either the personal server or the network server.

  ☞ For the file names on different operating systems, see "Introduction" on page 4.

  In this chapter, unless discussing network-specific options, the personal server is used in sample commands. The network server takes a very similar set of options.

  - **Server options** These options control the behavior of the database server for all running databases.

- ♦ **Database file** You can specify zero, one, or more database file names. Each of these databases starts and remains available for applications.

  > ***Caution***
  > *The database file and the transaction log file must be located on the same physical machine as the database server. Database files and transaction log files located on a network drive can lead to poor performance and data corruption.*

  - **Database options** For each database file you start, you can provide database options that control certain aspects of its behavior.

☞ For full reference information on each of these options, see "The database server" on page 116.

In examples throughout this chapter where there are several options, they appear on separate lines for clarity, as they could be written in a configuration file. If you type them directly at the command prompt, you must type them all on one line.

Case sensitivity

Database and server options are generally case sensitive. You should type all options in lower case.

Listing available options

❖ To list the database server options

1. Open a command prompt.
2. Type the following command:

```
dbeng9 -?
```

Starting the database server on Windows CE

The database server supplied for Windows CE is the network database server (*dbsrv9.exe*). The network server supports communications over a TCP/IP network link.

The usual client/server arrangement has the database server running on a machine with more power and resources than the client applications. Clearly, this is not the case with Windows CE; instead, the less powerful machine is running the database server.

The advantage to supplying a network server on Windows CE is that you can run database applications on your desktop computer to carry out tasks on your Windows CE database. For example:

- ♦ You can use Sybase Central on your desktop PC to manage your database.
- ♦ You can use Interactive SQL on your desktop to load and unload data, and carry out queries.
- ♦ You can use InfoMaker to produce reports.

The Windows CE database server does not start the TCP/IP network link unless it is explicitly requested.

☞ For more information about starting a database server on Windows CE, see "Lesson 1: Starting the database server" [*Introducing SQL Anywhere Studio*, page 64].

On Windows CE, attempting to start a second Adaptive Server Anywhere database server while a first server is already running brings the first server to the foreground. This is standard behavior for Windows CE applications. Because of this, two database servers cannot be run at the same time on a Windows CE device. As an alternative to running multiple database servers, one database server can run more than one database if necessary. Note that **-gd all** is required to autostart a database on a running server.

☞ For more information about the -gd option, see "-gd server option" on page 142.

Some common options

This section describes some of the most common options, and points out when you may wish to use them. They are:

- ♦ Using configuration files
- ♦ Naming the server and the databases
- ♦ Performance
- ♦ Permissions
- ♦ Maximum page size
- ♦ Special modes
- ♦ Threading
- ♦ Network communications (network server only)

Using configuration files to store server startup options

If you use an extensive set of options, you can store them in a configuration file and invoke that file in a server command. The configuration file can contain options on several lines. For example, the following configuration file starts the personal database server and the sample database. It sets a cache of 10 Mb, and names this instance of the personal server **Elora**. Lines with # as the first character in the line are treated as comments.

```
# Configuration file for server Elora
-n Elora
-c 10M
path\asademo.db
```

In the example, *path* is the name of your SQL Anywhere directory. On UNIX, you would use a forward slash instead of the backslash in the file path.

If you name the file *sample.cfg*, you could use these options as follows:

```
dbeng9 @sample.cfg
```

☞ For more information, see "@data server option" on page 123 and "Using configuration files" on page 495.

Naming the server and the databases

You can use –n as a server option (to name the server) or as a database option (to name the database).

The server and database names are among the connection parameters that client applications may use when connecting to a database. The server name appears on the desktop icon and in the title bar of the server window.

The following are invalid for database server and database names:

- ♦ names that begin with white space or single or double quotes
- ♦ names that end with white space
- ♦ names that contain semicolons

Naming the server

Providing a database server name helps avoid conflicts with other server names on your network. It also provides a meaningful name for users of client applications. The server keeps its name for its lifetime (until it is shut down). If you don't provide a server name, the server is given the name of the first database started.

You can name the server by supplying a –n option before the first database file. For example, the following command starts a server on the **asademo** database and gives the server the name **Cambridge**:

```
dbeng9 -n Cambridge asademo.db
```

If you supply a server name, you can start a database server with no database started. The following command starts a server named **Galt** with no database started:

```
dbeng9 -n Galt
```

☞ For more information about starting databases on a running server, see "Starting and stopping databases" on page 20.

Naming databases

You may want to provide a meaningful database name for users of client applications. The database is identified by that name until it is stopped.

If you don't provide a database name, the default name is the root of the database file name (the file name without the *.db* extension). For example, in the following command the first database is named **asademo**, and the second **sample**.

```
dbeng9 asademo.db sample.db
```

You can name databases by supplying a –n option following the database file. For example, the following command starts the sample database and names it **MyDB**:

```
dbeng9 asademo.db -n MyDB
```

Case sensitivity

Server names and database names are case insensitive as long as the character set is single-byte.

☞ For more information, see "Connection strings and character sets" on page 343.

Controlling performance and memory from the command line

Several options can have a major impact on database server performance, including:

♦ **Cache size** The –c option controls the amount of memory that Adaptive Server Anywhere uses as a cache. This can be a major factor in affecting performance.

  Generally speaking, the more memory made available to the database server, the faster it performs. The cache holds information that may be required more than once. Accessing information in cache is many times faster than accessing it from disk. The default initial cache size is computed based on the amount of physical memory, the operating system, and the size of the database files. On Windows and UNIX operating systems, the database server automatically grows the cache when the available cache is exhausted.

  ☞ For more information about performance tuning, see "Monitoring and Improving Performance" [*ASA SQL User's Guide*, page 157].

  ☞ For information on controlling cache size, see "Cache size" on page 121.

♦ **Number of processors** If you are running on a multi-processor machine using a network database server, you can set the number of processors with the -gt option.

  ☞ For more information, see "-gt server option" on page 147 and "Controlling threading from the command line" on page 14.

♦ **Other performance-related options** There are several options available for tuning network performance, including -gb (database process priority), and -u (buffered disk I/O).

  ☞ For more information about startup options, see "The database server" on page 116.

Controlling permissions from the command line

Some options control the permissions required to carry out certain global operations, including permissions to start and stop databases, load and unload data, and create and delete database files.

☞ For more information, see "Running the database server in a secure fashion" [*SQL Anywhere Studio Security Guide*, page 13].

Setting a maximum page size

The database server cache is arranged in **pages**—fixed-size areas of memory. Since the server uses a single cache for its lifetime (until it is shut down), all pages must have the same size.

A database file is also arranged in pages, with a size that is specified on the command line. Every database page must fit into a cache page. By default, the server page size is the same as the largest page size of the databases on the command line. Once the server starts, you cannot start a database with a larger page size than the server.

To allow databases with larger page sizes to be started after startup, you can force the server to start with a specified page size using the –gp option. If you use larger page sizes, remember to increase your cache size. A cache of the same size will accommodate only a fraction of the number of the larger pages, leaving less flexibility in arranging the space.

The following command starts a server that reserves an 8 Mb cache and can accommodate databases of page sizes up to 4096 bytes.

```
dbsrv9 -gp 4096 -c 8M -n myserver
```

Running in special modes

You can run Adaptive Server Anywhere in special modes for particular purposes.

♦ **Read-only** You can run databases in read-only mode by supplying the -r option.

  ☞ For more information, see "-r server option" on page 154.

♦ **Bulk load** This is useful when loading large quantities of data into a database using the Interactive SQL INPUT command. Do not use the –b option if you are using LOAD TABLE to bulk load data.

  ☞ For more information, see "-b server option" on page 125, and "Importing and Exporting Data" [*ASA SQL User's Guide*, page 555].

♦ **Starting without a transaction log** Use the -f database option for recovery—either to force the database server to start after the transaction log has been lost, or to force the database server to start using a transaction log it would otherwise not find. Note that -f is a database option, not a server option.

  Once the recovery is complete, you should stop your server and restart without the -f option.

☞ For more information, see “The database server” on page 116.

Controlling threading from the command line

Adaptive Server Anywhere can use multiple operating system threads to handle requests. This allows different requests to run simultaneously on separate CPUs. Each request runs on a single thread and no request is executed concurrently on multiple CPUs.

☞ For a list of options that control threading, see “Controlling threading behavior” on page 15.

Threading in Adaptive Server Anywhere

To understand how threading support works, you need to understand **requests** and **tasks.**

Requests

Suppose Adaptive Server Anywhere is being used concurrently by two users, or by two connections from a single application. Each connection submits a query (or other SQL statement) to Adaptive Server Anywhere. Each of these SQL statements is a separate request to the server.

Tasks

A task picks up a request and handles that request until it is complete. On Windows NT/2000/XP, tasks are lightweight threads called fibers; on all other platforms, tasks are threads. The number of tasks determines the number of requests that Adaptive Server Anywhere can handle concurrently.

Tasks on Windows 95/98/Me, NetWare, and UNIX

On Windows 95/98/Me, NetWare, and UNIX machines, each task is an operating system thread. When Adaptive Server Anywhere receives a request, an operating system thread picks up the request. The operating system thread runs until the request is complete.

If the server receives a request while the first request is running, the second request is assigned to a different thread. If a thread is blocked while completing a task, it does not pick up another request; instead, it waits until it can complete the current request.

Tasks on Windows NT/XP/2000

On Windows NT/2000/XP, tasks are lightweight threads called fibers. When Adaptive Server Anywhere receives a request, a fiber picks up the request and runs until the request is complete. If the database server receives another request while the first request is running, the second request is assigned to a different fiber. If a fiber blocks while processing a request, it yields control to another fiber until it can proceed: it does not pick up another request.

Threads host fibers, and the fibers are scheduled co-operatively. A fiber must explicitly yield control to another fiber when it is waiting to process a request, for example, while waiting for an I/O operation to complete. If a fiber blocks and does not yield control, it blocks the thread that is hosting it and prevents other fibers from running on that thread. If more than one thread is hosting fibers, only the thread that is hosting the waiting fiber is blocked: other threads are still free to run fibers.

You can set the number of threads used to run fibers with the -gx option. Unless you are using Java or remote data access, you should need only one thread per CPU.

Setting the number of threads when using Java or remote data access

When you use Java or remote data access and the fiber running Java or remote data access blocks, that fiber may not yield control to another fiber, which in turn blocks the thread. For this reason, the number of operating system threads assigned to the database server defaults to one more thread than the number of CPUs on the machine. This ensures that there is at least one thread available to host fibers if the thread being used by Java or remote data access is blocked. Specifying a higher number of threads than the default by using the -gx option has a minimal impact on performance.

Controlling threading behavior

There are four database server options that control threading behavior. Not all of these options are required on every platform.

- **Number of tasks** The -gn option controls the number of tasks used to process requests. Effectively, this is the number of requests that can be handled concurrently. Each request uses a task. When there are more requests than there are tasks, any outstanding requests must wait until a currently-running task completes. By default, there are 20 tasks for the network database server and 10 tasks for the personal database server. There is no benefit to setting a number of tasks that is greater than the maximum number of server connections.

  ☞ For more information, see "-gn server option" on page 145.

- **Stack size per internal execution thread** You can set the stack size per internal execution thread in the server using the -gss option. The -gss option allows you to lower the memory usage of the database server in environments with limited memory. This option has no effect on Windows operating systems.

  ☞ For more information, see "-gss server option" on page 147.

- **Number of processors** If you have more than one processor, you can control how many processors the threads exploit by specifying the -gt option. By default, all processors available on the machine are used.

☞ For more information, see "-gt server option" on page 147.

- **Number of threads assigned to the database server process** On Windows NT/2000/XP, the -gx option controls the number of threads that are dedicated to hosting fibers to service requests. By default, this is set to one more than the number of CPUs on the machine. On UNIX, NetWare, and Windows 95/98/Me where each task is its own thread, this option is not needed.

  ☞ For more information, see "-gx server option" on page 148.

Selecting communications protocols

Any communication between a client application and a database server requires a communications protocol. Adaptive Server Anywhere supports a set of communications protocols for communications across networks and for same-machine communications.

By default, the database server starts up all available protocols. You can limit the protocols available to a database server using the –x option. On the client side, many of the same options can be controlled using the CommLinks (LINKS) connection parameter.

☞ For more information on running the server using these options, see "Supported network protocols" on page 86.

Available protocols for the personal server

The personal database server (*dbeng9.exe*) supports the following protocols:

- **Shared memory** This protocol is for same-machine communications, and always remains available. It is available on all platforms.
- **TCP/IP** This protocol is for same-machine communications only from TDS clients, Open Client, or the jConnect JDBC driver. You must run TCP/IP if you wish to connect from Open Client or jConnect.

  ☞ For more information on TDS clients, see "Adaptive Server Anywhere as an Open Server" on page 101.
- **Named Pipes** This protocol is provided on Windows NT only. Named Pipes is for same-machine communications for applications that wish to run under a certified security environment.

Available protocols for the network server

The network database server (*dbsrv9.exe*) supports the following protocols:

- **Shared memory** This protocol is for same-machine communications, and always remains available. It is available on all platforms.
- **SPX** This protocol is supported on all platforms except for UNIX.
- **TCP/IP** This protocol is supported on all platforms.

- **Named Pipes** This protocol is supported on Windows NT only. Named Pipes is for same-machine communications for applications that wish to run under a certified security environment.

Specifying protocols

You can instruct a server to use only some of the available network protocols when starting up using the –x option. The following command starts the asademo database using the TCP/IP and SPX protocols:

```
dbsrv9 -x "tcpip,spx" path\asademo.db
```

Although not strictly required in this example, the quotes are necessary if there are spaces in any of the arguments to –x.

You can add additional parameters to tune the behavior of the server for each protocol. For example, the following command (typed all on one line) instructs the server to use two network cards, one with a specified port number.

```
dbsrv9 -x "tcpip{MyIP=192.75.209.12:2367,192.75.209.32}" path\
         asademo.db
```

☞ For more information about available network protocol options that can serve as part of the –x option, see "Network protocol options" on page 206.

Stopping the database server

You can stop the database server by:

- ♦ Clicking Shutdown on the database server window.
- ♦ Using the dbstop utility.

  The dbstop utility is particularly useful in batch files, or for stopping a server on another machine. It requires a connection string in its command.

  ☞ For more information on dbstop options, see "Stopping a database server using the dbstop command-line utility" on page 577.
- ♦ Letting it shut down automatically by default when the application disconnects. (This only works if the server is a personal server started by an application connection string.)
- ♦ Pressing Q when the server display window has the focus on UNIX or NetWare machines.

Examples

❖ To stop a server using the dbstop utility

1. Start a server. For example, the following command executed from the Adaptive Server Anywhere installation directory starts a server named Ottawa using the sample database:

   ```
   dbsrv9 -n Ottawa asademo.db
   ```

2. Stop the server using dbstop:

   ```
   dbstop -c "eng=Ottawa;uid=DBA;pwd=SQL"
   ```

Who can stop the server?

When you start a server, you can use the –gk option to set the level of permissions required for users to stop the server with dbstop. The default level of permissions required is **DBA**, but you can also set the value to **all** or **none**. (Interactively, of course, anybody at the machine can click Shutdown on the server window.)

Shutting down operating system sessions

If you close an operating system session where a database server is running, or if you use an operating system command to stop the database server, the server shuts down, but not cleanly. The next time the database loads, recovery is required, and happens automatically.

☞ For more information about recovery, see "Backup and Data Recovery" on page 373.

It is better to stop the database server explicitly before closing the operating system session. On NetWare, however, shutting down the NetWare server machine properly does stop the database server cleanly.

Examples of commands that will not stop a server cleanly include:

- Stopping the process in the Windows Task Manager.
- Using a UNIX **slay** or **kill** command.

Starting and stopping databases

A database server can have more than one database loaded at a time. You can start databases and start the server at the same time, as follows:

```
dbeng9 asademo sample
```

> **Caution**
> *The database file must be on the same machine as the database server. Managing a database file that is located on a network drive can lead to file corruption.*

Starting a database on a running server

You can also start databases after starting a server in one of the following ways:

- Connect to a database using a DatabaseFile (DBF) connection parameter while connected to a server. The DatabaseFile (DBF) connection parameter specifies a database file for a new connection. The database file is started on the current server.

  ☞ For more information, see "Connecting to an embedded database" on page 47, or "DatabaseFile connection parameter [DBF]" on page 185.

- Use the START DATABASE statement, or choose Start Database from the File menu in Sybase Central when you have a server selected.

  ☞ For more information, see "START DATABASE statement [Interactive SQL]" [*ASA SQL Reference*, page 623].

Limitations

- The server holds database information in memory using pages of a fixed size. Once a server has been started, you cannot start a database that has a larger page size than the server.
- The -gd server option decides the permissions required to start databases.

Stopping a database

You can stop a database by:

- Disconnecting from a database started by a connection string. Unless you explicitly set the AutoStop (ASTOP) connection parameter to **NO**, this happens automatically.

  ☞ For more information, see "AutoStop connection parameter [ASTOP]" on page 179.

- Using the STOP DATABASE statement from Interactive SQL or embedded SQL.

  ☞ For more information, see "STOP DATABASE statement" [*ASA SQL Reference*, page 632].

Running the server outside the current session

When you log on to a computer using a user ID and a password, you establish a **session**. When you start a database server, or any other application, it runs within that session. When you log off the computer, all applications associated with the session terminate.

It is common to require database servers to be available all the time. To make this easier, you can run Adaptive Server Anywhere for Windows NT/2000/XP and for UNIX in such a way that, when you log off the computer, the database server remains running. The way you do this depends on your operating system.

- **UNIX daemon** You can run the UNIX database server as a daemon using the -ud option, enabling the database server to run in the background, and to continue running after you log off.

  ☞ For more information, see "Running the UNIX database server as a daemon" on page 21.

- **Windows service** You can run the Windows database server as a service. This has many convenient properties for running high availability servers.

  ☞ For more information, see "Understanding Windows services" on page 23.

Running the UNIX database server as a daemon

To run the UNIX database server in the background, and to enable it to run independently of the current session, you run it as a **daemon**.

Do not use '&' to run the database server in the background
If you use the UNIX & (ampersand) command to run the database server in the background, it will not work—the server will hang. You must instead run the database server as a daemon.

As well, attempting to start a server in the background from within a program using the typical `fork()-exec()` sequence will not work.

You can run the UNIX database server as a daemon in one of the following ways:

1. Use the -ud option when starting the database server. For example:

```
dbsrv9 -ud asademo
```

2. Use the dbspawn tool to start the database server, for example:

```
dbspawn dbsrv9 asademo
```

One advantage of using dbspawn is that the dbspawn process does not terminate until it has confirmed that the daemon has started and is ready to accept requests. If for any reason the daemon fails to start, the exit code for dbspawn will be non-zero.

When you start the daemon directly using the -ud option, the dbeng9 and dbsrv9 commands create the daemon process and return immediately (exiting and allowing the next command to be executed) before the daemon initializes itself or attempts to open any of the databases specified in the command.

Using dbspawn can be useful when writing a script or other automated process that needs to start a server as a daemon prior to running one or more applications that will user the server because you want to ensure that the daemon is running before starting the applications. The following is an example of how to test this using a `csh` script.

```
#!/bin/csh
# start the server as a daemon and ensure that it is
running before we start any applications
dbspawn dbsrv9 asademo
if ( $status != 0 ) then
   echo Failed to start asademo server
   exit
endif
# ok, now we can start the applications
...
```

This example uses an `sh` script to test whether the daemon is running before starting the applications.

```
#!/bin/sh
# start the server as a daemon and ensure that it is
running before we start any applications
dbspawn dbsrv9 asademo
if [ $? != 0 ]; then
   echo Failed to start asademo server
   exit
fi
# ok, now we can start the applications
...
```

3. Spawn a daemon from within a C program, for example:

```
...
if( fork() == 0 ) {
      /* child process = start server daemon */
      execl( "/opt/sybase/SYBSsa9/bin/dbsrv9",
"dbsrv9", "-ud", "asademo" );
   exit(1);
}
/* parent process */
...
```

Note that the -ud option is used.

☞ For more information, see "-ud server option" on page 161 and "The Spawn utility" on page 575.

Understanding Windows services

Although you can run the database server like any other Windows NT/2000/XP program rather than as a service, there are limitations to running it as a standard program, particularly in multi-user environments.

Limitations of running as a standard executable

When you start a program, it runs under your Windows NT/2000/XP login session, which means that if you log off the computer, the program terminates. Only one person logs onto Windows NT/2000/XP (on any one computer) at one time. This restricts the use of the computer if you wish to keep a program running much of the time, as is commonly the case with database servers. You must stay logged onto the computer running the database server for the database server to keep running. This can also present a security risk as the Windows NT/2000/XP computer must be left in a logged on state.

Advantages of services

Installing an application as a Windows service enables it to run even when you log off.

When you start a service, it logs on using a special system account called LocalSystem (or another account that you specify). Since the service is not tied to the user ID of the person starting it, the service remains open even when the person who started it logs off. You can also configure a service to start automatically when the Windows computer starts, before a user logs on.

Managing services

Sybase Central provides a more convenient and comprehensive way of managing Adaptive Server Anywhere services than the Windows services manager.

Programs that can be run as Windows services

You can run the following programs as services:

- Network database server (*dbsrv9.exe*)

- Personal database server (*dbeng9.exe*)
- SQL Remote Message Agent (*dbremote.exe*)
- The MobiLink synchronization server (*dbmlsrv9.exe*)
- A sample application

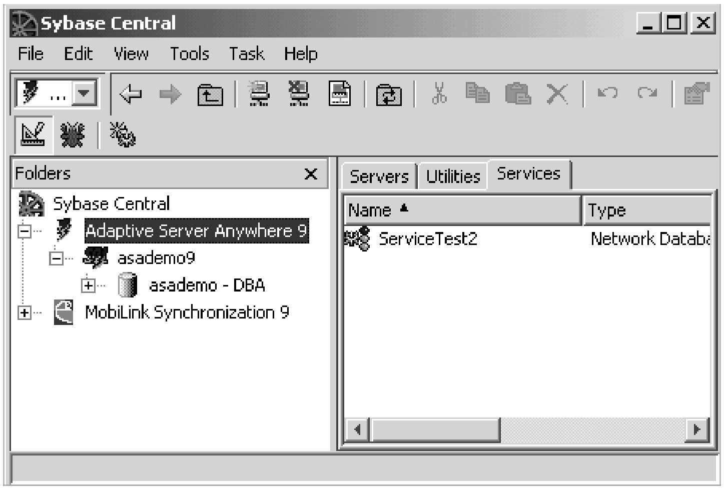
Not all these applications are supplied in all editions of Adaptive Server Anywhere.

Managing services

You can carry out the following service management tasks from the command line, or on the Services tab in Sybase Central:

- Add, edit, and remove services.
- Start, stop, and pause services.
- Modify the parameters governing a service.
- Add databases to a service so you can run several databases at one time.

The service icons in Sybase Central display the current state of each service using a traffic light icon (running, paused, or stopped).



Name	Type
ServiceTest2	Network Databa

Adding a service

This section describes how to set up services using Sybase Central and the Service Creation utility.

❖ To add a new service (Sybase Central)

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Services tab.
3. From the File menu, choose New ➤ Service.
4. Follow the instructions in the wizard.

❖ To add a new service (Command line)

1. Open a command prompt.
2. Execute the Service Creation utility using the -w option.

   For example, to create a personal server service called myserv, which starts the specified server with the specified parameters type the following. The server runs as the LocalSystem user, enter the following command, all on one line:

   ```
   dbsvc -as -w myserv "C:\Program Files\Sybase\SQL Anywhere 9\
            win32\dbeng9.exe" -n william -c 8m "C:\Program
            Files\Sybase\SQL Anywhere 9\sample.db"
   ```

☞ For more information about the Service Creation utility and options, see "The Service Creation utility" on page 569.

Notes

♦ Service names must be unique within the first eight characters.

♦ If you choose to start a service automatically, it starts whenever the computer starts Windows. If you choose to start the service manually, you need to start the service from Sybase Central each time. You may want to select Disabled if you are setting up a service for future use.

♦ Type options for the executable, without the executable name itself, in the window. For example, if you want a network server to run using the sample database with a cache size of 20 Mb and the name **myserver**, you would type the following in the Parameters box of the Service Creation wizard in Sybase Central:

  ```
  -c 20M
  -n myserver c:\Program Files\Sybase\SQL Anywhere 9\
           asademo.db
  ```

  Line breaks are optional.

  ☞ For information on valid options, see the description of each program in "Database Administration Utilities" on page 493.

- Choose the account under which the service will run: the special LocalSystem account or another user ID.

  ☞ For more information about this choice, see "Setting the account options" on page 28.

- If you want the service to be accessible from the Windows desktop, check Allow Service to Interact with Desktop. If this option is cleared, no icon appears in the System Tray and neither do any windows appear on the desktop.

☞ For more information on the configuration options, see "Configuring services" on page 26.

Removing a service

Removing a service removes the server name from the list of services. Removing a service does not remove any software from your hard disk.

If you wish to re-install a service you previously removed, you need to re-type the options.

❖ To remove a service (Sybase Central)

1. In the left pane, select the Adaptive Server Anywhere plug-in.

   In the right pane, click the Services tab.

2. In the right pane, select the service you want to remove and from the File menu, choose Delete.

❖ To remove a service (Command line)

1. Open a command prompt.

2. Execute the Service Creation utility using the -d option.

   For example, to delete the service called myserv, without prompting for confirmation, type the following command:

   ```
   dbsvc -y -d myserv
   ```

☞ For more information about the Service Creation utility and options, see "The Service Creation utility" on page 569.

Configuring services

A service runs a database server or other application with a set of options.

☞ For a full description of the options for each of the administration utilities, see "Database Administration Utilities" on page 493.

In addition to the options, services accept other parameters that specify the account under which the service runs and the conditions under which it starts.

❖ To change the parameters for a service

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, select the service you want to change.
3. From the File menu, choose Properties.
4. Alter the parameters as needed on the tabs of the Service property sheet.
5. Click OK when finished.

Changes to a service configuration take effect the next time someone starts the service. The Startup option is applied the next time Windows is started.

Setting the startup option

The following options govern startup behavior for Adaptive Server Anywhere services. You can set them on the General tab of the Service property sheet.

- ♦ **Automatic** If you choose the Automatic setting, the service starts whenever the Windows operating system starts. This setting is appropriate for database servers and other applications running all the time.
- ♦ **Manual** If you choose the Manual setting, the service starts only when a user with Administrator permissions starts it. For information about Administrator permissions, see your Windows documentation.
- ♦ **Disabled** If you choose the Disabled setting, the service will not start.

Specifying options

The Configuration tab of the Service property sheet provides a Parameters text box for specifying options for a service. *Do not type the name of the program executable in this box.*

Examples

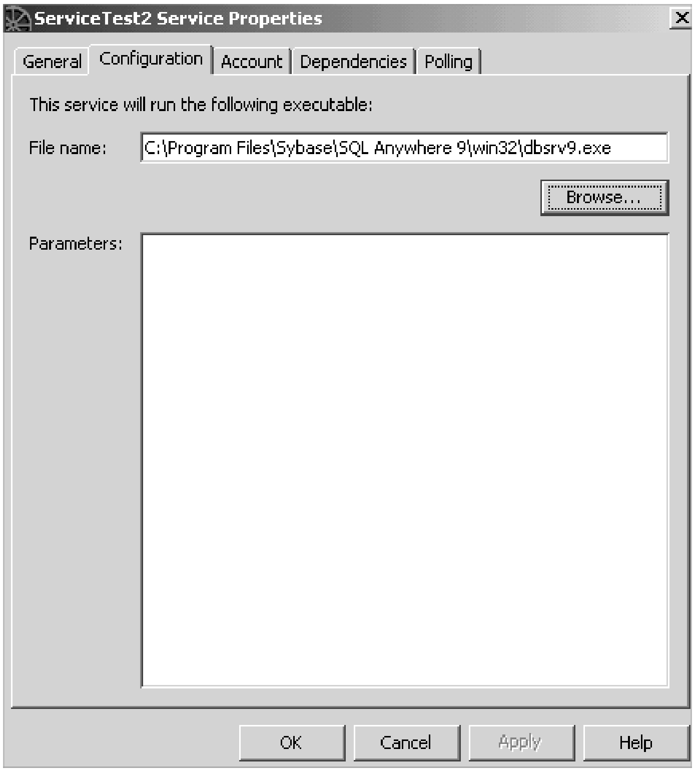
- ♦ To start a network server service named my_server running two databases, with a cache size of 20 Mb, you would type the following in the Parameters box:

```
-c 20M
-n my_server
c:\Program Files\Sybase\SQL Anywhere 9\db_1.db
c:\Program Files\Sybase\SQL Anywhere 9\db_2.db
```

- To start a SQL Remote Message Agent service connecting to the sample database as user ID DBA, you would type the following:

```
-c "uid=DBA;pwd=SQL;dbn=asademo"
```

The following figure illustrates a sample Service property sheet.



This service will run the following executable:

File name: C:\Program Files\Sybase\SQL Anywhere 9\win32\dbsrv9.exe

Parameters:

☞ The options for a service are the same as those for the executable. For a full description of the options for each program, see "The Database Server" on page 115.

Setting the account options

You can choose under which account the service runs. Most services run under the special LocalSystem account, which is the default option for services. You can set the service to log on under another account by opening the Account tab on the Service property sheet, and typing the account information.

If you choose to run the service under an account other than LocalSystem, that account must have the Log On As A Service privilege. This can be granted from the Windows User Manager application under Advanced Privileges.

When an icon appears in the System Tray

Whether or not an icon for the service appears in the System Tray or desktop depends on the account you select, and whether Allow Service to Interact with Desktop is selected, as follows:

- If a service runs under LocalSystem, and Allow Service to Interact with Desktop is selected on the Service property sheet, an icon appears on the desktop of every user logged in to Windows on the computer running the service. Consequently, any user can open the application window and stop the program running as a service.
- If a service runs under LocalSystem, and Allow Service to Interact with Desktop is cleared on the Service property sheet, no icon appears on the desktop for any user. Only users with permissions to change the state of services can stop the service.
- If a service runs under another account, no icon appears on the desktop. Only users with permissions to change the state of services can stop the service.

Changing the executable file

To change the program executable file associated with a service, click the Configuration tab on the Service property sheet and type the new path and file name in the File Name text box.

If you move an executable file to a new directory, you must modify this entry.

Adding new databases to a service

Each network server or personal server can run more than one database. If you wish to run more than one database at a time, we recommend that you do so by attaching new databases to your existing service, rather than by creating new services.

❖ To add a new database to an existing service

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Services tab.

3. Select the service and then from the File menu, choose Properties.
4. Click the Configuration tab.
5. Add the path and filename of the new database to the end of the list of options in the Parameters box.
6. Click OK to save the changes.

   The new database starts the next time the service starts.

Databases can be started on running servers by client applications, such as Interactive SQL.

☞ For more information about how to start a database on a server from Interactive SQL, see "START DATABASE statement [Interactive SQL]" [*ASA SQL Reference,* page 623].

☞ For more information about how to implement this function in an embedded SQL application, see "db_start_database function" [*ASA Programming Guide,* page 217].

Starting a database from an application does not attach it to the service. If the service is stopped and restarted, the additional database will not be started automatically.

Setting the service polling frequency

Sybase Central can poll at specified intervals to check the state (started, stopped, or paused) of each service, and update the icons to display the current state. By default, polling is off. If you leave it off, you must click Refresh Folder to see changes to the state.

❖ To set the Sybase Central polling frequency

1. In the left pane, click the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Services tab.
3. Select the service and then from the File menu, choose Properties.
4. Click the Polling tab.
5. Select Enable Polling.
6. Set the polling frequency.

   The frequency applies to all services, not just the one selected. The value you set in this window remains in effect for subsequent sessions, until you change it.

7. Click OK.

Starting, stopping, and pausing services

❖ **To start, stop, or pause a service**

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Services tab.
3. Select the service and then from the File menu, choose Start, Stop, or Pause.

   To resume a paused service, choose Continue from the popup menu.

If you start a service, it keeps running until you stop it. Closing Sybase Central or logging off does not stop the service.

Stopping a service closes all connections to the database and stops the database server. For other applications, the program shuts down.

Pausing a service prevents any further action being taken by the application. It does not shut the application down or (in the case of database server services) close any client connections to the database. Most users do not need to pause their services.

The Windows Service Manager

You can use Sybase Central to carry out all the service management for Adaptive Server Anywhere. Although you can use the Windows NT Service Manager in the Control Panel for some tasks, you cannot install or configure an Adaptive Server Anywhere service from the Windows NT Service Manager.

If you open the Windows Service Manager (from the Windows Control Panel), a list of services appears. The names of the Adaptive Server Anywhere services are formed from the Service Name you provided when installing the service, prefixed by Adaptive Server Anywhere. All the installed services appear together in the list.

Running more than one service

This section describes some topics specific to running more than one service at a time.

Service dependencies

In some circumstances you may wish to run more than one executable as a service, and these executables may depend on each other. For example, you may wish to run a server and a SQL Remote Message Agent or Log Transfer Manager to assist in replication.

In cases such as these, the services must start in the proper order. If a SQL Remote Message Agent service starts up before the server has started, it fails because it cannot find the server.

You can prevent these problems by using **service groups**, which you manage from Sybase Central.

Service groups overview

You can assign each service on your system to be a member of a service group. By default, each service belongs to a group, as listed in the following table.

Service	Default group
Network server	ASANYServer
Personal server	ASANYEngine
SQL Remote Message Agent	ASANYRemote
MobiLink synchronization server	ASANYMobiLink
Replication Agent	ASANYLTM

Before you can configure your services to ensure that they start in the correct order, you must check that your service is a member of an appropriate group. You can check which group a service belongs to, and change this group, from Sybase Central.

❖ To check and change which group a service belongs to

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Services tab.
3. Select the service and then from the File menu, choose Properties.
4. Click the Dependencies tab. The top text box displays the name of the group the service belongs to.
5. Click Change to display a list of available groups on your system.
6. Select one of the groups, or type a name for a new group.
7. Click OK to assign the service to that group.

Managing service dependencies

With Sybase Central you can specify **dependencies** for a service. For example:

- ♦ You can ensure that at least one member of each of a list of service groups has started before the current service.
- ♦ You can ensure that any number of services start before the current service. For example, you may want to ensure that a particular network server has started before a SQL Remote Message Agent that is to run against that server starts.

❖ To add a service or group to a list of dependencies

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. Right-click the service and choose Properties from the popup menu.
3. Click the Dependencies tab.
4. Click Add Services or Add Service Groups to add a service or group to the list of dependencies.
5. Select one of the services or groups from the list.
6. Click OK to add the service or group to the list of dependencies.

Troubleshooting server startup

This section describes some common problems that may occur when starting the database server.

Ensure that your transaction log file is valid

The server won't start if the existing transaction log is invalid. For example, during development you may replace a database file with a new version, without deleting the transaction log at the same time. This causes the transaction log file to be different than the database, and results in an invalid transaction log file.

Ensure that you have sufficient disk space for your temporary file

Adaptive Server Anywhere uses a temporary file to store information while running. This file is usually stored in the directory pointed to by the ASTMP environment variable, typically *c:\temp*.

If you do not have sufficient disk space available to the temporary directory, you will have problems starting the server.

☞ For more information, see "ASTMP environment variable" on page 279.

Ensure that network communication software is running

Appropriate network communication software must be installed and running before you run the database server. If you are running reliable network software with just one network installed, this should be straightforward.

☞ If you experience problems, if you are running non-standard software, or if you are running multiple networks, you may want to read the full discussion of network communication issues in "Client/Server Communications" on page 85.

You should confirm that other software requiring network communications is working properly before running the database server.

If you are running under the TCP/IP protocol, you may want to confirm that ping and Telnet are working properly. The ping and Telnet applications are provided with many TCP/IP protocol stacks.

Debugging network communications startup problems

If you are having problems establishing a connection across a network, you can use debugging options at both the client and the server to diagnose problems. On the server, you use the -z option. The startup information appears on the database server window: you can use the -o option to log the results to an output file.

☞ For more information, see "-z server option" on page 166 and "-o server option" on page 151.

Make sure you're using the right asasrv.ini file

If you are having problems establishing a connection to the correct server across a network, try deleting the *asasrv.ini* file. This file contains server information, including server name, protocol, and address. It is possible that the server information in this file is overriding information you specified in the connection string. Deleting this file causes Adaptive Server Anywhere to create a new *asasrv.ini* file containing the information you specify in the connection string. The *asasrv.ini* file should be located in your Adaptive Server Anywhere executable directory (the same directory as the ODBC/DBLib DLL).

If you continue to experience problems establishing a connection, you should also delete any copy of *asasrv.ini* located in any of the following places:

- ♦ the *win32* or *win64* subdirectory of your SQL Anywhere installation directory (you can find this in the directory listed in the *HKEY_LOCAL_MACHINE\SOFTWARE\Sybase\Adaptive Server Anywhere\9.0\Location* registry key)
- ♦ Windows directory
- ♦ Windows system directory
- ♦ anywhere else in your path

☞ For more information about the *asasrv.ini* file, see ."Server name caching for faster connections" on page 71.

Create a debug log file

You can use the LogFile connection parameter to create a debug log file. Log files can provide more detailed information about where a connection failure occurred, thereby helping you troubleshoot and correct the problem.

☞ For more information about log files, see "Logfile connection parameter [LOG]" on page 199.

CHAPTER 2

Connecting to a Database

About this chapter

This chapter describes how client applications connect to databases. It contains information about connecting to databases from ODBC, OLE DB, ADO.NET, iAnywhere JDBC driver, and embedded SQL applications. It also describes connecting from Sybase Central and Interactive SQL.

☞ For more information on connecting to a database from Sybase Open Client applications, see "Adaptive Server Anywhere as an Open Server" on page 101.

☞ For more information on connecting via JDBC (if you are not working in Sybase Central or Interactive SQL), see "JDBC Programming" [*ASA Programming Guide*, page 103].

Contents

Introduction to connections

Any client application that uses a database must establish a **connection** to that database before any work can be done. The connection forms a channel through which all activity from the client application takes place. For example, your user ID determines permissions to carry out actions on the database—and the database server has your user ID because it is part of the request to establish a connection.

When a user connects to a database, the database server assigns the connection a unique **connection ID**. For each new connection to the database server, the server increments the connection ID value by 1. These connection IDs are logged in the -z server output and the client LOGFILE connection parameter output. The connection ID can be used to filter request logging information, identify which connection has a lock on the database, or track the total number of connections to a server since it started and the order in which those connections were made.

☞ For information about request logging, see "Request logging" [*ASA SQL User's Guide*, page 158].

☞ For information about locks, see "How locking works" [*ASA SQL User's Guide*, page 135].

How connections are established

To establish a connection, the client application calls functions in one of the Adaptive Server Anywhere interfaces. Adaptive Server Anywhere provides the following interfaces:

♦ **ODBC** ODBC connections are discussed in this chapter.

♦ **ADO.NET** ADO.NET connections are discussed in this chapter.

  ☞ For more information about ADO.NET connections, see "The ADO.NET programming interface" [*ASA Programming Guide*, page 3].

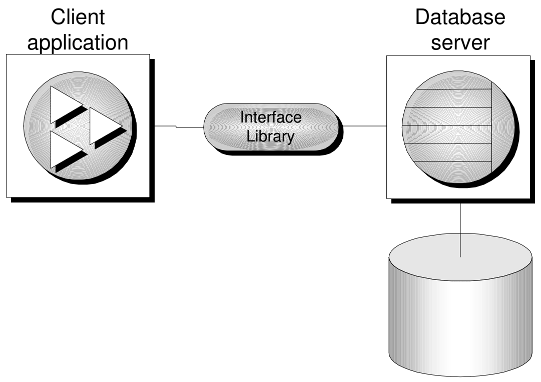
♦ **Sybase Open Client** Open Client connections are not discussed in this chapter.

  ☞ For information on connecting from Open Client applications, see "Adaptive Server Anywhere as an Open Server" on page 101.

♦ **jConnect JDBC driver** Sybase Central and Interactive SQL have the connection logic described in this chapter built into them. Other applications using jCOnnect cannot use the connection logic discussed in this chapter.

  ☞ For more information on connecting via JDBC, see "JDBC Programming" [*ASA Programming Guide*, page 103].

The interface uses connection information included in the call from the client application, perhaps together with information held in a data source, the SQLCONNECT environment variable, or the server address cache, to locate and connect to a server running the required database. The following figure is a simplified representation of the pieces involved.



What to read

The following table identifies where you can find answers to questions.

If you want...	**Consider reading...**
An overview of connecting from Sybase Central or Interactive SQL (including a description of the drivers involved)	"Connecting from Sybase Central or Interactive SQL" on page 42
Some examples to get started quickly, including Sybase Central and Interactive SQL scenarios	"Simple connection examples" on page 45
To learn about data sources	"Working with ODBC data sources" on page 53
To learn what connection parameters are available	"Connection parameters" on page 176
To see an in-depth description of how connections are established	"Troubleshooting connections" on page 66
To learn about network-specific connection issues	"Client/Server Communications" on page 85
To learn about character set issues affecting connections	"Connection strings and character sets" on page 343

How connection parameters work

When an application connects to a database, it uses a set of **connection parameters** to define the connection. Connection parameters include information such as the server name, the database name, and a user ID.

A keyword-value pair (of the form *parameter*=*value*) specifies each connection parameter. For example, you specify the password connection parameter for the default password as follows:

```
Password=SQL
```

Connection parameters are assembled into **connection strings**. In a connection string, a semicolon separates each connection parameter, as follows:

```
ServerName=asademo9;DatabaseName=asademo
```

There are a number of connection parameters that affect how a server is started. It is recommended that you use the following connection parameters instead of providing the corresponding server options within the StartLine (START) connection parameter:

- EngineName (ENG)
- DatabaseFile (DBF)
- DatabaseSwitches (DBS)
- DatabaseName (DBN)

Representing connection strings

This chapter has many examples of connection strings represented in the following form:

```
parameter1=value1
parameter2=value2
...
```

This is equivalent to the following connection string:

```
parameter1=value1;parameter2=value2
```

You must type a connection string on a single line with the parameter settings separated by semicolons.

Connection parameters passed as connection strings

Connection parameters are passed to the interface library as a **connection string**. This string consists of a set of parameters, separated by semicolons:

```
parameter1=value1;parameter2=value2;...
```

In general, the connection string built by an application and passed to the interface library does not correspond directly to the way a user enters the information. Instead, a user may fill in a dialog, or the application may read connection information from an initialization file.

Many of the Adaptive Server Anywhere utilities accept a connection string as the -c option and pass the connection string on to the interface library without change. For example, the following is a typical Collation utility (dbcollat) command line (which should be typed all on one line):

```
dbcollat -c "uid=DBA;pwd=SQL;dbn=asademo" c:\temp\asademo.col
```

Saving connection parameters in ODBC data sources

Many client applications, including application development systems, use the ODBC interface to access Adaptive Server Anywhere. When connecting to the database, ODBC applications typically use ODBC data sources. An ODBC data source is a set of connection parameters, stored in the registry or in a file.

☞ For more information on ODBC data sources, see "Working with ODBC data sources" on page 53.

For Adaptive Server Anywhere, ODBC data sources can be used by all client interfaces, except Open Client and jConnect. On UNIX and Windows CE, the data source is stored in a file. ODBC data sources cannot be used on NetWare.

Interactive SQL, Sybase Central, and the Adaptive Server Anywhere Console utility (dbconsole) can use ODBC data sources, even when connecting over jConnect.

Connecting from Sybase Central or Interactive SQL

To use Sybase Central or Interactive SQL for managing your database, you must first connect to it. In the Connect dialog, you tell Sybase Central or Interactive SQL what database you want to connect to, where it is located, and how you want to connect to it.

The connection process depends on your situation. For example, if you have a server already running on your machine and this server contains only one database, all you have to do in the Connect dialog is provide a user ID and a password for that database. Sybase Central or Interactive SQL then knows to connect immediately to the database on the running server.

If this running server has more than one database loaded on it, if it is not yet running, or if it is running on another machine, you need to provide more detailed information in the Connect dialog so that Sybase Central or Interactive SQL knows which database to connect to.

This section describes how to access the Connect dialog in Sybase Central and Interactive SQL.

☞ For more information about connection examples, including examples for Sybase Central and Interactive SQL, see "Simple connection examples" on page 45.

Opening the Connect dialog

A common Connect dialog is available in both Sybase Central and Interactive SQL to let you connect to a database.

When you start Sybase Central, you need to manually display this dialog. When you start Interactive SQL, the dialog automatically appears; you can also make it appear by choosing SQL ➤ Connect.

❖ To open the Connect dialog (Sybase Central)

1. In Sybase Central, choose Tools ➤ Connect.

   If you have more than one Sybase Central plug-in installed, choose Adaptive Server Anywhere 9 from the list.

   You can also click the Connect button on the main toolbar or press F11 to open the Connect dialog.

Tip
You can make subsequent connections to a given database easier and faster using a **connection profile**.

❖ To open the Connect dialog (Interactive SQL)

1. In Interactive SQL, choose SQL ➤ Connect.

   Alternatively, you can press F11 to open the Connect dialog.

Once the Connect dialog appears, you must specify the connection parameters you need to connect. For example, you can connect to the Adaptive Server Anywhere sample database by choosing ASA 9.0 Sample from the ODBC Data Source Name list on the Identification tab and then clicking OK.

Specifying a driver for your connection

When you are working with a database, all your requests and commands go through a driver to the database itself. Sybase Central and Interactive SQL support two main JDBC drivers: jConnect and the iAnywhere JDBC driver. Both are included with Adaptive Server Anywhere.

The iAnywhere JDBC driver

The iAnywhere JDBC driver is, in many cases, the preferred driver. It is the default driver for Interactive SQL and Sybase Central. Sybase jConnect is a platform-independent JDBC driver.

When you connect to a database in the Connect dialog, you can choose which driver you want to use for the connection on the Advanced tab. This is an optional configuration; the iAnywhere JDBC driver is the preferred driver and is automatically used for all connections unless you specify otherwise.

☞ For more information on JDBC drivers, see "Choosing a JDBC driver" [*ASA Programming Guide*, page 104], "Using the jConnect JDBC driver" [*ASA Programming Guide*, page 110], and "Working with ODBC data sources" on page 53.

Data sources and the jConnect driver

As a general rule, the jConnect driver cannot use ODBC data sources. However, Sybase Central and Interactive SQL are special cases. When you use the jConnect driver in either of them, you can specify an ODBC data source to establish a connection. For example, you can connect to the sample database using the ASA 9.0 Sample data source, even if you are using the jConnect driver.

This customized functionality is only available while you are working in Sybase Central or Interactive SQL. If you are constructing a JDBC application using jConnect, do not try to use an ODBC data source to connect to a database.

❖ To specify a driver for the connection (Sybase Central)

1. In Sybase Central, choose Tools ➤ Connect to open the Connect dialog.
2. Configure the necessary settings on the Identification and Database tabs of the dialog.
3. On the Advanced tab of the dialog, select either jConnect 5 or iAnywhere JDBC driver.

Working with the Connect dialog

The Connect dialog lets you define parameters for connecting to a server or database. The same dialog is used in both Sybase Central and Interactive SQL. The information entered in the Connect dialog is not preserved between sessions.

The Connect dialog has the following tabs:

- ♦ The Identification tab lets you identify yourself to the database and specify a data source.
- ♦ The Database tab lets you identify a server and/or database to connect to.
- ♦ The Advanced tab lets you add additional connection parameters and specify a driver for the connection.

After you connect successfully in Sybase Central, the database name appears in the left pane of the main window, under the server that it is running on. The user ID for the connection appears after the database name.

In Interactive SQL, the connection information (including the database name, your user ID, and the database server) appears in the title bar above the SQL Statements pane.

Simple connection examples

Although the connection model for Adaptive Server Anywhere is configurable, and can become complex, in many cases connecting to a database is very simple.

Who should read this section?

This section describes some simple cases of applications connecting to an Adaptive Server Anywhere database. This section may be all you need to get started.

☞ For more information about available connection parameters and their use, see "Connection parameters" on page 176.

Connecting to the sample database from Sybase Central or Interactive SQL

Many examples and exercises throughout the documentation start by connecting to the sample database from Sybase Central or Interactive SQL.

❖ To connect to the sample database (Sybase Central)

1. Start Sybase Central: from the Start menu, choose Programs ➤ Sybase SQL Anywhere 9 ➤ Sybase Central.

2. Open the Connect dialog: from the Tools menu, choose Connect.

3. Select the ODBC Data Source Name option and click Browse.

4. Select ASA 9.0 Sample and click OK.

❖ To connect to the sample database (Interactive SQL)

1. Start Interactive SQL: from the Start menu, choose Programs ➤ Sybase ➤ SQL Anywhere 9 ➤ Adaptive Server Anywhere ➤ Interactive SQL.

2. Open the Connect dialog: from the SQL menu, choose Connect.

3. Select the ODBC Data Source Name option and click Browse.

4. Select ASA 9.0 Sample and click OK.

   The database name, user ID, and server name appear in the title bar above the SQL Statements pane.

Note

You do not need to type a user ID and a password for this connection because the data source already contains this information.

Connecting to a database on your own machine from Sybase Central or Interactive SQL

The simplest connection scenario is when the database you want to connect to resides on your own machine. If this is the case for you, ask yourself the following questions:

- ♦ Is the database already running on a server? If not, you need to identify the database file so that Sybase Central or Interactive SQL can start it for you. If so, you can specify fewer parameters in the Connect dialog.
- ♦ Are there multiple databases running on your machine? If there is only one, Sybase Central or Interactive SQL assumes that it is the one you want to connect to, and you don't need to specify it in the Connect dialog. If so, you need to tell Sybase Central or Interactive SQL which database in particular to connect to.

The procedures below depend on your answers to these questions.

❖ To connect to a database on an already-running local server

1. Start Sybase Central or Interactive SQL and open the Connect dialog (if it doesn't appear automatically).
2. On the Identification tab of the dialog, type a user ID and a password for the currently-running database.
3. Do one of the following:
   - ♦ If the server only contains the one database, click OK to connect to it.
   - ♦ If the server contains multiple databases, click the Database tab of the dialog and specify a database name. This is usually the database file name, without the path or extension.

❖ To start and connect to a database

1. Start Sybase Central or Interactive SQL and open the Connect dialog (if it doesn't appear automatically).
2. On the Identification tab of the dialog, type a user ID and a password.
3. Click the Database tab.
4. Specify a file in the Database File field (including the full path, name, and extension). You can search for a file by clicking Browse.

5. If you want the database name for subsequent connections to be different from the file name, type a name in the Database Name field (without including a path or extension).

Tips

If the database is already loaded (started) on the server, you only need to provide a database name for a successful connection. The database file is not necessary.

You can connect using a data source (a stored set of connection parameters) for either of the above scenarios by selecting the appropriate data source option at the bottom of the Identification tab of the Connect dialog.

☞ For information about using data sources in conjunction with the jConnect JDBC driver, see "Specifying a driver for your connection" on page 43.

See also

♦ "Simple connection examples" on page 45

Connecting to an embedded database

An **embedded database**, designed for use by a single application, runs on the same machine as the application and is largely hidden from the application's user.

When an application uses an embedded database, the personal server is generally not running when the application connects. In this case, you can start the database using the connection string, and by specifying the database file in the DatabaseFile (DBF) parameter of the connection string.

Using the DBF parameter

The DatabaseFile (DBF) parameter specifies which database file to use. The database file automatically loads onto the default server, or starts a server if none are running.

The database unloads when there are no more connections to the database (generally when the application that started the connection disconnects). If the connection started the server, it stops once the database unloads.

The following connection parameters show how to load the sample database as an embedded database:

```
dbf=path\asademo.db
uid=DBA
pwd=SQL
```

where *path* is the name of your Adaptive Server Anywhere installation directory.

Using the StartLine [Start] parameter

The following connection parameters show how you can customize the startup of the sample database as an embedded database. This is useful if you wish to use options, such as the cache size:

```
Start=dbeng9 -c 8M
dbf=path\asademo.db
uid=DBA
pwd=SQL
```

See also

- ♦ “Opening the Connect dialog” on page 42
- ♦ “Simple connection examples” on page 45

Connecting using a data source

You can save sets of connection parameters in a **data source**. All Adaptive Server Anywhere interfaces, except Open Client and jConnect, can use data sources. The only exception is that Sybase Central, Interactive SQL, and the Adaptive Server Anywhere Console utility (dbconsole) can use data sources when connecting using jConnect.

☞ For more information, see “Specifying a driver for your connection” on page 43.

❖ To connect from using a data source (Sybase Central or Interactive SQL)

1. Start Sybase Central or Interactive SQL and open the Connect dialog (if it doesn’t appear automatically).

2. On the Identification tab, type a user ID and password.

3. On the lower half of the Identification tab, do one of the following:
   - ♦ Select the ODBC Data Source Name option and specify a data source name (equivalent to the DataSourceName (DSN) connection parameter, which references a data source in the Windows registry). You can view a list of data sources by clicking Browse.
   - ♦ Select the ODBC Data Source File option and specify a data source file (equivalent to the FileDataSourceName (FILEDSN) connection parameter, which references a data source held in a file). You can search for a file by clicking Browse.

The ASA 9.0 Sample data source holds a set of connection parameters, including the database file and a StartLine (START) parameter to start the database.

See also

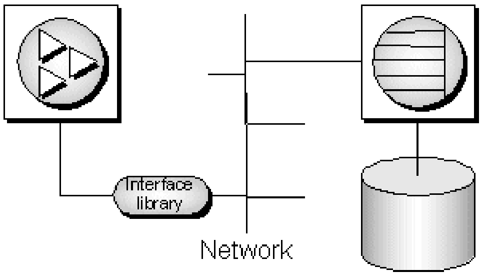
- ♦ “Opening the Connect dialog” on page 42
- ♦ “Simple connection examples” on page 45

Connecting to a server on a network

To connect to a database running on a network server somewhere on a local or wide area network, the client software must locate and connect to the database server. Adaptive Server Anywhere provides a network library to handle this task.

Network connections occur over a **network protocol**. TCP/IP is available on all platforms, and SPX is available on some platforms.

☞ For more information about client/server communications over a network, see "Client/Server Communications" on page 85.



Specifying the server

Adaptive Server Anywhere server names must be unique on a local domain for a given network protocol. The following connection parameters provide a simple example for connecting to a server running elsewhere on a network:

```
eng=svr_name
dbn=db_name
uid=user_id
pwd=password
CommLinks=all
```

When `CommLinks=all` is specified, the client library first looks for a personal server of the given name, and then looks on the network for a server of the given name.

☞ For information, see "CommLinks connection parameter [LINKS]" on page 181.

Specifying the protocol

If several protocols are available, you can instruct the network library which ones to use to improve performance. The following parameters use only the TCP/IP protocol:

```
eng=svr_name
dbn=db_name
uid=user_id
pwd=password
CommLinks=tcpip
```

The network library searches for a server by broadcasting over the network, which can be a time-consuming process. Once the network library locates a server, the client library stores its name and network address in a file (*asasrv.ini*), and reuses this entry for subsequent connection attempts to that server using the specified protocol. Subsequent connections are normally faster than a connection achieved by broadcast.

Many other connection parameters are available to assist Adaptive Server Anywhere in locating a server efficiently over a network.

☞ For more information, see "Network protocol options" on page 206.

☞ For more information, see "Connecting across a firewall" on page 87.

❖ To connect to a database on a network server (Sybase Central or Interactive SQL)

1. Start Sybase Central or Interactive SQL and open the Connect dialog (if it does not appear automatically).
2. On the Identification tab of the dialog, type a user ID and a password.
3. On the Database tab of the dialog, type the Server Name. You can search for a server by selecting the Search Network for Database Servers option and then clicking Find.
4. Identify the database by specifying a Database Name.

Tips

You can connect using a data source (a stored set of connection parameters) by selecting the appropriate data source option at the bottom of the Identification tab of the Connect dialog.

☞ For information about using data sources in conjunction with the jConnect JDBC driver, see "Specifying a driver for your connection" on page 43.

By default, all network connections in Sybase Central and Interactive SQL use the TCP/IP network protocol.

See also

- ♦ "Opening the Connect dialog" on page 42
- ♦ "Simple connection examples" on page 45

Using default connection parameters

You can leave many connection parameters unspecified, and instead use the default behavior to make a connection. Be cautious about relying on default behavior in production environments, especially if you distribute your application to customers who may install other Adaptive Server Anywhere applications on their machine.

Default database server and database

If a single personal server is running, with a single loaded database, you can connect using entirely default parameters:

```
uid=user_id
pwd=password
```

Default database server

If more than one database is loaded on a single personal server, you can leave the server as a default, but you need to specify the database you wish to connect to:

```
dbn=db_name
uid=user_id
pwd=password
```

Default database

If more than one server is running, you need to specify which server you wish to connect to. If only one database is loaded on that server, you do not need to specify the database name. The following connection string connects to a named server, using the default database:

```
eng=server_name
uid=user_id
pwd=password
```

No defaults for a local server

The following connection string connects to a named local server, using a named database:

```
eng=server_name
dbn=db_name
uid=user_id
pwd=password
```

☞ For more information about default behavior, see "Troubleshooting connections" on page 66.

No defaults for a network server

To connect to a network server running on a different machine:

```
eng=server_name
dbn=dbn
uid=user_id
pwd=password
CommLinks=tcpip
```

If CommLinks is not specified, only local shared memory connections are attempted.

If you are connecting from Sybase Central, Interactive SQL, or the Adaptive Server Anywhere Console utility (dbconsole), you can select the Search Network for Database Servers option on the Connect dialog to attempt a network connection.

☞ For more information about default behavior, see "Troubleshooting connections" on page 66.

Connecting from Adaptive Server Anywhere utilities

All Adaptive Server Anywhere database utilities that communicate with the server (rather than acting directly on database files) do so using embedded SQL. They follow the procedure outlined in "Troubleshooting connections" on page 66 when connecting to a database.

How database tools obtain connection parameter values

Many of the administration utilities obtain the connection parameter values by:

1. Using values specified on the command line (if there are any). For example, the following command starts a backup of the default database on the default server using the user ID DBA and the password SQL:

   ```
   dbbackup -c "uid=DBA;pwd=SQL" c:\backup
   ```

   ☞ For more information about options for each database tool, see the chapter "Database Administration Utilities" on page 493.

2. Using the SQLCONNECT environment variable settings if any values are missing. Adaptive Server Anywhere does not set this variable automatically.

   ☞ For more information about the SQLCONNECT environment variable, see "Environment variables" on page 276.

Working with ODBC data sources

Microsoft Corporation defines the **Open Database Connectivity** (**ODBC**) interface, which is a standard interface for connecting client applications to database-management systems in the Windows 95/98/Me and Windows NT/2000/XP environments. Many client applications, including application development systems, use the ODBC interface to access a wide range of database systems.

Where data sources are held

You connect to an ODBC database using an ODBC data source. You need an ODBC data source on the client computer for each database you want to connect to.

The ODBC data source contains a set of connection parameters. You can store sets of Adaptive Server Anywhere connection parameters as an ODBC data source, in either the Windows registry or as files.

If you have a data source, your connection string can simply name the data source to use:

- **Data source** Use the DataSourceName (DSN) connection parameter to reference a data source in the Windows registry:

  ```
  DSN=my data source
  ```

- **File data source** Use the FileDataSourceName (FILEDSN) connection parameter to reference a data source held in a file:

  ```
  FileDSN=mysource.dsn
  ```

For Adaptive Server Anywhere, the use of ODBC data sources goes beyond Windows applications using the ODBC interface:

- Adaptive Server Anywhere client applications on UNIX can use ODBC data sources, as well as those on Windows operating systems.
- ODBC data sources can be used by all Adaptive Server Anywhere client interfaces except jConnect and Open Client.
- Interactive SQL, Sybase Central, and the Adaptive Server Anywhere Console utility (dbconsole) can use ODBC data sources, even when using jConnect.

Creating an ODBC data source

You can create ODBC data sources on Windows 95/98/Me and Windows NT/2000/XP operating systems using the ODBC Administrator, which provides a central place for creating and managing ODBC data sources.

Adaptive Server Anywhere also includes a cross-platform utility named dbdsn to create data sources.

Before you begin

This section describes how to create an ODBC data source. Before you create a data source, you need to know which connection parameters you want to include in it.

☞ For more information, see “Simple connection examples” on page 45.

ODBC Administrator

On Windows 95/98/Me and Windows NT/2000/XP, you can use the Microsoft ODBC Administrator to create and edit data sources. You can work with User Data Sources, File Data Sources, and System Data Sources in this utility.

❖ To create an ODBC data source (ODBC Administrator)

1. Start the ODBC Administrator: In Sybase Central, choose Tools ➤ Adaptive Server Anywhere 9 ➤ Open ODBC Administrator.

   Alternatively, from Start menu, choose Programs ➤ Sybase ➤ SQL Anywhere 9 ➤ Adaptive Server Anywhere ➤ ODBC Administrator.

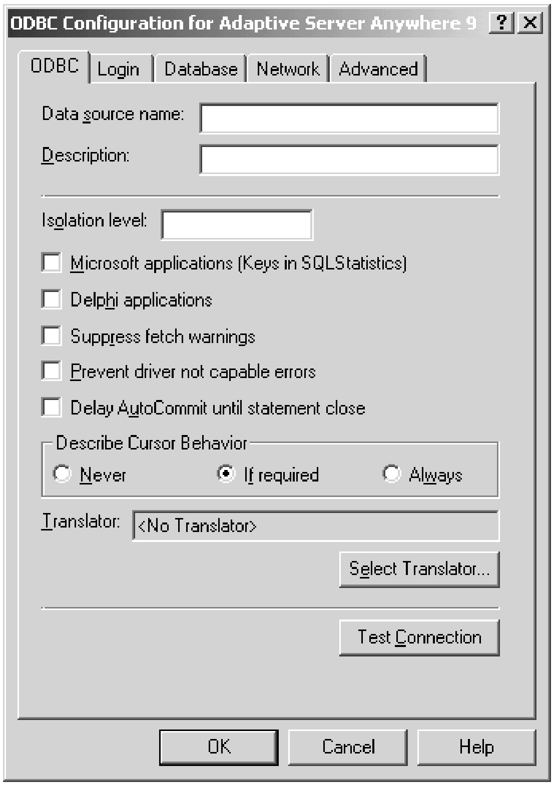
   The ODBC Data Source Administrator dialog appears.

2. Click Add.

   The Create New Data Source wizard appears.

3. From the list of drivers, choose Adaptive Server Anywhere 9.0, and click Finish.

   The ODBC Configuration for Adaptive Server Anywhere dialog appears.


ODBC Configuration for Adaptive Server Anywhere 9

Data source name:

Description:

Isolation level:

- [ ] Microsoft applications (Keys in SQLStatistics)
- [ ] Delphi applications
- [ ] Suppress fetch warnings
- [ ] Prevent driver not capable errors
- [ ] Delay AutoCommit until statement close

Describe Cursor Behavior: Never [✓ If required] Always

Translator: <No Translator>

Most of the fields in this window are optional. Click the Help button to find more information about the fields on each tab.

☞ For more information about the fields in the dialog, see "ODBC Configuration Dialog Help" [*SQL Anywhere Studio Help*, page 11].

4. When you have specified the parameters you need, click OK to close the window and create the data source.

To edit a data source, find and select the desired data source in the ODBC Administrator main window and click Configure.

Creating an ODBC data source from the command line

You can create User and System Data Sources using the dbdsn utility. You cannot create File Data Sources or System Data Sources in this way. File and System Data Sources are limited to Windows operating systems only, and you can use the ODBC Administrator to create File Data Sources.

❖ To create an ODBC data source (Command line)

1. Open a command prompt.

2. Type a `dbdsn` command, specifying the connection parameters you wish to use.

   For example, the following command creates a data source for the Adaptive Server Anywhere sample database. The command must be typed on one line:

```
dbdsn -w "My DSN" "uid=DBA;pwd=SQL;dbf=c:\Program Files\Sybase\
         SQL Anywhere 9\asademo.db"
```

☞ For more information on the dbdsn utility, see "The Data Source utility" on page 510.

Using file data sources on Windows

On Windows operating systems, ODBC data sources are typically stored in the system registry. File data sources are an alternative, which are stored as files. In Windows, file data sources typically have the extension *.dsn*. They consist of sections, each section starting with a name enclosed in square brackets.

To connect using a File Data Source, use the FileDataSourceName (FILEDSN) connection parameter. You cannot use both DataSourceName (DSN) and FileDataSourceName (FILEDSN) in the same connection.

File data sources can be distributed

One benefit of file data sources is that you can distribute the file to users. If the file is placed in the default location for file data sources, it is picked up automatically by ODBC. In this way, managing connections for many users can be made simpler.

❖ To create an ODBC file data source (ODBC Administrator)

1. Start the ODBC Administrator, click the File DSN tab and then click Add.

2. Select Adaptive Server Anywhere 9.0 from the list of drivers, and then click Next.

3. Follow the instructions to create the data source.

Using ODBC data sources on UNIX

On UNIX operating systems, ODBC data sources are held in a file named *.odbc.ini*. A sample file looks like this:

```
[My Data Source]
ENG=myserver
CommLinks=tcpip(Host=hostname)
uid=DBA
pwd=SQL
```

You can type any connection parameter in the *.odbc.ini* file.

☞ For a complete list or connection parameters, see "Connection parameters" on page 176.

Network protocol options are added as part of the CommLinks (LINKS) parameter.

☞ For a complete list of network protocol options, see "Network protocol options" on page 206.

You can create and manage ODBC data sources on UNIX using the dbdsn utility.

☞ For more information, see "Creating an ODBC data source" on page 53, and "The Data Source utility" on page 510.

Caution
You should not add simple encryption to the .odbc.ini system information file with the File Hiding utility (dbfhide) on UNIX unless you will only be using Adaptive Server Anywhere data sources. If you plan to use other data sources (for example, for MobiLink synchronization), then obfuscating the contents of the .odbc.ini file may prevent other drivers from functioning properly.

File location

The database server looks for the *.odbc.ini* file in the following locations:

1. ODBCINI environment variable
2. ODBCHOME and HOME environment variables
3. The user's home directory
4. The path

Using ODBC data sources on Windows CE

Windows CE does not provide an ODBC driver manager or an ODBC Administrator. On this platform, Adaptive Server Anywhere uses ODBC data sources stored in files. You can specify either the DSN or the FILEDSN keyword to use these data source definitions—on Windows CE (only), DSN and FILEDSN are synonyms.

Data source location

Windows CE searches for the data source files in the following locations:

1. The directory from which the ODBC driver (*dbodbc9.dll*) was loaded. This is usually the Windows directory.

2. The directory specified in Location key of the Adaptive Server Anywhere section of the registry. This is usually the same as the Adaptive Server Anywhere installation directory. The default installation directory is:

   ```
   \Program Files\Sybase\ASA
   ```

Each data source itself is held in a file. The file has the same name as the data source, with an extension of *.dsn*.

☞ For more information about file data sources, see "Using file data sources on Windows" on page 56.

Connecting from desktop applications to a Windows CE database

You can connect from applications running on a desktop PC, such as Sybase Central or Interactive SQL, to a database server running on a Windows CE device. The connection uses TCP/IP over the ActiveSync link between the desktop machine and the Windows CE device.

If you are using an Ethernet connection between your desktop machine and the Windows CE device the following procedure works. Connecting from the desktop to the CE device is not supported if ActiveSync is using a USB connection.

☞ For information about using a serial cable connection and ActiveSync 3.x, see "Using ActiveSync 3.0 and a serial cable" on page 60.

❖ To connect from a desktop application to a Windows CE database server

1. Determine the IP address of the server. Start the server on the Windows CE device with the -z option (output extra debug information).

   For example:

   ```
   dbsrv9 -z -x tcpip -n TestServer asademo.db
   ```

   With the -z option, the server writes out its IP address during startup. The address may change if you disconnect your CE device from the network and then re-connect it.

   To change between static and dynamic IP assignment for the CE device, configure the settings in the Windows Control Panel. Open Network, and choose the Services tab. Select Remote Access Service and click Properties ➤ Network ➤ TCP/IP Configuration.

2. Create an ODBC profile on your desktop machine.

   Open the ODBC Administrator, and click Add. Choose Adaptive Server Anywhere 9.0 from the list of drivers and click Finish. The ODBC Configuration for Adaptive Server Anywhere dialog appears.

   - On the Login tab, type a user ID and password.
   - On the Database tab, type the server name.
   - On the Network tab, check TCP/IP, and type the following in the adjacent field:

     ```
     dobroadcast=DIRECT;host=XXX.XXX.XXX.XXX
     ```

     where *XXX.XXX.XXX.XXX* is the server IP address.

- On the ODBC tab, click Test Connection to confirm that your ODBC data source is properly configured.

3. Exit the ODBC Administrator.

4. Ensure the database server is running on your Windows CE machine.

5. On your desktop machine, start an application such as Interactive SQL and select the ODBC data source you have created. The application connects to the Windows CE database.

Using ActiveSync 3.0 and a serial cable

To connect to a Windows CE device from your desktop over a serial cable, Adaptive Server Anywhere uses the TCP/IP protocol. For TCP/IP to work in this context, your ActiveSync installation must be set up to provide a Remote Access Service (RAS) link between desktop machine and Windows CE device.

☞ For information, see "Connecting from desktop applications to a Windows CE database" on page 59.

ActiveSync 3.x does not install and configure RAS. You must install RAS yourself to obtain TCP/IP connectivity to your device over a serial connection.

Instructions for installing RAS are provided by Microsoft. They are available at *http://support.microsoft.com/support/kb/articles/Q241/2/16.ASP* (Microsoft Knowledge Base article Q241216). You must follow these instructions *exactly*, including re-installing any Windows NT service packs, and granting your user account dial-in access using User Manager.

As you follow the instructions, where it says to install your modem, choose Dial-up Networking Serial Cable between 2 PCs instead.

Using RAS with ActiveSync

The following list includes suggestions for enabling ActiveSync 3.0 connections over a serial connection using RAS:

1. In the ActiveSync Connection Settings, select the checkbox Allow network (Ethernet) and Remote Access Service (RAS) server connection with desktop computer. You may need to turn off the checkbox Allow serial cable or infrared connection to this COM port.

2. On the desktop, using Remote Access Administrator (under Administrative Tools on Windows NT), start RAS on COM1.

3. On the Windows CE device, run the ActiveSync client (*repllog.exe* on a Windows CE PC). Choose Serial Connection.

4. Wait for a connection to be established.
5. As a test, run the *ipconfig* utility on Windows NT, and see the 192.168.55.100 static IP of the device. This is the IP you would use when connecting to an Adaptive Server Anywhere database server (for example) running on the CE device.
6. If you switch devices, Stop and Restart the RAS service (or reboot).
7. If everything is set up as above, but you still fail to get a connection from the device to the desktop, you should make sure your Port settings match the baud rates in the Modems Control Panel applet.

Connecting to a database using OLE DB

OLE DB uses the Component Object Model (COM) to make data from a variety of sources available to applications. Relational databases are among the classes of data sources that you can access through OLE DB.

This section describes how to connect to an Adaptive Server Anywhere database using OLE DB from the following environments:

- ♦ Microsoft ActiveX Data Objects (ADO) provides a programming interface for OLE DB data sources. You can access Adaptive Server Anywhere from programming tools such as Microsoft Visual Basic.
- ♦ Sybase PowerBuilder can access OLE DB data sources, and you can use Adaptive Server Anywhere as a PowerBuilder OLE DB database profile.

This section is an introduction to how to use OLE DB from Sybase PowerBuilder and Microsoft ADO environments such as Visual Basic. It is not complete documentation on how to program using ADO or OLE DB. The primary source of information on development topics is your development tool documentation.

☞ For more information about OLE DB, see “Introduction to OLE DB” [*ASA Programming Guide*, page 328].

OLE DB providers

You need an OLE DB provider for each type of data source you wish to access. Each **OLE DB provider** is a dynamic-link library. There are two OLE DB providers you can use to access Adaptive Server Anywhere:

- ♦ **Sybase ASA OLE DB provider** The Adaptive Server Anywhere OLE DB provider provides access to Adaptive Server Anywhere as an OLE DB data source without the need for ODBC components. The short name for this provider is **ASAProv**.

  When the **ASAProv** provider is installed, it registers itself. This registration process includes making registry entries in the COM section of the registry so that ADO can locate the DLL when the **ASAProv** provider is called. If you change the location of your DLL, you must re-register it.

  ☞ For more information about OLE DB providers, see “Introduction to OLE DB” [*ASA Programming Guide*, page 328].

- ♦ **Microsoft OLE DB provider for ODBC** Microsoft provides an OLE DB provider with a short name of **MSDASQL**.

The **MSDASQL** provider makes ODBC data sources appear as OLE DB data sources. It requires the Adaptive Server Anywhere ODBC driver.

Connecting from ADO

ADO is an object-oriented programming interface. In ADO, the **Connection** object represents a unique session with a data source.

You can use the following Connection object features to initiate a connection:

- ♦ The Provider property that holds the name of the provider. If you do not supply a Provider name, ADO uses the MSDASQL provider.
- ♦ The ConnectionString property that holds a connection string. This property holds an Adaptive Server Anywhere connection string, which is used in the same way as the ODBC driver. You can supply ODBC data source names, or explicit UserID, Password, DatabaseName, and other parameters, just as in other connection strings.
- ♦ The Open method initiates a connection.

☞ For more information about ADO, see "ADO programming with Adaptive Server Anywhere" [*ASA Programming Guide,* page 329].

Example

The following Visual Basic code initiates an OLE DB connection to Adaptive Server Anywhere:

```
' Declare the connection object
Dim myConn as New ADODB.Connection
myConn.Provider = "ASAProv"
myConn.ConnectionString = "Data Source=ASA 9.0 Sample"
myConn.Open
```

Connection parameter tips

Connection parameters often provide more than one way of accomplishing a given task. This is particularly the case with embedded databases, where the connection string starts a database server. For example, if your connection starts a database, you can specify the database name using the DatabaseName (DBN) connection parameter or using the DatabaseSwitches (DBS) parameter. The Database Name (DBN) connection parameter is recommended.

Here are some recommendations and notes for situations where connection parameters conflict:

- **Specify database files using DBF** You can specify a database file on the StartLine (START) parameter or using the DatabaseFile (DBF) connection parameter (recommended).
- **Specify database names using DBN** You can specify a database name on the StartLine (START) parameter, the DatabaseSwitches (DBS) connection parameter, or using the DatabaseName (DBN) connection parameter (recommended).
- **Use the Start parameter to specify cache size** Even though you use the DatabaseFile (DBF) connection parameter to specify a database file, you may still want to tune the way in which it starts. You can use the StartLine (START) connection parameter to do this.

  For example, if you are using the Java features of Adaptive Server Anywhere, you should provide additional cache memory on the StartLine (START) connection parameter. The following sample set of embedded database connection parameters describes a connection that may use Java features:

  ```
  DBF=path\asademo.db
  dbn=Sample
  ENG=Sample Server
  uid=DBA
  pwd=SQL
  Start=dbeng9 -c 8M
  ```

☞ For a list of connection parameters, see “Connection Parameters and Network Protocol Options” on page 175.

☞ For character set issues in connection strings, see “Connection strings and character sets” on page 343.

Notes about connection parameters

- **Boolean values** Boolean (true or false) arguments are either YES, ON, 1, or TRUE if true, or NO, OFF, 0, or FALSE if false.

- ♦ **Case sensitivity** Connection parameters are case insensitive.
- ♦ The connection parameters used by the interface library can be obtained from the following places (in order of precedence):
  - **Connection string** You can pass parameters explicitly in the connection string.
  - **SQLCONNECT environment variable** The SQLCONNECT environment variable can store connection parameters.
  - **Data sources** ODBC data sources can store parameters.
- ♦ **Character set restrictions** The server name must be composed of the ASCII character set in the range 1 to 127. There is no such limitation on other parameters.

  ☞ For more information on the character set issues, see "Connection strings and character sets" on page 343.
- ♦ **Priority** The following rules govern the priority of parameters:
  - The entries in a connect string are read left to right. If the same parameter is specified more than once, the last one in the string applies. ODBC and OLE DB are an exception to this: if the same parameter is specified more than once, the first string applies.
  - If a string contains a data source or file data source entry, the profile is read from the configuration file, and the entries from the file are used if they are not already set. For example, if a connection string contains a data source name and sets some of the parameters contained in the data source explicitly, then in case of conflict the explicit parameters are used.

Troubleshooting connections

Who needs to read this section?

In many cases, establishing a connection to a database is straightforward using the information presented in the first part of this chapter.

However, if you are having problems establishing connections to a server, you may need to understand the process by which Adaptive Server Anywhere establishes connections in order to resolve your problems. This section describes how Adaptive Server Anywhere connections work.

☞ For more information about network-specific issues, including connections across firewalls, see "Client/Server Communications" on page 85.

The software follows exactly the same procedure for each of the following types of client application:

- ♦ **ODBC** Any ODBC application using the SQLDriverConnect function, which is the common method of connection for ODBC applications. Many application development systems, such as Sybase PowerBuilder, belong to this class of application. The SQLConnect function is also available to ODBC applications.
- ♦ **Embedded SQL** Any client application using embedded SQL and using the recommended function for connecting to a database (db_string_connect).

  In addition, The SQL CONNECT statement is available for embedded SQL applications and in Interactive SQL. It has two forms: CONNECT AS… and CONNECT USING. All the database administration tools, including Interactive SQL, use db_string_connect.
- ♦ **OLE DB** Any ADO application using the ADODB Connection object. The Provider property is used to locate the OLE DB driver. The Connection String property may use **DataSource** as an alternative to **DataSourceName** and **User ID** as an alternative to **UserID**.
- ♦ **JDBC** Applications using the iAnywhere JDBC driver pass the URL **jdbc:odbc:** followed by a standard connection string as a parameter to the Driver Manager.GetConnection method. The connection string must include **DataSource=** and name an Adaptive Server Anywhere data source or include **Driver=Adaptive Server Anywhere 9.0** (this parameter is specified as **Driver=libdbodb9** on UNIX and Linux).

☞ For more troubleshooting tips, see "Troubleshooting server startup" on page 34, and "Troubleshooting network communications" on page 98.

The steps in establishing a connection

To establish a connection, Adaptive Server Anywhere carries out the following steps in order:

1. **Locate the interface library** The client application must locate the interface library.
2. **Assemble a list of connection parameters** Since connection parameters may appear in several places (such as data sources, a connection string assembled by the application, or an environment variable) Adaptive Server Anywhere assembles the parameters into a single list.
3. **Locate a server** Using the connection parameters, Adaptive Server Anywhere locates a database server on your machine or over a network.
4. **Locate the database** Adaptive Server Anywhere locates the database you want to connect to.
5. **Start a personal server** If Adaptive Server Anywhere fails to locate a server, it attempts to start a personal database server and load the database.

The following sections describe each of these steps in detail.

Locating the interface library

The client application makes a call to one of the Adaptive Server Anywhere interface libraries. In general, the location of this DLL or shared library is transparent to the user. Here we describe how to locate the library in case of problems.

ODBC driver location

For ODBC, the interface library is also called an ODBC driver. An ODBC client application calls the ODBC driver manager, and the driver manager locates the Adaptive Server Anywhere driver.

The ODBC driver manager looks in the supplied data source in the *.odbc.ini* file or registry to locate the driver. When you create a data source using the ODBC Administrator, Adaptive Server Anywhere fills in the current location for your ODBC driver.

Embedded SQL interface library location

Embedded SQL applications call the interface library by name. The name of the Adaptive Server Anywhere embedded SQL interface library is as follows:

♦ **Windows NT/2000/XP and Windows 95/98/Me** *dblib9.dll*

- **UNIX** *dblib9* with an operating-system-specific extension
- **NetWare** *dblib9.nlm*

OLE DB driver location

The provider name (ASAProv) is used to locate the ASAOLEDB DLL based on entries in the registry. The entries are created when the ASAProv provider is installed or if it is reregistered.

iAnywhere JDBC driver location

The Java package *jodbc.jar* must be in the classpath when you run your application. The native DLLs or shared objects must be in the system path.

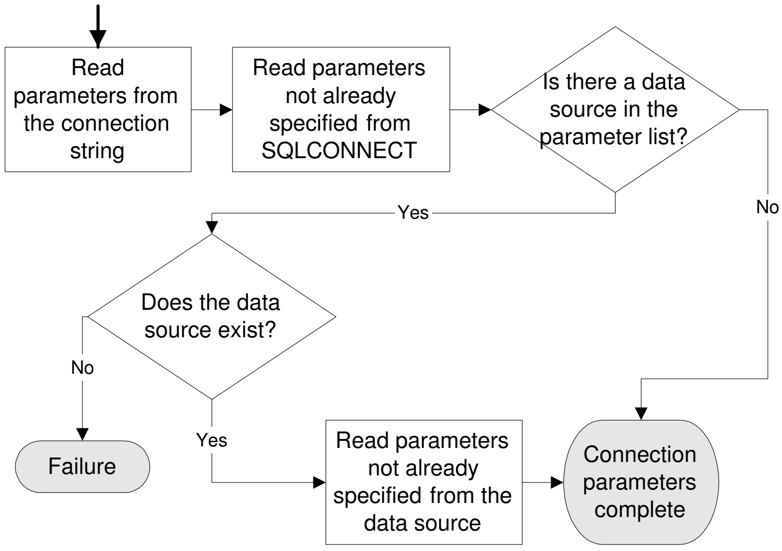
- **PC operating systems** On PC operating systems such as Windows, files are looked for in the current directory, in the system path, and in the *Windows* and *Windows\system* directories.
- **UNIX operating systems** On UNIX, files are looked for in the system path and the user library path.
- **NetWare** On NetWare, files are looked for in the search path, and in the *sys:system* directory.

When the library is located

Once the client application locates the interface library, it passes a connection string to it. The interface library uses the connection string to assemble a list of connection parameters, which it uses to establish a connection to a server.

Assembling a list of connection parameters

The following figure illustrates how the interface libraries assemble the list of connection parameters they use to establish a connection.


Read parameters from the connection string
Read parameters not already specified from SQLCONNECT
Is there a data source in the parameter list?
Yes
No
Does the data source exist?
No
Yes
Failure
Read parameters not already specified from the data source
Connection parameters complete

Notes

Key points from the figure include:

- ♦ **Precedence** Parameters held in more than one place are subject to the following order of precedence:
  1. Connection string
  2. SQLCONNECT
  3. Data source

  That is, if a parameter is supplied both in a data source and in a connection string, the connection string value overrides the data source value.

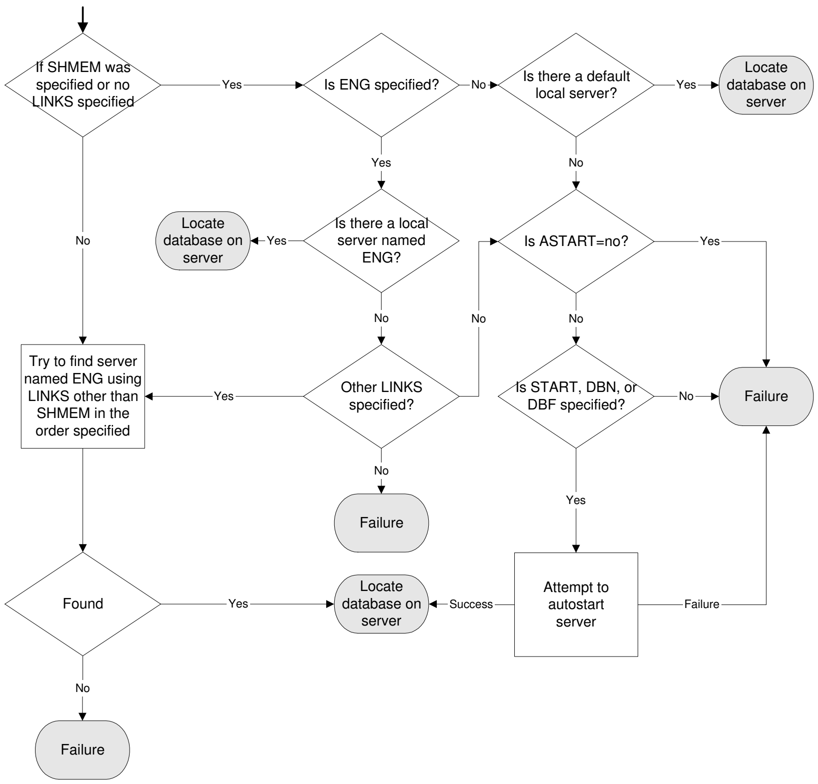
- ♦ **Failure** Failure at this stage occurs only if you specify in the connection string or in SQLCONNECT a data source that does not exist.

- ♦ **Common parameters** Depending on other connections already in use, some connection parameters may be ignored, including:
  - **AutoStop** Ignored if the database is already loaded.
  - **DatabaseFile** Ignored of DatabaseName is specified and a database with this name is already running.

The interface library uses the completed list of connection parameters to attempt to connect.

Locating a server

In the next step towards establishing a connection, Adaptive Server Anywhere attempts to locate a server. If the connection parameter list includes a server name (EngineName (ENG) connection parameter), it carries out a search for a server of that name. If no EngineName (ENG) connection parameter is supplied, and LINKS is not specified or LINKS includes Shared Memory, Adaptive Server Anywhere looks for a default server.


If SHMEM was specified or no LINKS specified
Yes
Is ENG specified?
No
Is there a default local server?
Yes
Locate database on server
Yes
No
No
Locate database on server
Yes
Is there a local server named ENG?
Is ASTART=no?
Yes
No
No
No
Try to find server named ENG using LINKS other than SHMEM in the order specified
Yes
Other LINKS specified?
Is START, DBN, or DBF specified?
No
Failure
No
Yes
Failure
Found
Yes
Locate database on server
Success
Attempt to autostart server
Failure
No
Failure

If Adaptive Server Anywhere locates a server, it tries to locate or load the required database on that server.

☞ For information, see "Locating the database" on page 71.

If Adaptive Server Anywhere cannot locate a server, it may attempt to start a personal server, depending on the connection parameters.

Notes

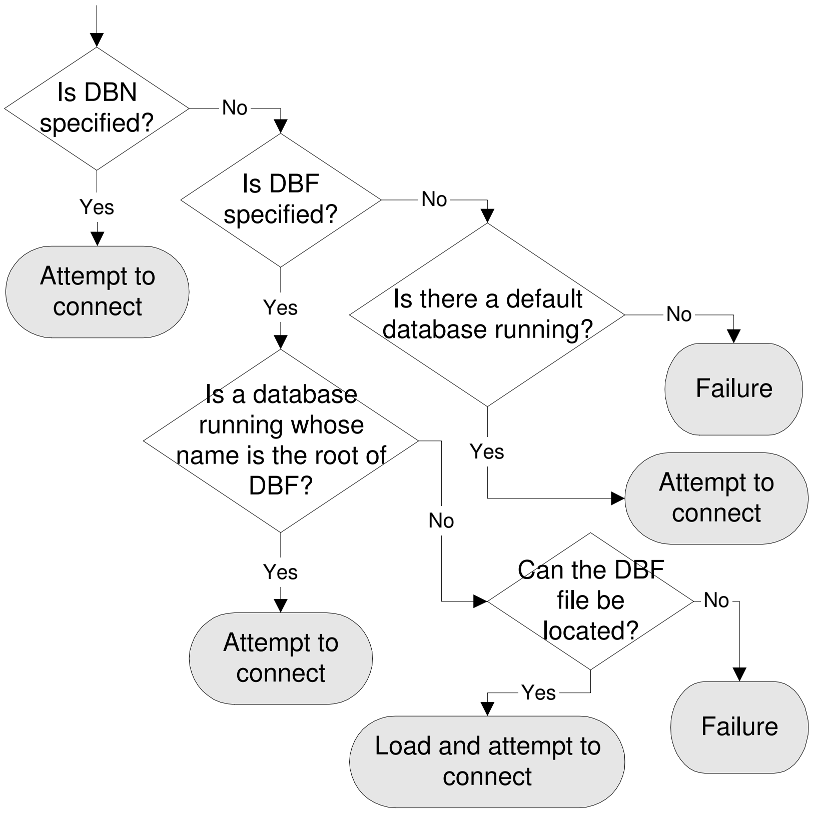
- For local connections, locating a server is simple. For connections over a network, you can use the CommLinks (LINKS) connection parameter to tune the search in many ways by supplying network protocol options.
- The network search involves a search over one or more of the protocols supported by Adaptive Server Anywhere. For each protocol, the network library starts a single port. All connections over that protocol at any one time use a single port.
- You can specify a set of network protocol options for each network port in the argument to the CommLinks (LINKS) connection parameter.
- Each attempt to locate a server involves two steps. First, Adaptive Server

Anywhere looks in the server name cache to see if a server of that name is available (this step is skipped if the value of DoBroadcast is none). Second, it uses the available connection parameters to attempt a connection.

- ♦ If the server is autostarted, information from the START, DBF, DBKEY, DBS, DBN, ENG, and AUTOSTOP connection parameters are used to construct the options for the autostarted server.

Locating the database

If Adaptive Server Anywhere successfully locates a server, it then tries to locate the database. For example:


Is DBN specified?
No
Yes
Attempt to connect
Is DBF specified?
No
Yes
Is there a default database running?
No
Failure
Yes
Attempt to connect
Is a database running whose name is the root of DBF?
No
Yes
Attempt to connect
Can the DBF file be located?
No
Failure
Yes
Load and attempt to connect

Server name caching for faster connections

When the DoBroadcast (DOBROAD) protocol option is set to DIRECT or ALL, the network library looks for a database server on a network by broadcasting over the network using the CommLinks (LINKS) connection parameter.

Tuning the broadcast

The CommLinks (LINKS) parameter takes as argument a string listing the protocols to use and, optionally for each protocol, a variety of network protocol options that tune the broadcast.

☞ For more information about network protocol options, see "Network protocol options" on page 206.

Caching server information

Broadcasting over large networks searching for a server of a specific name can be time-consuming. Caching server addresses speeds up network connections by saving the protocol the first connection to a server was found on, and its address, to a file and using that information for subsequent connections.

The server information is saved in a cached file named *asasrv.ini*. The file contains a set of sections, each of the following form:

```
[Server name]
Link=protocol_name
Address=address_string
```

The *asasrv.ini* file is located in your Adaptive Server Anywhere executable directory (the same directory as the ODBC/ DBLib DLL).

You can obfuscate the contents of the *asasrv.ini* file with simple encryption using the File Hiding utility.

☞ For more information, see "Hiding the contents of .ini files" on page 524.

Note:
It is very important that each server has a unique name. Giving different servers the same name can lead to identification problems.

How the cache is used

If the server name and protocol in the cache match the connection string, Adaptive Server Anywhere tries to connect using the cached address first. If that fails, or if the server name and protocol in the cache do not match the connection string, the connection string information is used to search for the server using a broadcast. If the broadcast is successful, the server name entry in the cache is overwritten. If no server is found, the server name entry in the cache is removed. If the DoBroadcast protocol option is set to none, any cached addresses are ignored.

Interactive SQL connections

The Interactive SQL utility has a different behavior from the default embedded SQL behavior when a CONNECT statement is issued while already connected to a database. If no database or server is specified in the CONNECT statement, Interactive SQL connects to the current database, rather than to the default database. This behavior is required for database reloading operations.

☞ For more information, see “CONNECT statement [ESQL] [Interactive SQL]” [*ASA SQL Reference*, page 332].

Testing that a server can be found

The dbping utility is provided to help in troubleshooting connections. In particular, you can use it to test if a server with a particular name is available on your network.

The dbping utility takes a connection string as an option, but by default only those pieces required to locate a server are used. By default, it does not attempt to start a server, but may if the -d option is specified.

Examples

The following command line tests to see if a server named Waterloo is available over a TCP/IP connection:

```
dbping -c "eng=Waterloo;CommLinks=tcpip"
```

The following command tests to see if a default server is available on the current machine.

```
dbping
```

☞ For more information on dbping options, see “The Ping utility” on page 563.

Using integrated logins

The **integrated login** feature allows you to maintain a single user ID and password for both database connections and operating system and/or network logins. This section describes the integrated login feature.

Operating systems supported

Integrated login capabilities are available for Windows NT/2000/XP servers only. It is possible for Windows 95/98/Me clients, as well as Windows NT/2000/XP clients, to use integrated logins to connect to a network server running on Windows NT/2000/XP.

Benefits of an integrated login

An integrated login is a mapping from one or more Windows users or Windows user group profiles to an existing user in a database. A user who has successfully navigated the security for that user profile or group and logged in to a machine can connect to a database without providing an additional user ID or password.

To accomplish this, the database must be configured to use integrated logins and a mapping must have been granted between the user or group profile used to log in to the machine and/or network, and a database user.

Using an integrated login is more convenient for the user and permits a single security system for database and network security. Its advantages include:

- When connecting to a database using an integrated login, the user does not need to type a user ID or password.
- If you use an integrated login, the user authentication is done by the operating system, not the database: a single system is used for database security and machine or network security.
- Multiple user or group profiles can be mapped to a single database user ID.
- The name and password used to login to the Windows NT/2000/XP machine do not have to match the database user ID and password.

Caution
Integrated logins offer the convenience of a single security system, but there are important security implications that database administrators should be familiar with.

☞ For more information about security and integrated logins, see "Security concerns: unrestricted database access" on page 82.

Creating integrated logins for Windows user groups

In addition to creating integrated logins for individual Windows users, you can create integrated logins for Windows user groups.

When a user logs in, if the supplied Windows user does not have an explicit integrated login mapping, but belongs to a Windows user group for which there is an integrated login mapping, the user connects to the database as the database user or group specified in the Windows user group's integrated login mapping.

Caution

Creating an integrated login for a Windows user group allows any user that is a member of the group to connect to the database without knowing a user ID or password.

☞ *For ways to prevent particular users that are members of a Windows user group from connecting to a database, see "Preventing members of Windows user groups from connecting to a database" on page 76.*

Members of multiple groups

If the Windows user belongs to more than one Windows user group, and more than one Windows user group on the machine has an integrated login mapping in the database, then the integrated login only succeeds if all of the Windows user groups on the machine have integrated login mappings to the same database user ID or group. If multiple Windows user groups have integrated login mappings to different database user IDs or groups, an error is returned and the integrated login fails.

For example, consider a database with two user IDs, dbuserA and dbuserB, and the Windows user windowsuser who belongs to the Windows user groups ntgroupA and ntgroupB.

This SQL statement...	Allows...
`GRANT INTEGRATED LOGIN windowsuser AS USER dbuserA`	windowsuser to connect to the database using the integrated login mapping set explicitly for windowsuser.
`GRANT INTEGRATED LOGIN ntgroupA AS USER dbuserB`	windowsuser to connect to the database using the integrated login mapping granted to ntgroupA.

This SQL statement...	Allows...
`GRANT INTEGRATED LOGIN ntgroupA AS USER dbuserB GRANT INTEGRATED LOGIN ntgroupb AS USER dbuserB`	windowsuser to connect to the database because both Windows user groups that windowsuser belongs to have an integrated login mapping to the same database user.
`GRANT INTEGRATED LOGIN ntgroupA AS USER dbuserA GRANT INTEGRATED LOGIN ntgroupb AS USER dbuserB`	no connection to the database. When windowsuser attempts to connect to the database, the integrated login fails because each Windows user group has an integrated login mapping to a different database user and windowsuser is a member of both Windows user groups.

Domain Controller locations

By default, the machine the Adaptive Server Anywhere database server is running on is used to verify Windows user group membership. If the Domain Controller server is a different machine than the one the database server is running on, you can specify the name of the Domain Controller server using the INTEGRATED_SERVER_NAME option. For example:

```
SET PUBLIC.OPTION INTEGRATED_SERVER_NAME = '\\myserver-1'
```

☞ For more information, see "INTEGRATED_SERVER_NAME option [database]" on page 658.

Preventing members of Windows user groups from connecting to a database

Creating an integrated login for a Windows user group allows any user that is a member of the group to connect to the database without knowing a database user ID or password. There are two methods you can use to prevent a user who is a member of a Windows user group that has an integrated login from connecting to a database using the group integrated login:

1. Create an integrated login for the user to a database user ID that does not have a password.
2. Created a stored procedure that is called by the LOGIN_PROCEDURE option to check whether a user is allowed to log in, and raise an exception when a disallowed user tries to connect.

You can use either of these methods to prevent members of Windows user groups from connecting to a database.

Creating an integrated login to a user ID with no password

When a user is a member of a Windows user group that has an integrated login, but also has an explicit integrated login for their user ID, the user's integrated login is used to connect to the database. To prevent a user from connecting to a database using their Windows user group integrated login, you can create an integrated login for the Windows user to a database user ID without a password. Database user IDs that do not have a password cannot connect to a database.

❖ To create an integrated login to a user ID with no password

1. Add a user to the database without a password. For example:

   ```
   GRANT CONNECT TO db_user_no_password
   ```

2. Create an integrated login for the Windows user that maps to the database user without a password. For example:

   ```
   GRANT INTEGRATED LOGIN TO WindowsUser
   AS USER db_user_no_password
   ```

Creating a procedure to prevent Windows users from connecting

The LOGIN_PROCEDURE option specifies a stored procedure to be called each time a connection to the database is attempted. By default, the dbo.sp_login_environment procedure is called. You can set the LOGIN_PROCEDURE option to call a procedure you have written that prevents specific users from connecting to the database.

The following example creates a procedure named login_check that is called by the LOGIN_PROCEDURE option. The login_check procedure checks the supplied user name against a list of users that are not allowed to connect to the database. If the supplied user name is found in the list, the connection fails. In this example, users named Joe, Harry, or Martha are not allowed to connect. If the user is not found in the list, the database connection proceeds as usual and calls the sp_login_environment procedure.

```
CREATE PROCEDURE DBA.user_login_check()
   BEGIN
      DECLARE INVALID_LOGON EXCEPTION FOR SQLSTATE '28000';
      // Disallow certain users
      IF( CURRENT USER IN ('Joe','Harry','Martha') ) THEN
        SIGNAL INVALID_LOGON;
      ELSE
         CALL sp_login_environment;
      END IF;
   END
go
GRANT EXECUTE ON DBA.user_login_check TO PUBLIC
go
SET OPTION PUBLIC.LOGIN_PROCEDURE='DBA.user_login_check'
go
```

Setting up integrated logins

Several steps must be implemented in order to connect successfully via an integrated login.

❖ To use an integrated login

1. Enable the integrated login feature in a database by setting the value of the LOGIN_MODE database option to either Mixed or Integrated (the option is case insensitive), in place of the default value of Standard. This step requires DBA authority.

2. Create an integrated login mapping between a Windows user or group profile and an existing database user. This can be done using a SQL statement or a wizard in Sybase Central. In Sybase Central, all users with integrated login permission appear in the Integrated Logins folder.

3. Connect from a client application in such a way that the integrated login facility is triggered.

Each of these steps is described in the sections below.

Enabling the integrated login feature

The LOGIN_MODE database option determines whether the integrated login feature is enabled. As database options apply only to the database in which they are found, different databases can have a different integrated login setting even if they are loaded and running within the same server.

The LOGIN_MODE database option accepts one of following three values (which are case insensitive).

- **Standard** With this setting, integrated logins are not permitted. This is the default setting. An error occurs if an integrated login connection is attempted.
- **Mixed** With this setting, both integrated logins and standard logins are allowed.
- **Integrated** With this setting, all logins to the database must be made using integrated logins.

Caution
Setting the LOGIN_MODE database option to Integrated restricts connections to only those users who have been granted an integrated login mapping. Attempting to connect using a user ID and password generates an error. The only exception to this is users with DBA authority (full administrative rights).

Example

The following SQL statement sets the value of the LOGIN_MODE database option to Mixed, allowing both standard and integrated login connections:

```
SET OPTION Public.LOGIN_MODE = Mixed
```

Creating an integrated login

User profiles can only be mapped to an existing database user ID. When that database user ID is removed from the database, all integrated login mappings based on that database user ID are automatically removed.

A user or group profile does not have to exist for it to be mapped to a database user ID. More than one user profile can be mapped to the same user ID.

Only users with DBA authority are able to create or remove an integrated login mapping.

An integrated login mapping is made either using a wizard in Sybase Central or a SQL statement.

❖ To map an integrated login (Sybase Central)

1. Connect to a database as a user with DBA authority.
2. Click the Integrated Logins folder in the left pane.
3. From the File menu, choose New ➤ Integrated Login.
4. On the first page of the wizard, specify the name of the system (computer) user or group for whom the integrated login is to be created.

Also, select the database user ID this user maps to. The wizard displays the available database users. You must select one of these. You cannot add a new database user ID.

5. Follow the remaining instructions in the wizard.

❖ To map an integrated login (SQL)

1. Connect to a database with DBA authority.
2. Execute a GRANT INTEGRATED LOGIN TO statement.

Example

The following SQL statement allows Windows NT users fran_whitney and matthew_cobb to log in to the database as the user DBA, without having to know or provide the DBA user ID or password.

```
GRANT INTEGRATED LOGIN
TO fran_whitney, matthew_cobb
AS USER DBA
```

☞ For more information, see "GRANT statement" [*ASA SQL Reference,* page 503].

The following SQL statement allows Windows NT users who are members of the Windows NT group mywindowsusers to log in to the database as the user DBA, without having to know or provide the DBA user ID or password.

```
GRANT INTEGRATED LOGIN
TO mywindowsusers
AS USER DBA
```

☞ For more information about creating integrated logins for Windows NT groups, see .

Revoking integrated login permission

You can remove an integrated login mapping using either Sybase Central or Interactive SQL.

❖ To revoke an integrated login permission (Sybase Central)

1. Connect to a database with DBA authority.
2. Open the Integrated Logins folder.
3. In the right pane, right-click the user/group you would like to remove the integrated login permission from and choose Delete from the popup menu.

❖ To revoke an integrated login permission (SQL)

1. Connect to a database with DBA authority.
2. Execute a REVOKE INTEGRATED LOGIN FROM statement.

Example

The following SQL statement removes integrated login permission from the Windows NT user pchin.

```
REVOKE INTEGRATED LOGIN
FROM pchin
```

☞ For more information, see the "REVOKE statement" [*ASA SQL Reference*, page 585].

Connecting from a client application

A client application can connect to a database using an integrated login in one of the following ways:

- Set the Integrated (INT) parameter in the list of connection parameters to YES.
- Specify neither a user ID nor a password in the connection string or Connect dialog.

If `Integrated=YES` is specified in the connection string, an integrated login is attempted. If the connection attempt fails and the LOGIN_MODE database option is set to Mixed, the server attempts a standard login.

If an attempt to connect to a database is made without providing a user ID or password, an integrated login is attempted. The attempt succeeds or fails depending on whether the current user profile name matches an integrated login mapping in the database.

Interactive SQL examples

For example, a connection attempt using the following Interactive SQL statement will succeed, providing the user has logged on with a user profile name that matches a integrated login mapping in a default database of a server:

```
CONNECT USING 'INTEGRATED=yes'
```

The Interactive SQL statement `CONNECT` can connect to a database if all the following are true:

- A server is currently running.
- The default database on the current server is enabled to accept integrated login connections.

- An integrated login mapping has been created that matches the current user's user profile name or for a Windows user group to which the user belongs.
- If the user is prompted with a dialog by the server for more connection information (such as occurs when using the Interactive SQL utility), the user clicks OK *without* providing more information.

Security concerns: unrestricted database access

The integrated login feature works using the login control system of Windows NT/2000/XP in place of the Adaptive Server Anywhere security system. Essentially, the user passes through the database security if they can log in to the machine hosting the database, and if other conditions, outlined in "Using integrated logins" on page 74, are met.

If the user successfully logs in to the Windows NT/2000/XP server as "dsmith", they can connect to the database without further proof of identification provided there is either an integrated login mapping or a default integrated login user ID.

When using integrated logins, database administrators should give special consideration to the way Windows NT/2000/XP enforces login security in order to prevent unwanted access to the database.

In particular, be aware that by default a "Guest" user profile is created and enabled when Windows NT Workstation or Server is installed.

Caution
Leaving the user profile Guest enabled can permit unrestricted access to a database that is hosted by that server.

If the Guest user profile is enabled and has a blank password, any attempt to log in to the server will be successful. It is not required that a user profile exist on the server, or that the login ID provided have domain login permissions. Literally any user can log in to the server using any login ID and any password: they are logged in by default to the Guest user profile.

This has important implications for connecting to a database with the integrated login feature enabled.

Consider the following scenario, which assumes the Windows NT server hosting a database has a Guest user profile that is enabled with a blank password.

- An integrated login mapping exists between the user fran_whitney and the database user ID DBA. When the user fran_whitney connects to the server with her correct login ID and password, she connects to the database as DBA, a user with full administrative rights.

But anyone else attempting to connect to the server as fran_whitney will successfully log in to the server regardless of the password they provide because Windows NT will default that connection attempt to the Guest user profile. Having successfully logged in to the server using the fran_whitney login ID, the unauthorized user successfully connects to the database as DBA using the integrated login mapping.

Disable the Guest user profile for security
The safest integrated login policy is to disable the Guest user profile on any Windows NT/2000/XP machine hosting an Adaptive Server Anywhere database. This can be done using the Windows User Manager utility.

Setting temporary public options for added security

Setting the value of the LOGIN_MODE option for a given database to Mixed or Integrated using the following SQL statement permanently enables integrated logins for that database.

```
SET OPTION Public.LOGIN_MODE = Mixed
```

If the database is shut down and restarted, the option value remains the same and integrated logins are still enabled.

Changing the LOGIN_MODE option temporarily will still allow user access via integrated logins. The following statement will change the option value temporarily:

```
SET TEMPORARY OPTION Public.LOGIN_MODE = Mixed
```

If the permanent option value is Standard, the database will revert to that value when it is shut down.

Setting temporary public options can be considered an additional security measure for database access since enabling integrated logins means that the database is relying on the security of the operating system on which it is running. If the database is shut down and copied to another machine (such as a user's machine) access to the database reverts to the Adaptive Server Anywhere security model and not the security model of the operating system of the machine where the database has been copied.

☞ For more information on using the SET OPTION statement see "SET OPTION statement" [*ASA SQL Reference*, page 613].

Network aspects of integrated logins

If the database is located on a network server, then one of two conditions must be met for integrated logins to be used:

- ♦ The user profile used for the integrated login connection attempt must exist on both the local machine and the server. As well as having identical user profile names on both machines, the passwords for both user profiles must also be identical.

  For example, when the user jsmith attempts to connect using an integrated login to a database loaded on a network server, identical user profile names and passwords must exist on both the local machine and application server hosting the database. jsmith must be permitted to login to both the local machine and the server hosting the network server.

- ♦ If network access is controlled by a Microsoft Domain, the user attempting an integrated login must have domain permissions with the Domain Controller server and be logged in to the network. A user profile on the network server matching the user profile on the local machine is not required.

Creating a default integrated login user

A default integrated login user ID can be created so that connecting via an integrated login will be successful even if no integrated login mapping exists for the user profile currently in use.

For example, if no integrated login mapping exists for the user profile name JSMITH, an integrated login connection attempt will normally fail when JSMITH is the user profile in use.

However, if you create a user ID named Guest in a database, an integrated login will successfully map to the Guest user ID if no integrated login mapping explicitly identifies the user profile JSMITH.

The default integrated login user permits anyone attempting an integrated login to successfully connect to a database if the database contains a user ID named Guest. The authorities granted to the Guest user ID determine the permissions and authorities granted to the newly-connected user.

CHAPTER 3

Client/Server Communications

About this chapter

Each network environment has its own peculiarities. This chapter describes those aspects of network communication that are relevant to the proper functioning of your database server, and provides some tips for diagnosing network communication problems. It describes how networks operate, and provides hints on running the network database server under each protocol.

Network database server only
The material in this chapter applies only to the network server. You do not need to read this chapter if you are using the personal database server.

Contents

Supported network protocols

Properly configured Adaptive Server Anywhere servers run under the following networks and protocols:

- **Windows NT/2000/XP and Windows 95/98/Me** TCP/IP or SPX protocols.
- **Windows CE** TCP/IP protocol.
- **NetWare** TCP/IP or SPX protocol.
- **UNIX** TCP/IP protocol.

In addition, the Named Pipes protocol is provided for communication from a client application running on the same machine.

- **Windows NT/2000/XP/Server 2003** Named Pipes is used for same-machine communication between any application and the server, but is generally not recommended for this purpose. Named Pipes is not used for network communications, but can be used to run Adaptive Server Anywhere in a manner equivalent to a C2-security-certified configuration.

The client library for each platform supports the same protocols as the corresponding server. In order for Adaptive Server Anywhere to run properly, the network protocols (TCP/IP and/or SPX) must be installed and configured properly on both the client and server computers.

Using the TCP/IP protocol

TCP/IP is a suite of protocols originally implemented by the University of California at Berkeley for BSD UNIX. TCP/IP has gained widespread use with the expansion of the Internet and the world wide web.

UDP is a transport layer protocol that sits on top of IP. Adaptive Server Anywhere uses UDP on top of IP to do initial server name resolution and TCP for connection and communication after that.

When you use the TCP/IP protocol, you can secure client/server communications using transport-layer security and ECC_TLS (formerly Certicom) or RSA_TLS encryption technology.

☞ For more information, see "Adaptive Server Anywhere Transport-Layer Security" [*SQL Anywhere Studio Security Guide*, page 27].

Using TCP/IP with Windows

The TCP/IP implementation for database servers on all Windows platforms uses Winsock 2.0. The implementation for clients on Windows NT/2000/XP also uses Winsock 2.0. Clients on Windowsă95/98/Me use Winsock 1.1 by default. Clients on Windows CE use the Winsock 1.1 standard.

If you do not have TCP/IP installed, you can install the TCP/IP protocol from the Control Panel by double-clicking Network Settings.

☞ For more information, see "DLL protocol option" on page 210.

Tuning TCP/IP performance

Increasing the packet size may improve query response time, especially for queries transferring a large amount of data between a client and a server process. You can set the packet size using the -p option in the database server command, or by setting the CommBufferSize (CBSIZE) connection parameter in your connection profile.

☞ For more information, see "-p server option" on page 152, or "CommBufferSize connection parameter [CBSIZE]" on page 180.

Connecting across a firewall

There are restrictions on connections when the client application is on one side of a firewall, and the server is on the other. Firewall software filters network packets according to network port. Also, it is common to disallow UDP packets from crossing the firewall.

When connecting across a firewall, you must use a set of protocol options in the CommLinks (LINKS) connection parameter of your application’s connection string.

- **Host** Set this parameter to the host name on which the database server is running. You can use the short form IP.
- **ServerPort** If your database server is not using the default port of 2638, you must specify the port it is using. You can use the short form Port. This option may not be necessary depending on the firewall’s configuration.
- **ClientPort** Set this parameter to a range of allowed values for the client application to use. You can use the short form CPort.
- **DoBroadcast=NONE** Set this parameter to prevent UDP from being used when connecting to the server.

The firewall must be configured to allow TCP/IP traffic between the Adaptive Server Anywhere server’s address and all the Adaptive Server Anywhere clients’ addresses. The Adaptive Server Anywhere server’s address is the IP address of the machine running the Adaptive Server Anywhere server (the HOST parameter) and the Adaptive Server Anywhere server’s IP port number (the ServerPort protocol option, default 2638). Each Adaptive Server Anywhere client’s address consists of the IP address of the client machine, and the range of the client IP ports (the ClientPort protocol option). For simplest configuration, all client ports can be allowed. If only specific client ports are allowed, specify a range with more ports than the maximum number of concurrent connections from each client machine, since there is a several minute timeout before a client port can be refused.

☞ For more information, see “ClientPort protocol option [CPORT]” on page 209.

Example

The following connection string fragment restricts the client application to ports 5050 through 5060, and connects to a server named **myeng** running on the machine at address **myhost** using the server port 2020. No UDP broadcast is carried out because of the DoBroadcast option.

```
Eng=myeng;Links=tcpip(ClientPort=5050-
        5060;Host=myhost;Port=2020;DoBroadcast=NONE)
```

See also

- “CommLinks connection parameter [LINKS]” on page 181
- “ClientPort protocol option [CPORT]” on page 209
- “ServerPort protocol option [PORT]” on page 220
- “Host protocol option [IP]” on page 212
- “DoBroadcast protocol option [DOBROAD]” on page 211

Connecting on a dial-up network connection

You can use connection and protocol options to assist with connecting to a database across a dial-up link.

On the client side, you should specify the following protocol options:

♦ **Host parameter** You should specify the host name or IP address of the database server using the Host (IP) protocol option.

☞ For more information, see "Host protocol option [IP]" on page 212.

♦ **DoBroadcast parameter** If you specify the Host (IP) protocol option, there is no need to do a broadcast search for the database server. For this reason, use direct broadcasting.

☞ For more information, see "DoBroadcast protocol option [DOBROAD]" on page 211.

♦ **MyIP parameter** You should set **MyIP=NONE** on the client side.

☞ For more information, see "MyIP protocol option [ME]" on page 218.

♦ **TIMEOUT parameter** Set the TIMEOUT (TO) protocol option to increase the time the client will wait while searching for a server.

☞ For more information, see the "Timeout protocol option [TO]" on page 222.

A typical CommLinks (LINKS) connection parameter may look as follows:

```
Links=tcpip(MyIP=NONE;DoBroadcast=DIRECT;Host=server_ip)
```

Encrypting client/server communications over TCP/IP

By default, communication packets are not encrypted; this poses a potential security risk. You can secure communications between client applications and the database server over TCP/IP using simple encryption or transport-layer security. Transport-layer security provides server authentication, strong encryption using ECC_TLS (formerly Certicom) or RSA_TLS encryption technology, and other features for protecting data integrity.

☞ For more information, see "Adaptive Server Anywhere Transport-Layer Security" [*SQL Anywhere Studio Security Guide*, page 27].

Connecting using an LDAP server

You can specify a central LDAP server to keep track of all servers in an enterprise if you are operating on a Windows (except CE and Win64), UNIX or NetWare platform. When the database server registers itself with an LDAP server, clients can query the LDAP server and find the server they are looking for regardless of whether they are on a WAN or LAN, or going through firewalls. They do not need to specify an IP address (HOST=), either. The Server Location utility (dblocate) can use the LDAP server to find other such servers as well.

LDAP is only used with TCP/IP, and only on network servers.

To enable this feature, a file containing information on how to find and connect to the LDAP server must be created on both the server machine and on each client machine. By default the name of this file is *asaldap.ini*, but it is configurable. If this file doesn't exist, LDAP support is silently disabled.

The file must be located in the same directory as the Adaptive Server Anywhere executables (for example, *%asany%\win32* on Windows) unless a full path is specified with the LDAP parameter. The file must be in the following format:

```
[LDAP]
server=<machine running LDAP server>
port=<port number of LDAP server>
basedn=<Base DN>
authdn=<Authentication DN>
password=<password for authdn>
search_timeout=<age of timestamps to be ignored>
update_timeout=<frequency of timestamp updates>
```

You can add simple encryption to obfuscate the contents of the *asaldap.ini* file using the File Hiding utility (dbfhide).

☞ For more information, see "Hiding the contents of .ini files" on page 524.

server the name or IP address of the machine running the LDAP server. This value is required on NetWare and UNIX. If this entry is missing on Windows, Windows looks for an LDAP server running on the local domain controller.

port the port number used by the LDAP server. The default is 389.

basedn the domain name of the subtree where the Adaptive Server Anywhere entries are stored. This defaults to the root of the tree.

authdn the authentication domain name. The domain name must specify an existing user object in the LDAP directory that has write access to the basedn. This is required for the server, and ignored on the client.

password the password for authdn. This is required for the server, and ignored on the client.

search_timeout the age of timestamps at which the timestamp will be ignored by the client and/or the Server Location [dblocate] utility. A value of 0 disables this option so that all entries are assumed to be current. The default is 600 seconds (10 minutes).

update_timeout the frequency of timestamp updates in the LDAP directory. A value of 0 disables this option so that the server never updates the timestamp. The default is 120 seconds (2 minutes).

Example

The following is a sample *asaldap.ini* file:

```
[LDAP]
server=ldapserver
basedn=dc=iAnywhere,dc=com
authdn=cn=ASAServer,ou=iAnywhereASA,dc=iAnywhere,dc=com
password=secret
```

The entries are stored in a subtree of the basedn called iAnywhereASA. This entry must be created before Adaptive Server Anywhere can use LDAP. To create the subtree, you can use the LDAPADD utility, supplying the following information:

```
dn: ou=iAnywhereASA,<basedn>
objectClass: organizationalUnit
objectClass: top
ou: iAnywhereASA
```

When the server starts up, it checks for an existing entry with the same name in the LDAP file. If one is found, it is replaced if either

- the location entries in LDAP match the server attempting to start, or
- the timestamp field in the LDAP entry is more than 10 minutes old (the timeout value is configurable)

If neither of these is true, then there is another server running with the same name as the one attempting to start, and startup fails.

To ensure that entries in LDAP are up-to-date, the server updates a timestamp field in the LDAP entry every 2 minutes. If an entry's timestamp is older than 10 minutes, clients ignore the LDAP entry. Both of these values are configurable.

On the client, the LDAP directory is searched before doing any broadcasting, so if the server is found, no broadcasts are sent. The LDAP search is very fast, so if it fails, there is no discernible delay.

The Server Location utility (dblocate) also uses LDAP—all servers listed in LDAP are added to the list of servers returned. This allows the Server Location utility (dblocate) to list servers that wouldn’t be returned normally, for example, those which broadcasts wouldn’t reach. Entries with timestamps older than 10 minutes are not included.

Using the SPX protocol

SPX is a protocol from Novell. Adaptive Server Anywhere for NetWare, Windows NT/2000/XP, and Windows 95/98/Me can all employ the SPX protocol. This section provides some tips for using SPX under different operating systems.

SPX is not supported on 64-bit operating systems.

Connecting via SPX

Some machines can use the NetWare bindery. These machines are NetWare servers or Windows NT/2000/XP or Windows 95/98/Me machines where the Client Service for NetWare is installed. A client application on one of these machines does not need to use broadcasts to connect to the server if the server to which it is connecting is also using the bindery.

Applications running on machines not using the bindery must connect using one of the following:

- **An explicit address** You can specify an explicit address using the HOST (IP) protocol option.

  ☞ For more information, see the "Host protocol option [IP]" on page 212.

- **Broadcast** If a server is not found in the bindery, the client will attempt to find it using a broadcast. This can be disabled with by setting the DoBroadcast (DOBROAD) protocol option to **NONE** (client-side) or **NO** (server-side).

  ☞ For more information, see the "DoBroadcast protocol option [DOBROAD]" on page 211.

Using Named Pipes

Named Pipes is a facility for interprocess communication. Named Pipes can be used for communication between processes on the same computer or on different computers.

Adaptive Server Anywhere for Windows NT/2000/XP can use local Named Pipes for same-machine communication.

☞ For more information, see "Supported network protocols" on page 86.

Note that shared memory (the default protocol) is more efficient than Named Pipes. The Named Pipes protocol should be used when using Adaptive Server Anywhere in a manner equivalent to a C2-security-certified configuration.

Adaptive Server Anywhere does not use Named Pipes for client/server communications between different machines.

Adjusting communication compression settings to improve performance

Enabling compression for one or all connections, as well as setting the minimum size at which packets are compressed, can significantly improve Adaptive Server Anywhere performance in some circumstances.

To determine if enabling compression will help in your particular situation, we recommend that you conduct a performance analysis on your particular network and using your particular application before using communication compression in a production environment. Performance results will vary according to the type of network you are using, your applications, and the data you transfer.

The most basic way of tuning compression is as simple as enabling or disabling the Compression (COMP) connection parameter on either the connection or server level. More advanced methods of fine tuning compression performance include adjusting the CommBufferSize (CBSIZE) and/or the CompressionThreshold (COMPTH) connection parameters.

Enabling compression increases the quantity of information stored in data packets, thereby reducing the number of packets required to transmit a particular set of data. By reducing the number of packets, the data can be transmitted more quickly.

☞ For more information about performance analysis, see "Performance Monitor statistics" on page 704, or "sa_conn_compression_info system procedure" [*ASA SQL Reference*, page 779].

Enabling compression

Enabling compression for a connection (or all connections) can significantly improve Adaptive Server Anywhere performance under some circumstances, including:

- ♦ when used over slow networks such as some wireless networks, some modems, serial links and some WANs.
- ♦ when used in conjunction with Adaptive Server Anywhere encryption over a slow network with built-in compression, since packets are compressed before they are encrypted.

Enabling compression, however, can sometimes also cause slower performance. For instance,

- ♦ communication compression uses more memory and more CPU. It may cause slower performance, especially for LANs and other fast networks.
- ♦ most modems and some slow networks already have built-in compression. In these cases, Adaptive Server Anywhere communication

compression will not likely provide additional performance benefits unless you are also encrypting the data.

☞ For more information about compression, see "Compress connection parameter [COMP]" on page 183, "-pc server option" on page 152.

Modifying the CommBufferSize setting

While CommBufferSize (CBSIZE) is not a compression parameter in itself, adjusting this parameter can benefit compression, especially if you are transferring a large amount of data between a client and server. A larger packet size can improve performance for multi-row fetches and fetches of larger rows, as well as inserting or retrieving BLOBs.

As always, however, there are tradeoffs with performance improvements. Since each connection has its own pool of buffers, a large buffer size also increases the memory usage. Be sure to analyze your particular situation to make sure that the benefits of increasing the CommBufferSize outweigh the costs.

☞ For more information see "Tuning TCP/IP performance" on page 87, the "CommBufferSize connection parameter [CBSIZE]" on page 180, or the "-p server option" on page 152.

Modifying the compression threshold

You can also adjust the compression threshold to improve Adaptive Server Anywhere performance. For most networks, the compression threshold does not need to be changed.

When compression is enabled, individual packets may or may not be compressed, depending on their size. For example, Adaptive Server Anywhere does not compress packets smaller than the compression threshold, even if communication compression is enabled. As well, small packets (less than about 100 bytes) usually do not compress at all. Since CPU time is required to compress packets, attempting to compress small packets could actually decrease performance.

Generally speaking, lowering the compression threshold value may improve performance on very slow networks, while raising the compression threshold may improve performance by reducing CPU usage. However, since lowering the compression threshold value will increase CPU usage on both the client and server, a performance analysis should be done to determine whether or not changing the compression threshold is beneficial.

☞ For more information see "CompressionThreshold connection parameter [COMPTH]" on page 184 and "-pt server option" on page 153.

❖ To adjust Adaptive Server Anywhere compression settings

1. Enable communication compression.

   Large data transfers with highly compressible data and larger packet sizes tend to get the best compression rates.

   ☞ For more information about enabling compression, see "Compress connection parameter [COMP]" on page 183 and "-pc server option" on page 152.

2. Adjust the CommBufferSize setting.

   Increasing Adaptive Server Anywhere's packet size can improve compression performance.

   ☞ For more information about adjusting the CommBufferSize setting, see "CommBufferSize connection parameter [CBSIZE]" on page 180 or "-p server option" on page 152.

3. Adjust the CompressionThreshold setting.

   Lowering the compression threshold value may improve performance on very slow networks, while raising the compression threshold may improve performance by reducing CPU usage.

   ☞ For more information about adjusting the CompressionThreshold (COMPTH) connection parameter, see "CompressionThreshold connection parameter [COMPTH]" on page 184 and "-pt server option" on page 153.

Troubleshooting network communications

Network software involves several different components, increasing the likelihood of problems. Although we provide some tips concerning network troubleshooting here, the primary source of assistance in network troubleshooting should be the documentation and technical support for your network communications software, as provided by your network communications software vendor.

Ensure that you are using compatible protocols

Ensure that the client and the database server are using the same protocol. The -x option for the server selects a list of protocols for the server to use, and the **CommLinks (LINKS)** connection parameter does the same for the client application.

You can use these options to ensure that each application is using the same protocol.

By default, the network database server uses all available protocols. The server supports client requests on any active protocol. By default, the client only uses the shared memory protocol. The client can use all available protocols by setting the CommLinks (LINKS) connection parameter to **all**.

☞ For more information about the -x option, see the "-x server option" on page 163.

☞ For information about the CommLinks (LINKS) connection parameter, see "CommLinks connection parameter [LINKS]" on page 181.

Ensure that you have current drivers

Old network adapter drivers can be a source of communication problems. You should ensure that you have the latest version of your network adapter. You should be able to obtain current network adapter drivers from the manufacturer or supplier of the card.

Testing the TCP/IP protocol

The **ping** utility can be useful for testing that TCP/IP is installed and configured properly.

Using ping to test the IP layer

Each IP layer has an associated address—a four-integer, period-separated number (such as 191.72.109.12). Ping takes as an argument an IP-address and attempts to send a single packet to the address.

First, determine if your own machine is configured correctly by pinging yourself. If your IP-address is 191.72.109.12, you would type:

```
ping 191.72.109.12
```

at the command prompt and wait to see if the packets are routed at all. If they are, the output will appear similar to the following:

```
c:> ping 191.72.109.12
Pinging 191.72.109.12 with 32 bytes of data:
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
...
```

If this works, it means that the computer is able to route packets to itself. This is reasonable assurance that the IP layer is set up correctly. You could also ask someone else running TCP/IP for their IP address and try pinging them.

You should ensure that you can ping the computer running the database server from the client computer before proceeding.

Diagnosing wiring problems

Faulty network wiring or connectors can cause problems that are difficult to track down. Try recreating problems on a similar machine with the same configuration. If a problem occurs on only one machine, it may be a wiring problem or a hardware problem.

A checklist of common problems

The following list presents some common problems and their solutions.

☞ For information about troubleshooting connections to the database or database server, see "Troubleshooting connections" on page 66 and "Troubleshooting server startup" on page 34.

If you receive the message `Database server not found` when trying to connect, the client cannot find the database server on the network. Check for the following problems:

♦ Under the TCP/IP protocol, clients search for database servers by broadcasting a request. Such broadcasts will typically not pass through gateways, so any database server on a machine in another (sub)network, will not be found. If this is the case, you must supply the host name of the machine on which the server is running using the HOST (IP) protocol option.

♦ A firewall between the client and server may be preventing the connection.

☞ For more information, see "Connecting across a firewall" on page 87.

- Your network drivers are not installed properly or the network wiring is not installed properly.
- If you receive the message `Unable to initialize requested communication links`, one ore more links failed to start. The probable cause is that your network drivers have not been installed. Check your network documentation to find out how to install the driver you wish to use.
- The server use the TCP/IP protocol if you are connecting via jConnect.
- If you are trying to connect to a database server on your local machine, make sure the Search Network For Database Servers option on the Database tab of the Connect dialog is cleared. You can select this option if you are trying to connect to a database server running on a machine other than your local machine.

☞ For more information about network protocol options, see "Network protocol options" on page 206.

Adjusting timeout values

If you are experiencing problems with connections unexpectedly terminating, consider adjusting either the liveness or the idle timeout values.

☞ For more information about liveness timeout values, see "LivenessTimeout connection parameter [LTO]" on page 199, and the "-tl server option" on page 158.

☞ For more information about connection timeouts, see the "Idle connection parameter [IDLE]" on page 196, and the "-ti server option" on page 157.

CHAPTER 4

Adaptive Server Anywhere as an Open Server

About this chapter

Adaptive Server Anywhere can appear to client applications as an Open Server. This feature enables Sybase Open Client applications to connect natively to Adaptive Server Anywhere databases.

This chapter describes how to use Adaptive Server Anywhere as an Open Server, and how to configure Open Client and Adaptive Server Anywhere to work together.

☞ For information on developing Open Client applications for use with Adaptive Server Anywhere, see "The Open Client Interface" [*ASA Programming Guide*, page 461].

Contents

Open Clients, Open Servers, and TDS

This chapter describes how Adaptive Server Anywhere fits into the Sybase Open Client/Open Server client/server architecture. This section describes the key concepts of this architecture, and provides the conceptual background for the rest of the chapter.

If you simply wish to use a Sybase application with Adaptive Server Anywhere, you do not need to know any details of Open Client, Open Server, or TDS. However, an understanding of how these components fit together may be helpful for configuring your database and setting up applications. This section explains how the components fit together, but avoids any discussion of the internal features of the components.

Open Clients and Open Servers

Adaptive Server Anywhere and other members of the Adaptive Server family act as **Open Servers**. This means you can develop client applications using the **Open Client** libraries available from Sybase. Open Client includes both the Client Library (CT-Library) and the older DB-Library interfaces.

Tabular Data Stream

Open Clients and Open Servers exchange information using an application protocol called the **tabular data stream** (TDS). All applications built using the Sybase Open Client libraries are also TDS applications because the Open Client libraries handle the TDS interface. However, some applications (such as Sybase jConnect) are TDS applications even though they do not use the Sybase Open Client libraries—they communicate directly using the TDS protocol.

While many Open Servers use the Sybase Open Server libraries to handle the interface to TDS, some applications have a direct interface to TDS of their own. Sybase Adaptive Server Enterprise and Adaptive Server Anywhere both have internal TDS interfaces. They appear to client applications as an Open Server, but do not use the Sybase Open Server libraries.

Programming interfaces and application protocols

Adaptive Server Anywhere supports two application protocols. Open Client applications and other Sybase applications such as Replication Server and OmniConnect use TDS. ODBC and embedded SQL applications use a separate application protocol specific to Adaptive Server Anywhere.

TDS uses TCP/IP

Application protocols such as TDS sit on top of lower-level communications protocols that handle network traffic. Adaptive Server Anywhere supports TDS only over the TCP/IP network protocol. In contrast, the Adaptive Server Anywhere-specific application protocol supports several network protocols, as well as a shared memory protocol designed for same-machine communication.

Sybase applications and Adaptive Server Anywhere

The ability of Adaptive Server Anywhere to act as an Open Server enables Sybase applications such as Replication Server and OmniConnect to work with Adaptive Server Anywhere.

Replication Server support

The Open Server interface enables support for Sybase Replication Server: Replication Server connects through the Open Server interface, enabling Adaptive Server Anywhere databases to act as replicate sites in Replication Server installations.

For your database to act as a primary site in a Replication Server installation, you must also use the Replication Agent for Sybase Adaptive Server Anywhere, also called a **Log Transfer Manager**.

☞ For information on the Replication Agent, see "Replicating Data with Replication Server" on page 463.

OmniConnect support

Sybase OmniConnect provides a unified view of disparate data within an organization, allowing users to access multiple data sources without having to know what the data looks like or where to find it. In addition, OmniConnect performs heterogeneous joins of data across the enterprise, enabling cross-platform table joins of targets such as DB2, Sybase Adaptive Server Enterprise, Oracle, and VSAM.

Using the Open Server interface, Adaptive Server Anywhere can act as a data source for OmniConnect.

Setting up Adaptive Server Anywhere as an Open Server

This section describes how to set up an Adaptive Server Anywhere server to receive connections from Open Client applications.

System requirements

There are separate requirements at the client and server for using Adaptive Server Anywhere as an Open Server.

Server-side requirements

You must have the following elements at the server side to use Adaptive Server Anywhere as an Open Server:

- **Adaptive Server Anywhere server components** You must use the network server (*dbsrv9.exe*) if you want to access an Open Server over a network. You can use the personal server (*dbeng9.exe*) as an Open Server only for connections from the same machine.
- **TCP/IP** You must have a TCP/IP protocol stack to use Adaptive Server Anywhere as an Open Server, even if you are not connecting over a network.

Client-side requirements

You need the following elements to use Sybase client applications to connect to an Open Server (including Adaptive Server Anywhere):

- **Open Client components** The Open Client libraries provide the network libraries your application needs to communicate via TDS if your application uses Open Client.
- **jConnect** If your application uses JDBC, you need jConnect and a Java runtime environment.
- **DSEdit** You need *DSEdit*, the directory services editor, to make server names available to your Open Client application. On UNIX platforms, this utility is called *sybinit*.

  DSEdit is not included with SQL Anywhere Studio, but is included with Open Server software.

Starting the database server as an Open Server

If you wish to use Adaptive Server Anywhere as an Open Server, you must ensure that you start it using the TCP/IP protocol. By default, the server starts all available communications protocols, but you can limit the protocols started by listing them explicitly in the command. For example, the following commands are both valid:

```
dbsrv9 -x tcpip,spx asademo.db
dbsrv9 -x tcpip -n myserver asademo.db
```

The first command uses both TCP/IP and SPX protocols, of which TCP/IP is available for use by Open Client applications. The second line uses only TCP/IP.

You can use the personal database server as an Open Server for communications on the same machine because it supports the TCP/IP protocol.

The server can serve other applications through the TCP/IP protocol or other protocols using the Adaptive Server Anywhere-specific application protocol at the same time as serving Open Client applications over TDS.

Port numbers

Every application using TCP/IP on a machine uses a distinct TCP/IP **port** so that network packets end up at the right application. The default port for Adaptive Server Anywhere is port 2638. It is recommended that you use the default port number as Adaptive Server Anywhere has been granted that port number by the Internet Assigned Numbers Authority (IANA). If you wish to use a different port number, you can specify which one using the ServerPort (PORT) protocol option:

```
dbsrv9 -x tcpip(ServerPort=2629) -n myserver asademo.db
```

You may also need to supply an EngineName if more than one local database server is running, or if you wish to connect to a network server.

Open Client settings

To connect to this server, the interfaces file at the client machine must contain an entry specifying the machine name on which the database server is running, and the TCP/IP port it uses.

☞ For details on setting up the client machine, see “Configuring Open Servers” on page 106.

Configuring Open Servers

Adaptive Server Anywhere can communicate with other Adaptive Servers, Open Server applications, and client software on the network. Clients can talk to one or more servers, and servers can communicate with other servers via remote procedure calls. For products to interact with one another, each needs to know where the others reside on the network. This network service information is stored in the interfaces file.

The interfaces file

The **interfaces file** is usually named *sql.ini* on PC operating systems and *interfaces*, or *interfac* on UNIX operating systems.

Like an address book, the interfaces file lists the name and address of every database server known to Open Client applications on your machine. When you use an Open Client program to connect to a database server, the program looks up the server name in the interfaces file and then connects to the server using the address.

The name, location, and contents of the interfaces file differ between operating systems. Also, the format of the addresses in the interfaces file differs between network protocols.

When you install Adaptive Server Anywhere, the setup program creates a simple interfaces file that you can use for local connections to Adaptive Server Anywhere over TCP/IP. It is the System Administrator’s responsibility to modify the interfaces file and distribute it to users so that they can connect to Adaptive Server Anywhere over the network.

Using the DSEdit utility

The *DSEdit* utility is a Windows utility that allows you to configure the interfaces file (*SQL.ini*). The following sections explain how to use the *DSEdit* utility to configure the interfaces file.

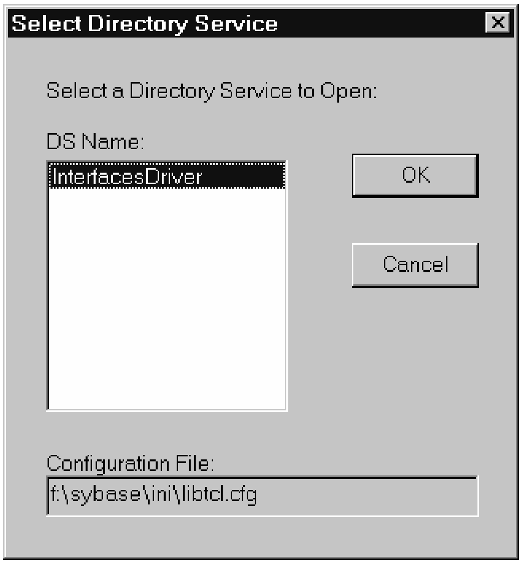
These sections describe how to use *DSEdit* for those tasks required for Adaptive Server Anywhere. It is not complete documentation for the *DSEdit* utility.

☞ For more information on *DSEdit*, see the *Utility Programs* book for your platform, included with other Sybase products.

Starting DSEdit

The *DSEdit* executable is located in the *SYBASE\bin* directory, which is added to your path on installation. You can start *DSEdit* either from a command prompt or from the Explorer in the standard fashion.

When you start *DSEdit*, the Select Directory Service dialog appears.



```
f:\sybase\ini\libtcl.cfg
```

Opening a Directory Services session

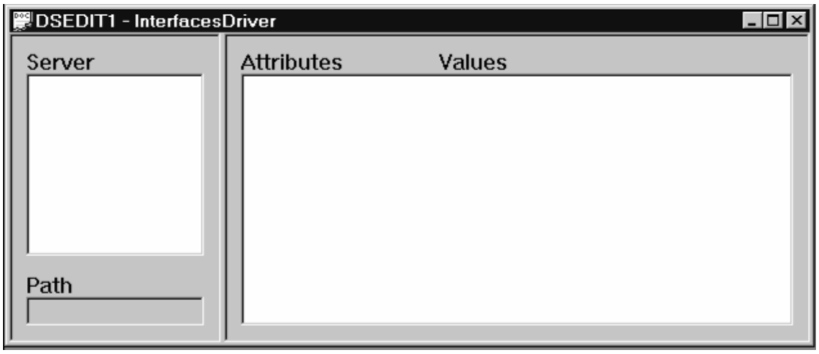
The Select Directory Service window allows you to open a session with a directory service. You can open a session to edit the interfaces file (*SQL.ini*), or any directory service that has a driver listed in the *libtcl.cfg* file.

❖ To open a session

1. Click the local name of the directory service you want to connect to, as listed in the DS Name box, and click OK.

   For Adaptive Server Anywhere, select the InterfacesDriver.

SYBASE environment variable must be set
The *DSEdit* utility uses the SYBASE environment variable to locate the *libtcl.cfg* file. If the SYBASE environment variable is incorrect, *DSEdit* cannot locate the *libtcl.cfg* file.



You can add, modify, or delete entries for servers, including Adaptive Server Anywhere servers, in this window.

Adding a server entry

❖ To add a server entry

1. Choose Add from the Server Object menu.

   The Input Server Name window appears.

2. Type a server name in the Server Name box, and click OK to enter the server name.

   The server name entry must match the server name you used to start Adaptive Server Anywhere. The server *address* is used to identify and locate the server. The server name field is an identifier for Open Client. For Adaptive Server Anywhere, if the server has more than one database loaded, the *DSEdit* server name entry identifies which database to use.

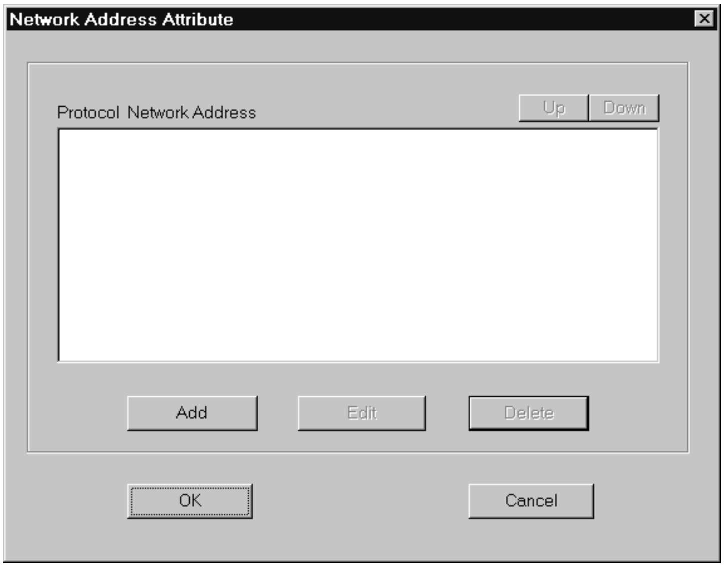
   The server entry appears in the Server box. To specify the attributes of the server, you must modify the entry.

Adding or changing the server address

Once you have entered a Server Name, you need to modify the Server Address to complete the interfaces file entry.

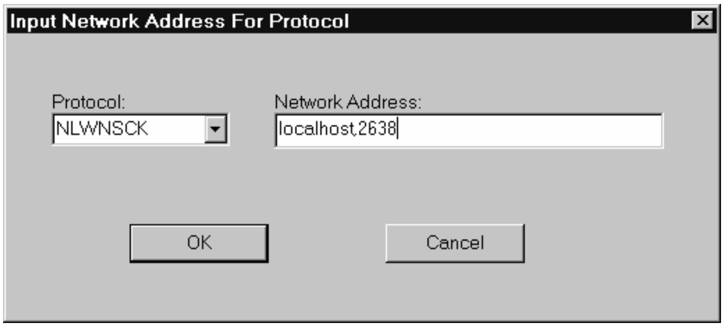
❖ To enter a Server Address

1. Select a server entry in the Server box.
2. Right-click the Server Address in the Attributes box.
3. Choose Modify Attribute from the popup menu. A window appears, showing the current value of the address. If you have no address entered, the box is empty.



4. Click Add.

   The Input Network Address for Protocol window appears.
5. Select NLWNSCK from the Protocol list box (this is the TCP/IP protocol) and enter a value in the Network Address text box.



Protocol: NLWNSCK

Network Address: localhost,2638

For TCP/IP addresses, use one of the following two forms:

- computer name, port number
- IP-address, portnumber

The address or computer name is separated from the port number by a comma.

Machine name

A name (or an IP address) identifies the machine on which the server is running. On Windows operating systems, you can find the machine name in Network Settings, in the Control Panel.

If your client and server are on the same machine, you must still enter the machine name. In this case, you can use **localhost** to identify the current machine.

Port number

The port number you enter must match the port number you used to start the Adaptive Server Anywhere database server with, as described in "Starting the database server as an Open Server" on page 104. The default port number for Adaptive Server Anywhere servers is 2638. This number has been assigned to Adaptive Server Anywhere by the Internet Adapter Number Authority (IANA), and use of this port is recommended unless you have good reasons for explicitly using another port.

The following are valid server address entries:

```
elora,2638
123.85.234.029,2638
```

Verifying the server address

You can verify your network connection using the Ping command from the Server Object menu.

Database connections not verified
Verifying a network connection confirms that a server is receiving requests on the machine name and port number specified. It does not verify anything about database connections.

❖ To ping a server

1. Ensure that the database server is running.
2. Click the server entry in the Server box of the *DSEdit* session window.
3. Choose Ping Server from the Server Object menu.

   The Ping window appears.
4. Select the address you want to ping. Click Ping.

   A message box appears, notifying you whether or not the connection is successful. A message box for a successful connection states that both Open Connection and Close Connection succeeded.

Renaming a server entry

You can rename server entries from the *DSEdit* session window.

❖ To rename a server entry

1. Select a server entry in the Server box.
2. Choose Rename from the Server Object menu.

   The Input Server Name window appears.
3. Type a new name for the server entry in the Server Name box.
4. Click OK to make the change.

Deleting server entries

You can delete server entries from the *DSEdit* session window.

❖ To delete a server entry

1. Click a server entry in the Server box.
2. Choose Delete from the Server Object menu.

Configuring servers for JDBC

The JDBC connection address (URL) contains all the information required to locate the server.

☞ For information on the JDBC URL, see "Supplying a URL for the server" [*ASA Programming Guide,* page 112].

Characteristics of Open Client and jConnect connections

When Adaptive Server Anywhere is serving applications over TDS, it automatically sets relevant database options to values compatible with Adaptive Server Enterprise default behavior. These options are set temporarily, for the duration of the connection only. The client application can override them at any time.

Default settings

The database options set on connection using TDS include:

Option	Set to
ALLOW_NULLS_BY_DEFAULT	OFF
ANSI_BLANKS	ON
ANSINULL	OFF
AUTOMATIC_TIMESTAMP	ON
CHAINED	OFF
CLOSE_ON_ENDTRANS	OFF
DATE_FORMAT	YYYY-MM-DD
DATE_ORDER	MDY
ESCAPE_CHARACTER	OFF
FLOAT_AS_DOUBLE	ON
ISOLATION_LEVEL	1
ON_TSQL_ERROR	CONTINUE
QUOTED_IDENTIFIER	OFF
TIME_FORMAT	HH:NN:SS.SSS
TIMESTAMP_FORMAT	YYYY-MM-DD HH:NN:SS.SSS
TSQL_HEX_CONSTANT	ON
TSQL_VARIABLES	ON

How the startup options are set

The default database options are set for TDS connections using a system procedure named sp_tsql_environment. This procedure sets the following options:

```
SET TEMPORARY OPTION Allow_nulls_by_default='OFF';
SET TEMPORARY OPTION Ansi_blanks='ON';
SET TEMPORARY OPTION Ansinull='OFF';
SET TEMPORARY OPTION Automatic_timestamp='ON';
SET TEMPORARY OPTION Chained='OFF';
SET TEMPORARY OPTION Close_on_endtrans='OFF';
SET TEMPORARY OPTION Date_format='YYYY-MM-DD';
SET TEMPORARY OPTION Date_order='MDY';
SET TEMPORARY OPTION Escape_character='OFF';
SET TEMPORARY OPTION Float_as_double='ON';
SET TEMPORARY OPTION Isolation_level='1';
SET TEMPORARY OPTION On_tsql_error='CONTINUE';
SET TEMPORARY OPTION Quoted_identifier='OFF';
SET TEMPORARY OPTION Time_format='HH:NN:SS.SSS';
SET TEMPORARY OPTION Timestamp_format='YYYY-MM-DD HH:NN:SS.SSS';
SET TEMPORARY OPTION Tsql_hex_constant='ON';
SET TEMPORARY OPTION Tsql_variables='ON';
```

Do not edit the sp_tsql_environment procedure
Do not alter the sp_tsql_environment procedure yourself. It is for system use only.

The procedure sets options only for connections that use the TDS communications protocol. This includes Open Client and JDBC connections using jConnect. Other connections (ODBC and embedded SQL) have the default settings for the database.

You can change the options for TDS connections.

❖ To change the option settings for TDS connections

1. Create a procedure that sets the database options you want. For example, you could use a procedure such as the following:

   ```
   CREATE PROCEDURE my_startup_procedure()
   BEGIN
     IF connection_property('CommProtocol')='TDS' THEN
       SET TEMPORARY OPTION QUOTED_IDENTIFIER='OFF';
     END IF
   END
   ```

   This particular procedure example changes only the QUOTED_IDENTIFIER option from the default setting.

2. Set the LOGIN_PROCEDURE option to the name of a new procedure:

   ```
   SET OPTION LOGIN_PROCEDURE= 'DBA.my_startup_procedure'
   ```

Future connections will use the procedure. You can configure the procedure differently for different user IDs.

☞ For more information about database options, see “Database Options” on page 613.

CHAPTER 5

The Database Server

About this chapter

This chapter describes the command line options for the Adaptive Server Anywhere database server.

Contents

The database server

Function	Start a personal database server or network database server.
Syntax	{ **dbeng9** | **dbsrv9** } [*server-options*] [*database-file* [*database-options*] . . .]
NetWare syntax	**load dbsrv9** [*server-options*] [*database-file* [*database-options*] . . .]

Server options

Server option	Description
@*@data*	Read in options from a configuration file or environment variable. See "@data server option" on page 123.
-?	Display usage information. See "-? server option" on page 125.
-b	Run in bulk operations mode. See "-b server option" on page 125.
-c *size*	Set initial cache size. See "-c server option" on page 126.
-ca 0	Disable dynamic cache sizing [Windows NT/2000/XP, Windows 95/98/Me, UNIX]. See "-ca server option" on page 127.
-cc { + | - }	Collect information about database pages to be used for cache warming. See "-cc server option" on page 128.
-ch *size*	Set the cache size upper limit [Windows NT/2000/XP, Windows 95/98/Me]. See "-ch server option" on page 128.
-cl *size*	Set the cache size lower limit [Windows NT/2000/XP]. See "-cl server option" on page 129.
-cr { + | - }	Warm the cache with database pages. See "-cr server option" on page 130.
-cs	Display cache usage in database server window. See "-cs server option" on page 130.
-ct { + | - }	Turn character-set translation on and off [not NetWare or Windows CE]. See "-ct server option" on page 131.
-cv { + | - }	Control the appearance of messages about cache warming in the database server window. See "-cv server option" on page 131.

Server option	Description
-cw	Enable use of Address Windowing Extensions on Windows 2000, Windows XP, and Windows Server 2003 for setting the size of the database server cache. See "-cw server option" on page 132.
-d	Use POSIX I/O [NetWare]. See "-d server option" on page 135.
-ec *encryption-options*	Enable packet encryption [network server]. See "-ec server option" on page 135.
-ep	Prompt for encryption key. See "-ep server option" on page 138.
-fc	Specify the filename of a DLL containing the file system full callback function. See "-fc server option" on page 139.
-fips	Requires that only FIPS-approved algorithms should be used for strong database and communication encryption. See "-fips server option" on page 140.
-ga	Automatically unload the database after the last connection closed. In addition, shut down after the last database is closed [not NetWare]. See "-ga server option" on page 141.
-gb *level*	Set database process priority class to *level* [Windows NT/2000/XP]. See "-gb server option" on page 142.
-gc *num*	Set maximum checkpoint timeout period to *num* minutes. See "-gc server option" on page 142.
-gd *level*	Set database starting permission. See "-gd server option" on page 142.
-ge *size*	Set the stack size for threads that run external functions. See "-ge server option" on page 144.
-gf	Disable firing of triggers. See "-gf server option" on page 144.
-gk *level*	Set the permission required to stop the server. See "-gk server option" on page 144.
-gl *level*	Set the permission required to load or unload data. See "-gl server option" on page 144.

Server option	Description
-gm *num*	Set the maximum number of connections. See "-gm server option" on page 145.
-gn *num*	Set the maximum number of concurrent requests the database server can handle at one time. See "-gn server option" on page 145.
-gp *size*	Set the maximum page size to *size* bytes. See "-gp server option" on page 146.
-gr *minutes*	Set the maximum recovery time to *num* minutes. See "-gr server option" on page 147.
-gss *size*	Set the thread stack size to *size* bytes [not applicable to Windows]. See "-gss server option" on page 147.
-gt *num*	Set the number of operating processors used by the database server to *num* processors. See "-gt server option" on page 147.
-gu *level*	Set the permission level for utility commands: **utility_-db**, **all**, **none**, or **DBA**. See "-gu server option" on page 148.
-gx	Set the number of operating system threads assigned to the database server process [Windows NT/2000/XP, Windows 95/98/Me]. See "-gx server option" on page 148.
-m	Truncate the transaction log after each checkpoint for all databases. See "-m server option" on page 149.
-n *name*	Use *name* as the name of the database server. Note that the -n option is positional. See "-n server option" on page 150.
-o *filename*	Output messages to the specified file. See "-o server option" on page 151.
-oe *filename*	Specify file to log startup errors, fatal errors and assertions to. See "-oe server option" on page 151.
-os *size*	Limit the size of the log file for messages. See "-os server option" on page 151.
-p *packet-size*	Set the maximum network packet size [network server]. See "-p server option" on page 152.

Server option	Description
-pc	Compress all connections except same-machine connections. See "-pc server option" on page 152.
-pt *size_in_bytes*	Set the minimum network packet size to compress. See "-pt server option" on page 153.
-qi	Do not display database server tray icon or screen. See "-qi server option" on page 153.
-qp	Suppress messages about performance in the database server window. See "-qp server option" on page 153.
-qs	Suppress startup error dialogs. See "-qs server option" on page 154.
-qw	Do not display the database server screen. See "-qw server option" on page 154.
-r	Opens database in read-only mode. See "-r server option" on page 154.
-s	Set the syslog facility ID [UNIX]. See "-s server option" on page 155.
-sb { **0** | **1** }	Specify how the server reacts to broadcasts. See "-sb server option" on page 156.
-sc	Disable the shared memory port, and enable Named Pipes. [Windows NT/2000/XP]. See "-sc server option" on page 157.
-ti *minutes*	Client idle time before shutdown—default 240 minutes. See "-ti server option" on page 157.
-tl *seconds*	Default liveness timeout for clients in seconds—default 120 seconds. See "-tl server option" on page 158.
-tmf	Force transaction manager recovery for distributed transactions [Windows NT/2000/XP]. See "-tmf server option" on page 158.
-tmt *milliseconds*	Set the reenlistment timeout for distributed transactions [Windows NT/2000/XP]. See "-tmt server option" on page 159.
-tq *time*	Set quitting time [network server]. See "-tq server option" on page 159.

Server option	Description
-u	Use buffered disk I/O. See "-u server option" on page 159.
-ua	Turn off use of asynchronous I/O [Linux]. See "-ua server option" on page 160.
-uc	
-ud	Run as a daemon [UNIX]. See "-ud server option" on page 161.
-ui	
-ut *minutes*	Touch temporary files every *min* minutes [UNIX]. See "-ut server option" on page 162.
-ux	Display the Server Messages window and Server Startup Options dialog [Linux and Solaris only]. See "-ux server option" on page 162.
-v	Display database server version and stop. See "-v server option" on page 163.
-x *list*	Comma-separated list of communication links to try. See "-x server option" on page 163.
-xs	Specify server side web services communications protocols. See "-xs server option" on page 165.
-y	Run as a Windows 95/98/Me service [Windows 95/98/Me]. See "-y server option" on page 166.
-z	Provide diagnostic information on communication links [network server]. See "-z server option" on page 166.
-zl	Turn on capturing of the most recently-prepared SQL statement for each connection. See "-zl server option" on page 166.
-zn	Specifies the number of request log file copies to retain. See "-zn server option" on page 167.
-zo *filename*	Redirect request logging information to a separate file. See "-zo server option" on page 168.
-zr { **all** | **SQL** | **none** }	Turn on logging of SQL operations. The default is NONE. See "-zr server option" on page 168.
-zs *size*	Limit the size of the log file used for request logging. See "-zs server option" on page 169.

Recovery options

Recovery Option	Description
-a *filename*	Apply the named transaction log file. See "-a recovery option" on page 170.
-f	Force the database to start without a transaction log. See "-f recovery option" on page 170.

Database options

Database Option	Description
-ek *key*	Specify encryption key. See "-ek database option" on page 171.
-m	Truncate (delete) the transaction log after each checkpoint for the specified database. See "-m database option" on page 171.
-n *name*	Name the database. Note that the -n option is positional. See "-n database option" on page 172.
-r	Open the specified database(s) in read-only mode. Database modifications not allowed. See "-r database option" on page 173.

Description

The **dbeng9** command starts a personal database server. The **dbsrv9** command starts a network database server.

Cache size

The amount of cache memory available to the database server can be a key factor in affecting performance. The database server takes an initial amount of cache memory that is either specified by the -c option or is a default value.

☞ For information on the default cache size, see "-c server option" on page 126.

On Windows NT/2000/XP, Windows 95/98/Me, and UNIX, the database server automatically takes more memory for use in the cache as needed, as determined by a heuristic algorithm.

☞ For more information, see "Using the cache to improve performance" [*ASA SQL User's Guide*, page 180].

You can use database options to configure the upper cache limit: see "-ch server option" on page 128. As well, you can force the cache to remain at its initial amount: see "-ca server option" on page 127.

Server differences

The personal database server has a maximum of ten concurrent connections, uses at most one CPU for request processing, and does not support network client/server connections.

In addition, there are other minor differences, such as the default permission level that is required to start new databases, or the permissions required to execute the CHECKPOINT statement.

Platform availability

Both personal and network database servers are supplied for each supported operating system, with the following exceptions:

- **Novell NetWare** Only the network server is supplied.
- **Windows CE** Only the network server is supplied. The support for TCP/IP in the network server enables you to carry out tasks from your desktop machine, including database management, with Sybase Central.

> **Note**
> It is recommended that you run the network database server on Windows NT/2000/XP or Windows 2003, rather than Windows 95/98/Me, to ensure better performance and reliability.

NetWare notes

In NetWare, the database file and the transaction log file must be on a NetWare volume, and the paths must be fully specified. NetWare allows you to have volumes that span two or more hard disks.

Database file The *database-file* specifies the database filename. If *database-file* is specified without a file extension, Adaptive Server Anywhere looks first for *database-file* with extension *.wrt* (a write file), followed by *database-file* with extension *.db*.

If you use a relative path, it is read relative to the current working directory. You can supply a full path. Also, you can supply a path that conforms to the Universal Naming Convention (UNC) format:

```
\\server\volume\path\file.ext
```

Suppressing Windows event log messages

If you run the database server as a Windows service, you can suppress Windows event log entries by setting a registry entry. The registry entry is

```
Software\Sybase\Adaptive Server Anywhere\9.0
```

To control event log entries, set the EventLogMask key, which is of type REG_DWORD. The value is a bit mask containing the internal bit values for the different types of event messages:

```
errors       EVENTLOG_ERROR_TYPE          0x0001
warnings     EVENTLOG_WARNING_TYPE        0x0002
information  EVENTLOG_INFORMATION_TYPE    0x0004
```

For example, if the EventLogMask key is set to zero, no messages appear at all. A better setting would be 1, so that informational and warning messages do not appear, but errors do. The default setting (no entry present) is for all message types to appear.

See also

- ♦ "Running the Database Server" on page 3
- ♦ "Network protocol options" on page 206

Operating quietly

The database server supports quiet mode. You determine how quiet you want the server to operate, ranging from suppressing messages or the icon in the system tray, to complete silence. To operate a completely silent database server on Windows, specify the -qi and -qs options. With these options set, there is no visual indication that the server is running as all icons and all possible startup error messages are suppressed. If you run the database server in quiet mode, you can use either (or both) the -o or -oe options to diagnose errors.

Note that the -qi and -qs options do not suppress error dialogs caused by the -v (version) and -ep (prompt for database encryption password) server options.

Database server options

These options apply to the server as a whole, not just to an individual database.

@data server option

Function

Read in options from the specified environment variable or configuration file

Syntax

{ **dbsrv9** | **dbeng9** } *@data* . . .

Applies to

All operating systems and servers, as well as all database utilities except Interactive SQL (dbisql), the Language utility (dblang), the Adaptive Server Anywhere Console utility (dbconsole), the Rebuild utility (rebuild), the Certificate Generation utility (gencert), the Certificate Reader utility (readcert), the ActiveSync provider install utility (dbasinst), and the File Hiding utility (dbfhide).

Description

Use this option to read in command-line options from the specified environment variable or configuration file. If both exist with the same name that is specified, the environment variable is used.

Configuration files can contain line breaks, and can contain any set of options.

☞ For more information about using configuration files, see "Using configuration files" on page 495.

If you want to protect the information in a configuration file (for example, because it contains passwords) you can use the File Hiding (dbfhide) utility to obfuscate the contents of configuration files.

☞ For more information about the File Hiding utility, see "The File Hiding utility" on page 524.

The @data parameter can occur at any point in the command line, and parameters contained in the file are inserted at that point. Multiple files can be specified, and the file specifier can be used with command line options.

All @data parameters will be left untouched if they occur after the -w option. This can be useful, for example, if you want to use the Service Creation utility [dbsvc] to create a service where the database server reads the parameter file itself.

Examples

The following reads all parameters from *dbinit.in*.

```
dbinit @dbinit.in
```

The following example is the same as `dbinit -q -ja -p 2048 -b new.db`

```
echo "-ja -p 2048" > dbinit.in
dbinit -q @dbinit.in -b new.db
```

In the following example, dbsvc_parms will be read by dbsvc, but dbsrv_parms will not. The service will be created with the command line `dbsrv9.exe @dbsrv_parms`.

```
dbsvc @dbsvc_parms -w mysvc dbsrv9.exe @dbsrv_parms
```

The following configuration file holds a set of options for a server named **myserver** that starts with a cache size of 4 Mb and loads the sample database:

```
-c 4096
-n myserver
"c:\Program Files\Sybase\SQL Anywhere 9\asademo.db"
```

If this configuration file is saved as *c:\config.txt*, it can be used in a command as follows:

```
dbsrv9 @c:\config.txt
```

The following configuration file contains comments:

```
#This is the server name:
-n MyServer
#These are the protocols:
-x tcpip
#This is the database file
my.db
```

The following statement, entered all on one line, sets an environment variable that holds options for a database server that starts with a cache size of 4 Mb and loads the sample database.

```
set envvar=-c 4096 "c:\Program Files\Sybase\SQL Anywhere 9\
         asademo.db"
```

This statement starts the database server using an environment variable named **envvar**.

```
dbsrv9 @envvar
```

-? server option

Function

Display usage information.

Syntax

{ **dbsrv9** | **dbeng9** } **-?**

Applies to

All operating systems and servers.

Description

Displays a short description of each server option. The database does not carry out any other task.

-b server option

Function

Use bulk operation mode.

Syntax

{ **dbsrv9** | **dbeng9** } **-b** . . .

Applies to

All operating systems and servers.

Description

This is useful for using the Interactive SQL INPUT command to load large quantities of data into a database.

The -b option should not be used if you are using LOAD TABLE to bulk load data.

When you use this option, the database server allows only one connection by one application. It keeps a rollback log, but it does not keep a transaction log. The multi-user locking mechanism is turned off.

When you first start the database server after loading data with the -b option, you should use a new log file.

Bulk operation mode does not disable the firing of triggers.

-c server option

Function

Set the initial memory reserved for caching database pages and other server information.

Syntax

{ **dbsrv9** | **dbeng9** } **-c** { *integer* | *integer* **G** | *integer* **K** | *integer* **M** | *integer* **P** } . . .

Applies to

All operating systems and servers.

Description

The amount of memory available for use as a database server cache is one of the key factors controlling performance. You can set the initial amount of cache memory using the -c server option

The more cache memory that can be given the server, the better its performance.

The units **G**, **K**, and **M** can be either lower case or upper case. If **G**, **K**, or **M** is not supplied, any integer less than 10 000 is assumed to be in kilobytes, and any integer 10 000 or greater is assumed to be in bytes. For example, **-c 4096** means 4096 kb or 4 194 304 bytes, whereas **-c 200 000** means (an unreasonably small) cache of 200 000 bytes.

The unit **P** is a percentage of the physical system memory, and if you use this, the argument is a percentage. You can use % as an alternative to P, but as most non-UNIX operating systems use % as an environment variable escape character, you must escape the % character. To use 50 percent of the physical system memory, you would use the following:

```
dbeng9 -c 50%% ...
```

On NetWare and UNIX operating systems, if the cache size specified with -c is greater than the amount of available memory, the database server uses a maximum cache size that is calculated as follows:

```
95% of (available memory - 5 MB)
```

On Windows CE, if the cache size specified with -c is greater than the amount of available memory, the database server uses a maximum cache size that is calculated as follows:

```
95% of (available memory - 2 MB)
```

If no -c option is provided, the database server computes the initial cache allocation as follows:

1. It uses the following operating-system-specific default cache sizes:
   - **Windows CE** 600K

- **Windows NT/2000/XP, Windows 95/98/Me, NetWare** 2 Mb
- **UNIX** 8 Mb

2. It computes a runtime-specific minimum default cache size, which is the lesser of the following items:
   - 25% of the machine's physical memory
   - The sum of the sizes of the main database files specified on the command line. Additional dbspaces apart from the main database files are not included in the calculation. If no files are specified, this value is zero.
3. It allocates the greater of the two values computed.

NetWare database server

There is a tradeoff between memory for the database server and memory for the NetWare file system buffers. A larger database server cache will improve database server performance at the expense of NetWare file system performance. If the database server cache is too big, NetWare will report an error that there is insufficient memory for cache buffers.

NetWare memory requirements increase with every new directory and file on the file server. To track memory usage on the NetWare server, load *monitor.nlm* (if it is not already loaded) and select "Resource Utilization". Extra memory for your NetWare server computer could improve database performance and/or file server performance dramatically.

Example

The following example, entered all on one line, starts a server named **myserver** that starts with a cache size of 3 Mb and loads the sample database:

```
dbeng9 -c 3M -n myserver "C:\Program Files\Sybase\SQL Anywhere
         9\asademo.db"
```

See also

- "-ch server option" on page 128

-ca server option

Function

Enforce a static cache size. The zero argument is required.

Syntax

{ **dbsrv9** | **dbeng9** } **-ca 0** . . .

Applies to

Windows NT/2000/XP, Windows 95/98/Me, UNIX

Description

You can disable automatic cache increase because of high server load by specifying **-ca 0** on the command line. If you do not set **-ca 0**, the database server automatically takes additional cache as needed. The cache size still increases if the database server would otherwise run into the error `Fatal`

`Error:  dynamic memory exhausted`, or if the Java VM requires memory that would lead to a fatal error.

This server option should be used only in the form -ca 0. If **0** is omitted, automatic cache growth is enabled.

Example

The following example starts a server named **myserver** that has a static cache that is 40% of the available physical memory and loads the sample database:

```
dbsrv9 -c 40P -ca 0 -n myserver "C:\Program Files\Sybase\SQL
         Anywhere 9\asademo.db"
```

See also

- ♦ "-c server option" on page 126
- ♦ "-ch server option" on page 128

-cc server option

Function

Collect information about database pages to be used for cache warming the next time the database is started.

Syntax

{ **dbsrv9** | **dbeng9** } **-cc** { **+** | **-** } ...

Applies to

All operating systems and servers.

Description

By default, page collection is turned on. When collection is turned on, the database server keeps track of each database page that is requested. Collection stops when the maximum number of pages has been collected, the database is shut down, or the collection rate falls below the minimum value. Note that you cannot configure the maximum number of pages collected or specify the value for the collection rate (the value is based on cache size and database size). Once collection stops, information about the requested pages is recorded in the database so those pages can be used to warm the cache the next time the database is started with the -cr option. Collection of referenced pages is turned on by default.

See also

- ♦ "-cr server option" on page 130
- ♦ "-cv server option" on page 131
- ♦ "Using cache warming" [*ASA SQL User's Guide*, page 184]

-ch server option

Function

Sets a maximum cache size, as a limit to automatic cache growth.

Syntax

{ **dbsrv9** | **dbeng9** } **-ch** { *integer* | *integer* **G** | *integer* **K** | *integer* **M** | *integer* **P** } . . .

Applies to

Windows NT/2000/XP, Windows 95/98/Me, UNIX

Description

This option limits the cache that the database server can take during automatic cache growth. By default the upper limit is approximately the lower of 256 Mb and 90% of the physical memory of the machine.

The units **G**, **K**, and **M** can be either lower case or upper case. If **G**, **K**, or **M** is not supplied, any integer less than 10 000 is assumed to be in kilobytes, and any integer 10 000 or greater is assumed to be in bytes. For example, **-ch 4096** means 4096 kb or 4 194 304 bytes, whereas **-ch 200 000** means (an unreasonably small) maximum cache of 200 000 bytes.

The unit **P** is a percentage of the physical system memory, and if you use this, the argument is a percentage. You can use % as an alternative to P, but as most non-UNIX operating systems use % as an environment variable escape character, you must escape the % character. To use limit the cache to 50 percent of the physical system memory, you would use the following:

```
dbeng9 -ch 50%% ...
```

Example

The following example starts a server named silver that has a maximum cache size of 2 Mb and loads the sample database:

```
dbeng9 -ch 2M -n silver "C:\Program Files\Sybase\SQL Anywhere 9\
        asademo.db"
```

See also

- “-c server option” on page 126
- “-ca server option” on page 127
- “-cl server option” on page 129

-cl server option

Function

Set a minimum cache size as a lower limit to automatic cache resizing.

Syntax

{ **dbsrv9** | **dbeng9** } **-cl** { *integer* | *integer***G** | *integer***K** | *integer***M** | *integer***P** } . . .

Applies to

Windows NT/2000/XP, Windows 95/98/Me, UNIX

Description

This option sets a lower limit to the cache. The default minimum cache size is the initial cache size.

The units **G**, **K**, and **M** can be either lower case or upper case. If **G**, **K**, or **M** is not supplied, any integer less than 10 000 is assumed to be in kilobytes, and any integer 10 000 or greater is assumed to be in bytes. For example, **-cl 4096** means 4096 kb or 4 194 304 bytes, whereas **-cl 200 000** means a minimum cache of 200 000 bytes.

The unit **P** is a percentage of the physical system memory, and if you use this, the argument is a percentage. You can use % as an alternative to P, but as most non-UNIX operating systems use % as an environment variable escape character, you must escape the % character. To set the minimum cache size to 50 percent of the physical system memory, you would use the following:

```
dbeng9 -cl 50%% ...
```

Example

The following example starts a server named silver that has a minimum cache size of 5 Mb and loads the sample database:

```
dbeng9 -cl 5M -n silver "C:\Program Files\Sybase\SQL Anywhere 9\
         asademo.db"
```

See also

- ♦ "-c server option" on page 126
- ♦ "-ca server option" on page 127
- ♦ "-ch server option" on page 128

-cr server option

Function

Reload (warm) the cache with database pages using information collected the last time the database was run.

Syntax

{ **dbsrv9** | **dbeng9** } **-cr** { **+** | **-** } ...

Applies to

All operating systems and servers.

Description

You can instruct the database server to warm the cache using pages that were referenced the last time the database was started (page collection is turned on using the -cc option). Cache warming is turned on by default. When a database is started, the server checks the database to see if it contains a collection of pages requested the last time the database was started. If the database contains this information, the previously-referenced pages are then loaded into the cache.

Warming the cache with pages that were referenced the last time the database was started can improve performance when the same query or similar queries are executed against a database each time it is started.

See also

- ♦ "-cc server option" on page 128
- ♦ "-cv server option" on page 131
- ♦ "Using cache warming" [*ASA SQL User's Guide*, page 184]

-cs server option

Function

Display cache size changes in the database server window.

Syntax

{ **dbsrv9** | **dbeng9** } **-cs** . . .

Applies to

All platforms with dynamic cache sizing.

Description	For troubleshooting purposes, display cache information in the database server window whenever the cache size changes.

-ct server option

Function	Turn character set translation on and off.
Syntax	{ **dbsrv9** | **dbeng9** } **-ct** { **+** | **-** } . . .
Applies to	All operating systems except Windows CE.
Description	By default, character set translation is turned on. Character set translation converts strings between character sets that represent the same characters, but at different values. This is useful when the client machine and the database use different character sets. Character set translation is disabled with the -ct- server option. If this argument is supplied as -ct+, character set translation is turned on.

In version 7.x and earlier of Adaptive Server Anywhere, the **+** or **-** values are not accepted: specifying the -ct option enabled character set translation.

Example	The following example starts a server with character set translation disabled and loads the sample database:

```
dbeng9 -ct- "C:\Program Files\Sybase\SQL Anywhere 9\asademo.db"
```

See also

- "Turning off character set translation on a database server" on page 356
- "Starting a database server using character set translation" on page 356

-cv server option

Function	Control the appearance of messages about cache warming in the database server window.
Syntax	{ **dbsrv9** | **dbeng9** } **-cv** { **+** | **-** } ...
Applies to	All operating systems and servers.
Description	When -cv+ is specified, a message appears in the database server window when any of the following cache warming activities occur:

- collection of requested pages starts or stops (controlled by the -cc server option)
- page reloading starts or stops (controlled by the -cr server option)

By default, this option is turned off.

Example	The following command starts a database called *mydatabase.db* with database page collection and page loading turned on, and logs messages

about these activities to the database server window:

```
dbsrv9 -cc+ -cr+ -cv+ mydatabase.db
```

See also

- “-cc server option” on page 128
- “-cr server option” on page 130
- “Using cache warming” [*ASA SQL User’s Guide*, page 184]

-cw server option

Function

Enable use of Address Windowing Extensions (AWE) on Windows 2000, Windows XP, and Windows Server 2003 for setting the size of the database server cache.

Syntax

{ **dbsrv9** | **dbeng9** } **-cw** . . .

Applies to

Windows 2000, Windows XP, Windows Server 2003, and later.

Description

The amount of memory available for use as a database server cache is one of the key factors controlling performance. Because Windows 2000, Windows XP, and Windows Server 2003 support Address Windowing Extensions, you can use the -cw option to take advantage of large cache sizes based on the maximum amount of physical memory in the system.

Operating system	**Maximum non-AWE cache size**	**Maximum amount of physical memory supported by Windows**
Windows 2000 Professional	1.8 Gb	4 Gb
Windows 2000 Server	1.8 Gb	4 Gb
Windows 2000 Advanced Server	2.7 Gb*	8 Gb
Windows 2000 Datacenter Server	2.7 Gb*	64 Gb
Windows XP Home Edition	1.8 Gb	2 Gb
Windows XP Professional	1.8 Gb	4 Gb
Windows Server 2003, Web Edition	1.8 Gb	2 Gb
Windows Server 2003, Standard Edition	1.8 Gb	4 Gb

Operating system	**Maximum non-AWE cache size**	**Maximum amount of physical memory supported by Windows**
Windows Server 2003, Enterprise Edition	2.7 Gb*	32 Gb
Windows Server 2003, Datacenter Edition	2.7 Gb*	64 Gb

*You must boot the operating system using the `/3GB` option to use a cache of this size.

When using an AWE cache, almost all of the available physical memory in the system can be allocated for the cache.

If you can set a cache of the desired size using a non-AWE cache, this is recommended because AWE caches allocate memory that can only be used by the database server. This means that while the database server is running, the operating system and other applications cannot use the memory allocated for the database server cache. AWE caches do not support dynamic cache sizing. Therefore, if an AWE cache is used and you specify the -ch or -cl options to set the upper and lower cache size, they are ignored.

☞ For information about specifying the cache size, see "-c server option" on page 126.

To start a database server with an AWE cache, you must do the following:

- ♦ Have at least 130 Mb of memory available on your system.
- ♦ If your system has between 2 Gb and 16 Gb of memory, add the `/3GB` option to the Windows boot line in the "[operating systems]" section of the *boot.ini* file.

  If your system has more than 16 Gb of memory, do not add the `/3GB` option to the Windows boot line in the "[operating systems]" section of the *boot.ini* file because Windows will not be able to address memory beyond 16 Gb.
- ♦ If your system has more than 4 Gb of memory, add the `/PAE` option to the Windows boot line in the "[operating systems]" section of the *boot.ini* file.
- ♦ Grant the "Lock pages in memory" privilege to the user ID under which the server is run. The following steps explain how to do this on Windows 2000.
  1. Log on to Windows as Administrator.

2. From the Start menu, choose Settings ➤ Control Panel.
3. Open the Administrative Tools folder.
4. Double-click Local Security Policy.
5. Open Local Policies in the left pane.
6. Double-click User Rights Assignment in the left pane.
7. Double-click the Lock Pages In Memory policy in the right pane.
   The Local Security Policy Setting dialog appears.
8. In the Local Security Policy Setting dialog, click Add.
   The Select Users or Groups dialog appears.
9. Select the user ID from the list and click Add.
10. In the Local Security Policy Setting dialog, click OK.
11. Restart the computer for the setting to take effect.

If you specify the -cw option and the -c option on the command line, the database server attempts the initial cache allocation as follows:

1. The AWE cache is no larger than the cache size specified by the -c option. If the value specified by the -c option is less than 2 Mb, AWE is not used.

2. The AWE cache is no larger than all available physical memory less 128 Mb.

3. The AWE cache is no smaller than 2 Mb. If this minimum amount of physical memory is not available, an AWE cache is not used.

When you specify the -cw option and do not specify the -c option, the database server attempts the initial cache allocation as follows:

1. The AWE cache uses 100% of all available memory except for 128 Mb that is left free for the operating system.

2. The AWE cache is no larger than the sum of the sizes of the main database files specified on the command line. Additional dbspaces apart from the main database files are not included in the calculation. If no files are specified, this value is zero.

3. The AWE cache is no smaller than 2 Mb. If this minimum amount of physical memory is not available, an AWE cache is not used.

When the server uses an AWE cache, the cache page size is at least 4 kb and dynamic cache sizing is disabled. On 64-bit Windows platforms, the cache page size is at least 8 kb.

☞ For more information about dynamic cache sizing, see "Using the cache to improve performance" [*ASA SQL User's Guide*, page 180].

Example

The following example starts a server named **myserver** that starts with a cache size of 12 Gb and loads the sample database:

```
dbeng9 -c 12G -cw asademo.db
```

See also

- ♦ "-c server option" on page 126

-d server option

Function

Use POSIX I/O.

Syntax

{ **dbsrv9** | **dbeng9** } **-d** . . .

Applies to

NetWare. The use of this option on Windows platforms has been deprecated.

Description

The -d option forces the use of POSIX I/O rather than DFS (Direct File System) I/O. This option is provided as a workaround for bugs in old versions of DFS. Note that asynchronous I/O is not available when using POSIX I/O.

This option applies to NetWare systems only.

-ec server option

Function

Use transport-layer security or simple encryption to encrypt all native Adaptive Server Anywhere packets (DBLib, ODBC, and OLE DB) transmitted to and from all clients. TDS packets are not encrypted.

Syntax

{ **dbsrv9** | **dbeng9** } **-ec** *encryption-options* . . .

encryption-options:

{ **NONE**
|**SIMPLE**
|**ECC_TLS (CERTIFICATE=***filename*;
  CERTIFICATE_PASSWORD=*password* **)**
|**RSA_TLS (CERTIFICATE=***filename*;
  CERTIFICATE_PASSWORD=*password* **)**
| **RSA_TLS_FIPS (CERTIFICATE=***filename*;
  CERTIFICATE_PASSWORD=*password* **)** } , . . .

Description

You can use this option to secure communication packets between client applications and the database server using transport-layer security.

☞ For more information, see "Adaptive Server Anywhere Transport-Layer Security" [*SQL Anywhere Studio Security Guide*, page 27].

Separately licensable option required
Transport-layer security requires that you obtain the separately-licensable SQL Anywhere Studio security option and is subject to export regulations.

☞ To order this component, see "Separately-licensable components" [*Introducing SQL Anywhere Studio*, page 5].

The -ec option instructs the database server to accept *only* connections from ODBC, OLE DB, or embedded SQL interfaces that are encrypted using one of the specified types. Connections over the TDS protocol, which include Java applications using jConnect, are always accepted regardless of the usage of the -ec option, and are never encrypted. Setting the TDS connection parameter to NO disallows these unencrypted TDS connections.

☞ See "TDS protocol option" on page 222 for more information.

By default, communication packets are not encrypted, which poses a potential security risk. If you are concerned about the security of network packets, use the -ec option. Encryption affects performance only marginally. The -ec option controls the server's encryption settings and requires one of the following parameters in a comma-separated list:

none accepts connections that are not encrypted.

simple accepts connections that are encrypted with simple encryption. This type of encryption is supported on all platforms, as well as on previous versions of Adaptive Server Anywhere. Simple encryption does not provide server authentication, strong elliptic-curve or RSA encryption, or other features of transport-layer security.

ECC_TLS accepts connections that are encrypted using the elliptic curve-based Certicom encryption technology. For backwards compatibility, ecc_tls can also be specified as **CERTICOM**. To use this type of encryption, both the server and the client must be operating on Solaris, Linux, NetWare, Mac OS X, or any supported Windows platform except Windows CE, and the connection must be over the TCP/IP port. UNIX platforms, except Solaris, Linux, and Mac OS X, do not recognize the client or server ECC_TLS parameter. This parameter accepts the following required arguments:

- **certificate** the file name of the server certificate.

  This is the server certificate containing the server's private key. This server certificate can be self-signed or signed by an enterprise root certificate or Certificate Authority.

- **certificate_password** the password for the server certificate's private key.

In order to use ECC_TLS, you must generate your certificates using the ECC cipher.

☞ For more information about creating certificates, see "Creating digital certificates" [*SQL Anywhere Studio Security Guide,* page 31].

RSA_TLS accepts connections that are encrypted using RSA-based encryption technology. To use this type of encryption, both the server and the client must be operating on Solaris, Linux, NetWare, Mac OS X, or any supported Windows platform except Windows CE, and the connection must be over the TCP/IP port. UNIX platforms, except for Solaris, Linux, and Mac OS X, do not recognize the client or server RSA_TLS parameter. This parameter accepts the following required arguments:

- **certificate** the file name of the server certificate.

  This is the server certificate containing the server's private key. This server certificate can be self-signed or signed by an enterprise root certificate or Certificate Authority.

- **certificate_password** the password for the server certificate's private key.

You must generate your certificates using the RSA cipher.

☞ For more information about creating certificates, see "Creating digital certificates" [*SQL Anywhere Studio Security Guide,* page 31].

RSA_TLS_FIPS accepts connections that are encrypted using FIPS-approved RSA encryption technology. RSA_TLS_FIPS uses a separate approved library, but is compatible with clients specifying RSA_TLS with Adaptive Server Anywhere 9.0.2 or later. To use this type of encryption, both the server and the client must be operating on a supported 32-bit Windows operating system, and the connection must be over the TCP/IP port. This parameter accepts the following required arguments:

- **certificate** the file name of the server certificate.

  This is the server certificate containing the server's private key. This server certificate can be self-signed or signed by an enterprise root certificate or Certificate Authority.

- **certificate_password** the password for the server certificate's private key.

If you are using FIPS-approved RSA encryption, you must generate your certificates using the RSA cipher.

☞ For more information about creating certificates, see "Creating digital certificates" [*SQL Anywhere Studio Security Guide,* page 31].

The *dbecc9.dll* and *dbrsa9.dll* files contain the ECC and RSA code used for encryption and decryption. *dbrsa9f.dll* contains the code for the FIPS-approved RSA algorithm. When you connect to the server, if the appropriate file cannot be found, or if an error occurs, a message appears on the server window. The server does not start if the types of encryption specified cannot be initiated.

The client's and the server's encryption settings must match or the connection will fail. However, if **-ec simple** is specified, on the server, but **-ec none** is not, then connections that do not request encryption can connect and automatically user simple encryption.

Examples

The following example specifies the elliptic-curve server certificate *sample.crt* at the dbsrv9 command line.

```
dbsrv9 -ec ecc_tls(certificate=sample.crt; certificate_
         password=tJ1#m6+W) -x tcpip asademo.db
```

The following example specifies the RSA server certificate *rsaserver.crt* at the dbsrv9 command line.

```
dbsrv9 -ec rsa_tls(certificate=rsaserver.crt; certificate_
         password=test) -x tcpip asademo.db
```

The following example specifies the RSA server certificate *rsaserver.crt* at the dbsrv9 command line. The rsa_tls_fips parameter is used for the FIPS-approved RSA algorithm.

```
dbsrv9 -ec rsa_tls_fips(certificate=rsaserver.crt; certificate_
         password=test) -x tcpip asademo.db
```

See also

- ♦ "Starting the database server with transport-layer security" [*SQL Anywhere Studio Security Guide*, page 39]
- ♦ "Encryption connection parameter [ENC]" on page 191

-ep server option

Function

Prompt the user for the encryption key upon starting a strongly encrypted database.

Syntax

{ **dbsrv9** | **dbeng9** } **-ep** . . .

Description

The -ep option instructs the database server to display a dialog box for the user to enter the encryption key. This server option provides an extra measure of security by never allowing the encryption key to be seen in clear text.

When used with supported tools, this option always prompts the user for the encryption key, even if a key is not necessary. If the dialog box prompt appears and you know that a key is not necessary, you can click Cancel to continue.

When used with the server, this option prompts the user for the encryption key only if *all* of the following are true:

- the -ep option is specified
- the server is a Windows personal server, or the server is just starting up
- a key is required to start a database
- the server is either not a Windows service, or it is a Windows service with the interact with desktop option turned ON
- the server is not a daemon (UNIX)

If you want to secure communication packets between client applications and the database server use the -ec server option and transport-layer security.

☞ For more information, see "Adaptive Server Anywhere Transport-Layer Security" [*SQL Anywhere Studio Security Guide*, page 27].

Example The user is prompted for the encryption key when the *myencrypted.db* database is started:

```
dbsrv9 -ep -x tcpip myencrypted.db
```

See also

- "-ek database option" on page 171
- "DatabaseKey connection parameter [DBKEY]" on page 187

-fc server option

Function Specify the filename of a DLL (or shared object on UNIX) containing the File System Full callback function.

Syntax { **dbsrv9** | **dbeng9** } **-fc** *filename* . . .

Applies to All operating systems and servers.

Description This option can be used to notify users, and possibly take corrective action, when a file system full condition is encountered. If you use the -fc option, the database server attempts to load the specified DLL and resolve the entry point of the callback function during startup. If the Adaptive Server Anywhere database server cannot find both the DLL and the entry point, the database server returns an error and shuts down. The DLL is user-supplied and can use the callback to, among other things, invoke a batch file (or shell script on UNIX) you have supplied to take diagnostic or corrective action. Alternatively, the callback function itself can perform such an action.

A sample disk full callback function is located in *Samples\Asa\DiskFull*.

Adaptive Server Anywhere searches for the callback function DLL in the same locations as it searches for other DLLs and files.

☞ For information about where Adaptive Server Anywhere searches for files, see "How Adaptive Server Anywhere locates files" on page 274.

When the database server detects a disk full condition, it invokes the callback function (if one has been provided), passing it the following information:

- the name of the dbspace where the condition was triggered
- the operating system-specific error code from the failed operation

The return code from the call to xp_out_of_disk indicates whether the operation that caused the condition should be aborted or retried. If a non-zero value is returned, the operation is aborted, otherwise it is retried. The callback function is invoked repeatedly as long as it returns zero and the file system operation fails.

On Windows platforms, if the database server has been started with a Server Messages window (neither -qi nor -qw have been specified), and a callback DLL has not been provided, a dialog appears when a disk full condition occurs. This dialog contains the dbspace name and error code, and allows the user to choose whether the operation that caused the disk full condition should be retried or aborted.

On all other operating systems, when -fc is not specified and a disk full condition is encountered, a fatal error occurs.

You can create system events to track the available disk space of devices holding the database file, the log file, or the temporary file and alert administrators in case of a disk space shortage.

☞ For more information, see "CREATE EVENT statement" [*ASA SQL Reference,* page 351].

Example

When the database server starts, it attempts to load the *diskfull.dll* DLL.

```
dbeng9 -fc diskfull.dll
```

See also

- "Using callback functions" [*ASA Programming Guide,* page 263]
- "Choosing a system event" on page 307
- "TEMP_SPACE_LIMIT_CHECK option [database]" on page 694

-fips server option

Function

Requires that only FIPS-approved algorithms should be used for strong database and communication encryption.

Syntax

{ **dbsrv9** | **dbeng9** } **-fips** . . .

Description

Specifying this option enforces all server encryption to use FIPS-approved algorithms. This option applies to strong database encryption, client/server transport-layer security, and web services transport-layer security. You can still use unencrypted connections and databases when the -fips option is specified, but you cannot use simple encryption.

Separately licensable option required
Transport-layer security and strong database encryption using AES_FIPS requires that you obtain the separately-licensable SQL Anywhere Studio security option and is subject to export regulations.

☞ To order this component, see "Separately-licensable components" [*Introducing SQL Anywhere Studio*, page 5].

For strong database encryption, the -fips option causes new databases to use the AES_FIPS type, even if AES is specified in the ALGORITHM clause of the CREATE DATABASE statement. When the database server is started with -fips, you can run databases encrypted with AES or AES_FIPS strong encryption, but not databases encrypted with simple encryption. Unencrypted databases can also be started on the server when -fips is specified.

The SQL Anywhere Studio security option must be installed on any machine used to run a database encrypted with AES_FIPS.

For Adaptive Server Anywhere transport-layer security, the -fips option causes the server to use the RSA_TLS_FIPS, even if RSA_TLS is specified. If ECC_TLS is specified, an error occurs because a FIPS-approved elliptic-curve algorithm is not available.

For transport-layer security for web services, the -fips option causes the server to use HTTPS_FIPS, even if HTTPS is specified.

See also

- "Strong encryption" [*SQL Anywhere Studio Security Guide*, page 15]
- "Adaptive Server Anywhere Transport-Layer Security" [*SQL Anywhere Studio Security Guide*, page 27]
- "Using transport-layer security for web services" [*SQL Anywhere Studio Security Guide*, page 44]

-ga server option

Function

Unload database after last connection is dropped.

Syntax

{ **dbsrv9** | **dbeng9** } **-ga** . . .

Applies to	All operating systems except NetWare.
Description	Specifying this option on the network server causes each database to be unloaded after the last connection to it is dropped. In addition to unloading each database after the last connection is dropped, the personal server shuts down when the last database is stopped.

-gb server option

Function	Set the database process priority class.
Syntax	{ **dbsrv9** | **dbeng9** } **-gb** { **idle** | **normal** | **high** | **maximum** } . . .
Applies to	Windows NT/2000/XP
Description	Set the database process priority class. The value idle is provided for completeness, and maximum may interfere with the running of your computer. Normal and high are the commonly-used settings.

-gc server option

Function	Set maximum desired interval between checkpoints.
Syntax	{ **dbsrv9** | **dbeng9** } **-gc** *integer* . . .
Applies to	All operating systems and servers.
Description	Set the *maximum* desired length of time in minutes that the database server runs without doing a checkpoint on each database.

The default value is 60 minutes.

When a database server is running with multiple databases, the checkpoint time specified by the first database started is used unless overridden by this option. If a value of 0 is entered, the default value of 60 minutes is used.

Checkpoints generally occur more frequently than the specified time.

☞ For more information, see "How the database server decides when to checkpoint" on page 399.

See also

- "CHECKPOINT_TIME option [database]" on page 641
- "Checkpoints and the checkpoint log" on page 396

-gd server option

Function	Set permissions required to start or stop a database.
Syntax	{ **dbsrv9** | **dbeng9** } **-gd** { **DBA** | **all** | **none** } . . .

Applies to

All operating systems and servers.

Description

This is the permission required for a user to cause a new database file to be loaded by the server, or to stop a database on a running database server. The level can be one of the following:

- **DBA** Only users with DBA authority can start or stop databases.
- **all** All users can start or stop databases.
- **none** Starting and stopping databases is not allowed apart from when the database server itself is started and stopped.

The default setting is **ALL** for the personal database server and **DBA** for the network database server. Both upper case and lower case syntax is acceptable.

Note that when this option is set to DBA, the client application must already have a connection to the server in order to start or stop a database. Providing a DBA username and password on a new connection is not sufficient.

Example

The following steps illustrates how to use the -gd option for the network database server.

1. Enter a password in the *util_db.ini* file in the directory that holds the database server executable:

   ```
   [UTILITY_DB]
   pwd=mypwd
   ```

2. Start the network database server:

   ```
   dbsrv9 -x tcpip -n myserver -gd DBA
   ```

3. Connect to the utility database from Interactive SQL.

   The following command assumes that **myserver** is the default database server:

   ```
   dbisql -c "uid=DBA;pwd=mypwd;dbn=utility_db "
   ```

4. Start a database:

   ```
   start database asademo
   on myserver;
   ```

5. Connect to the database you have started:

   ```
   connect
   to myserver
   database asademo
   user DBA identified by SQL
   ```

-ge server option

Function	Set stack size for external functions.
Syntax	{ **dbsrv9** | **dbeng9** } **-ge** *integer* ...
Applies to	Windows 95/98/Me, Windows NT/2000/XP, NetWare
Description	Sets the stack size for threads running external functions, in bytes. The default is 32K.

-gf server option

Function	Disable firing of triggers by the server.
Syntax	{ **dbsrv9** | **dbeng9** } **-gf** ...
Applies to	All operating systems and servers.
Description	The -gf server option instructs the server to disable the firing of triggers.
See also	♦ "FIRE_TRIGGERS option [compatibility]" on page 654

-gk server option

Function
: Set the permission required to stop the network server and personal server using dbstop.

Syntax
: { **dbsrv9** | **dbeng9** } **-gk** { **DBA** | **all** | **none** } ...

Applies to
: All operating systems and servers.

Description
: The allowed values include:

- **DBA** Only users with DBA authority can use dbstop to stop the server. This is the default for the network server.
- **all** All users can use dbstop to stop the server. This is the default for the personal server.
- **none** The server cannot be stopped using dbstop.

Both upper case and lower case syntax is acceptable.

-gl server option

Function
: Set the permission required to load data using LOAD TABLE, and to unload data using UNLOAD or UNLOAD TABLE.

Syntax
: { **dbsrv9** | **dbeng9** } **-gl** { **DBA** | **all** | **none** } ...

Applies to	All operating systems and servers.

Description

Using the UNLOAD TABLE or UNLOAD statement places data in files on the database server machine, and the LOAD TABLE statement reads files from the database server machine.

To control access to the file system using these statements, the -gl server option allows you to control the level of database permission that is required to use these statements.

The allowed values are as follows:

- **DBA** Only users with DBA authority can load or unload data from the database.
- **all** All users can load or unload data from the database.
- **none** Data cannot be unloaded or loaded.

Both upper case and lower case syntax is acceptable.

The default settings are **all** for personal database servers on non-UNIX operating systems, and **DBA** for the network database server and the UNIX personal server. These settings reflect the fact that, on non-UNIX platforms, the personal database server is running on the current machine, and so the user already has access to the file system.

-gm server option

Function	Limit the number of concurrent connections to the server.
Syntax	{ **dbsrv9** | **dbeng9** } **-gm** *integer* . . .
Applies to	All operating systems and servers.

Description

Defines the connection limit for the server. If this number is greater than the number that is allowed under licensing and memory constraints, it has no effect.

The database server allows one extra DBA connection above the connection limit to allow a DBA to connect to the server and drop other connections in an emergency.

-gn server option

Function	Set the maximum number of active requests (both user and system) that the database server can handle concurrently.
Syntax	{ **dbsrv9** | **dbeng9** } **-gn** *integer* . . .

Applies to	All operating systems and servers.
Description	Set the number of active user and system requests that the database server can handle concurrently. If the maximum number of active requests are already in use when the database server receives an additional request, the new request must wait until another request is completed.

Each connection uses a thread for each request, and when the request is completed the thread is returned to the pool for use by other connections. As no connection can have more than one request in progress at one time, no connection uses more than one thread at a time. An exception to this rule is if a Java application uses threads. Each thread in the Java application is a database server execution thread.

The default is 20 threads for the network database server and for the personal database server, except on Windows CE where the default is 3 threads.

On Windows NT/2000/XP, you may want investigate the Performance Monitor readings for **Requests: Active** and **Requests: Unscheduled**. If the number of active requests is always less than -gn, you can lower -gn. If the number of total requests (active + unscheduled) is often larger than -gn, then you might want to increase -gn. The Performance Monitor is not available for UNIX or Linux platforms.

See also

- "Controlling threading from the command line" on page 14

-gp server option

Function	Set the maximum allowed database page size.
Syntax	{ **dbsrv9** | **dbeng9** } **-gp** { **1024** | **2048** | **4096** | **8192** | **16384** | **32768** } . . .
Applies to	All operating systems and servers.
Description	Database files with a page size larger than the page size of the server cannot be loaded. This option explicitly sets the page size of the server, in bytes.

If you do not use this option, then the page size of the first database on the command line is used.

The minimum page size on all UNIX platforms is 2048 bytes. You can still use databases with smaller page sizes, but cache memory is used very inefficiently. If you do not use this option and start a server with no databases loaded, the default value is 2048.

On all other platforms, if you do not use this option and start a server with no databases loaded, the default value is 1024.

-gr server option

Function	Set the maximum length of time (in minutes) for recovery from system failure.
Syntax	{ **dbsrv9** | **dbeng9** } **-gr** *integer* . . .
Applies to	All operating systems and servers.
Description	When a database server is running with multiple databases, the recovery time that is specified by the first database started is used unless overridden by this option. ☞ For more information, see "RECOVERY_TIME option [database]" on page 686.

-gss server option

Function	Set the stack size per internal execution thread in the server.
Syntax	{ **dbsrv9** | **dbeng9** } **-gss** { *integer* | *integer* **K** | *integer* **M** } . . .
Applies to	This option has no effect on Windows operating systems.
Description	The number of internal execution threads is controlled by the -gn option, and has a default value of 20. The -gss option allows you to lower the memory usage of the database server in environments with limited memory. On NetWare, the default, and minimum, stack size per internal execution thread is 128 kb, and the maximum stack size is 1 Mb. On UNIX, the default stack size per internal execution thread is 64 kb, the minimum stack size is 500 kb, and the maximum stack size is 4 Mb.
See also	♦ "Controlling threading from the command line" on page 14

-gt server option

Function	Set the maximum number of requests that can be run simultaneously by the database server. This option is only useful on multiprocessor systems.
Syntax	{ **dbsrv9** | **dbeng9** } **-gt** *integer* . . .
Applies to	All operating systems and servers except NetWare.
Description	With per-seat licensing, the network database server uses all CPUs available on the machine (the default). With CPU-based licensing, the network database server uses only the number of processors you are licensed for. In addition, the personal database server and runtime database server are both limited to a single processor.

See also

- "Controlling threading from the command line" on page 14

-gu server option

Function
Set permission levels for utility commands.

Syntax
{ **dbsrv9** | **dbeng9** } **-gu** { **all** | **none** | **DBA** | **utility_db** } . . .

Applies to
All operating systems and servers.

Description
Sets permission levels for utility commands such as CREATE DATABASE and DROP DATABASE. The level can be set to one of the following: **utility_db**, **all**, **none**, **DBA**. The default is **DBA**.

The **utility_db** level restricts the use of these commands to only those users who can connect to the utility database. The **all**, **none**, and **DBA** levels permit all users, no users, or users with DBA authority to execute utility commands.

-gx server option

Function
Set the maximum number of requests that can concurrently execute blocking system calls.

Syntax
{ **dbsrv9** | **dbeng9** } **-gx** *integer* . . .

Applies to
Windows 95/98/Me, Windows NT/2000/XP

Description
This option is only of use on Windows NT. On other platforms the -gx value is equivalent to the -gt value.

Meaningful values are in the range specified by the -gt option value and the -gn option value. By default, this option is set to a value one greater than the number of CPUs on the machine. The additional process permits the use of remote data access between databases on a single database server.

The primary cost of increasing -gx is reduced address space (not physical memory) available for the cache. This option is only of use if you are using remote data access, Java, or external stored procedures (and in the case of Java/external stored procedures, if those features are being used to implement servers). If you are using remote data access/Java/external stored procedures and the server hangs for no apparent reason, increasing -gx might solve the problem. Setting the -gx option beyond the -gn option has no benefit.

You may want to increase the option setting beyond the default to reserve capacity for external tasks, separate from standard database tasks. For example, you may want to increase the value of -gx by one for each concurrent connection using remote data access, or for connections using

Java in the database to listen on an external port. In other circumstances, this option should be left at the default value.

On UNIX, each task is executed in its own thread, so that the number of tasks (-gn) also determines the number of threads.

See also

- "Controlling threading from the command line" on page 14

-m server option

Function

Delete the transaction log when a checkpoint is done.

Syntax

{ **dbsrv9** | **dbeng9** } **-m** . . .

Applies to

All operating systems and servers.

Description

This option deletes the transaction log when a checkpoint is done, either at shutdown or as a result of a checkpoint scheduled by the server.

Caution
When this option is selected, there is no protection against media failure on the device that contains the database files.

This provides a way to automatically limit the growth of the transaction log. Checkpoint frequency is still controlled by the CHECKPOINT_TIME and RECOVERY_TIME options (which you can also set on the command line).

The -m option is useful for limiting the size of the transaction log in situations where high volume transactions requiring fast response times are being processed, and the contents of the transaction log are not being relied upon for recovery or replication. The -m option provides an alternative to operating without a transaction log at all, in which case a checkpoint would be required following each COMMIT and performance would suffer as a result. When the -m option is selected, there is no protection against media failure on the device that contains the database files. Other alternatives for managing the transaction log (for example, using the BACKUP statement and events) should be considered before using the -m option.

To avoid database file fragmentation, it is recommended that where this option is used, the transaction log be placed on a separate device or partition from the database itself.

Replicated and synchronized databases
Do not use the -m option with databases that are being replicated or synchronized. Replication and synchronization, used by SQL Remote and MobiLink, inherently rely on transaction log information.

-n server option

Function	Set the name of the database server.
Syntax	{ **dbsrv9** | **dbeng9** } **-n** *database-file-name*. . .
Applies to	All operating systems and servers.

Description

By default, the database server receives the name of the database file with the path and extension removed. For example, if the server is started on the file *c:\Program Files\Sybase\SQL Anywhere 9\asademo.db* and no -n option is specified, the name of the server is **asademo**.

The server name is interpreted according to the character set of the machine, as no database collation exists at startup time. Multi-byte characters are not recommended in server names.

Names must be a valid identifier. Long server names are truncated to different lengths depending on the protocol.

Protocol	**Truncation Length**
UNIX shared memory	31 bytes
non-UNIX shared memory	40 bytes
TCP/IP	40 bytes
SPX	32 bytes
Named Pipes	8 bytes

The following are invalid for database server names:

- ♦ names that begin with white space or single or double quotes
- ♦ names that end with white space
- ♦ names that contain semicolons

The server name specifies the name to be used in the EngineName (ENG) connection parameter of client application connection strings or profiles. With shared memory, there is a default database server that will be used if no server name is specified provided that at least one database server is running on the computer.

Running multiple servers with the same name is not recommended.

There are two -n options

The -n option is positional. If it appears before a database file name, it is a server option and names the server. If it appears after a database file name, it is a database option and names the database.

For example, the following command names the server SERV and the database DATA:

```
dbsrv9 -n SERV asademo.db -n DATA
```

☞ For more information, see "-n database option" on page 172.

See also

- ♦ "Identifiers" [*ASA SQL Reference*, page 7]
- ♦ "EngineName connection parameter [ENG]" on page 194

-o server option

Function Print all server window output to a file.

Syntax { **dbsrv9** | **dbeng9** } **-o** *filename* . . .

Applies to All operating systems and servers.

Description Print all server message window output to a file. You can obtain the name of the file by executing the following command:

```
SELECT property ( 'ConsoleLogFile' )
```

-oe server option

Function Specify a filename to log startup errors, fatal errors, and assertions.

Syntax { **dbsrv9** | **dbeng9** } **-oe** *filename* ...

Applies to All operating systems and servers.

Description Each line in this file is prefixed with the date and time. Startup errors include such errors as:

- ♦ Could not open/read database file: <database file>
- ♦ A database server with that name has already started

Fatal errors and assertions are logged to the Windows Application Event Log (except on Window CE) or the UNIX system log regardless of whether -oe is specified.

-os server option

Function Limit the file size for the server window output.

Syntax	{ **dbsrv9** | **dbeng9** } **-os** { *integer* | *integer* **G** | *integer* **K** | *integer* **M** } ...
Applies to	All operating systems and servers.
Description	Limits the size of the log file used by the `-o` option. The units **G**, **K**, and **M** can be either lower case or upper case. If **G**, **K**, or **M** is not supplied, any integer less than 10 000 is assumed to be in kilobytes, and any integer 10 000 or greater is assumed to be in bytes.
See also	♦ "-o server option" on page 151

-p server option

Function	Set the maximum size of communication packets.
Syntax	{ **dbsrv9** | **dbeng9** } **-p** *integer* . . .
Applies to	All operating systems and servers.
Description	The default is 1460 bytes. The minimum value is 300 bytes and the maximum is 16 000.
See also	♦ "CommBufferSize connection parameter [CBSIZE]" on page 180

-pc server option

Function	Compress all connections except for same-machine connections.
Syntax	{ **dbsrv9** } **-pc** . . .
Applies to	All operating systems and network servers.
Description	The packets sent between an Adaptive Server Anywhere client and server can be compressed using the -pc option. Compressing a connection may improve performance under some circumstances. Large data transfers with highly compressible data tend to get the best compression rates. This option can be overridden for a particular client by specifying **COMPRESS=NO** in the client's connection parameters. The default is not to compress connections. Specifying the -pc option compresses all connections except same-machine connections and TDS connections. TDS connections (including jConnect) do not support Adaptive Server Anywhere communication compression. Same-machine connections over any communication link will not enable compression, even if the -pc option or **COMPRESS=YES** connection parameter is used.
See also	♦ "Adjusting communication compression settings to improve performance" on page 95

- "Compress connection parameter [COMP]" on page 183
- "Try using Adaptive Server Anywhere's compression features" [*ASA SQL User's Guide*, page 170]

-pt server option

Function
: Increase or decrease the size limit at which packets are compressed.

Syntax
: { **dbsrv9** } **-pt** *size_in_bytes* . . .

Applies to
: All operating systems and network servers.

Description
: This parameter takes an integer value representing the minimum byte-size of packets to be compressed. Values less than 80 are not recommended. The default is 120 bytes.

Under some circumstances, changing the compression threshold can help performance of a compressed connection by allowing you to compress packets only when compression will increase the speed at which the packets are transferred. The default setting should be appropriate for most cases.

If both client and server specify different compression threshold settings, the client setting applies.

See also

- "Adjusting communication compression settings to improve performance" on page 95
- "CompressionThreshold connection parameter [COMPTH]" on page 184
- "Try using Adaptive Server Anywhere's compression features" [*ASA SQL User's Guide*, page 170]

-qi server option

Function
: Control whether database server tray icon and window appear.

Syntax
: { **dbsrv9** | **dbeng9** } **-qi** . . .

Applies to
: Windows platforms, excluding Windows CE.

Description
: This option leaves no visual indication that the server is running, other than possible startup error dialogs. You can use either (or both) the -o or -oe logs to diagnose errors.

-qp server option

Function
: Do not display messages about performance in the database server window.

Syntax
: { **dbsrv9** | **dbeng9** } **-qp** . . .

Applies to
: All operating systems and servers.

Description

Do not display messages about performance in the database server window. Messages that are suppressed include the following:

- ♦ No unique index or primary key for table '*table_name*'
- ♦ Database file "*mydatabase.db*" consists of *nnn* fragments

-qs server option

Function

Suppress startup error dialogs.

Syntax

{ **dbsrv9** | **dbeng9** } **-qs** . . .

Applies to

Windows only.

Description

This option suppresses startup error dialogs. Startup errors include errors such as:

- ♦ Could not open/read database file: <*database file*>
- ♦ A database server with that name has already started

On Windows platforms, if the server is not being autostarted, these errors appear in a dialog and must be cleared before the server stops. These dialogs do not appear if the -qs option is used.

If there is an error loading the language DLL, no dialog appears if -qs was specified on the command line and not in @data. This error is not logged to the -o or -oe logs, but rather to the Windows Application Event Log (except on Windows CE).

Usage errors are suppressed if -qs is on the command line, but not in @data expansion.

-qw server option

Function

Do not display database server screen.

Syntax

{ **dbsrv9** | **dbeng9** } **-qw** . . .

Applies to

All operating systems and servers except NetWare.

Description

This option suppresses the database server window (Windows platforms) and display messages on the console (non-Windows platforms).

-r server option

Function

Force all databases that start on the server to be read-only. No changes to the database(s) are allowed: the server does not modify the database file(s) and transaction log files.

Syntax	{ **dbsrv9** | **dbeng9** } **-r** . . .
Applies to	All operating systems and servers.
Description	Opens all database files as read-only with the exception of the temporary file when the option is specified before any database names on the command line. If the -r server option is specified after a database name, only that specific database is read-only. You can make changes on temporary tables, but ROLLBACK has no effect, since the transaction and rollback logs are disabled.

Databases distributed on CD-ROM devices, and compressed databases are examples of database files that cannot be modified. You can either create a write file to allow changes to the database outside the database file, or run in read-only mode.

If you attempt to modify the database, for example with an INSERT or DELETE statement, a SQLSTATE_READ_ONLY_DATABASE error is returned.

Databases that require recovery cannot be started in read-only mode. For example, database files created using an online backup cannot be started in read-only mode if there were any open transactions when the backup was started, since these transactions would require recovery when the backup copy is started.

Example

To open two databases in read-only mode

```
dbeng9 -r database1.db database2.db
```

To open only the first of two databases in read-only mode.

```
dbeng9 database1.db -r database2.db
```

-s server option

Function	Set the user ID for syslog messages.
Syntax	{ **dbsrv9** | **dbeng9** } **-s** { **none** | **user** | **daemon** | **local*n*** } . . .
Applies to	UNIX
Description	Sets the system user ID used in messages to the **syslog** facility. The default is **user** for database servers that are started in the foreground, and **daemon** for those that are run in the background (for example, started by dbspawn, autostarted by a client, or started with the -ud database server option).

A value of **none** prevents any syslog messages from being logged. The **local*n*** argument allows you to use a facility identifier to redirect messages to a file. You can specify a number between 0 and 7, inclusive, for *n*. Refer to the UNIX syslog(3) man page for more information.

The following steps illustrate how to redirect messages on Solaris, but you can also do this on Linux, AIX, and Mac OS X. Note that on other platforms, including DEC and HP, the *syslog.conf* file is found in a different location. You can place the */var/adm/sqlanywhere* file in whatever location you want.

❖ To redirect messages to a file using a facility identifier

1. Choose a unique facility identifier that is not already being used by another application that is running on your system.

   You can do this by looking in the */etc/syslog.conf* file to see of any of the **local*n*** facilities are referenced.

2. Edit the */etc/syslog.conf* file and add the following line, where **local*n*** is the facility identifier you chose in step 1:

   ```
   localn.err;localn.info;localn.notice  /var/adm/sqlanywhere
   ```

3. Create the */var/adm/sqlanywhere* file:

   ```
   touch /var/adm/sqlanywhere
   ```

4. Tell the syslogd process that you have modified the *syslog.conf* file by finding the process ID of syslogd:

   ```
   ps -ef | grep syslogd
   ```

   and then executing the following command where <*pid*> is the process ID of syslogd:

   ```
   kill -HUP <pid>
   ```

5. Start your Adaptive Server Anywhere database server with the following command, where **local*n*** is the facility identifier you chose in step 1:

   ```
   dbeng9 -s localn ...
   ```

   Now any messages that the Adaptive Server Anywhere database server reports to syslog are redirected to the */var/adm/sqlanywhere* file.

-sb server option

Function	Specify how the server reacts to broadcasts.
Syntax	{ **dbsrv9** | **dbeng9** } **-sb** { **0** | **1** } . . .
Applies to	SPX, TCP/IP

Description	Using **-sb 0** causes the server not to start up any TCP/UDP broadcast listeners. In addition to forcing clients to use the **DoBroadcast=NONE** and **HOST=** options to connect to the server, this option causes the server to be unlisted when using dblocate. Using **-sb 1** causes the server to not respond to broadcasts from dblocate, while leaving connection logic unaffected. You can connect to the server by specifying **LINKS=tcpip** and **ENG=**<*name*>.

-sc server option

Function	Disable the shared memory communications protocol and use Named Pipes.
Syntax	{ **dbsrv9** | **dbeng9** } **-sc** . . .
Applies to	Windows NT/2000/XP
Description	This option disables the shared memory communications protocol that is used for same-machine communications, and starts the Named Pipes protocol. This option is implemented as part of an initiative to obtain C2 security certification. It is likely to be of general use only for customers wanting to run in a C2-certified environment.

-ti server option

Function	Disconnect inactive connections.
Syntax	{ **dbsrv9** | **dbeng9** } **-ti** *minutes* . . .
Applies to	All operating systems and servers.
Description	Disconnect connections that have not submitted a request for the specified number of *minutes*. The default is 240 (4 hours). The maximum value is 32767. A client machine in the middle of a database transaction holds locks until the transaction is ended or the connection is terminated. The -ti option is provided to disconnect inactive connections, freeing their locks. The -ti option is very useful when used in conjunction with *dbsrv9* since most connections will be over network links (TCP or SPX). The -ti option is useful with *dbeng9* only for local TCP/IP connections. Using -ti has no effect on connections to a local server using shared memory. Setting the value to zero disables checking of inactive connections, so that no connections are disconnected.
See also	♦ "sa_server_option system procedure" [*ASA SQL Reference*, page 830]

-tl server option

Function	Set the period at which to send liveness packets.
Syntax	{ **dbsrv9** | **dbeng9** } **-tl** *seconds* . . .
Applies to	All database servers using TCP/IP or SPX.
See also	♦ "sa_server_option system procedure" [*ASA SQL Reference*, page 830]

Description

A liveness packet is sent periodically across a client/server TCP/IP or SPX communications protocol to confirm that a connection is intact. If the server runs for a LivenessTimeout period (default 2 minutes) without detecting a liveness packet on a connection, the communication is severed, and the server drops the connection associated with that client. UNIX non-threaded clients and TDS connections do not do liveness checking.

The -tl option on the server sets the LivenessTimeout value for all clients that do not specify a liveness period.

Liveness packets are sent when a connection has not sent any packets for between one third and two thirds of the LivenessTimeout value.

When there are more than 200 connections, the server automatically calculates a higher LivenessTimeout value based on the stated LivenessTimeout value. This enables the server to handle a large number of connections more efficiently. Liveness packets are sent between one third and two thirds of the LivenessTimeout on each idle connection. Large numbers of liveness packets are not sent at the same time. If liveness packets take a long time to send (depending on the network, the machine's hardware, and the CPU and network load on the machine), it is possible that liveness packets will sent after two thirds of the LivenessTimeout. A warning appears in the server console if the liveness sends take a long time. If this warning occurs, consider increasing the LivenessTimeout value.

Although it is not generally recommended, you can disable liveness by specifying the following:

```
dbsrv9 -tl 0
```

Rather than disabling the LivenessTimeout option, consider increasing the value to 1 hour as follows:

```
dbsrv9 -tl 3600
```

-tmf server option

Function	Use for recovery from distributed transactions in unusual circumstances.

Syntax	{ **dbsrv9** | **dbeng9** } **-tmf** . . .
Applies to	Windows NT/2000/XP only
Description	Used during recovery of distributed transactions when the distributed transaction coordinator is not available. It could also be used if starting a database with distributed transactions in the transaction log, on a platform where the distributed transaction coordinator is not available.

Caution
If you use this option, distributed transactions are not recovered properly. It is not for routine use.

-tmt server option

Function	Set a reenlistment timeout for participation in distributed transactions.
Syntax	{ **dbsrv9** | **dbeng9** } **-tmt** *milliseconds* . . .
Applies to	Windows NT/2000/XP only
Description	Used during recovery of distributed transactions. The value specifies how long the database server should wait to be reenlisted. By default there is no timeout (the database server waits indefinitely).

-tq server option

Function	Shut down the server at a specified time.
Syntax	{ **dbsrv9** | **dbeng9** } **-tq** { *datetime* | *time* } . . .
Applies to	All operating systems and servers.
Description	This is useful for setting up automatic off-line backup procedures (see "Backup and Data Recovery" on page 373). The format for the time is in *hh*:*mm* (24 hour clock), and can be preceded by an optional date. If a date is specified, the date and time must be enclosed in double quotes and be in the format *YYYY/MM/DD HH*:*MM*.
See also	♦ "sa_server_option system procedure" [*ASA SQL Reference*, page 830]

-u server option

Function	Open files using the operating system disk cache.
Syntax	{ **dbsrv9** | **dbeng9** } **-u** . . .
Applies to	Windows NT/2000/XP, Windows 95/98/Me, UNIX

Description	Files are opened using the operating system disk cache in addition to the database cache. While the operating system disk cache may improve performance in some cases, in general better performance is obtained without this option, using the database cache only. If the server is running on a dedicated machine, you should not use the -u option, as the database cache itself is generally more efficient. You may want to use the -u option if the server is running on a machine with several other applications (so that a large database cache may interfere with other applications) and yet IO-intensive tasks are run intermittently on the server (so that a large cache will improve performance).

-ua server option

Function	Turn off use of asynchronous I/O.
Syntax	{ **dbsrv9** | **dbeng9** } **-ua** ...
Applies to	Linux
Description	By default, the database server uses asynchronous I/O on Linux when possible. In order to use asynchronous I/O, the following conditions must be met: 1. The library *libaio.so* can be loaded at run time. 2. The kernel has asynchronous I/O support. If you want to turn off the use of asynchronous I/O, specify the -ua option on the database server command line.

-uc server option

Function	Start the database server in console mode. This is the default.
Syntax	{ **dbsrv9** | **dbeng9** } **-uc**
Applies to	UNIX
Description	Starts the database server in console mode. You should only specify one of -uc, -ui, or -ux. ☞ For information about starting the database server as a daemon, see “-ud server option” on page 161.
See also	♦ “-ui server option” on page 161 ♦ “-ux server option” on page 162

Example

-ud server option

Function	Run as a daemon.
Syntax	{ **dbsrv9** | **dbeng9** } **-ud** . . .
Applies to	UNIX
Description	Using this option lets you run the server so that it continues running after the current operating system session ends.

-ui server option

Function

Open the Server Startup Options dialog and display the Server Messages window, or start the database server in console mode if a usable display is not available on Linux and Solaris.

Syntax

{ **dbsrv9** | **dbeng9** } **-ui**

Applies to

Linux and Solaris, with X Windows Server support

Description

The -ui option allows you to use the Server Startup Options dialog to specify server options when starting the database server and display the Server Messages window once the server has started.

When the -ux option is the only option specified on the server command line, the Server Startup Options dialog appears where you can enter options for starting the database server. The Server Messages dialog appears once the server starts.

If you specify other server options in addition to -ux, then the Server Messages window appears once the database server starts.

When -ui is specified, the server attempts to find a usable display. If it cannot find one, for example because the DISPLAY environment variable is not set or because X Windows Server is not running, then the database server starts in console mode. You should specify the -ux option if you do not want the database server to start when it cannot locate a usable display. You should only specify one of -uc, -ui, or -ux.

☞ For information about starting the database server as a daemon, see "-ud server option" on page 161.

See also

- "-uc server option" on page 160
- "-ux server option" on page 162

-ut server option

Function	Touch temporary files.
Syntax	{ **dbsrv9** | **dbeng9** } **-ut** *minutes* . . .
Applies to	UNIX
Description	This option causes the server to touch temporary files at specified intervals.

-ux server option

Function

Open the Server Startup Options dialog or display the Server Messages window on Linux and Solaris.

Syntax

{ **dbsrv9** | **dbeng9** } **-ux**

Applies to

Linux and Solaris, with X Windows Server support

Description

The -ux option allows you to do two things when starting the database server: use the Server Startup Options dialog to specify server options when starting the database server and display the Server Messages window once the server has started.

When the -ux option is the only option specified on the server command line, the Server Startup Options dialog appears where you can enter options for starting the database server.

If you specify other server options in addition to -ux, then the Server Messages window appears once the database server is started. You should only specify one of -uc, -ui, or -ux.

The server must be able to find a usable display when -ux is specified. If it cannot find one, for example because the DISPLAY environment variable is not set or because X Windows Server is not running, then the database server fails to start.

☞ For information about starting the database server as a daemon, see "-ud server option" on page 161.

See also

- "-uc server option" on page 160
- "-ux server option" on page 162

Example

The following command displays the Server Startup Options dialog where you can enter options for starting the database server:

```
dbeng9 -ux
```

The following command starts the database server and displays the Server

Messages window:

```
dbeng9 -ux sample.db
```

-v server option

Function	Display the software version.
Syntax	{ **dbsrv9** | **dbeng9** } **-v** . . .
Applies to	All operating systems and servers.
Description	Supplies the database server version in a message box, and then stops.

-x server option

Function
Specify server side network communications protocols.

Syntax 1
dbsrv9 -x { **all** | **none** | *srv-protocols* } . . .

srv-protocols:
  { [**namedpipes** | **spx** | **tcpip**] *parmlist* },. . .
parmlist:
  (*parm=value*;. . . **)**

Syntax 2
dbeng9 -x { **all** | **none** | *eng-protocols* } . . .

eng-protocols:
  { **namedpipes** | **tcpip** [*parmlist*] },. . .
parmlist:
  (*parm=value*;. . . **)**

Applies to
All operating systems and servers.

Description
Use the -x option to specify which communications protocols, in addition to shared memory, you want to use to listen for client connection broadcasts.

If you do not specify the -x option, the server attempts to listen for client connection broadcasts using all protocols supported by the database server running on your operating system, including shared memory.

If you specify the -x option with one or more protocols, the server attempts to listen for client connection broadcasts using the specified protocol(s) and also using a shared-memory protocol.

If you are running Windows CE and specify the -x option, the server only attempts to listen for client connection broadcasts using the TCP/IP protocol unless you explicitly request otherwise.

Regardless of which settings you choose for the -x option, the server always listens for connection broadcasts using the shared memory protocol. In addition to the shared memory protocol, you can also specify the following:

- **ALL** Listen for connection attempts by the client using all communications protocols that are supported by the server on this platform, including shared memory. This is the default.
- **NamedPipes (NP)** Listen for connection attempts by the client using the NamedPipes protocol. NamedPipes is supported on Windows NT/2000/XP as an alternative means of same-machine communication.
- **NONE** Listen for connection attempts by the client using only the shared memory protocol.
- **SPX** Listen for connection attempts by the client using the SPX protocol. The SPX protocol is supported by NetWare, Windows NT/2000/XP, and Windows 95/98/Me network servers.
- **TCPIP (TCP)** Attempt to connect to the client using the TCP/IP protocol. The TCP/IP protocol is supported by the network server on all operating systems, and by the personal database server for same-machine communications.

By default, the database server listens for broadcasts on port 2638, and redirects them to the appropriate port. This ensures a connection in most cases.

You can override this default and cause the server not to listen on port 2638 by setting the option **-sb 0**, or by turning off the BroadcastListener option (**BroadcastListener=0**). Additionally, if the client and server are communicating through a firewall, the client must send the packet to the exact port the server is listening on by specifying **DoBroadcast=None** and **Host=**.

☞ For information, see "ServerPort protocol option [PORT]" on page 220.

For some protocols, additional parameters may be provided, in the format

```
-x tcpip(PARM1=value1;PARM2=value2;...)
```

☞ For a description of available parameters, see "Network protocol options" on page 206.

For UNIX, quotation marks are required if more than one parameter is supplied:

```
-x "tcpip(PARM1=value1;PARM2=value2;...)"
```

Example

Allow only shared memory, TCP/IP, and SPX communications:

```
-x tcpip,spx
```

See also

♦ "CommLinks connection parameter [LINKS]" on page 181

-xs server option

Function

Specify server-side web services communications protocols.

Syntax

{ **dbeng9** | **dbsrv9** } **-xs** { **none** | *web-protocols* } . . .

web-protocols :
{ **http** (*parmlist*) | **https** (*parmlist*) |**https_fips** (*parmlist*) } ,
...
parmlist:
(*parm=value*;. . . **)**

Applies to

All operating systems and servers.

Description

Use the -xs option to specify which web protocols you want to use to listen for requests.

If you do not specify the -xs option, the server does not attempt to listen for web requests.

If you specify the -xs option with one or more protocols, the server attempts to listen for web requests using the specified protocol(s).

You can use the https or the FIPS-approved HTTPS_FIPS protocols for transport-layer security.

☞ For more information, see "Using transport-layer security for web services" [*SQL Anywhere Studio Security Guide*, page 44].

Separately licensable option required
Transport-layer security requires that you obtain the separately-licensable SQL Anywhere Studio security option and is subject to export regulations.

☞ To order this component, see "Separately-licensable components" [*Introducing SQL Anywhere Studio*, page 5].

Regardless of which settings you specify with the -xs option, the server always listens for connection broadcasts using the shared memory protocol. You can specify any of the following:

- **HTTP** Listen for web requests by the client using the HTTP protocol. The default port on which to listen is 80.
- **HTTPS** Listen for web requests by the client using the HTTPS protocol. The default port on which to listen is 443.

- **HTTPS_FIPS** Listen for web requests by the client using the HTTPS protocol and FIPS-approved algorithms for encryption. HTTPS_FIPS uses a separate approved library but is compatible with HTTPS. The default port on which to listen is 443.

  Note
  If you want to start both https and https_fips at the same time, then you must change the port for one of them since they both have the same default port.

- **NONE** Do not listen for web requests. This is the default.

☞ For a description of available parameters, see "Network protocol options" on page 206.

For UNIX, quotation marks are required if more than one parameter is supplied:

```
-xs "http(PARM1=value1;PARM2=value2;...)"
```

Example Listen for HTTP web requests on port 80:

```
dbeng9 web.db -xs http(port=80)
```

-y server option

Function Run as a Windows service.

Syntax { **dbsrv9** | **dbeng9** } **-y** . . .

Applies to Windows 95/98/Me

Description If the server registered as a Windows service, it continues to operate whether users log on or off, and shutdown commands are ignored.

-z server option

Function Display communications operations on startup for troubleshooting purposes.

Syntax { **dbsrv9** | **dbeng9** } **-z** . . .

Applies to All operating systems and servers.

Description This should only be used when tracking problems. The information appears in the database server window.

-zl server option

Function Turn on capturing of the most recently-prepared SQL statement for each connection to databases on the server.

Syntax { **dbsrv9** | **dbeng9** } **-zl** . . .

Applies to All operating systems and servers.

Description This feature can also be turned on using the RememberLastStatement server setting. You can obtain the most recently-prepared SQL statement for a connection using the LastStatement value of the connection_property function. The sa_conn_activity stored procedure allows you to obtain the most recently-prepared SQL statement for all current connections to databases on the server.

For stored procedure calls, only the outermost procedure call appears, not the statements within the procedure.

Caution
When -zl is specified or when the RememberLastStatement server setting is turned on, any user can call the sa_conn_activity system procedure or obtain the value of the LastStatement connection property to find out the most recently-prepared SQL statement for any other user. This option should be used with caution and turned off when it is not required.

See also

- "Connection-level properties" on page 713
- "sa_conn_activity system procedure" [*ASA SQL Reference*, page 778]
- "sa_server_option system procedure" [*ASA SQL Reference*, page 830]

-zn server option

Function Specifies the number of request log file copies to retain.

Syntax { **dbsrv9** | **dbeng9** } **-zn** *integer*

Applies to All operating systems and servers.

Description If request logging is enabled over a long period of time, the request log file can become large. The -zn option allows you to specify the number of request log file copies to retain. It only takes effect if -zs is also specified. The -zs option allows you to create a new log file and rename the original log file when the original log file reaches a specified size.

☞ For more information about the -zs option, see "-zs server option" on page 169.

For example, if you redirect request logging information to the file *req.out*, and specify five request log file copies using the -zn option, the server creates files in the following order: *req.out.1*, *req.out.2*, *req.out.3*, *req.out.4*, and *req.out.5*. When these files exist and the active request log fills again, the following happens:

- *req.out.1* is deleted
- the files *req.out.2* to *req.out.5* are renamed *req.out.1* to *req.out.4*
- the copy of the active log is renamed *req.out.5*

Request logging is turned on using the -zr option and redirected to a separate file using the -zo option. You can also set the number of request logs using the sa_server_option system procedure where *nn* specifies the number of request log file copies:

```
CALL sa_server_option('RequestLogNumFiles',nn)
```

Example

In the following example, entered all on one line, request logging information is output to a request log file named *mydatabase.log*, which has a maximum size of 10 KB, and three copies of the request log are kept:

```
dbeng9 "C:\Program Files\Sybase\SQL Anywhere 9\asademo.db" -zr
         all -zn 3
 -zs 10 -zo mydatabase.log
```

See also

- "-zo server option" on page 168
- "-zr server option" on page 168
- "-zs server option" on page 169
- "sa_server_option system procedure" [*ASA SQL Reference*, page 830]

-zo server option

Function

Redirect request logging information to a file separate from the regular log file.

Syntax

{ **dbsrv9** | **dbeng9** } **-zo** *filename*. . .

Applies to

All operating systems and servers.

Description

Request logging is turned on using the -zr option. You can direct the output from this file to a separate file from that specified with the -zo option.

This option also prevents request logging from appearing in the console.

See also

- "-zn server option" on page 167
- "-zr server option" on page 168
- "-zs server option" on page 169

-zr server option

Function

Enable request logging of operations.

Syntax

{ **dbsrv9** | **dbeng9** } **-zr** { **all** | **SQL** | **none** | **SQL+hostvars** } . . .

Applies to All operating systems and servers.

Description This should only be used when tracking problems. The information appears in the database server window or is sent to the logging file.

See also

- ♦ "sa_server_option system procedure" [*ASA SQL Reference*, page 830]
- ♦ "-zn server option" on page 167
- ♦ "-zo server option" on page 168

-zs server option

Function Limit the size of the request logging file.

Syntax { **dbsrv9** | **dbeng9** } **-zs** { *integer* | *integer* **G** | *integer* **K** | *integer* **M** } . . .

Applies to All operating systems and servers.

Description Request logging is turned on using the -zr option, and redirected to a separate file using the -zo option. You can limit the size of the file using the -zs option.

The units **G**, **K**, and **M** can be either lower case or upper case. If **G**, **K**, or **M** is not supplied, any integer less than 10 000 is assumed to be in kilobytes, and any integer 10 000 or greater is assumed to be in bytes.

When the request log file reaches the size specified by either the -zs option or the sa_server_option system procedure, the file is renamed with the extension *.old* appended (replacing an existing file with the same name if one exists). The request log file is then restarted.

See also

- ♦ "-zn server option" on page 167
- ♦ "-zo server option" on page 168
- ♦ "-zr server option" on page 168
- ♦ "sa_server_option system procedure" [*ASA SQL Reference*, page 830]

Example The following example shows how the -zs option is used to control log file size. Suppose you start a database server with the following command line:

```
dbeng9 -zr all -zs 10 -zo mydatabase.log
```

A new log file *mydatabase.log* is created. When this file reaches 10K in size, any existing *mydatabase.old* files are deleted, *mydatabase.log* is renamed to *mydatabase.old*, and a new *mydatabase.log* file is started. This process is repeated each time the *mydatabase.log* file reaches the specified size (in this case 10K).

Recovery options

These options are for use in recovery situations only.

-a recovery option

Function	Apply the named transaction log.
Syntax	{ **dbsrv9** | **dbeng9** } **-a** *log-filename* . . .
Applies to	All operating systems and servers.

Description

This is used to recover from media failure on the database file. When this option is specified, the database server applies the log and then terminates—it will not continue to run.

Specifying a cache size when starting the server can reduce recovery time.

☞ For information on recovery, see "Backup and Data Recovery" on page 373.

Example

The following example, entered all on one line, applies the log file *asademo.log* to the sample database.

```
dbeng9 "C:\Program Files\Sybase\SQL Anywhere 9\asademo.db" -a
         "C:\backup\asademo.log"
```

-f recovery option

Function	Force the database server to start after the transaction log has been lost.
Syntax	{ **dbsrv9** | **dbeng9** } **-f** . . .
Applies to	All operating systems and servers.

Description

If there is no transaction log, the database server carries out a checkpoint recovery of the database and then terminates—it does not continue to run. You can then restart the database server without the -f option for normal operation.

If there is a transaction log in the same directory as the database, the database server carries out a checkpoint recovery, and a recovery using the transaction log, and then terminates—it does not continue to run. You can then restart the database server without the -f option for normal operation.

During recovery, you should use the same server that you use in production. For example, if you use the network database server (*dbsrv9.exe*) during production, then you should use the network server for recovery because you may encounter problems as a result of the personal server's 10 connection limit. Specifying a cache size when starting the server can reduce recovery time.

☞ For more information, see "Backup and Data Recovery" on page 373.

Example

The following command forces the database server to start and carry out a recovery of the sample database.

```
dbeng9 "c:\Program Files\Sybase\SQL Anywhere 9\asademo.db" -f
```

Database options

These options are entered after the database name, and apply only to that database.

-ek database option

Function

Specify the key for a strongly encrypted database.

Syntax

{ **dbsrv9** | **dbeng9** } [*server-options*] *database-file* **-ek** *key* . . .

Description

The -ek database option must be provided after a database filename on the command line. You must provide the KEY value with the -ek option to start an encrypted database. The KEY is a string, including mixed cases, numbers, letters, and special characters.

If you want to type the encryption key in a dialog so it cannot be seen in clear text, use the -ep server option.

☞ For more information, see “-ep server option” on page 138.

If you want to secure communication packets between client applications and the database server use the -ec server option and transport-layer security.

☞ For more information, see “Adaptive Server Anywhere Transport-Layer Security” [*SQL Anywhere Studio Security Guide*, page 27].

Example

The following example starts the sample database and specifies the encryption key on the command line.

```
dbsrv9 -x tcpip asademo.db -ek "Akmm9u70y"
```

See also

- “-ep server option” on page 138
- “DatabaseKey connection parameter [DBKEY]” on page 187

-m database option

Function

Truncate the transaction log when a checkpoint is done.

Syntax

{ **dbsrv9** | **dbeng9** } [*server-options*] *database-file* **-m** . . .

Applies to

All operating systems and servers.

Description

Truncate (delete) the transaction log when a checkpoint is done, either at shutdown or as a result of a checkpoint scheduled by the server. This provides a way to automatically limit the growth of the transaction log. Checkpoint frequency is still controlled by the CHECKPOINT_TIME and RECOVERY_TIME options (which you can also define on the command line).

The -m option is useful where high volume transactions requiring fast response times are being processed, and the contents of the transaction log are not being relied upon for recovery or replication. When this option is selected, there is no protection against media failure on the device that contains the database files.

To avoid database file fragmentation, it is recommended that where this option is used, the transaction log be placed on a separate device or partition from the database itself.

This option is the same as the -m server option, but applies only to the current database or the database identified by the *database-file* variable.

Replicated databases
Do not use the -m option with databases that are being replicated. Replication inherently relies on transaction log information.

Example

The following example starts a database server named silver and loads the sample database. When a checkpoint is done, the transaction log is truncated.

```
dbsrv9 -n silver "c:\Program Files\Sybase\SQL Anywhere 9\
         asademo.db" -m
```

-n database option

Function

Set the name of the database.

Syntax

{ **dbsrv9** | **dbeng9** } [*server-options*] *database-file* **-n** *string* . . .

Applies to

All operating systems and servers.

Description

Both database servers and databases can be named. Since a database server can load several databases, the database name is used to distinguish the different databases.

By default, the database receives the name of the database file with the path and extension removed. For example, if the database is started on *c:\asa\asademo.db* and no -n option is specified, the name of the database is **asademo**.

The following are invalid for database server names:

- names that begin with white space or single or double quotes

- names that end with white space
- names that contain semicolons

Example

The following example starts the database server with a cache size of 3 Mb, loads the sample database, and names the sample database “test”.

```
dbsrv9 -c 3Mb "c:\Program Files\Sybase\SQL Anywhere 9\
         asademo.db" -n "test"
```

There are two -n options

The -n option is positional. If it appears before a database file name, it is a server option and names the server. If it appears after a database file name, it is a database option and names the database.

For example, the following command names the server SERV and the database DATA:

```
dbsrv9 -n SERV asademo.db -n DATA
```

☞ For more information, see “-n server option” on page 150.

-r database option

Function

Start the named database as read-only. No changes to the database(s) are allowed: the server does not modify the database file(s) and transaction log files. This option is position dependent.

Syntax

{ **dbsrv9** | **dbeng9** } **-r** . . .

Applies to

All operating systems and servers.

Description

Opens all database files as read-only with the exception of the temporary file when the option is specified before any database names on the command line. If the -r server option is specified after a database name, only that specific database is read-only. You can make changes on temporary tables, but ROLLBACK has no effect, since the transaction and rollback logs are disabled.

Databases distributed on CD-ROM devices, and compressed databases are examples of database files that cannot be modified. You can either create a write file to allow changes to the database outside the database file, or run in read-only mode.

If you attempt to modify the database, for example with an INSERT or DELETE statement, a SQLSTATE_READ_ONLY_DATABASE error is returned.

Databases that require recovery cannot be started in read-only mode. For example, database files created using an online backup cannot be started in read-only mode if there were any open transactions when the backup was started, since these transactions would require recovery when the backup copy is started.

Example

To open two databases in read-only mode

```
dbeng9 -r database1.db database2.db
```

To open only the first of two databases in read-only mode.

```
dbeng9 database1.db -r database2.db
```

CHAPTER 6

Connection Parameters and Network Protocol Options

About this chapter

This chapter provides a reference for the parameters that establish and describe connections from client applications to a database.

Contents

Connection parameters

This section describes each connection parameter. Connection parameters are included in connection strings. They can be entered in the following places:

♦ In an application's connection string.

  ☞ For more information, see "Assembling a list of connection parameters" on page 68.

♦ In an ODBC data source.

  ☞ For more information, see "Working with ODBC data sources" on page 53.

♦ In the Adaptive Server Anywhere Connect dialog.

  ☞ For more information, see "Connecting from Adaptive Server Anywhere utilities" on page 52.

The ODBC configuration dialog and the Adaptive Server Anywhere Connect dialog for Windows operating systems share a common format. Some of the parameters correspond to checkboxes and fields in these dialogs, while others can be entered in the text box on the Advanced tab.

Notes

♦ Connection parameters are case insensitive.

♦ The Usage for each connection parameter describes the circumstances under which the parameter is to be used. Common usage entries include the following:

  - **Embedded databases** When Adaptive Server Anywhere is used as an embedded database, the connection starts a personal server and loads the database. When the application disconnects from the database, the database is unloaded and the server stops.
  - **Running local databases** This refers to the case where an Adaptive Server Anywhere personal server is already running, and the database is already loaded on the server.
  - **Network servers** When Adaptive Server Anywhere is used as a network server, the client application must locate a server already running somewhere on the network and connect to a database.

♦ You can use the dbping utility to test connection strings. For example, if a personal server with the name asademo is running the database *asademo* (which can be started with the command `dbeng9 asademo.db`)

  The following string returns the message `Ping database successful`:

```
dbping -d -c "eng=asademo;dbn=asademo;uid=db;pwd=sql"
```

The following command, however, returns the message `Ping database failed - Database server not running`:

```
dbping -d -c "eng=other_engine;uid=dba;pwd=sql"
```

☞ For more information, see "The Ping utility" on page 563.

AppInfo connection parameter [APP]

Function	To assist administrators in identifying the origin of particular client connections from a database server.
Usage	Anywhere
Values	*String*
Default	*Empty string*

Description

This connection parameter is sent to the database server from embedded SQL, ODBC, OLE DB, or ADO.NET clients and from applications using the iAnywhere JDBC driver. It is not available from Open Client or jConnect applications.

It consists of a generated string that holds information about the client process, such as the IP address of the client machine, the operating system it is running on, and so on. The string is associated in the database server with the connection, and you can retrieve it using the following statement:

```
SELECT connection_property( 'AppInfo' )
```

Clients can also specify their own string, which is appended to the generated string. The AppInfo property string is a sequence of semicolon-delimited **key**=*value* pairs. The valid keys are as follows:

- **API** DBLIB, ODBC, OLEDB, or ADO.NET (ODBC is returned of the iAnywhere JDBC driver).
- **APPINFO** If you specified AppInfo in the connection string, the string entered.
- **EXE** The name of the client executable (Windows and NetWare only).
- **HOST** The host name of the client machine.
- **IP** The IP address of the client machine (UNIX and NetWare only).
- **OS** The operating system name and version number (for example, Windows 2000, NetWare 5.1).

- **PID** The process ID of the client (Windows and UNIX only).
- **THREAD** The thread ID of the client (Windows and UNIX only).
- **TIMEZONEADJUSTMENT** The number of minutes that must be added to the Coordinated Universal Time (UTC) to display time local to the connection.
- **VERSION** The version of the connection protocol in use, including major and minor values, and a build number (for example 9.0.00.3642).

If you specify a debug log file in your client connection parameters, the APPINFO string is added to the file.

See also

- "How connection parameters work" on page 40
- "Connection parameter tips" on page 64

Examples

- Connect to the sample database from Interactive SQL (the iAnywhere JDBC driver is used by default):

  ```
  dbisql -c "uid=DBA;pwd=SQL;dbf=C:\Program Files\Sybase\SQL
           Anywhere 9\asademo.db"
  ```

  View the application information:

  ```
  SELECT connection_property('appinfo')
  ```

  The result is as follows (in a single string):

  ```
  HOST=machine-name;OS=Windows 2000 Build 2195 Service Pack
           3;PID=0x724;THREAD=0x6bc;EXE=C:\Program Files\
           Sybase\SQL Anywhere 9\win32\
           dbisqlg.exe;VERSION=9.0.00.3642;API=ODBC;TIMEZONEAD
           JUSTMENT=-300
  ```

- Connect to the sample database from Interactive SQL, appending your own information to the AppInfo property:

  ```
  dbisql -c "uid=DBA;pwd=SQL;dbf=C:\Program Files\Sybase\SQL
           Anywhere 9\asademo.db;app=ISQL connection"
  ```

  View the application information:

  ```
  SELECT connection_property('appinfo')
  ```

  The result is as follows (in a single string):

  ```
  HOST=machine-name;OS=Windows 2000 Build 2195 Service Pack
           3;PID=0x8d0;THREAD=0xd74;EXE=C:\Program Files\
           Sybase\SQL Anywhere 9\win32\
           dbisqlg.exe;VERSION=9.0.00.3642;API=ODBC;TIMEZONEAD
           JUSTMENT=-300;APPINFO=ISQL connection
  ```

AutoStart connection parameter [ASTART]

Function	To prevent a local database server from being started if no connection is found.
Usage	Anywhere
Values	**YES**, **NO**
Default	**YES**
Description	By default, if no server is found during a connection attempt, and a database file, database name, or the START connection parameter is specified, then a database server is started on the same machine. You can turn this behavior off by setting the AutoStart (ASTART) connection parameter to **NO** or **OFF** in the connection string.
See also	♦ "How connection parameters work" on page 40 ♦ "Connection parameter tips" on page 64

AutoStop connection parameter [ASTOP]

Function	To prevent a database from being stopped as soon as there are no more open connections.
Usage	Embedded databases
Values	**YES**, **NO**
Default	**YES**
Description	By default, any server that is started from a connection string is stopped when there are no more connections to it. Also, any database that is loaded from a connection string is unloaded as soon as there are no more connections to it. This behavior is equivalent to `AutoStop=YES`. If you supply `AutoStop=NO`, any database that you start in that connection remains running when there are no more connections to it. As a consequence, the database server remains operational as well. The AutoStop (ASTOP) connection parameter is used only if you are connecting to a database that is not currently running. It is ignored if the database is already started.
See also	♦ "How connection parameters work" on page 40 ♦ "Connection parameter tips" on page 64

CharSet connection parameter [CS]

Function	To specify the character set to be used on this connection.
Usage	Anywhere
Values	*String*
Default	The local character set.
	☞ For information on how this is determined, see "Determining locale information" on page 353.
Description	If you supply a value for CharSet, the specified character set is used for the current connection. Setting **CharSet=none** disables character set conversion for the connection.
	☞ For a list of valid character set values, see "Character set labels" on page 369.
See also	♦ "How connection parameters work" on page 40 ♦ "Connection parameter tips" on page 64

CommBufferSize connection parameter [CBSIZE]

Function	To set the maximum size of communication packets, in bytes.
Usage	Anywhere
Values	*Integer*
Default	If no CommBufferSize value is set, the CommBufferSize is controlled by the setting on the server, which defaults to **1460** bytes.
Description	The CommBufferSize (CBSIZE) connection parameter specifies the size of communication packets, in bytes. The minimum value of CommBufferSize is 300, and the maximum is 16 000.

The protocol stack sets the maximum size of a packet on a network. If you set the CommBufferSize to be larger than that permitted by your network, the largest buffers are broken up by the network software. You should set the buffer size to be somewhat smaller than that allowed by your network because the network software may add information to each buffer before sending it over the network. The default of 1460 allows an ethernet packet to be completely filled when using TCP/IP.

A larger packet size may improve performance for multi-row fetches and fetches of larger rows, but it also increases memory usage for both the client and the server.

If CommBufferSize is not specified on the client, the connection uses the server's buffer size. If CommBufferSize is specified on the client, the connection uses the CommBufferSize value.

Using the -p database server option to set the CommBufferSize causes all clients that do not specify their own CommBufferSize to use the size specified by the -p database server option.

See also

- "How connection parameters work" on page 40
- "Connection parameter tips" on page 64

Example

- To set the buffer size to 400 bytes:

```
...
CommBufferSize=400
...
```

  Alternatively, you can set this parameter by entering its value in the Buffer Size text box on the Network tab of the ODBC Configuration dialog.

CommLinks connection parameter [LINKS]

Function

To specify client side network protocol options.

Usage

Anywhere. The CommLinks (LINKS) connection parameter is optional for connections to a personal server, and required for connections to a network server.

Values

String

Default

Use only the shared memory communication protocol to connect.

Description

If you do not specify a CommLinks (LINKS) connection parameter, the client searches for a server on the current machine only, and only using a shared memory connection. This is the default behavior, and is equivalent to `CommLinks=ShMem`. The shared memory protocol is the fastest communication link between a client and server running on the same machine, as is typical for applications connecting to a personal database server.

If you specify `CommLinks=ALL`, the client searches for a server using all available communication protocols. Since there may be an impact on performance if you specify `CommLinks=ALL`, use this setting only when you don't know which protocol to use.

If you specify one or more protocols in the CommLinks (LINKS) connection parameter, the client uses the named communication protocol(s), *in the order specified*, to search for a network database server. A connection error appears and the connection attempt aborts if the connection fails to connect using a specified protocol, even if there are protocols remaining in the list to try.

CommLinks (LINKS) connection parameter values are case insensitive, and include:

- **SharedMemory (ShMem)** Start the shared memory protocol for same-machine communication. This is the default setting. The client tries shared memory first if it appears in a list of protocols, regardless of the order in which protocols appear.
- **ALL** Attempt to connect using the shared memory protocol first, followed by all remaining and available communication protocols. Use this setting if you are unsure of which communication protocol(s) to use.
- **NamedPipes (NP)** For C2 security purposes, connect from a Windows NT/2000/XP client to a database server on the same machine that was started with the –sc option.
- **TCPIP (TCP)** Start the TCP/IP communication protocol. TCP/IP is supported on all operating systems.
- **SPX** Start the SPX communication protocol. The SPX protocol is supported for Windows and NetWare clients.

Each of these values can have additional network protocol options supplied.

☞ For a list of parameters, see "Network protocol options" on page 206.

You may want to use a specific protocol, as opposed to ALL, for the following reasons:

- The network library starts slightly faster if the client uses only necessary network protocols.
- Connecting to the database may be faster.
- You must specify the protocol explicitly if you want to tune the broadcast behavior of a particular protocol by providing additional network protocol options.

The CommLinks (LINKS) connection parameter corresponds to the database server –x option.

See also

- "Network protocol options" on page 206
- "Client/Server Communications" on page 85
- "-x server option" on page 163
- "How connection parameters work" on page 40
- "Connection parameter tips" on page 64

Examples

- The following connection string fragment starts the TCP/IP protocol only:

  ```
  CommLinks=tcpip
  ```

- The following connection string fragment starts the shared memory protocol and searches for the database server over shared memory. If the search fails, it then starts the TCP/IP port and searches for the server on the local network. If that fails, it starts the SPX port and searches for the server over SPX.

  ```
  CommLinks=tcpip,shmem,spx
  ```

- The following connection string fragment starts the SPX port and searches for the server over SPX. If the search fails, it then starts the TCP port and searches for the server on the local network, as well as the host kangaroo. Note that if the server is found over SPX, the TCP port is *not* started.

  ```
  CommLinks=spx,tcpip(HOST=kangaroo)
  ```

Compress connection parameter [COMP]

Function

To turn compression ON or OFF for a connection. Compressing a connection may improve performance under some circumstances.

Usage

Anywhere except with TDS connections. TDS connections (including jConnect) do not support Adaptive Server Anywhere communication compression.

Values

YES, **NO**

In the case of a difference between client and server settings, the client setting applies.

Default

NO

If no Compress value is set, the compression status is controlled by the setting on the server, which defaults to no compression.

Description

The packets sent between an Adaptive Server Anywhere client and server can be compressed using the Compress (COMP) connection parameter. Large data transfers with highly compressible data tend to get the best compression rates.

Available values of the Compress (COMP) connection parameter are case insensitive, and include:

- **YES** Turn communication compression on for this connection.

- **NO** Turn communication compression off for this connection.

To save yourself time and possible disappointment, it is wise to conduct a performance analysis on the particular network and using the particular application before using communication compression in a production environment.

To enable compression for all remote connections on the server, use the -pc server option.

Note that same-machine connections over any communication link will not enable compression, even if the -pc option or `COMPRESS=YES` parameter is used.

See also

- "-pc server option" on page 152
- "Adjusting communication compression settings to improve performance" on page 95
- "How connection parameters work" on page 40
- "Connection parameter tips" on page 64

Examples

- The following connection string fragment turns packet compression ON:

```
Compress=YES
```

- The following connection string fragment turns packet compression OFF:

```
Compress=NO
```

CompressionThreshold connection parameter [COMPTH]

Function

To increase or decrease the size limit at which packets are compressed. Changing the compression threshold can help performance of a compressed connection by allowing you to only compress packets when compression will increase the speed at which the packets are transferred.

Usage

Anywhere except TDS. Only applies to compressed connections.

Values

Integer, representing the minimum byte-size of packets to be compressed. Values less than 80 are not recommended.

If both client and server specify different compression threshold settings, the client setting applies.

Default

120

If no CompressionThreshold value is set, the compression threshold value is controlled by the setting on the server, which defaults to 120 bytes.

Description

When compression is enabled, individual packets may or may not be compressed, depending on their size. For example, Adaptive Server

Anywhere does not compress packets smaller than the compression threshold, even if communication compression is enabled. As well, small packets (less than about 100 bytes) usually do not compress at all. Since CPU time is required to compress packets, attempting to compress small packets could actually decrease performance.

Generally speaking, lowering the compression threshold value may improve performance on very slow networks, while raising the compression threshold may improve performance by reducing CPU. However, since lowering the compression threshold value will increase CPU usage on both the client and server, a performance analysis should be done to determine whether or not changing the compression threshold is beneficial.

See also

- “-pt server option” on page 153
- “Adjusting communication compression settings to improve performance” on page 95
- “How connection parameters work” on page 40
- “Connection parameter tips” on page 64

Example

- Connect, with a compression threshold of 100 bytes.

```
CompressionThreshold=100
```

ConnectionName connection parameter [CON]

Function

Names a connection, to make switching to it easier in multi-connection applications.

Usage

Anywhere.

Values

String

Default

No connection name.

Description

An optional parameter, providing a name for the particular connection you are making. You may leave this unspecified unless you are going to establish more than one connection, and switch between them.

The connection name is not the same as the data source name.

See also

- “How connection parameters work” on page 40
- “Connection parameter tips” on page 64

Example

- Connect, naming the connection First_Con:

```
CON=First_Con
```

DatabaseFile connection parameter [DBF]

Function

For use when starting a database that is not already running. The

DatabaseFile connection parameter indicates which database file you want to load and connect to.

If you want to connect to an already running database, use the **DatabaseName (DBN)** parameter.

Usage

Embedded databases

Values

String

Default

There is no default setting.

Description

The DatabaseFile (DBF) connection parameter is used to load and connect to a specific database file that is not already running on a database server.

- ♦ If the database you want to connect to is not already running, use the DatabaseFile (DBF) connection parameter so the database can be started.
- ♦ If the filename does not include an extension, Adaptive Server Anywhere looks for a file with the *.db* extension.
- ♦ The path of the file is relative to the working directory of the database server. If you start the server from a command prompt, the working directory is the directory that you are in when entering the command. If you start the server from an icon or shortcut, it is the working directory that the icon or shortcut specifies. It is recommended that you supply a complete path and file name.
- ♦ If you specify both the database file and the database name, an attempt is made to connect to a running database with the specified name (the database file is ignored), and if that fails, an attempt is made to autostart a database using both the database file and database name.

You can also use UNC filenames and Novell NetWare Directory Services file names.

☞ For information about using UNC filenames and Novell NetWare Directory Services file names, see "The database server" on page 116.

Caution
The database file must be on the same machine as the database server. Starting a database file that is located on a network drive can lead to file corruption.

See also

- ♦ "DatabaseName connection parameter [DBN]" on page 188
- ♦ "How connection parameters work" on page 40
- ♦ "Connection parameter tips" on page 64

Example

- The DatabaseFile (DBF) connection parameter in the following example loads and connects to the sample database, *asademo.db*, installed in the directory *c:\Program Files\Sybase\SQL Anywhere 9*:

  ```
  DBF=c:\Program Files\Sybase\SQL Anywhere 9\asademo.db
  ```

- The following two examples assume that you have started a database file named *cities.db*, and renamed the database Kitchener as follows:

  ```
  dbeng9 cities.db -n Kitchener
  ```

  - To successfully start and connect to a database and name it Kitchener:

    ```
    DBN=Kitchener;DBF=cities.db
    ```

  - Specifying `DBF=cities.db` would fail to connect to the running database named Kitchener.

DatabaseKey connection parameter [DBKEY]

Function	To start an encrypted database with a connect request.
Usage	Anywhere
Values	*String*
Default	None

Description

You must specify this parameter when you start an encrypted database with a connect request. You do not need to specify this parameter if you are connecting to an encrypted database that is already running.

The encryption key is a string, including mixed cases, numbers, letters, and special characters.

If you want to secure communication packets between client applications and the database server use the -ec server option and transport-layer security.

☞ For more information, see "Adaptive Server Anywhere Transport-Layer Security" [*SQL Anywhere Studio Security Guide*, page 27].

See also

- "-ek database option" on page 171
- "-ep server option" on page 138
- "How connection parameters work" on page 40
- "Connection parameter tips" on page 64

Example

The following fragment illustrates the use of the DatabaseKey (DBKEY) connection parameter:

```
"UID=dba;PWD=sql;ENG=myeng;DBKEY=V3moj3952B;DBF=C:\Program
         Files\Sybase\SQL Anywhere 9\asademo.db"
```

DatabaseName connection parameter [DBN]

Function	For use when connecting to a database that is already running. Identifies a loaded database to which a connection needs to be made. If you want to connect to a database that is not already running, use the DatabaseFile (DBF) parameter.
Usage	Running local databases or network servers.
Values	*String*
Default	There is no default setting.

Description

Whenever a database is started on a server, it is assigned a database name, either by the administrator using the -n option, or by the server using the base of the filename with the extension and path removed.

> **Note**
> The DatabaseName (DBN) connection parameter is recommended for naming databases, rather than using the -n option with the DatabaseSwitches (DBS) connection parameter.

If the database you want to connect to is already running, you should specify the database name rather than the database file.

A connection will only occur if the name of the running database matches the name that is specified in the DatabaseName (DBN) parameter.

> **Note**
> If you specify both the database file and the database name, an attempt is made to connect to a running database with the specified name (the database file is ignored), and if that fails, an attempt is made to autostart a database using both the database file and database name.

See also

- "How connection parameters work" on page 40
- "Connection parameter tips" on page 64

Example

- To start a database file named *cities.db* and rename the database Kitchener, you can use the following command:

```
dbeng9 cities.db -n Kitchener
```

- Assuming you have run the above command, you can successfully connect to the running database named Kitchener as follows:

```
DBN=Kitchener
```

- Alternatively, you could use the following to successfully connect to the running database named Kitchener:

  ```
  DBN=Kitchener;DBF=cities.db
  ```

- However, specifying the following would fail to connect to the database named Kitchener:

  ```
  DBF=cities.db
  ```

DatabaseSwitches connection parameter [DBS]

Function

To provide database-specific options (switches) when starting a database.

Usage

Connecting to a server when the database is not loaded. This connection parameter autostarts a server with the specified database and options if a server is not already running.

Values

String

Default

No options

Description

You should supply DatabaseSwitches only if you are connecting to a database that is not currently running. When the server starts the database specified by DatabaseFile, the server uses the supplied DatabaseSwitches to determine startup options for the database.

Only *database* options (switches) can be supplied using this parameter. Server options must be supplied using the StartLine connection parameter.

☞ For information about database options, see "Database options" on page 171.

Note
The DatabaseName (DBN) connection parameter is recommended for naming databases, rather than using the -n option with the DatabaseSwitches (DBS) connection parameter.

See also

- "The database server" on page 116
- "StartLine connection parameter [START]" on page 204
- "How connection parameters work" on page 40
- "Connection parameter tips" on page 64

Example

- The following command, entered all on one line at a command prompt, connects to the default database server, loads the database file *asademo.db* (DatabaseFile (DBF) connection parameter), names it my_db (DatabaseName (DBN) connection parameter) and starts it in read-only mode (-r option).

```
dbisql -c "uid=DBA;pwd=SQL;dbf=c:\Program Files\Sybase\SQL
        Anywhere 9\asademo.db;dbn=my_db;dbs=-r"
```

DataSourceName connection parameter [DSN]

Function	Tells the ODBC driver manager or embedded SQL library where to look in the *.odbc.ini* file or registry to find ODBC data source information.
Usage	Anywhere
Values	*String*
Default	There is no default data source name.

Description

It is common practice for ODBC applications to send only a data source name to ODBC. The ODBC driver manager and ODBC driver locate the data source, which contains the remainder of the connection parameters.

In Adaptive Server Anywhere, embedded SQL applications can also use ODBC data sources to store connection parameters.

See also

- ♦ "FileDataSourceName connection parameter [FILEDSN]" on page 195
- ♦ "How connection parameters work" on page 40
- ♦ "Connection parameter tips" on page 64

Example

- ♦ The following parameter uses a data source name:

```
DSN=Dynamo Demo
```

DisableMultiRowFetch connection parameter [DMRF]

Function	To turn off multi-row fetches across the network
Usage	Anywhere
Values	**YES**, **NO**
Default	**NO**

Description

By default, when the database server gets a simple fetch request, the application asks for extra rows. You can disable this behavior by setting this parameter to **ON**.

☞ For more information, see "Using cursors in procedures and triggers" [*ASA SQL User's Guide*, page 692].

Setting the **DisableMultiRowFetch (DMRF)** connection parameter to **ON** is equivalent to setting the **PREFETCH** database option to **OFF**.

☞ For more information, see "Prefetching rows" [*ASA Programming Guide*, page 42].

See also	♦ "How connection parameters work" on page 40 ♦ "Connection parameter tips" on page 64
Example	♦ The following connection string fragment prevents prefetching: `DMRF=YES`

EncryptedPassword connection parameter [ENP]

Function	To provide a password, stored in an encrypted fashion in a data source.
Usage	Anywhere
Values	*String*
Default	None
Description	Data sources are stored on disk as a file or in the registry. Storing passwords on disk may present a security problem. For this reason, when you enter a password into a data source, it is stored in an encrypted form. If both the Password (PWD) connection parameter and the EncryptedPassword (ENP) connection parameter are specified, Password (PWD) takes precedence.
See also	♦ "How connection parameters work" on page 40 ♦ "Connection parameter tips" on page 64

Encryption connection parameter [ENC]

Function	To encrypt packets sent between the client application and the server using transport-layer security or simple encryption.
Usage	For **ECC_TLS** (Certicom), **RSA_TLS**, **RSA_TLS**, **RSA_TLS_FIPS**, TCP/IP only. For **NONE** or **SIMPLE**, anywhere.
Values	*String*
Default	**NONE** If an Encryption value is not set, encryption is controlled by the setting on the server. ☞ For information about the server's encryption setting, see "-ec server option" on page 135.
Description	You can use this parameter if you want to secure communications between client applications and the database server using transport-layer security.

☞ For more information, see "Adaptive Server Anywhere Transport-Layer Security" [*SQL Anywhere Studio Security Guide,* page 27].

Separately licensable option required
Transport-layer security requires that you obtain the separately-licensable SQL Anywhere Studio security option and is subject to export regulations.

☞ To order this component, see "Separately-licensable components" [*Introducing SQL Anywhere Studio,* page 5].

The Encryption (ENC) connection parameter accepts the following arguments:

- ♦ **none** accepts communication packets that are not encrypted. This value is equivalent to NO in previous versions of Adaptive Server Anywhere.

- ♦ **simple** accepts communication packets that are encrypted with simple encryption supported on all platforms and on previous versions of Adaptive Server Anywhere. This value is equivalent to YES in previous versions of Adaptive Server Anywhere. Simple encryption does not provide server authentication, strong elliptic-curve or RSA encryption, or other features of transport-layer security.

- ♦ **ECC_TLS** accepts communication packets that are encrypted using elliptic-curve based Certicom encryption technology. For backwards compatibility, ecc_tls can also be specified as **CERTICOM**. To use this type of encryption, both the server and the client must be operating on Solaris, Linux, NetWare, Mac OS X, or any supported Windows operating system except Windows CE, and the connection must be over the TCP/IP port. UNIX platforms, except for Solaris, Linux, and Mac OS X, do not recognize the client or server ECC_TLS parameter. The client can use the following arguments to verify the field values in the server's public certificate:
  - **trusted_certificates** specify the certificate file the client uses to authenticate the server. This is the only required parameter.
  - **certificate_company** specify the value for the organization field of the certificate.
  - **certificate_unit** specify the value for the organization unit field of the certificate.
  - **certificate_name** specify the certificate's common name.

    ☞ For more information about verifying certificate fields for server authentication, see "Verifying certificate fields" [*SQL Anywhere Studio Security Guide,* page 42].

    ☞ For more information about using digital certificates, see "Creating digital certificates" [*SQL Anywhere Studio Security Guide,* page 31].

- ♦ **RSA_TLS** accepts communication packets that are encrypted using RSA encryption technology. To use this type of encryption, both the server and the client must be operating on Solaris, Linux, NetWare, Mac OS X, or any supported Windows operating system except Windows CE, and the connection must be over the TCP/IP port. UNIX platforms, except for Solaris, Linux, and Mac OS X, do not recognize the client or server RSA_TLS parameter. The client can use the following arguments to verify the field values in the server's public certificate:
  - **trusted_certificates** specify the certificate file the client uses to authenticate the server. This is the only required parameter.
  - **certificate_company** specify the value for the organization field of the certificate
  - **certificate_unit** specify the value for the organization unit field of the certificate.
  - **certificate_name** specify the certificate's common name.

    ☞ For more information about verifying certificate fields for server authentication, see "Verifying certificate fields" [*SQL Anywhere Studio Security Guide*, page 42].

    ☞ For more information about using digital certificates, see "Creating digital certificates" [*SQL Anywhere Studio Security Guide*, page 31].

- ♦ **RSA_TLS_FIPS** accepts communication packets that are encrypted using FIPS-approved RSA encryption technology. RSA_TLS_FIPS uses a separate approved library, but is compatible with servers specifying RSA_TLS with Adaptive Server Anywhere 9.0.2 or later. To use this type of encryption, both the client and the server must be operating on a supported 32-bit Windows operating system, and the connection must be over the TCP/IP port. The client can use the following arguments to verify the field values in the server's public certificate:
  - **trusted_certificates** specify the certificate file the client uses to authenticate the server. This is the only required parameter.
  - **certificate_company** specify the value for the organization field of the certificate.
  - **certificate_unit** specify the value for the organization unit field of the certificate.
  - **certificate_name** specify the certificate's common name.

    ☞ For more information about verifying certificate fields for server authentication, see "Verifying certificate fields" [*SQL Anywhere Studio Security Guide*, page 42].

    ☞ For more information about using digital certificates, see "Creating digital certificates" [*SQL Anywhere Studio Security Guide*, page 31].

If you are using FIPS-approved RSA encryption, you must generate your certificates using the RSA cipher.

☞ For information about certificates, see "Creating digital certificates" [*SQL Anywhere Studio Security Guide*, page 31].

You can use the connection_property system function to retrieve the encryption settings for the current connection. The function returns one of five values: none, simple, ecc_tls, rsa_tls, or rsa_tls_fips depending which type of encryption is being used.

☞ For information about using the connection_property system function, see "CONNECTION_PROPERTY function [System]" [*ASA SQL Reference*, page 117].

See also

- ♦ "Configuring client applications to use transport-layer security" [*SQL Anywhere Studio Security Guide*, page 41]
- ♦ "-ec server option" on page 135
- ♦ "How connection parameters work" on page 40
- ♦ "Connection parameter tips" on page 64

Examples

The following connection string fragment connects to a database server myeng with a TCP/IP link, using transport-layer security and elliptic-curve encryption:

```
"ENG=myeng;LINKS=tcpip;Encryption=ECC_TLS (trusted_
         certificates=sample.crt)"
```

The following connection string fragment connects to a database server myeng with a TCP/IP link, using transport-layer security and RSA encryption:

```
"ENG=myeng;LINKS=tcpip;Encryption=RSA_TLS (trusted_
         certificates=sample.crt)"
```

The following connection string fragment connects to a database server myeng with a TCP/IP link, using simple encryption:

```
"ENG=myeng;LINKS=tcpip;Encryption=SIMPLE"
```

EngineName connection parameter [ENG]

Function	Synonym for ServerName. The name of a running database server to which you want to connect.
Usage	Network servers or running personal servers.
Values	*String*
Default	The default local database server.

Description

EngineName is not needed if you want to connect to the default local database server.

You only need to supply an EngineName if more than one local database server is running, or if you want to connect to a network server. In the Connect dialog, and in the ODBC Administrator, this is the Server Name field.

If you are autostarting a server, you can provide a server name using this parameter.

The server name is interpreted according to the character set of the client machine. Multi-byte characters are not recommended in server names.

Names must be a valid identifier. Long server names are truncated to different lengths depending on the protocol.

Protocol	**Truncation Length**
UNIX shared memory	31 bytes
non-UNIX shared memory	40 bytes
TCP/IP	40 bytes
SPX	32 bytes
Named Pipes	8 bytes

Note
It is recommended that you include the EngineName parameter in connection strings for deployed applications. This ensures that the application connects to the correct server in case a machine is running multiple Adaptive Server Anywhere database servers and can help prevent timing-dependent connection failures.

See also

- ♦ "Identifiers" [*ASA SQL Reference*, page 7]
- ♦ "-n server option" on page 150
- ♦ "How connection parameters work" on page 40
- ♦ "Connection parameter tips" on page 64

Example

- ♦ Connect to a server named Guelph:

```
ENG=Guelph
```

FileDataSourceName connection parameter [FILEDSN]

Function

Tells the client library there is an ODBC file data source holding information about the database to which you want to connect.

Usage	Anywhere
Values	*String*
Default	There is no default name.
Description	File data sources hold the same information as ODBC data sources stored in the registry. File data sources can be easily distributed to end users so that connection information does not have to be reconstructed on each machine. Both ODBC and embedded SQL applications can use File data sources.
See also	♦ "DataSourceName connection parameter [DSN]" on page 190 ♦ "How connection parameters work" on page 40 ♦ "Connection parameter tips" on page 64

ForceStart connection parameter [FORCESTART]

Function	To start a server without attempting to connect to one.
Usage	Only with the db_start_engine function
Values	**YES**, **NO**
Default	**NO**
Description	By setting ForceStart=YES, the db_start_engine function starts a server without attempting to connect to one, even if there is one already running.
See also	♦ "db_start_engine function" [*ASA Programming Guide,* page 218] ♦ "How connection parameters work" on page 40 ♦ "Connection parameter tips" on page 64

Idle connection parameter [IDLE]

Function	To specify the connection's idle timeout period.
Usage	Anywhere except with TDS and Shared Memory connections. Shared Memory and TDS connections (including jConnect) ignore the Adaptive Server Anywhere Idle (IDLE) connection parameter.
Values	*Integer*
Default	240
Description	The Idle (IDLE) connection parameter applies only to the current connection. You can have multiple connections on the same server set to different timeout values. If no connection idle timeout value is set, the idle timeout value is controlled by the setting on the server, which defaults to 240 minutes. In case of a

conflict between timeout values, the connection timeout value supercedes any server timeout value whether specified or unspecified.

See also

- "-ti server option" on page 157
- "How connection parameters work" on page 40
- "Connection parameter tips" on page 64

Example

- The following connection string fragment sets the timeout value for this connection to 10 minutes:

```
"ENG=myeng;LINKS=tcpip;IDLE=10"
```

Integrated connection parameter [INT]

Function

To use the integrated login facility.

Usage

Anywhere

Values

YES, **NO**

Default

NO

Description

The Integrated (INT) connection parameter has the following settings:

- **YES** An integrated login is attempted. If the connection attempt fails and the LOGIN_MODE option is set to Mixed, a standard login is attempted.
- **NO** This is the default setting. No integrated login is attempted.

For a client application to use an integrated login, the server must be running with the LOGIN_MODE database option set to Mixed or Integrated.

See also

- "LOGIN_MODE option [database]" on page 665
- "How connection parameters work" on page 40
- "Connection parameter tips" on page 64

Example

- The following data source fragment uses an integrated login:

```
INT=YES
```

Language connection parameter [LANG]

Function

Specifies the language of the connection.

Usage

Anywhere

Values

The two-letter combination representing a language. For example, setting **LANG=DE** sets the default language to German.

Default	The language specified by (in order) the ASLANG environment variable, the dblang utility, or the installer.
Description	This connection parameter establishes the language for the connection. Any errors or warnings from the server are delivered in the specified language, assuming that the server supports the language. If no language is specified, the default language is used. The default language is the language specified by, in order, the ASLANG environment variable, the dblang utility, or the installer. ☞ For a list of language codes, see "Understanding the locale language" on page 331. This connection parameter only affects the connection. Messages returned from Adaptive Server Anywhere's various tools and utilities appear in the default language, while the messages returned from the server appear in the connection's language.
See also	♦ "How connection parameters work" on page 40 ♦ "Connection parameter tips" on page 64

LazyClose connection parameter [LCLOSE]

Function	Enabling this option causes the `CLOSE` *cursor-name* database request to be queued, and then sent to the server with the next database request. This eliminates a network request each time a cursor is closed.
Usage	Anywhere
Values	**YES**, **NO**
Default	**NO**
Description	When this parameter is enabled, cursors are not actually closed until the next database request. Any isolation level 1 cursor stability locks still apply to the cursor while the `CLOSE` *cursor-name* database request is queued. Enabling this option can improve performance if your: ♦ network exhibits poor latency ♦ application sends many cursor open and close requests Note that in rare circumstances, canceling the next request after the `CLOSE` *cursor-name* database request can leave the cursor in a state where it appears to be closed on the client side, but is not actually closed on the server side. Subsequent attempts to open another cursor with the same name will fail. Using LazyClose is not recommended if your application cancels requests frequently.

See also	♦ “How connection parameters work” on page 40 ♦ “Connection parameter tips” on page 64

LivenessTimeout connection parameter [LTO]

Function	To control the termination of connections when they are no longer intact.
Usage	Network server only. All platforms except non-threaded UNIX applications.
Values	*Integer* (in seconds)
Default	**120** If no LivenessTimeout value is set, the LivenessTimeout is controlled by the setting on the server, which defaults to 120 seconds.
Description	A **liveness packet** is sent periodically across a client/server TCP/IP or SPX communication protocol to confirm that a connection is intact. If the client runs for the LivenessTimeout period without detecting a liveness request or response packet, the communication is severed. Liveness packets are sent when a connection has not sent any packets for between one third and two thirds of the LivenessTimeout value. When there are more than 200 connections to a server, the server automatically calculates a higher LivenessTimeout value based on the stated LivenessTimeout value. This enables the server to handle a large number of connections more efficiently. Alternatively, you can set this parameter by entering its value in the LivenessTimeout text box of the Network tab of the ODBC Configuration dialog.
See also	♦ “How connection parameters work” on page 40 ♦ “Connection parameter tips” on page 64
Example	♦ The following sets a LivenessTimeout value of 10 minutes `LTO=600`

Logfile connection parameter [LOG]

Function	To send client error messages and debugging messages to a file.
Usage	Anywhere
Values	*String*
Default	No log file

Description

If you want to save client error messages and debugging messages in a file, use the Logfile (LOG) connection parameter.

If the file name includes a path, it is relative to the current working directory of the client application.

The LogFile (LOG) connection parameter is connection-specific, so from a single application you can set different LogFile arguments for different connections.

Typical log file contents are as follows:

```
Fri Oct 27 2000 11:45
Application information:
"HOST=NAME-PC;OS=Windows NT 4.0 (Service Pack
         5);PID=0x14b;THREAD=0x148;EXE=C:\ASA90\WIN32\
         DBPING.EXE;VERSION=9.0.0.1271;API=DBLIB;TIMEZONEADJUSTM
         ENT=-300"
Attempting to connect using:
UID=dba;PWD=***;ENG=name;DBG=YES;LOG=c:\temp\
         cli.out;LINKS=shmem,tcpip
Attempting to connect to a running server...
Trying to start SharedMemory link ...
    SharedMemory link started successfully
Attempting SharedMemory connection (no asasrv.ini cached
         address)
Failed to connect over SharedMemory
Trying to start TCPIP link ...
Loading wsock32.dll
Loading ws2_32.dll
TCP using Winsock version 2.0
My IP address is 172.31.142.196
My IP address is 127.0.0.1
    TCPIP link started successfully
Attempting TCPIP connection (address 172.31.143.196:2638 found
         in asasrv.ini cache)
Trying to find server at cached address 172.31.143.196:2638
         without broadcasting
    Server not found (no reply received)
Looking for server with name NAME
I am in a class B network
Sending broadcast to find server
Using broadcast address of: 172.31.255.255:2638
I am in a class A network
Sending broadcast to find server
Using broadcast address of: 127.255.255.255:2638
Found database server at address 172.31.142.196:2638
Found database server NAME on TCPIP link
Connected to server over TCPIP at address 172.31.142.196:2638
Writing server address 172.31.142.196:2638 to asasrv.ini cache
   Liveness timeout 120, liveness retransmit period 30
Connected to the server, attempting to connect to a running
         database...
Connected to database successfully
```

See also	♦ “How connection parameters work” on page 40 ♦ “Connection parameter tips” on page 64
Example	The following command line starts Interactive SQL, connecting to the ASA 9.0 Sample data source with a LogFile (LOG) connection parameter:

```
dbisql -c "DSN=ASA 9.0 Sample;LOG=d:\logs\test.txt"
```

Password connection parameter [PWD]

Function	To provide a password for a connection.
Usage	Anywhere
Values	*String*
Default	No password provided.
Description	Every user of a database has a password. The password must be supplied for the user to be allowed to connect to the database. Passwords have a maximum length of 255 bytes. The Password (PWD) connection parameter is not encrypted. If you are storing passwords in a data source, you should use the EncryptedPassword (ENP) connection parameter. Sybase Central and the Adaptive Server Anywhere ODBC configuration tool both use encrypted parameters. If both Password (PWD) connection parameter and the EncryptedPassword (ENP) connection parameter are specified, the Password (PWD) connection parameter takes precedence. Alternatively, you can set this parameter in the Password text box in the Connect dialog and ODBC Administrator dialog.
See also	♦ “EncryptedPassword connection parameter [ENP]” on page 191 ♦ “GRANT statement” [*ASA SQL Reference*, page 503] ♦ “Case sensitivity” [*ASA SQL User’s Guide*, page 486] ♦ “How connection parameters work” on page 40 ♦ “Connection parameter tips” on page 64
Example	♦ The following connection string fragment supplies the user ID **DBA** and password **SQL**.

```
UID=DBA;PWD=SQL
```

PrefetchBuffer connection parameter [PBUF]

Function	Set the maximum amount of memory for buffering rows, in kilobytes.
Usage	Anywhere

Values

Integer

Default

16 (Windows CE) **64** (all other platforms)

Description

The **PrefetchBuffer (PBUF)** connection parameter controls the memory allocated on the client to store prefetched rows. In some circumstances, increasing the number of rows prefetched from the database server by the client can improve query performance. You can increase the number of rows prefetched using the **PrefetchRows (PROWS)** and **PrefetchBuffer (PBUF)** connection parameters.

Increasing the **PrefetchBuffer (PBUF)** connection parameter increases the amount of memory used to buffer GET DATA requests. This may improve performance for some applications that process many GET DATA (SQLGetData) requests.

☞ For more information, see "PrefetchRows connection parameter [PROWS]" on page 203.

See also

- "How connection parameters work" on page 40
- "Connection parameter tips" on page 64

Examples

- The following connection string fragment could be used to determine if the PrefetchBuffer memory limit is reducing the number of rows prefetched.

```
...PrefetchRows=100;Debug=YES;Logfile=c:\client.txt
```

- The following string could be used to increase the memory limit to 256 K:

```
...PrefetchRows=100;PrefetchBuffer=256
```

PreFetchOnOpen connection parameter

Usage

ODBC

Values

YES, **NO**

Default

NO

Description

Enabling this option sends a prefetch request with a cursor open request, thereby eliminating a network request to fetch rows each time a cursor is opened. Columns must already be bound in order for the prefetch to occur on the open. Rebinding columns between the cursor open and the first fetch when using PrefetchOnOpen will cause reduced performance.

Calling ODBC's SQLExecute or SQLExecDirect on a query or stored procedure which returns a result set causes a cursor open.

Enabling this option can improve performance if your:

- network exhibits poor latency
- application sends many cursor open and close requests

PrefetchRows connection parameter [PROWS]

Function	Set the maximum number of rows to prefetch when querying the database.
Usage	Anywhere
Values	*Integer*
Default	**10**

Description

Increasing the number of rows prefetched from the database server by the client can improve performance on cursors that only fetch relative 0 or 1, with either single row or wide fetches. Wide fetches include embedded SQL array fetches and ODBC block fetches.

Improvements occur particularly under the following conditions:

- The application fetches many rows (several hundred or more) with very few absolute fetches.
- The application fetches rows at a high rate, and the client and server are on the same machine or connected by a fast network.
- Client/server communication is over a slow network, such as a dial-up link or wide area network.

The number of rows prefetched is limited both by the **PrefetchRows (PROWS)** connection parameter and the **PrefetchBuffer (PBUF)** connection parameter, which limits the memory available for storing prefetched rows.

☞ For more information, see "PrefetchBuffer connection parameter [PBUF]" on page 201.

See also

- "How connection parameters work" on page 40
- "Connection parameter tips" on page 64

Example

- The following connection string fragment sets the number of prefetched rows to 100:

```
...PrefetchRows=100;...
```

ServerName connection parameter [ENG]

This is a synonym for the **EngineName (ENG)** connection parameter.

☞ For more information, see "EngineName connection parameter [ENG]" on page 194.

StartLine connection parameter [START]

Function	To start a personal database server running from an application.
Usage	Embedded databases
Values	*String*
Default	No StartLine parameter

Description

You should supply a **StartLine (START)** connection parameter only if you are connecting to a database server that is not currently running. The **StartLine (START)** connection parameter is a command line to start a personal database server.

☞ For a detailed description of available command line options, see "The database server" on page 116.

See also

- "How connection parameters work" on page 40
- "Connection parameter tips" on page 64

Example

- The following data source fragment starts a personal database server with a cache of 8 Mb.

```
StartLine=dbeng9 -c 8M;dbf=asademo.db
```

Unconditional connection parameter [UNC]

Function	To stop a server using dbstop, even when there are connections to the server.
Usage	Anywhere
Values	**YES**, **NO**
Default	**NO**

Description

The dbstop utility shuts down a database server. If you specify `UNC=YES` in the connection string, the server is shut down even if there are active connections. If Unconditional is not set to YES, then the server is shut down only if there are no active connections.

See also

- "Stopping a database server using the dbstop command-line utility" on page 577
- "How connection parameters work" on page 40
- "Connection parameter tips" on page 64

Example

- The following command line shuts down the server unconditionally:

```
dbstop -c "UID=DBA;PWD=SQL;ENG=server-name;UNC=YES"
```

Userid connection parameter [UID]

Function	The user ID with which you log on to the database.
Usage	Anywhere
Values	*String*
Default	*None*
Description	You must always supply a user ID when connecting to a database.
See also	♦ “How connection parameters work” on page 40 ♦ “Connection parameter tips” on page 64
Example	♦ The following connection string fragment supplies the user ID **DBA** and password **SQL**:

```
uid=DBA;pwd=SQL
```

Network protocol options

Network protocol options (for both the client and the server) enable you to work around peculiarities of different network protocol implementations.

You can supply the network protocol options in the server command. For example:

```
dbsrv9 -x tcpip(PARM1=value1;PARM2=value2;. . .),SPX
```

From the client side, you enter the protocol options as the CommLinks (LINKS) connection parameter:

```
CommLinks=tcpip(PARM1=value1;PARM2=value2;. . .),SPX
```

If there are spaces in a parameter, the network protocol options must be enclosed in quotation marks to be parsed properly by the system command interpreter:

```
dbsrv9 -x "tcpip(PARM1=value 1;PARM2=value 2;...),SPX"
CommLinks="tcpip(PARM1=value 1;PARM2=value 2;...),SPX"
```

The quotation marks are required under UNIX if more than one parameter is given because UNIX interprets the semicolon as a command separator.

Boolean parameters are turned on with YES, ON, TRUE, or 1, and are turned off with any of NO, OFF, FALSE, and 0. The parameters are case insensitive.

The examples provided should all be entered on a single line; you can also include them in a configuration file and use the @ server option to invoke the configuration file.

TCP/IP, HTTP, HTTPS, and SPX protocol options

The options currently available for TCP/IP, HTTP, HTTPS, and SPX are as follows.

TCP/IP	HTTP & HTTPS	SPX
Broadcast [BCAST]	Certificate	BroadcastListener [BLISTENER]
BroadcastListener [BLISTENER]	Certificate_Password	DLL
ClientPort [CPORT]	DatabaseName [DBN]	DoBroadcast [DO-BROAD]
DLL	LocalOnly [LOCAL]	ExtendedName [ENAME]

TCP/IP	HTTP & HTTPS	SPX
DoBroadcast [DO-BROAD]	LogFile [LOG]	Host [IP]
Host [IP]	LogMaxSize [LSize]	RegisterBindery [REG-BIN]
LocalOnly [LOCAL]	LogOptions [LOpt]	SearchBindery [BIN-SEARCH]
LDAP [LDAP]	LogFormat [LF]	Timeout [TO]
MyIP [ME]	MaxConnections [Max-Conn]	
ReceiveBufferSize [RCVBUFSZ]	MaxRequestSize [Max-Size]	
SendBufferSize [SND-BUFSZ]	MyIP [ME]	
ServerPort [PORT]	ServerPort [PORT]	
TDS	Timeout [TO]	
Timeout [TO]		
VerifyServerName [VERIFY]		

Broadcast protocol option [BCAST]

Usage	TCP/IP
Values	*String* (in the form of an IP address)
Default	Broadcasts to all addresses on the same subnet
Description	BROADCAST specifies the IP address that should be used to send broadcast messages. The default broadcast address is created using the local IP address and subnet mask. The subnet mask indicates which portion of the IP address identifies the network, and which part identifies the host.

For example, for a subnet of 10.24.98.x, with a mask of 255.255.255.0, the default broadcast address would be 10.24.98.255.

BroadcastListener protocol option [BLISTENER]

Usage	SPX, TCP/IP, Server side

Values	**YES**, **NO**
Default	**YES**
Description	This option allows you to turn broadcast listening OFF for this port. Using **-sb 0** is the same as specifying `BroadcastListener=NO` on both TCP/IP and SPX.
See also	♦ "-sb server option" on page 156
Example	♦ Start a server that accepts both TCP/IP and SPX connections, but require that TCP/IP connections use the HOST protocol option:

```
dbsrv9 -x tcpip(BroadcastListener=NO),spx ...
```

Certificate protocol option

Usage	HTTPS
Values	*String*
Default	There is no default certificate name.
Description	This option allows you to specify the name of an encryption certificate. The password for this certificate must be specified with the Certificate_Password parameter.
Example	♦ Start a server that requires web connections to use a particular encryption certificate.

```
dbsrv9 -xs https(Certificate=cert.file;Certificate_
         Password=secret) ...
```

Certificate_Password protocol option

Usage	HTTPS
Values	*String*
Default	There is no default certificate password.
Description	This option allows you to specify the password that matches the encryption certificate specified by the Certificate parameter.
Example	♦ Start a server that requires web connections to use a particular encryption certificate.

```
dbsrv9 -xs https(Certificate=cert.file;Certificate_
         Password=secret) ...
```

ClientPort protocol option [CPORT]

Usage TCP/IP (Client side only)

Values *Integer*

Default Assigned dynamically per connection by the networking implementation. If you do not have firewall restrictions, it is recommended that you do not use this parameter.

Description This option is provided for connections across firewalls, as firewall software filters according to TCP/UDP port. It is recommended that you do not use this parameter unless you need to for firewall reasons.

The ClientPort option designates the port number on which the client application communicates using TCP/IP. You may specify a single port number, or a combination of individual port numbers and ranges of port numbers.

It is best to specify a list or a range of port numbers if you want to make multiple connections using a given Data Source or a given connect string. If you specify a single port number, then your application will be able to maintain only one connection at a time. In fact, even after closing the one connection, there is a several minute timeout period during which no new connection can be made using the specified port. When you specify a list and/or range of port numbers, the application keeps trying port numbers until it finds one to which it can successfully bind.

See also

- "Host protocol option [IP]" on page 212
- "DoBroadcast protocol option [DOBROAD]" on page 211
- "ServerPort protocol option [PORT]" on page 220
- "Connecting across a firewall" on page 87

Examples

- The following connection string fragment makes a connection from an application using port 6000 to a server named my_server using port 5000:

```
CommLinks=tcpip(ClientPort=6000;ServerPort=5000); ServerName=my_
        server
```

- The following connection string fragment makes a connection from an application that can use ports 5050 through 5060, as well as ports 5040 and 5070 for communicating with a server named my_server using the default server port:

```
CommLinks=tcpip(ClientPort=5040,5050-5060,5070);
gServerName=my_server
```

DatabaseName protocol option [DBN]

Usage	HTTP, HTTPS
Values	**AUTO**, **REQUIRED**, *database-name*
Default	**AUTO**

Description

Specifies the name of a database to use when processing web requests, or uses the REQUIRED or AUTO keyword to specify whether database names are required as part of the URI.

If this parameter is set to REQUIRED, the URI must specify a database name.

If this parameter is set to AUTO, the URI may specify a database name, but does not need to do so. If the URI contains no database name, the default database on the server is used to process web requests. Since the server must guess whether or not the URI contains a database name when set to AUTO, you should design your web site so as to avoid ambiguity.

If this parameter is set to the name of a database, that database is used to process all web requests. The URI must not contain a database name.

Example

♦ The following command starts two databases, but permits only one of them to be accessed via HTTP.

```
dbsrv9 -xs http(dbn=web) asademo.db web.db
```

DLL protocol option

Usage	TCP/IP, SPX (Windows 95/98/Me, Windows NT/2000/XP)
Values	*String*

Default

♦ For database servers on all Windows platforms, including Windows CE, the default is **ws2_32.dll** (Winsock 2.0).

♦ For clients on Windows NT/2000/XP, the default is **ws2_32.dll** (Winsock 2.0).

♦ For clients on Windows 95/98/Me and Windows CE, the default is **wsock32.dll** (Winsock 1.1).

Description

To support untested TCP/IP protocol stacks where the required networking interface functions are in DLLs that differ from the default protocol stack. The client or server looks for its required functionality in the named DLLs.

Winsock 2.0 is required for database servers on all Windows platforms.

Example

- The following command on Windows 98 uses Winsock 2.0:

```
dbping -c "eng=my_eng;links=tcpip(dll=ws2_32.dll)"
```

DoBroadcast protocol option [DOBROAD]

Usage	TCP/IP, SPX
Values	**ALL**, **NONE**, **DIRECT** (Client side)
	YES, **NO** (Server side)
Default	**ALL** (Client side), **YES** (Server side)

Description

Client usage With `DoBroadcast=ALL` (formerly `DoBroadcast=YES`) a broadcast is performed to search for a server. The broadcast goes first to the local subnet. If `HOST=` is specified, broadcast packets are also sent to each of the hosts. For TCP, all broadcast packets are UDP packets. For SPX, the broadcast is only performed if the server is not found in the bindery. For SPX, all broadcast packets are IPX packets.

With `DoBroadcast=DIRECT` (formerly `DoBroadcast=NO`), no broadcast is performed to the local subnet to search for a database server. Broadcast packets are sent only to the hosts listed in the HOST (IP) protocol option. If you specify `DoBroadcast=DIRECT`, the HOST (IP) protocol option is required.

Specifying `DoBroadcast=NONE` causes no UDP or IPX broadcasts to be used and the server address cache (*asasrv.ini*) is ignored. A TCP/IP or SPX connection is made directly with the HOST/PORT specified, and the server name is verified. With TCP/IP, you can choose not to verify the server name by setting the VerifyServerName (VERIFY) protocol option to **NO**. The HOST (IP) protocol option is a required parameter, while the ServerPort (PORT) protocol option is optional.

For DIRECT and NONE, you must specify the server host with the HOST option.

Server usage Setting `DoBroadcast=NO` prevents the database server from broadcasting to find other servers with the same name when starting up. This is useful in certain rare circumstances, but it is not generally recommended.

Example

- The following command starts a client without broadcasting to search for a database server. Instead, the server is looked for only on the computer named silver.

```
CommLinks=tcpip(DOBROADCAST=DIRECT;HOST=silver) asademo
```

ExtendedName protocol option [ENAME]

Usage	SPX (platforms other than Windows 95/98/Me or Windows NT/2000/XP)
Values	**YES**, **NO**
Default	**NO**

Description

According to the Novell standard for legal SAP names, the following characters are not allowed:

```
\ / : ; , * ? + -
```

If you start a server named **asademo-1**, the default behavior is to strip out the **-** and try to start a server **asademo1**. By turning on ExtendedName, the name is left untouched.

This is an SPX port parameter and is only useful when starting a server.

> **Caution**
> *Users should be wary of using this option as it is contrary to the SAP standard.*

Example

- The following command starts a NetWare server with the name asademo-1.

```
load dbsrv9.nlm -x spx(ExtendedName=YES) asademo-1
```

Host protocol option [IP]

Usage	TCP/IP, SPX (all platforms) Server and client sides
Values	*String*
Default	No additional machines.

Description

HOST specifies additional machines outside the immediate network to be searched by the client library. On the server, the search is carried out to avoid starting a server with a duplicate name.

For TCP/IP, the HOST value the *hostname* or a dot-separated IP address may be used. You may optionally specify a PORT value as well.

For SPX, an address of the form *a*:*b*:*c*:*d*:*e*:*f*/*g*:*h*:*i*:*j* is used, where *a*:*b*:*c*:*d*:*e*:*f* is the node number (Ethernet card address) of the server, and *g*:*h*:*i*:*j* is the network number.

The server prints addressing information to the database server window during startup if the -z option is used. In addition, the application writes this information to its logfile if LogFile is specified.

You can use a comma-separated list of addresses to search for more than one machine. You can also append a port number to an IP address, using a colon as separator. Alternatively, you can specify the host and server ports explicitly, as in `HOST=myhost;PORT=5000`.

To specify multiple values for a single parameter, use a comma-separated list. When you specify multiple ports and servers, you can associate a particular port with a specific server by specifying the port in the HOST (IP) protocol option instead of the PORT parameter.

IP and **HOST** are synonyms when using TCP/IP. For SPX, you must use **HOST**.

See also

- "ClientPort protocol option [CPORT]" on page 209

Examples

- The following connection string fragment instructs the client to look on the machines kangaroo and 197.75.209.222 (port 2369) to find a database server:

  ```
  LINKS=tcpip(IP=kangaroo,197.75.209.222:2369)
  ```

- The following connection string fragment instructs the client to look on the machines my_server and kangaroo to find a database server. A connection is attempted to the first host that responds.

  ```
  LINKS=tcpip(HOST=my_server,kangaroo;PORT=2639)
  ```

- The following connection string fragment instructs the client to look for a server on host1 running on port 1234 and for a server on host2 running on port 4567. The client does not look on host1 on port 4567 or on host2 on port 1234.

  ```
  LINKS=tcpip(HOST=host1:1234,host2:4567)
  ```

LDAP protocol option [LDAP]

Usage

TCP/IP

Values

YES, **NO**, or *filename*

Default

ON

The default *filename* is *asaldap.ini*

Description

Having the database server register itself with an LDAP server allows clients (and the Locate utility [dblocate]) to query the LDAP server. This allows clients running over a WAN or through a firewall to find servers without specifying the IP address. It also allows the Locate utility [dblocate] to find such servers.

Specifying **LDAP**=*filename* turns LDAP support on and uses the specified file as the configuration file. Specifying **LDAP**=*YES* turns LDAP support on and uses *asaldap.ini* as the configuration file.

You can hide the contents of the *asaldap.ini* file with simple encryption using the File Hiding utility.

☞ For more information, see "Hiding the contents of .ini files" on page 524.

LDAP is only used with TCP/IP.

See also	♦ "Connecting using an LDAP server" on page 90

LocalOnly protocol option [LOCAL]

Usage TCP/IP, HTTP, HTTPS

Values **YES**, **NO**

Default **NO**

Description The LocalOnly (LOCAL) protocol option allows a client to choose to connect only to a server on the local machine, if one exists. If no server with the matching server name is found on the local machine, a server will not be autostarted.

The LocalOnly (LOCAL) protocol option is only useful if `DoBroadcast=ALL` (the default) is also specified.

`LocalOnly=YES` uses the regular broadcast mechanism, except that broadcast responses from servers on other machines are ignored.

You can use the LocalOnly (LOCAL) protocol option with the server to restrict connections to the local machine. Connection attempts from remote machines will not find this server, and the Locate [dblocate] utility will not see this server. Running a server with the LocalOnly (LOCAL) protocol option set to YES allows the network server to run as a personal server without experiencing connection or CPU limits.

See also ♦ "Broadcast protocol option [BCAST]" on page 207

LogFile protocol option [LOG]

Usage HTTP, HTTPS

Values *Filename*

Default	None
Description	Specify the name of the file to which the database server is to write information about web requests.
See also	♦ "LogFormat protocol option [LF]" on page 215 ♦ "LogMaxSize protocol option [LSIZE]" on page 216 ♦ "LogOptions protocol option [LOPT]" on page 216

LogFormat protocol option [LF]

Usage	HTTP, HTTPS
Values	*Format-string*
Default	**@T - @W - @I - @P - "@M @U @V" - @R - @L - @E**
Description	This parameter controls the format of messages written to the log file and which fields appear in them. If they appear in the string, the current values are substituted for the following codes as each message is written.

- **@@** The @ character.
- **@B** Date and time that processing of the request started, unless the request could not be queued due to an error.
- **@C** Date and time that the client connected.
- **@D** Name of the database associated with the request.
- **@E** Text of the error message, if an error occurred.
- **@F** Date and time that processing of the request finished.
- **@I** IP address of the client.
- **@L** Length of the response, in bytes, including headers and body.
- **@M** HTTP request method.
- **@P** Listener port associated with the request.
- **@Q** Date and time that the request was queued for processing, unless the request could not be queued due to an error.
- **@R** Status code and description of the HTTP response.
- **@S** HTTP status code.
- **@T** Date and time that the current log entry was written.
- **@U** Requested URI.

- **@V** Requested HTTP version.
- **@W** Time taken to process the request (@F – @B), or 0.000 if the request was not processed due to an error.

See also	♦ “LogFile protocol option [LOG]” on page 214 ♦ “LogMaxSize protocol option [LSIZE]” on page 216 ♦ “LogOptions protocol option [LOPT]” on page 216

LogMaxSize protocol option [LSIZE]

Usage	HTTP, HTTPS
Values	*Size*
Default	**0**
Description	When the log file reaches the stated size, it is renamed and another log file is created. If LogMaxSize is zero, the log file size is unlimited.
See also	♦ “LogFile protocol option [LOG]” on page 214 ♦ “LogFormat protocol option [LF]” on page 215 ♦ “LogOptions protocol option [LOPT]” on page 216

LogOptions protocol option [LOPT]

Usage	HTTP, HTTPS
Values	**NONE**, **OK**, **INFO**, **ERRORS**, **ALL**, *status-codes*, **REQHDRS**, **RESHDRS**, **HEADERS**
Default	**ALL**

Description

The values available include keywords that select particular types of messages, and HTTP status codes. Multiple values may be specified, separated by commas.

The following keywords control which categories of messages are logged:

- **NONE** Log nothing.
- **OK** Log requests that complete successfully (20x HTTP status codes).
- **INFO** Log requests that return over or not modified status codes (30x HTTP status codes).
- **ERRORS** Log all errors (40x and 50x HTTP status codes)
- **ALL** Log all requests.

The following common HTTP status codes are also available. They can be used to log requests that return particular status codes:

♦ **C200**	OK
♦ **C400**	Bad request
♦ **C401**	Unauthorized
♦ **C403**	Forbidden
♦ **C404**	Not found
♦ **C408**	Request timeout
♦ **C501**	Not implemented
♦ **C503**	Service unavailable

In addition, the following keywords may be used to obtain more information about the logged messages:

- ♦ **REQHDRS** When logging requests, also write request headers to the log file.
- ♦ **RESHDRS** When logging requests, also write response headers to the log file.
- ♦ **HEADERS** When logging requests, also write both request and response headers to the log file (same as REQHDRS,RESHDRS).

See also

- ♦ "LogFile protocol option [LOG]" on page 214
- ♦ "LogFormat protocol option [LF]" on page 215
- ♦ "LogMaxSize protocol option [LSIZE]" on page 216

MaxConnections protocol option [MAXCONN]

Usage	HTTP, HTTPS
Values	*Size*
Default	**5** (personal server) or number of licensed connections (network server)
Description	The number of simultaneous connections accepted by the server. The value 0 indicates no limit.
See also	♦ "MaxRequestSize protocol option [MAXSIZE]" on page 218

MaxRequestSize protocol option [MAXSIZE]

Usage	HTTP, HTTPS
Values	*Size* in bytes
Default	100 kb
Description	The size of the largest request accepted by the server. If the size of a request exceeds this limit, the connection is closed and a `413 ENTITY TOO LARGE` response is returned to the client. This value limits only the size of the request, not that of the response. The value 0 disables this limit, but should be used with extreme caution. Without this limit, a rogue client could overload the server or cause it to run out of memory.
See also	♦ "MaxConnections protocol option [MAXCONN]" on page 217

MyIP protocol option [ME]

Usage

TCP/IP, HTTP, HTTPS

Values

String

Description

The MyIP (ME) protocol option is provided for machines with more than one network adapter.

Each adapter has an IP address. By default, the database server uses every network interface it finds. If you don't want your database server to listen on all network interfaces, specify the address of each interface you want to use in the MyIP (ME) protocol option.

If the keyword NONE is supplied as the IP number, no attempt is made to determine the addressing information. The NONE keyword is intended for clients on machines where this operation is expensive, such as machines with multiple network cards or remote access (RAS) software and a network card. It is not intended for use on the server.

Under Windows 95/98/Me or Windows NT/2000/XP, this option can be used multiple times for machines with multiple IP addresses.

Separate multiple IP addresses with commas.

Example

♦ The following command line (entered all on one line) instructs the server to use two network cards.

```
dbsrv9 -x tcpip(MyIP=192.75.209.12,192.75.209.32) "c:\
          Program Files\Sybase\SQL Anywhere 9\asademo.db"
```

♦ The following connection string fragment instructs the client to make no attempt to determine addressing information.

```
LINKS= tcpip(MyIP=NONE)
```

ReceiveBufferSize protocol option [RCVBUFSZ]

Usage	TCP/IP
Values	*Integer* in bytes
Default	Machine-dependent
Description	Sets the size for a buffer used by the TCP/IP protocol stack. You may want to increase the value if BLOB performance over the network is important.

RegisterBindery protocol option [REGBIN]

Usage	SPX
	Server side only.
Values	**YES**, **NO**
Default	**YES**
Description	The database server attempts to register its name with any active binderies on the network when loading the SPX link. To disable this name registration, set RegisterBindery to NO. In this case, the client library must be able to locate the database server over SPX by broadcasting packets.

SearchBindery protocol option [BINSEARCH]

Usage	SPX
Values	**YES**, **NO**
Default	**YES**
Description	With `SEARCHBINDERY=NO`, no NetWare bindery is searched for a database server.

SendBufferSize protocol option [SNDBUFSZ]

Usage	TCP/IP
Values	*Integer* in bytes
Default	Machine-dependent
Description	Sets the size for a buffer used by the TCP/IP protocol stack. You may want to increase the value if BLOB performance over the network is important.

ServerPort protocol option [PORT]

Usage	TCP/IP, HTTP, HTTPS
Values	*Integer*
Default	The default value for TCP/IP is **2638**. The default value for HTTP is **80**. The default value for HTTPS is **443**.

Description

The Internet Assigned Numbers Authority has assigned the Adaptive Server Anywhere database server port number **2638** to use for TCP/IP communications. However, other applications are not disallowed from using this reserved port, and this may result in an addressing collision between the database server and another application.

In the case of the database server, the ServerPort protocol option designates the port number on which to communicate using TCP/IP.

The database server always listens on UDP port 2638 on most operating systems, even if you specify a different port using a network protocol option. Hence, applications can connect to the database server without specifying a port number.

For a client, the ServerPort protocol option informs the client of the port or ports on which database servers are listening for TCP/IP communication. The client broadcasts to every port that is specified by the ServerPort (PORT) protocol option to find the server.

If you using a web server, by default, the database server listens on the standard HTTP and HTTPS ports of 80 and 443, respectively.

If you start a database server using TCP/IP port number 2638 (the default), then the server also listens to UDP port 2638. The database server listens to UDP ports and responds to requests on these ports so that clients can locate the database server by server name.

If the database server's TCP/IP port number is not 2638, then the server listens to the same UDP port as the TCP/IP port. On Mac OS X, HP-UX, and Tru64, the server does *not* listen to UDP port 2638, while servers on other platforms do additionally listen on UDP port 2638. On the Mac OS X, HP-UX, and Tru64 platforms, UDP port 2638 must remain available in case a second database server starts on TCP/IP port 2638.

Differences on Mac OS X, HP-UX, and Tru64

Mac OS X, HP-UX, and Tru64 do not allow multiple processes to bind to the same UDP port. When the database server is running on one of these platforms, it only listens to the specified UDP port or port 2638 if no port is specified.

This means that clients *must* specify the TCP/IP port number if the server is not using the default port (2638).

For example if the database server is started with the command `dbsrv9 -n MyASAServer asademo.db`, a client on the same subnet can find the server using the following connection parameters `ENG=MyASAServer;LINKS=tcpip`. If another server is started on Mac OS X, HP-UX, or Tru64 with the following command `dbsrv9 -n SecondASAServer -x tcpip(PORT=7777) asademo.db`, a client on the same subnet can find the server using the connection parameters `ENG=SecondASAServer;LINKS=tcpip(PORT=7777)`. Note that if the database server was running on a platform other than Mac OS X, HP-UX, or Tru64, then the client would not need to specify the PORT parameter.

Additionally, on Mac OS X, HP-UX, and Tru64, if an Adaptive Server Anywhere database server is already using port 2638, and a second network database server was started without the PORT protocol option, the second network server would fail to start. The reason for this is users need to know and specify the server's port number in their connection parameters. Personal servers start successfully, even if port 2638 is in use, because shared memory is normally used to connect to personal servers.

Example

The following example shows how to use the PORT protocol option to specify the port the server starts on.

1. Start a network database server:

   ```
   dbsrv9 -x tcpip -n server1
   ```

   Port number 2638 is now taken.

2. Attempt to start another database server:

   ```
   dbsrv9 -x tcpip -n server2
   ```

   The default port is currently allocated, and so the server starts on another port. (On Mac OS X, HP-UX, and Tru64, this will fail).

3. If another web server on your machine is already using port 80 or you do not have permission to start a server on this low of a port number, you may want to start a server that listens on an alternate port, such as 8080:

   ```
   dbsrv9 -xs http(port=8080) -n server3 web.db
   ```

TDS protocol option

Usage	TCP/IP, NamedPipes Server side only
Values	**YES**, **NO**
Default	**YES**
Description	To disallow TDS connections to a database server, set TDS to NO. If you want to ensure that only encrypted connections are made to your server, these port options are the only way to disallow TDS connections.

Example

♦ The following command starts a database server using the TCP/IP protocol, but disallowing connections from Open Client or jConnect applications.

```
dbsrv9 -x tcpip(TDS=NO) ...
```

Timeout protocol option [TO]

Usage	TCP/IP, SPX, HTTP, HTTPS
Values	*Integer*, in seconds
Default	**5** for TCP/IP, **30** for HTTP and HTTPS
Description	Timeout specifies the length of time, in seconds, to wait for a response when establishing communications. It also specifies the length of time to wait for a response when disconnecting. You may want to try longer times if you are having trouble establishing TCP/IP or SPX communications.

When using HTTP or HTTPS on the server, this parameter specifies the maximum idle time permitted when receiving a request. If this limit is reached, the connection is closed and a `408 REQUEST TIMEOUT` is returned to the client. The value 0 disables idle timeout, but should be used with extreme caution. Without this limit, a rogue client could consume the server's resources and prevent other clients from connecting.

Example

♦ The following data source fragment starts a TCP/IP communication link only, with a timeout period of twenty seconds.

```
...
CommLinks=tcpip(TO=20)
...
```

VerifyServerName protocol option [VERIFY]

Usage	TCP/IP

Client side only

Values **YES**, **NO**

Default **YES**

Description When connecting over TCP using the `DoBroadcast=NONE` parameter, the client makes a TCP connection, then verifies that the name of the server found is the same as the one it's looking for. Specifying `VerifyServerName=NO` skips the verification of the server name. This allows Adaptive Server Anywhere clients to connect to an Adaptive Server Anywhere server if they know only an IP address/port.

The server name must still be specified in the connection string, but it is ignored. The VerifyServerName (VERIFY) protocol option is used only if `DoBroadcast=NONE` is specified.

Note
It is recommended that you only use this parameter in the rare circumstances when it is not possible to give each server a unique server name, and use that unique name to connect. Giving each server a unique server name, and connecting to the server using that name is still the best way to connect.

See also
- "DoBroadcast protocol option [DOBROAD]" on page 211

CHAPTER 7

Using Web Services

About this chapter

Adaptive Server Anywhere contains a built-in HTTP server that allows you to provide web services, and to access web services in other Adaptive Server Anywhere databases and standard web services available over the internet. SOAP is the standard used for this purpose, but the built-in HTTP server in Adaptive Server Anywhere also lets you handle standard HTTP and HTTPS requests from client applications. This chapter describes how to create and use both types of web services.

Contents

About web services

The term *web service* has been used to mean a variety of things. Commonly, it refers to software that facilitates inter-machine data transfer and interoperability. Essentially, web services make segments of business logic avalable over the internet. The *Simple Object Access Protocol* (SOAP) is the standard protocol used in this application and may be incorporated into any application capable of communicating with XML over HTTP.

The acronym SOAP refers to the flexible message exchange protocol accessible from applications such as those written in Java or Visual Studio .NET. SOAP messages define the services that a server provides. Actual data transfer generally takes places using HTTP to exchange XML documents structured so as to efficiently encode relevant information. Any application, such as a client or server, that participates in SOAP communication is called a SOAP node or SOAP endpoint. Such applications may transmit, receive, or process SOAP messages. You can create SOAP nodes with Adaptive Server Anywhere.

Web services and SQL Anywhere

In the context of SQL Anywhere, the term web services means that Adaptive Server Anywhere has the ability to listen for and handle standard SOAP requests. Web services in Adaptive Server Anywhere provide client applications an alternative to such traditional intefaces as JDBC and ODBC. They can be accessed from client applications written in a variety of languages and running on a variety of platforms. Even common scripting languages such as Perl and Python provide access to web services. You create web services in a database using the CREATE SERVICE statement.

Adaptive Server Anywhere can also function as a SOAP or HTTP client, permitting applications running within the database access to standard web services available over the internet, or provided by other Adaptive Server Anywhere databases. This client functionality is accessed through stored functions and procedures.

In addition, the term also refers to applications that use the built-in web server to handle HTTP requests from clients. These applications generally function like traditional database-backed web applications, but can be more compact and are easier to write as the data and the entire application can reside within a database. In this application, the web service typically returns documents in HTML format. GET, HEAD, and POST methods are supported.

The collection of web services within your database together define the available URLs. Each service provides a set of web pages. Typically, the content of these pages is generated by procedures that you write and store in your database, although they can be a single statement or, optionally, allow users to execute statements of their own. These web services become available when you start the database server with switches that enable it to listen for HTTP requests.

Since the HTTP server that handles web service requests is embedded in the database, performance is good. Applications that use web services are easily deployed, since no additional components are needed, beyond the database and database server.

☞ For more information about the SOAP standards, see www.w3.org/TR/SOAP.

Quick start

The following procedure describes how to create a new database, start an Adaptive Server Anywhere database with the HTTP server enabled, and access this database using any popular web browser.

❖ To create and access a simple XML web service

1. To create a new database to be used for this demonstration, execute the following statement at a command prompt.

   ```
   dbinit web.db
   ```

2. Execute the following statement to start a personal web server. The `-xs http(port=80)` switch tells the database engine to listen for HTTP requests. If you already have a web server running on port 80, use another port number such as 8080 for this demonstration.

   ```
   dbeng9 -xs http(port=80) web.db
   ```

3. Start Interactive SQL. Connect to the web database using the user name DBA and password SQL. Execute the following statement.

   ```
   CREATE SERVICE mytables
   TYPE 'XML'
   AUTHORIZATION OFF
   USER DBA
   AS SELECT * FROM SYSTABLE
   ```

   This statement creates a web service named mytables. This simple service returns the results of the statement SELECT * FROM SYSTABLE, automatically converting the output into XML format.

4. Start a web browser. Browse to the URL `http://localhost:80/web/mytables`. Use the port number you specified when starting the database server.

   Your web browser will show you the body of the XML document returned by the database server. As no formatting information has been included, you see the raw XML, including tags and attributes.

5. You can also access the mytables routine from common programming languages. For example, the following short C# program uses the mytable web service:

```
static void Main(string[] args) {
  XmlTextReader reader = new XmlTextReader(
         "http://localhost/mytables" );

  while( reader.Read() ) {
    switch( reader.NodeType ) {
    case XmlNodeType.Element:
      if( reader.Name == 'row' ) {
        Console.writeline(reader.GetAttribute("table_id");
        Console.writeline(reader.GetAttribute("table_name");
      }
      break;
    }
  }
```

6. In addition, you can access the same web service from Python, as in the following example code:

```
import xml.sax

class DocHandler( xml.sax.ContentHandler ):
  def startElement( self, name, attrs ):
    if name == 'row':
      table_id = attrs.getValue( 'table_id' )
      table_name = attrs.getValue( 'table_name' )
      print '%s %s' % ( table_id, table_name )

parser = xml.sax.makd_parser()
parser.setContentHandler( DocHandler() )
parser.parse( 'http://localhost/mytables' )
```

❖ To set up a simple HTML web service

1. Ensure that you have no web server already running on port 80, or choose a different port number to use for the rest of this procedure.
2. Navigate to a convenient directory and create a new database named web.

```
dbinit web.db
```

3. Start a personal database server to run this new database. The -xs option tells the database server to start all communication protocols, including HTTP.

```
dbeng9 web.db -xs http(port=80)
```

4. Start Interactive SQL and connect to your new database using the user ID **DBA** and the password **SQL**.
5. Using Interactive SQL, execute the following SQL statement, which adds an HTML service that exposes the SYS.SYSTABLE to HTTP users.

Because authorization is off, no permissions are required to access the table from a web browser.

```
CREATE SERVICE tables TYPE 'HTML'
AUTHORIZATION OFF USER DBA
AS SELECT * FROM SYS.SYSTABLE
```

6. Start a web browser. Browse to the URL `http://localhost:80/web/tables`. Use the port number you specified when starting the database server.

Many properties of the service are controlled by parameters to the -xs command-line option.

☞ For more information, see "Network protocol options" on page 206.

As well as starting the database server with the appropriate -xs option parameters, you must create web services to respond to incoming requests. These are defined using the CREATE SERVICE statement.

☞ For more information, see "Creating web services" on page 231 and "CREATE SERVICE statement" [*ASA SQL Reference,* page 395].

Examples

Samples are included in the *Samples\ASA\HTTP* subdirectory of your SQL Anywhere Studio installation.

Other examples are available on CodeXchange at *http://ianywhere.codexchange.sybase.com/*.

Creating web services

Services, created and stored in databases, define which URLs are valid and what they do. A single database can define multiple services. It is possible to define services in different databases so that they appear to be part of a single web site.

The following statements permit you to create, alter, and delete web services:

- **CREATE SERVICE**
- **ALTER SERVICE**
- **DROP SERVICE**
- **COMMENT ON SERVICE**

The general syntax of the CREATE SERVICE statement is as follows:

CREATE SERVICE *service-name* **TYPE** *service-type-string* [*attributes*] [**AS** *statement*]

Service names

As service names will form part of the URL used to access them, they are flexible in terms of what characters they can contain. In addition to the standard alpha-numeric characters, the following characters are permitted: "-", "_", ".", "!", "*", " ' ", "(", and ")".

In addition, service names may contain the slash, "/", but some restrictions apply because this character is a standard URL delimiter and affects how Adaptive Server Anywhere interprets your URLs. It cannot be the first character of a service name. In addition, service names cannot contain two consecutive slashes.

The same characters are permitted in group names, which apply to DISH services, as to service names.

Service types

The following service types are supported:

- **SOAP** The result set is returned as a SOAP response. The format of the data is determined by the FORMAT clause. A request to a SOAP service must be a valid SOAP request, not just a simple HTTP request.
- **DISH** A DISH service (Determine SOAP Handler) acts as a proxy for those SOAP services identified by the GROUP clause, and generates a WSDL (Web Services Description Language) file for each of these SOAP services.
- **XML** The result set is returned as XML. If the result set is already XML, no additional formatting is applied. If it is not already XML, it is

automatically formatted as XML. The effect is similar to that of using the FOR XML RAW clause in a SELECT statement.

- **HTML** The result set of a statement or procedure are automatically formatted into an HTML document that contains a table.

- **RAW** The result set is sent to the client without any further formatting. You can produce formatted documents by generating the required tags explicitly within your procedure.

Of all the service types, RAW gives you the most control over the output. However, it does require that you do more work as you must explicitly output all the necessary tags. The output of XML services can be adjusted by applying the FOR XML clause to the service's statement. The output of SOAP services can be adjusted using the FORMAT attribute of the CREATE or ALTER SERVICE statement.

☞ For more information, see "CREATE SERVICE statement" [*ASA SQL Reference*, page 395].

Statements

The statement is the command, usually a stored procedure, that is called when someone accesses the service. If you define a statement, this is the only statement that can be run through this service. The statement is mandatory for SOAP services, and ignored for DISH services. The default is NULL, which means no statement.

You can create services that do not include statements. The statement is taken from the URL. Services configured in this way can be useful when you are testing a service, or want a general way of accessing information. To do so, either omit the statement entirely or use the phrase AS NULL in place of the statement.

Services without statements are a serious security risk as they permit web clients to execute arbitrary commands. When creating such services, you must enable authorization, which forces all clients to provide a valid username and password. Even so, only services that define statements should be run in a production system.

Attributes

The following attributes are available. In general, all are optional. However, some are interdependent.

- **AUTHORIZATION** This attribute controls which users can use the service. The default setting is ON. Authorization must be ON if no statement is provided. In addition, the authorization setting affects how user names, defined by the USER attribute, are interpreted.

- **SECURE** When set to ON, only secure connections are permitted. All connections received on the HTTP port are automatically redirected to the HTTPS port. The default is OFF, which enables both HTTP and HTTPS requests, provided these ports are enabled using the appropriate options when the database server is started.

♦ **USER** The USER clause controls which database user accounts can be used to process service requests. However, the interpretation of this setting depends on whether authorization is ON or OFF.

  When authorization is set to ON, all clients must provide a valid user name and password when they connect. When authorization is ON, the USER option may be NULL, a database user name, or the name of a database group. If it is NULL, any database user can connect and make requests. Requests are run using the account and permissions of that user. If a group name is specified, only those users who belong to the group can run requests. All other database users are denied permission to use the service.

  If authorization is OFF, a statement must be provided. In addition, a user name must be provided. All requests are run using that user's account and permissions. Thus, if the server is connected to a public network, the permissions of the named user account should be minimal to limit the damage that could be caused through malicious use.

♦ **GROUP** The GROUP clause, which applies to DISH services only, determines which SOAP services are exposed by the DISH service. Only SOAP services whose names begin with the name of the group name of a DISH service are exposed by that DISH service. Thus, the group name is a common prefix among the exposed SOAP services. For example, specifying GROUP *xyz* exposes only SOAP services *xyz/aaaa*, *xyz/bbbb*, or *xyz/cccc*, but does not expose *abc/aaaa* or *xyzaaaa*. If no group name is specified, the DISH service exposes all the SOAP services in the database. The same characters are permitted in group names as in service names.

  SOAP services may be exposed by more than one DISH service. In particular, this feature permits a single SOAP service to supply data in multiple formats. The service type, unless specified in a SOAP service, is inherited from the DISH service. Thus, you can create a SOAP service that declares no format type, then include it in multiple DISH services, each of which specifies a different format.

♦ **FORMAT** The FORMAT clause, which applies to DISH and SOAP services only, controls the output format of the SOAP or DISH response. Output formats compatible with various types of SOAP clients, such as .NET or Java JAX-RPC, are available. If the format of a SOAP service is not specified, the format is inherited from the service's DISH service declaration. If the DISH service also does not declare a format, it defaults to DNET, which is compatible with .NET clients. A SOAP service that does not declare a format may be used with different types of SOAP clients by defining multiple DISH services, each having a different FORMAT type.

- **URL** The URL clause controls the interpretation of URLs and applies to XML, HTML, and RAW service types only. In particular, it determines whether URL paths are accepted and, if so, how they are processed. If the service name ends with the character "/", the URL must be set to OFF.

  ☞ For more information, see "CREATE SERVICE statement" [*ASA SQL Reference,* page 395].

Starting a database server that listens for web requests

When you want a database server to listen for web service requests over HTTP or HTTPS, you must specify for which types of web requests it is to listen on the command line when you start the server. By default, database servers do not listen for web service requests, leaving no way for clients to access any services that may be defined in your database.

You can also specify various properties of an HTTP or HTTPS service on the command line, such as on which port they are to listen.

☞ You must also create web services within the database. For more information, see “Creating web services” on page 231.

You use the -xs option to enable protocols. The two available web service protocols are http and https. Optional parameters, placed within parentheses after the protocol name, let you customize access to each type of web service.

The general syntax of the option is as follows:

-xs { *protocol* [**(***option***=***value*; . . . **)**], . . . }

Protocols

The following web service protocol values are available:

- **http** Listen for HTTP connections.
- **https** Listen for HTTPS connections.
- **https_fips** Listen for HTTPS FIPS connections.
- **none** Do not listen for web service requests. This is the default setting.

Options

The following are some of the options that are available:

- **ServerPort [PORT]** The port on which to listen for web requests. By default, Adaptive Server Anywhere listens on port 80 for HTTP requests and on port 443 for secure HTTP (HTTPS) requests.

  For example, if you already have a web server running on port 80, you could use the following options to start the a database server that listens for web requests on port 8080:

  ```
  dbeng9 web.db -xs http(port=8080)
  ```

  As another example, the following command starts the secure server with the sample certificate included with SQL Anywhere. It should be entered on a single line.

```
dbsrv9 -xs https(certificate=rsaserver.crt;
                 certificate_password=test)
```

Caution
The sample certificate is intended for use only during testing and development. It provides no protection because it is a standard part of SQL Anywhere. Replace it with your own certificate before deploying your application.

- **DatabaseName [DBN]** Specifies the name of a database to use when processing web requests, or uses the REQUIRED or AUTO keyword to specify whether database names are required as part of the URL.

  If this parameter is set to REQUIRED, the URL must specify the database name.

  If this parameter is set to AUTO, the URL may specify a database name, but does not need to do so. If the URL contains no database name, the default database on the server is used to process web requests.

  If this parameter is set to the name of a database, that database is used to process all web requests. The URL must not contain a database name.

- **LocalOnly [LOCAL]** When set to YES, this parameter causes a network database server to reject all connections from clients running on different machines. This option has no effect on personal database servers, which never accept web service requests from other machines. The default value is NO, which means accept requests from clients no matter where they are located.

- **LogFile [LOG]** The name of the file to which the database server is to write information about web requests.

- **LogFormat [LF]** Controls the format of messages written to the log file and which fields appear in them. If they appear in the string, current values are substituted for the codes, such as @T, when each message is written.

  The default value is **@T - @W - @I - @P - “@M @U @V" - @R - @L - @E**, which produces messages such as the following:

  ```
  06/15 01:30:08.114 - 0.686 - 127.0.0.1 - 80
   - "GET /web/show_table HTTP/1.1" - 200 OK - 55133 -
  ```

  The format of the log file is compatible with Apache, so the same tools may be used to analyze it.

- **LogOptions [LOPT]** Allows you to specify keyword and error numbers that control which messages, or types of messages, are written to the log file.

  ☞ For more information, see "LogOptions protocol option [LOPT]" on page 216.

☞ For a complete list of the available options and detailed information about them, see "Network protocol options" on page 206.

How URLs are interpreted

Universal Resrouce Locators, or URLs, identify documents, such as HTML pages, available from SOAP or HTTP web services. Those used in Adaptive Server Anywhere follow the patterns familiar to you from browsing the web. Users browsing through a database server need not be aware that their requests are not being handled by a traditional stand-alone web server.

Athough standard in format, Adaptive Server Anywhere database servers interpret URLs differently than standard web servers. The options you specify when you start the database server also affect their interpretation.

The general syntax of the URL is as follows:

{ **http** | **https** } **://** [*user*:*password*@] *host* [**:***port*] [**/***dbn*] **/***service-name* [*parameters*] [**?***query*]

User and password When a web service requires authentication, the user name and password may be passed directly as part of the URL by separating them with a colon and prepending them to the host name, much like an e-mail address.

Host and port Like all standard HTTP requests, the start of the URL contains the host name or IP number and, optionally, a port number. The IP address or host name, and port, should be the one on which your server is listening. The IP address will be the address of a network card in the machine running Adaptive Server Anywhere. The port number will be the port number you specified using the -xs option when you started the database server. If you did not specify a port number, the default port number for that type of service is used. For example, the server listens by default on port 80 for HTTP requests.

☞ For more information, see "Network protocol options" on page 206.

Database name The next token, between the slashes, is usually the name of a database. This database must be running on the server and must contain web services.

The default database is used if no database name appears in the URL and the database name was not specified using the DBN connection parameter to the -xs option.

The database name can be omitted only if the database server is running only one database, or if the database name was specified using the DBN connection parameter to the -xs option.

Service name The next portion of the URL is the service name. This service must exist in the specified database. The service name may extend beyond the next slash character because web service names can contain slash characters. Adaptive Server Anywhere matches the remainder of the URL with the defined services.

If the URL provides no service name, the database server looks for a service named root. If the named service, or the root service, is not defined, the server returns a `404 Not Found` error.

Parameters

Depending on the type of the target service, parameters can be supplied in different ways. Parameters to HTML, XML, and RAW services can be passed in any of the following ways:

- appended to the URL
- supplied as an explicit parameters list
- supplied as POST data in a POST request

Parameters to SOAP services must be included as part of a standard SOAP request. Values supplied in other ways are ignored.

To access parameter values, parameters must be given names. These host variable names, prefixed with a colon (:), may be included in the statement that forms part of the web service definition.

For example, a statement that calls the procedure place_order may require product identification number and quantity parameters:

```
call place_order( :product, :quantity )
```

☞ Parameters may also be accessed using the HTTP_VARIABLE function. For more information, see "HTTP_VARIABLE function [HTTP]" [*ASA SQL Reference*, page 167].

Query

The query is a list of *name*=*value* pairs separated by the & character. This list is introduced by a question mark. GET requests are formatted in this manner. If present, the named variables are defined and assigned the corresponding values.

If you have URL PATH set to ON or ELEMENTS, additional variables are defined. However, the two are otherwise independent. You can allow variables to be used in requested URLs by setting PATH to ON, or ELEMENTS, or both.

☞ For more information about variables, see "Working with variables" on page 263.

Creating SOAP and DISH web services

SOAP and Web services are the means by which you create standard SOAP web services that can be accessed by standard SOAP clients, such as those written with Microsoft .NET or Java JAX-RPC.

SOAP services

SOAP services are the mechanism for constructing web services in SQL Anywhere that accept and process standard SOAP requests.

To declare a SOAP service, specify that the service is to be of type SOAP. The body of a standard SOAP request is SOAP envelope, meaning an XML document with a specific format. Adaptive Server Anywhere parses and processes these requests using the procedures you provide. The response is automatically formatted in the form of a standard SOAP response, which is also an SOAP envelope, and returned to the client.

The syntax of the statement used to create SOAP services is as follows:

CREATE SERVICE *service-name*
TYPE ’SOAP’
[**FORMAT** { **’DNET’** | **’CONCRETE’** | **’XML’** | **NULL** }]
[*common-attributes*]
AS *statement*

DISH services

Dish services act as proxies for groups of SOAP services. In addition, they automatically construct WSDL documents that describe the SOAP services that they currently expose.

When you create a DISH service, the name given in the GROUP clause determines which SOAP services that DISH service exposes. Every SOAP service whose name is prefixed with the name of the DISH service is exposed. For example, specifying GROUP xyz exposes SOAP services xyz/aaaa, xyz/bbbb, or xyz/cccc. It does not expose SOAP services named abc/aaaa or xyzaaaa. SOAP services may be exposed by more than one DISH service. If no group name is specified, the DISH service exposes all the SOAP services in the database. The same characters are permitted in group names as in service names.

The syntax of the statment used to create DISH services is as follows:

CREATE SERVICE *service-name*
TYPE ’DISH’
[**GROUP** { *group-name* | **NULL** }]
[**FORMAT** { **’DNET’** | **’CONCRETE’** | **’XML’** | **NULL** }]
[*common-attributes*]

SOAP and DISH service formats

The FORMAT clause of the CREATE SERVICE statement customizes the SOAP service data payload to best suit the various types of SOAP clients, such as .NET and Java JAX-RPC. The FORMAT clause affects the content of the WDSL document returned by a DISH service and the format of data payloads returned in SOAP responses.

The default format, DNET, is a native format for use with .NET SOAP client applications, which expect a .NET DataSet format.

The CONCRETE format is for use with clients such as Java JAX-RPC and .NET that automatically generate interfaces based on the format of the returned data structures. When you specify this format, the WSDL document returned by Adaptive Server Anywhere exposes an ASADataset element that describes a result set in concrete terms. This element is a containment hierarchy of a rowset composed of an array of rows, each containing an array of column elements.

The XML format is for use with SOAP clients that accept the SOAP response as one large string, and use an XML parser to locate and extract the required elements and values. This format is generally the most portable between different types of SOAP clients.

If the format of a SOAP service is not specified, the format is inherited from the service's DISH service declaration. If the DISH service also does not declare a format, it defaults to DNET, which is compatible with .NET clients. A SOAP service that does not declare a format may be used with different types of SOAP clients by defining multiple DISH services, each having a different FORMAT type.

Creating Homogeneous DISH services

SOAP services need not specify a format type—in other words, set the format type to NULL. When they do so, the format is inherited from the DISH service that act as proxies for them. More than one DISH service can act as a proxy for each SOAP service, and these DISH services need not be of the same type. These facts mean that it is possible to use a single SOAP service with different types of SOAP clients, such as .NET and Java JAX-RPC, by using multiple DISH services, each of a different type. Such DISH services are said to be *homogeneous* because they expose the same data payloads for the same SOAP services, but in different formats.

For example, consider the following two SOAP services, neither of which specifies a format:

```
CREATE SERVICE "abc/hello"
TYPE 'SOAP'
AS CALL hello(:student)

CREATE SERVICE "abc/goodbye"
TYPE 'SOAP'
AS CALL goodbye(:student)
```

Since neither of these services includes a FORMAT clause, the format, by default, is NULL. It is thus inherited from the DISH service that is acting as a proxy. Now, consider the following to DISH services:

```
CREATE SERVICE "abc_xml"
TYPE 'DISH'
GROUP "abc"
FORMAT 'XML'

CREATE SERVICE "abc_concrete"
TYPE 'DISH'
GROUP "abc"
FORMAT 'CONCRETE'
```

Since both DISH services specify the same group name, “abc”, they act as proxies for the same SOAP services, namely all SOAP services whose names have the prefix “abc/”.

However, when either of the two SOAP services is accessed through the abc_xml dish service, the SOAP service inherits the XML format type; when accessed throught he abc_concrete SOAP service, the CONCRETE format.

Homogeneous DISH services provide a means of avoiding duplicate services whenever you want to give types of SOAP clients access to the SOAP web services you create.

Tutorial: Accessing web services from Java JAX-RPC

The following tutorial demonstrates how to access web services from Java JAX-RPC. To access SQL Anywhere SOAP web services from JAX-RPC, the service should be declared to be of format CONCRETE.

❖ Create SOAP and DISH services.

1. From a command prompt, run the following command to create a new database to be used for this tutorial.

   ```
   dbinit jaxrpc.db
   ```

2. Start this new database. Use the -xs http flag so that the database server will listen for and handle HTTP requests. The `-xs http(port=80)` switch tells the database engine to accept HTTP requests. If you already have a web server running on port 80, use another port number such as 8080 for this tutorial.

   ```
   dbsrv9 -xs http(port=80) jaxrpc.db
   ```

3. Start Interactive SQL. Connect to the web database using the user name DBA and password SQL. Execute the following statements:

   a. Create a simple hello procedure to be used by the SOAP service.

      ```
      CREATE PROCEDURE "hello_proc"(IN user_name char(64))
      RESULT (result_out long varchar)
      BEGIN
         SELECT 'hello' || ' ' || user_name || '\n';
      END
      ```

   b. Define a SOAP service that calls the hello stored procedure.

      ```
      CREATE SERVICE "sample/hello"
      TYPE 'SOAP'
      AUTORIZATION OFF
      SECURE OFF
      USER DBA
      AS CALL hello_proc(:user_name)
      ```

      As authorization has been turned off, anyone can use this service without supplying a user name and password. The commands will be run as user DBA. This arrangement is simple, but insecure.

   c. Create a DISH service to act as a proxy for the SOAP service and generate the WSDL.

```
CREATE SERVICE "sample/dish_service"
TYPE 'DISH'
GROUP "sample"
FORMAT 'CONCRETE'
AUTHORIZATION OFF
SECURE OFF
USER DBA
```

The SOAP and DISH service must be of type CONCRETE. In this example, the FORMAT clause was omitted when the SOAP service was created. As a result, the SOAP service inherits the concrete format from the DISH service.

4. Take a look at the WSDL that the dish service automatically creates. To do so, open a web browser and browse to the following URL: *http://localhost:80/jaxrpc/sample/dish_service*. The DISH service automatically generates the following WSDL document.

   In particular, observe the ASADataset object that is exposed because the format of this service is CONCRETE. In a later step, the wscompile.bat application will use this information to generate a SOAP 1.1 client interface for this these services.

```
<?xml version="1.0" ?>
<definitions
         xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
         xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
         xmlns:s="http://www.w3.org/2001/XMLSchema"
         xmlns:soapenc="http://schemas.xmlsoap.org/soap/enco
         ding/"
         xmlns:tm="http://microsoft.com/wsdl/mime/textMatchi
         ng/"
         xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
         name="sample/dish_service"
         targetNamespace="http://localhost/jax/sample/dish_
         service"
         xmlns:s3="http://localhost/jax/sample/dish_service"
         xmlns="http://schemas.xmlsoap.org/wsdl/">
 <types>
  <s:schema attributeFormDefault="qualified"
         elementFormDefault="qualified"
         targetNamespace="http://localhost/jax/sample/dish_
         service">
   <s:import namespace="http://www.w3.org/2001/XMLSchema" />
   <s:complexType name="ASADataset">
    <s:sequence>
     <s:element name="rowset">
      <s:complexType>
       <s:sequence>
```

```
<s:element name="row" minOccurs="0" maxOccurs="unbounded">
         <s:complexType>
          <s:sequence>
           <s:any minOccurs="0" maxOccurs="unbounded"/>
          </s:sequence>
         </s:complexType>
        </s:element>
       </s:sequence>
      </s:complexType>
</s:element>
    </s:sequence>
   </s:complexType>
   <s:element name="error" type="s:string" />
   <s:element name="hello">
    <s:complexType>
     <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="user_
         name" nillable="true" type="s:string" />
     </s:sequence>
</s:complexType>
   </s:element>
   <s:element name="helloResponse">
    <s:complexType>
     <s:sequence>
      <s:element minOccurs="1" maxOccurs="1"
         name="helloResult" type="s3:ASADataset" />
      <s:element name="sqlcode" type="s:int" />
     </s:sequence>
    </s:complexType>
</s:element>
  </s:schema>
 </types>
 <message name="helloIn">
  <part name="parameters" element="s3:hello" />
 </message>
 <message name="helloOut">
  <part name="parameters" element="s3:helloResponse" />
 </message>
<message name="ASAFaultMessage">
    <part name="error" element="s3:error" />
   </message>
 <portType name="sample/dish_serviceSoapPort" >
  <operation name="hello">
   <input message="s3:helloIn" />
   <output message="s3:helloOut" />
   <fault message="s3:ASAFaultMessage" name="error" />
  </operation>
```

```
</portType>
 <binding name="sample/dish_serviceSoapBinding"
         type="s3:sample/dish_serviceSoapPort">
  <soap:binding style="document"
         transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="hello">
   <soap:operation
         soapAction="http://localhost/jax/sample/hello"
         style="document" />
   <input>
    <soap:body use="literal" />
   </input>
   <output>
<soap:body use="literal" />
   </output>
   <fault name="error">
    <soap:fault name="error" use="literal" />
   </fault>
  </operation>
 </binding>
<service name="sample/dish_service">
 <port name="sample/dish_serviceSoap"
         binding="s3:sample/dish_serviceSoapBinding">
<soap:address location="http://localhost/jax/sample/dish_
         service" />
  </port>
 </service>
</definitions>
```

In the next section of this tutorial, you access these services from Java. To do so, you need to use the wscompile tool available from Sun.

☞ To download the Java JAX-RPC tools, including wscompile, see *http://java.sun.com/xml/downloads/jaxrpc.html*.

❖ To generate and use a JAX-RPC interface for these web services

1. The first step is to create a simple XML document that tells wscompile what to do. Samples of this configuration document are available from Sun. You need only replace the location attribute of the wsdl element with the URL of your dish service. Optionally, you may also specify a package name. The package name causes the generated .java and .class files to be placed in subdirectory of the same name.

   Using a text editor, create an XML document named *config.xml* with the following content:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <configuration
       xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
       <wsdl
           location="http://localhost:80/jaxrpc/sample/dish_
            service"
           packageName="asa" />
   </configuration>
   ```

   In this case, the location specified is the URL of the dish service. In addition, the optional packageName attribute has been added so that all the generated files will be placed in a new subdirectory named *asa*.

2. At a command prompt, enter the following command.

   ```
   wscompile -gen -keep config.xml
   ```

   The -gen flag tells wscompile to retrieve the WSDL document from the given URL and generate and compile an interface for it. The -keep flag tells wscompile not to delete the .java files. Without this flag, these files are deleted after the corresponding .class files have been generated. Saving these files makes it easer to examine the makeup of the interface.

   If the wscompile command is not found, make sure that you have wscompile on your machine and that it is in your path.

   Once this command completes, you should discover a new subdirectory named *asa* that contains the following Java files, along with the compiled .class versions of each.

   - ASADataset.java
   - ASADataset_LiteralSerializer.java
   - ASAFaultMessage.java
   - Dish_service.java
   - Dish_serviceSoapPort.java
   - Dish_serviceSoapPort_Stub.java
   - Dish_serviceSoapPort_hello_Fault_SOAPBuilder.java
   - Dish_serviceSoapPort_hello_Fault_SOAPSerializer.java

- Dish_service_Impl.java
- Dish_service_SerializerRegistry.java
- Hello.java
- HelloResponse.java
- HelloResponse_LiteralSerializer.java
- Hello_LiteralSerializer.java
- Row.java
- Row_LiteralSerializer.java
- Rowset.java
- Rowset_LiteralSerializer.java

3. Save the following Java source code into a text file, and compile.

```
// HelloTest.java illustrates a web service client
// that calls the dish_service and prints out the data:

import java.util.*;
import asa.*;

public class HelloTest
{
    public static void main( String[] args )
    {
        try {
            Dish_service_Impl service = new Dish_service_
         Impl();
            Dish_serviceSoapPort port = service.getDish_
         serviceSoap();
      // this is the soap service call to hello...
            HelloResponse response = port.hello("New User");
            ASADataset result = response.getHelloResult();
            Rowset rowset = result.getRowset();
            Row[] row = rowset.getRow();

for ( int i = 0; i < row.length; i++ ) {
          // column data is contained as a SOAPElement...
                javax.xml.soap.SOAPElement[] col =
         row[i].get_any();
                for ( int j = 0; j < col.length; j++ ) {
                    System.out.print("<" +
         col[j].getLocalName() + ">" +
                        col[j].getValue() );
                }
                System.out.println();
            }

}
        catch (Exception x) {
            x.printStackTrace();
        }
    }
}
```

4. Run the above application. The following output should be generated.

```
<result_out>hello New User
```

Creating web service client functions and procedures

As well as providing web services, an Adaptive Server Anywhere database can consume web services. These can be standard web services available over the internet, or can be provided by Adaptive Server Anywhere databases, as long as the web service is not in the same database as the client procedure or function.

Adaptive Server Anywhere can act as both HTTP and SOAP web service clients. This functionality is provided through stored functions and stored procedures.

Client functions and procedures are created and manipulated using the following SQL statements

- "CREATE FUNCTION statement" [*ASA SQL Reference,* page 362]
- "CREATE PROCEDURE statement" [*ASA SQL Reference,* page 373]
- "ALTER FUNCTION statement" [*ASA SQL Reference,* page 275]
- "ALTER PROCEDURE statement" [*ASA SQL Reference,* page 278]
- "DROP statement" [*ASA SQL Reference,* page 454]

For example, the syntax of the CREATE FUNCTION and CREATE PROCEDURE statements, as used to create web service client functions, are as follows:

CREATE PROCEDURE [*owner*.]*procedure-name* **(** [*parameter*, . . .] **)**
URL *url-string*
[*proc-attributes*]

CREATE FUNCTION [*owner*.]*procedure-name* **(** [*parameter*, . . .] **)**
RETURNS *data-type*
URL *url-string*
[*proc-attributes*]

Key to this syntax is the URL clause, which is used to provide the URL of the web service that you want the procedure to access. The basic syntax of the URL clause is as follows:

url-string :
'{**HTTP**|**HTTPS**}**://**[*user* **:***password* **@**]*hostname*[**:***port*][**/***path*]**'**

The optional user and password information permits you to access web services that require authentication. The hostname may be the name or the IP address of the machine providing the web service.

The port number is required only if the server is listening on a port number other than the default. The default port numbers are 80 for HTTP services and 443 for HTTPS services.

The path identifies the resource or web service on the server.

A request can be sent to any web service, whether it is provided by another Adaptive Server Anywhere database or available over the internet. The web service may be provided by the same database server, but must not reside in the same database as the client function. Attempting to access a web service in the same database results in the error 403 Forbidden.

☞ Because it is used for parameter substitution, exclamation marks that appear in strings anywhere in the procedure definition must be escaped. For more information, see "Escaping the ! character" on page 259.

Common clauses The additional clauses that let you supply more detailed information about the procedure call are as follows:

proc-attributes :
[**TYPE** { **'HTTP**[:{**GET**|**POST**}]**'** | **'SOAP**[:{**RPC**|**DOC**}]**'** }]
[**NAMESPACE** *namespace-string*]
[**CERTIFICATE** *certificate-string*]
[**CLIENTPORT** *clientport-string*]
[**PROXY** *proxy-string*]

The TYPE clause is important because it tells Adaptive Server Anywhere how to format the request to the web service provider. Standard SOAP types RPC and DOC are available. The standard HTTP methods of GET and POST are available, specified as HTTP:GET and HTTP:POST, respectively. Specifying HTTP implies HTTP:POST.

If type SOAP is chosen, the Adaptive Server Anywhere automatically formats the request as an XML document in the standard format necessary for SOAP requests. Since an SOAP request is always an XML document, this an HTTP POST request is always implicitly used to send the SOAP request document to the server whenever type SOAP is chosen. Type SOAP implies SOAP:RPC.

Names for web service client functions and procedures The procedure name is used as the SOAP operation name when building the outgoing SOAP request. In addition, the names of any parameters also appear in in tagnames in the SOAP request envelope. Therefore specifying these names correctly, as the SOAP server expects to see it, is an important part of defining a SOAP stored procedure. This fact places restrictions on the name of SOAP procedures and functions, beyond the general rules that apply to procedure and function names in SQL Anywhere.

The following procedure definition demonstrates what happens.

```
CREATE PROCEDURE MyOperation (a INTEGER, b CHAR(128) )
URL 'HTTP://localhost'
TYPE 'SOAP:DOC'
```

When this procedure is called, such as by the following statement, a SOAP request in generated:

```
CALL MyOperation( 123, 'abc' )
```

The procedure name, which in this example is MyOperation, is appears in the <`m:MyOperation`> tag name within the request body. Furthermore, the two parameters to the procedure, a and b, become <`m:a`> and <`m:b`>, respectively.

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:m="http://localhost">
  <SOAP-ENV:Body>
    <m:MyOperation>
      <m:a>123</m:a>
      <m:b>abc</m:b>
    </m:MyOperation>
  </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```

Namespace URIs

All SOAP requests require a method namespace URI. The SOAP processor on the server side uses this URI to understand the names of the various entities in the message body of the request.

When creating a SOAP function or procedure, of either SOAP:DOC or SOAP:RPC, you will likely be required to specify a namespace URI before the call will succeed. You can obtain the required namespace value from the WSDL description document, or from other available documentation for the service. The NAMESPACE clause applies only to SOAP functions and procedures. The default namespace value is the procedure's URI, up to but not including the optional path component and excluding any user and password values.

HTTPS requests

In order to issue a secure HTTPS request, the client must have access to the server's certificate, or the certificate used to sign the server's certificate. This certificate tells Adaptive Server Anywhere how to encrypt the request. Certificate values are also be required when a request directed to a insecure server may be redirected to a secure one.

There are two ways to provide the certificate information. You may either place the certificate in a file and provide the filename, or you may provide the entire certificate as a string value. You may not do both.

The certificate attributes are supplied as a string value constructed as key=value pairs separated by semicolons:

certificate-string :
{ **file=***filename* | **certificate=***string* } **; company=***company* **; unit=***company-unit* **; name=** *common-name*

The following keys are available:

Key	**Abbre-viation**	**Description**
file		The filename of the certificate.
certificate	cert	The certificate itself, Base64 encoded.
company	co	The company specified in the certificate.
unit		The company unit specified in the certificate.
name		The common name specified in the certificate.

For example, the following statement creates a procedure that makes a secure request to a web service located on the same machine as the client:

```
CREATE PROCEDURE test ()
URL 'HTTPS://localhost/myservice'
CERTIFICATE 'file=C:\srv_cert.crt;co=iAnywhere;
             unit=ASA;name=JohnSmith'
```

Since no TYPE clause was provided, the request is assumed to be of type SOAP:RPC. The server's public certificate is located in the file *C:\srv_cert.crt.*

Client ports

When accessing web services through a firewall, it is sometimes necessary to tell Adaptive Server Anywhere which ports to use when opening a connection to the server. Ordinarily, port numbers are obtained dynamically, and you should rely on the default behavior unless your firewall restricts access to a particular range of ports.

The ClientPort option designates the port number on which the client application communicates using TCP/IP. You may specify a single port number, or a combination of individual port numbers, and ranges of port numbers, as demonstrated in the following example:

```
CREATE PROCEDURE test ()
URL 'HTTPS://localhost/myservice'
CLIENTPORT '5040,5050-5060,5070'
```

It is best to specify a list or a range of port numbers. If you specify a single port number, then only one connection will be maintained at a time. In fact, even after closing the one connection, there is a several minute timeout period during which no new connection can be made to the same remote server and port. When you specify a list and/or range of port numbers, the application keeps trying port numbers until it finds one to which it can successfully bind.

☞ This feature is similar to the ClientPort Network Protocol option. For more information, see "ClientPort protocol option [CPORT]" on page 209.

Using proxies

Some web service requests may need to be made through a proxy server. When this is the case, the URL of the proxy server must be supplied using the PROXY clause.

The format of the value is the same as for URL clause, although any user, password or path values are ignored:

url-string :
'{**HTTP**|**HTTPS**}**://**[*user*:*password*@]*hostname*[:*port*][**/***path*]**'**

When a proxy is specified, Adaptive Server Anywhere formats the request and sends it to the proxy server using the supplied proxy URL. The proxy server forwards the request to the final destination, obtains the response, and forwards the response back to Adaptive Server Anywhere.

Return values and result sets

Web service client calls may be made with either stored functions, or stored procedures. If made from a function, the return type of the function must be of a character data type, such as CHAR, VARCHAR, or LONG VARCHAR. The value returned is the body of the HTTP response. No header information is included. Additional information about the request, including the HTTP status information, is returned by procedures. Thus, procedures are prefered when access to this additional information is desirable.

SOAP procedures

The response from a SOAP function is an XML document that contains the SOAP response.

Since SOAP responses are structured XML documents, Adaptive Server Anywhere by default attempts to exploit this information and construct a more useful result set. Each of the top-level tags within the returned response document is extracted and used as a column name. The contents of the subtree below each of these tags is used as the row value for that column.

For example, given the SOAP response shown below, Adaptive Server Anywhere would construct the shown data set:

```
<SOAP-ENV:Envelope
  xmlns:SOAPSDK1="http://www.w3.org/2001/XMLSchema"
  xmlns:SOAPSDK2="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAPSDK3="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
     <ElizaResponse xmlns:SOAPSDK4="SoapInterop">
        <Eliza>Hi, I’m Eliza. Nice to meet you.</Eliza>
     <ElizaResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Eliza
Hi, I’m Eliza. Nice to meet you.

In this example, the response document is delimited by the ElizaResponse tags that appear within the SOAP-ENV:Body tags.

Result sets have as many columns as there are top-level tags. This result set has only one column because there is only one top-level tag in the SOAP response. This single top-level tag, Eliza, becomes the name of the column.

XML processing facilities

Information within XML result sets, including SOAP responses, can also be accessed using the built-in Open XML processing capabilities.

The following example uses the OPENXML function to extract portions of a SOAP response. This example uses a web service to expose the contents of the SYSWEBSERVICE table as a SOAP service:

```
CREATE SERVICE get_webservices
TYPE ’SOAP’
AUTHORIZATION OFF
USER DBA
AS SELECT * FROM SYSWEBSERVICE
```

The following stored function, which must be created in a second Adaptive Server Anywhere database, issues a call to this webservice. The return value of this function is the entire SOAP response document. The response is in the .NET DataSet format, as DNET is the default SOAP service format.

```
CREATE FUNCTION get_webservices ()
RETURNS LONG VARCHAR
URL ’HTTP://localhost/get_webservices’
TYPE ’SOAP:DOC’
```

The following statement demonstrates how two columns of the result set can be extracted with the aid of the OPENXML function. These are the *service_name* and *secure_required* columns, which indicate which SOAP services are secure and hence require HTTPS.

```
SELECT *
FROM OPENXML( get_webservice(), '//row' )
WITH ("Name"    char(128) 'service_name',
      "Secure?" char(1)   'secure_required' )
```

This statement works by selecting the decendants of the *row* node. The WITH clause constructs the result set based on the two elements of interest. Assuming only the *get_webservices* service exists, this function returns the following result set:

Name	Secure?
get_webservices	N

☞ For more information about the XML processing facilities available in Adaptive Server Anywhere, see "Using XML in the Database" [*ASA SQL User's Guide,* page 513].

Other types of procedures

Procedures of other types return all the information about a response in a two-column result set. This result set includes the response status, header information and body. The first column, is named *Attribute* and the second *Value*. Both are of data type LONG VARCHAR.

The result set has one row for each of the response header fields, as well as a row for the HTTP status line (*Status* attribute) and a row for the response body (*Body* attribute).

The following example represents a typical response:

Attribute	Value
Status	HTTP /1.0 200 OK
Body	<!DOCTYPE HTML ... ><HTML> ... </HTML>
Content-Type	text/html
Server	GWS/2.1
Content-Length	2234
Date	Mon, 18 Oct 2004, 16:00:00 GMT

Selecting from result sets

The SELECT statement is used to retrieve values from results sets. Once retrieved, these values may be stored in tables or used to set variables.

```
CREATE PROCEDURE test( INOUT parm CHAR(128) )
URL 'HTTP://localhost/test'
TYPE 'HTTP'
```

As it is of type HTTP, this procedure returns the two-column result set described in the previous section. In the first column is an attribute name; in the second column the attribute value. The keywords are as in the HTTP response header fields. A Body attribute contains the body of the message, which is typically an HTML document.

One approach is to insert the result sets into a table, such as the following one:

```
CREATE TABLE StoredResults(
     Attribute LONG VARCHAR,
     Value       LONG VARCHAR
)
```

Result sets can be inserted into this table as follows:

```
INSERT INTO StoredResults SELECT *
FROM test('Storing into a table')
WITH (Attribute LONG VARCHAR, Value LONG VARCHAR)
```

You can add clauses according to the usual syntax of the SELECT statement. For example, if you want only a specific row of the result set you can add a WHERE clause to limit the results of the select to only one row:

```
SELECT Value
FROM test('Calling test for the status code')
WITH (Attribute LONG VARCHAR, Value LONG VARCHAR)
WHERE Attribute = 'Status'
```

This statement selects only the status information from the result set. It can thus be used to verify that the call was successful.

Parameters

Stored functions and stored procedures that act as web service clients may be declared with parameters, just as may other types of function or procedures. These parameters values, unless used during parameter substitution, are passed as part of the HTTP or SOAP request.

In addition, parameters can be used to replace placeholders within the body of the stored function or stored procedure at the time the function or procedure is called. If no placeholders for a particular variable exist, the parameter and its value are passed as part of the request. Parameters and values used for substitution in this manner are not passed as part of the web service request.

Passed parameters

All parameters to a function or procedure, unless used during parameter substitution, are passed as part of the web service request. The format in which they are passed depends on the type of the web service request.

HTTP requests

Parameters to requests of type HTTP:GET are encoded in the URL. For example, the following procedure declares two parameters:

```
CREATE PROCEDURE test ( a INTEGER, b CHAR(128) )
URL 'HTTP://localhost/myservice'
TYPE 'HTTP:GET'
```

If this procedure is invoked with the two values 123 and “xyz”, then URL used for the request is equivalent to that shown below:

```
HTTP://localhost/myservice?a=123&b=xyz
```

If the type is HTTP:POST, the parameters and their values instead become part of the body of the request. In the case of the two parameter and values, the following text appears, after the headers, in the body of the HTTP request:

```
a=123&b=xyz
```

SOAP requests

Parameters passed to SOAP requests are bundled as part of the request body, as required by the SOAP specifications:

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:m="http://localhost">
  <SOAP-ENV:Body>
    <m:test>
      <m:a>123</m:a>
      <m:b>abc</m:b>
    </m:test>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Parameter substitution

Declared parameters to the stored procedure or stored function are automatically substituted for placeholders within the stored function or stored procedure definition each time the procedure or function is executed. Any substrings that contain an exclamation mark (“!”) followed by the name of one of the declared parameters are replaced by that parameter’s value.

For example, the following procedure definition permits the entire URL to be passed as a parameter. Different values may be used each time this procedure is called.

```
CREATE PROCEDURE test ( url CHAR(128) )
URL '!url'
TYPE 'HTTP:POST'
```

For example, you could then use the procedure as follows:

```
CALL test ( 'HTTP://localhost/myservice' )
```

Hiding user and password values

One useful application of parameter substitution is to avoid making sensitive values, such as user names and passwords, part of a web service client function or procedure definition. When such values are specified as literals in the procedure or function definition, they are stored in the system tables, making them readily accessible to all users of the database. Passing these values as parameters circumvents this problem.

For example, the following procedure definition includes the user name and password as plain text as part of the procedure definition:

```
CREATE PROCEDURE test
URL 'HTTP://DBA:SQL@localhost/myservice'
```

This string appears as plain text in the system tables, permitting easy access to the authentication values. To avoid this problem, you can declare user and password as parameters. Doing so permits the user and password values to be supplied only when the procedure is invoked.

```
CREATE PROCEDURE test ( uid CHAR(128), pwd CHAR(128) )
URL 'HTTP://!uid:!pwd@localhost/myservice'
```

This procedure is called as follows:

```
CALL test ( 'DBA', 'SQL' )
```

As another example, you can use parameter substitution to pass encryption certificates from a file and pass them to a stored procedure or stored function:

```
CREATE PROCEDURE secure( cert LONG VARCHAR )
URL 'https://localhost/secure'
TYPE 'HTTP:GET'
CERTIFICATE 'cert=!cert;company=test;unit=test;name=RSA Server'
```

When you call this procedure, you supply the certificate as a string. In the following example call, the certificate is read from a file. This is done for illustration only, as the certificate can be read directly from a file using the **file=** keyword for the CERTIFICATE clause.

```
CALL secure( xp_read_file('C:\asainstalldir\win32\
         rsaserver.crt') )
```

Escaping the ! character

As the exclamation mark "!" is used to identify placeholders for parameter subsitution in the context of web service client stored functions and stored procedures, it must be escaped whenever you want to include this character as part of any of the procedure attribute strings string. To do so, prefix the the exclamation mark with a second exclamation mark. Hence, all occurances of “!!” in strings within a web service client or web service function definition are replaced by “!”.

Parameter names used as placeholders must contain only alphanumeric characters. In addition, placeholders must be followed by a non-alphanumeric character to avoid ambiguity. Placeholders with no matching parameter name are automatically deleted. For example, the parameter size would not be substituted for the placeholder in the following procedure:

```
CREATE PROCEDURE orderitem ( size CHAR(18) )
URL ’HTTP://salesserver/order?size=!sizeXL’
TYPE ’SOAP:RPC’
```

Instead, !sizeXL is always deleted because it is a valid placeholder for which there is no matching parameter.

Parameter data type conversion

Parameter values that are not of character or binary data types are converted to a string representation before being added to the request. This process is equivalent to casting the value to a character type. The conversion is done in accordance with the data type formatting option settings at the time the function or procedure is invoked. In particular, the conversion may be affected by such options as PRECISION, SCALE, and TIMESTAMP_FORMAT.

Procedures that provide HTML documents

Generally, it is easiest to write a procedure that handles the requests sent to a particular service. Such a procedure should return a web page. Optionally the procedure can accept arguments, passed as part of the URL, to customize its output.

The following example, however, is much simpler. It demonstrates how simple a service can be. This web service simply returns the phrase "Hello world!".

```
CREATE SERVICE hello TYPE 'RAW'
      AUTHORIZATION OFF USER DBA
      AS select 'Hello world!'
```

Start a server with the -xs option to enable handling of web requests, then request the URL **http://localhost/hello** from any web browser. The words Hello world! will appear in an otherwise plain page.

HTML pages

The above page appears in your browser in plain text. This happens because the default HTTP Content-Type is text/plain. To create a more normal web page, formatted in HTML, you must do two things:

- Set the HTTP Content-Type header field to text/html so that the browsers will expect HTML.
- Include HTML tags in the output.

You can write tags to the output in two ways. One way is to use the phrase TYPE HTML in the CREATE SERVICE statement. Doing so asks Adaptive Server Anywhere to add HTML tags for you. This can work quite well if, for example, you are returning a table.

The other way is to use TYPE RAW and write out all the necessary tags yourself. This second method provides the most control over the output. Note that specifying type RAW does not necessarily mean the output is not in HTML or XML format. It only tells Adaptive Server Anywhere that it can pass the return value directly to the client without adding tags itself.

The following procedure generates a fancier version of Hello world. For convenience, the body of the work is done in the following procedure, which formats the web page.

The built-in procedure sa_set_http_header is used to set the HTTP header type so browsers will correctly interpret the result. If you omit this statement, your browser will display all the HTML codes, rather than use them to format the document.

```
CREATE PROCEDURE hello_pretty_world ()
RESULT (html_doc long varchar)
BEGIN
  call dbo.sa_set_http_header( 'Content-Type', 'text/html' );
  select '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">\n'
    || '<html>\n'
    || '<head>\n'
    || '  <title>Hello Pretty World</title>\n'
    || '</head>\n'
    || '<body>\n'
    || '  <h1>Hello Pretty World!</h1>\n'
    || '  <p>(If you see the tags in your browser, check that\n'
    || '      the Content-Type header is set to text/html.)\n'
    || '</body>\n'
    || '</html>';
END
```

The following statement creates a service that uses this procedure. The statement calls the above procedure, which generates the Hello Pretty World web page.

```
CREATE SERVICE hello_pretty_world TYPE 'RAW'
      AUTHORIZATION OFF USER DBA
      AS call hello_pretty_world()
```

Once you have created the procedure and the service, you are ready to access the web page. Ensure that your database server was started with the correct -xs option values, then open the URL **http://localhost/hello_pretty_world** in a web browser.

You see the results formatted in a simple HTML page, with the title Hello Pretty World. You can make the web page as fancy as you wish by including more content, using more tags, using style sheets, or including scripts that run in the browser. In all cases, you create the necessary services to handle the browser's requests.

☞ For more information about built-in stored procedures, see "System and catalog stored procedures" [*ASA SQL Reference*, page 777].

Root services

When no service name is included in a URL, Adaptive Server Anywhere looks for a web service named root. The role of root pages is analogous to the role of index.html pages in many traditional web servers.

Root services are handy for creating home pages because they can handle URL requests that contain only the address of your web site. For example, the following procedure and service implement a simple web page to be displayed when you browse to the URL **http://localhost**.

```
CREATE PROCEDURE home_page
RESULT (html_doc long varchar)
BEGIN
  call dbo.sa_set_http_header( 'Content-Type', 'text/html' );
  select '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">\n'
      || '<html>\n'
      || '<head>\n'
      || '  <title>My Home Page</title>\n'
      || '</head>\n'
      || '<body>\n'
      || '  <h1>My home on the web<h1>\n'
      || '  <p>Thank you for visiting my web site!\n'
      || '</body>\n'
      || '</html>';
END
```

Now create a service that uses this procedure:

```
CREATE SERVICE root TYPE 'RAW'
AUTHORIZATION OFF USER DBA
AS call home_page()
```

You can access this web page by browsing to the URL **http://localhost**, as long as you do not specify that database names are mandatory when you start the database server.

Examples

For example, the request in the Quick Start example includes no database name and no parameters. In this request, **/tables** identifies the service.

```
http://localhost:80/tables
```

☞ More extensive examples are included in the *Samples/ASA/HTTP* subdirectory of your SQL Anywhere Studio installation.

Working with variables

Variables in HTTP requests come from one of two sources. First, the URL may include a query, which includes various name-value pairs. HTTP GET requests are formatted in this manner.

The second way is through the URL path. Setting the URL PATH to either ON or ELEMENTS causes the portion of the path following the service name to be interpreted as variable values. This option allows URLs to appear to be requesting a file in a particular directory, as would be the case on a traditional file-based web site, rather than something stored inside a database.

For example, the URL **http://localhost/gallery/sunset.jpg** appears to request the file *sunset.jpg* from a directory named gallery, but instead asks the gallery service to retrieve a picture from a database table.

The parameter passed in HTTP requests depends on the setting of URL PATH.

- **OFF** No path parameters are permitted after the service name.
- **ON** All path elements after the service name are assigned to the variable URL.
- **ELEMENTS** The remainder of the URL path is split at the slash characters into a list of up to 10 elements. These values are assigned the variables URL1, URL2, URL3, . . . , URL10. If there are fewer than 10 values, the remaining variables are set to NULL. Specifying more than ten variables causes an error.

Apart from the location in which they are defined, there is no difference between variables. You access and use all HTTP variables the same way. For example, the values of variables such as url1 are accessed in the same way as parameters that appear as part of a query, such as **?picture=sunset.jpg**.

Accessing parameters

There are two main ways to access variables. The first is to mention variables in the statement of the service declaration. For example, the following statement passes the value of multiple variables to the show_table stored procedure:

```
CREATE SERVICE show_table TYPE ’RAW’
AUTHORIZATION ON
AS CALL show_table( :user_name, :table_name, :limit, :start )
```

The other way is to use the built-in functions next_http_variable and http_variable within the stored procedure that handles the request. If you do not know which variables are defined, you can use the next_http_variable to find out. The http_variable function returns the variable values.

The next_http_variable function allows you to iterate through the names of the defined variables. The first time you call it, you pass in the NULL value. This returns the name of one variable. Calling it subsequent times, each time passing in the name of the previous variable, returns the next variable name. When the name of the last variable is passed to this function, it returns NULL.

Iterating through the variable names in this manner guarantees that each variable name will be returned exactly once. However, the order that the values are returned may not be the same as the order that they appear in the request. In addition, if you iterate through the names a second time, they may be returned in a different order.

To get the value of each variable, use the http_variable function. The first parameter is the name of the variable. Additional parameters are optional. In the case that multiple values were supplied for a variable, the function returns the first value if supplied with only one parameter. Supplying an integer as the second parameter allows you to retrieve additional values.

The third parameter allows you to retrieve variable header-field values from multi-part requests. Supply the name of a header field to retrieve its value. For example, the following SQL statements retrieve three variable values, then retrieve the header-field values of the image variable.

```
SET v_id    = http_variable( 'id' );
SET v_title = http_variable( 'title' );
SET v_descr = http_variable( 'descr' );

SET v_name = http_variable( 'image', NULL, 'Content-Disposition'
         );
SET v_type = http_variable( 'image', NULL, 'Content-Type' );
SET v_image = http_variable( 'image', NULL, '@BINARY' );
```

Automatic character set conversion

By default, character set conversion is performed automatically on outgoing result sets of type text. Result sets of other types, such as binary objects, are not translated. The character set of the request is converted to the database character set, and the result set is converted from the database character set to the client character set, as required, except on binary columns in the result set. When the request lists multiple character sets that it can handle, the server takes the first suitable one from the list.

Character-set conversion can be enabled or disabled by setting the HTTP option CharsetConversion. The allowed values are ON and OFF. The default value is ON. The following statement turns automatic character-set conversion off:

```
call dbo.sa_set_http_option( 'CharsetConversion', 'OFF' )
```

☞ For more information about built-in stored procedures, see "System and catalog stored procedures" [*ASA SQL Reference*, page 777].

Errors

When web service requests fail, the database server generates standard errors that appear in your browser. These errors are assigned numbers consistent with the protocol standards.

If the service is a SOAP service, faults are returned as to the client as SOAP faults, following the SOAP as defined in the SOAP version 1.1 standard:

- When an error in the application handling the request generates a SQLCODE, a SOAP Fault is returned with a **faultcode** of Client, possibly with a sub-category, such as Client.Procedure. The **faultstring** element within the SOAP Fault is set to a detailed explanation of the error and a **detail** element contains the numeric SQLCODE value.

- In the event of a transport protocol error, the **faultcode** is set to either **Client** or **Server**, depending on the error. faultstring is set to the HTTP transport message, such as `404 Not Found`, and the **detail** element contains the numeric HTTP error value.

- SOAP Fault messages generated due to application errors that return an SQLCODE value are returned with an HTTP status of `200 OK`.

If the client cannot be identified as a SOAP client, then the appropriate HTTP error is returned in a generated HTML document.

The following are some of the more typical errors that you may encounter:

Number	Name	SOAP fault	Description
301	Moved permanently	Server	The requested page has been permanently moved. The server will automatically redirect the request to the new location.
304	Not Modified	Server	The server has decided, based on information in the request, that the requested data has not been modified since the last request and so does not need to be sent again.
307	Temporary Redirect	Server	The requested page has been moved, but this change may not be permanent. The server will automatically redirect the request to the new location.

Number	Name	SOAP fault	Description
400	Bad Request	Client.-BadRequest	The HTTP request is incomplete or malformed.
401	Authorization Required	Client.-Authorization	Authorization is required to use the service, but a valid user name and password were not supplied.
403	Forbidden	Client.-Forbidden	You do not have permission to access the database.
404	Not Found	Client.-NotFound	The named database is not running on the server, or the named web service does not exist.
408	Request Timeout	Server.-RequestTimeout	The maximum connection idle time was exceeded while receiving the request.
411	HTTP Length Required	Client.-LengthRequired	The server requires that the client include a Content-Length specification in the request. Typically occurs when uploading data to the server.
413	Entity Too Large	Server	The request exceeds the maximum permitted size.
414	URI Too Large	Server	The length of the URI exceeds the maximum allowed length.
500	Internal Server Error	Server	An internal error occurred. The request could not be processed.
501	Not Implemented	Server	The HTTP request method is not GET, HEAD, or POST.
502	Bad Gateway	Server	The document requested resides on a third-party server and the server received an error from the third-party server.
503	Service Unavailable	Server	The number of connections exceeds the allowed maximum.

PART II

WORKING WITH DATABASE FILES

This section describes how to install, use, and backup database files. It also describes how to use international languages and character sets.

CHAPTER 8

File Locations and Installation Settings

About this chapter

This chapter describes the installation and operating system settings used by Adaptive Server Anywhere. Depending on the operating system, these settings may be stored as environment variables, initialization file entries, or registry settings.

Contents

Installation directory structure

When you install Adaptive Server Anywhere, several directories may be created. Some of the files in these directories are essential, and others are not. This section describes the directory structure.

Adaptive Server Anywhere software, whether you receive it as a product or bundled as part of another product, is installed under a single installation directory. The tools provided with the Adaptive Server Anywhere product, however, are installed in other directories. This section describes only the installation directory structure for Adaptive Server Anywhere itself.

The Adaptive Server Anywhere installation directory

The Adaptive Server Anywhere installation directory itself holds several items, including the following:

- **The sample database** The sample database is held in the file *asademo.db*.
- **Read Me First** A Read Me First file named *readme.txt* holds late-breaking information.

For platforms other than Novell NetWare and Windows CE, there are several directories under the installation directory:

- **Executable directories** There is a separate directory for each operating system, which holds executables, dynamic link libraries, and help files.

  On Windows, except Windows CE, these files are installed in the *win32* or *win64* directory. If you are using UNIX, they are installed in the *bin* and *lib* directories. On NetWare, the executables are stored in the installation directory itself.

  You will not have all these directories on your machine; you will have only the ones required for the operating system version you installed.
- **Java directory** Java base classes are stored in this directory.
- **Scripts directory** The scripts directory contains SQL scripts that are used by the database administration utilities and as examples. With the exception of the *custom.sql* script, *do not edit these scripts.* If the scripts directory is not present, the administration utilities will not work.
- **Samples directories** This holds separate directories for a variety of samples.
- **h directory** The *h* directory contains header files for ESQL and ODBC database development. ON UNIX, this directory is called *include*.

Novell NetWare file locations

On Novell NetWare, all files are installed to a single directory on the server. Throughout this documentation, when reference is made to files in subdirectories of the installation directory, the file on NetWare is in the installation directory itself.

Windows CE file locations

On Windows CE, all files are installed to the installation directory *\Program Files\Sybase\ASA9*. No subdirectories are created. The exception is that all DLLs are installed into the *\Windows* directory.

How Adaptive Server Anywhere locates files

The client library and the database server need to locate files for two main purposes:

- DLLs and initialization files are required to run Adaptive Server Anywhere. If an incorrect DLL is located, there is the possibility of version mismatch errors.
- Some files are specified in SQL statements and need to be located at run time, such as INSTALL or LOAD TABLE.

Examples of SQL statements that use file names include the following:

- **INSTALL JAVA statement** The name of the file that holds Java classes.
- **LOAD TABLE and UNLOAD TABLE statements** The name of the file from which data should be loaded or to which the data should be unloaded.
- **CREATE DATABASE statement** A file name is needed for this statement and similar statements that can create files.

In some cases, Adaptive Server Anywhere uses a simple algorithm to locate files. In other cases, a more extensive search is carried out.

Simple file searching

In many SQL statements (such as LOAD TABLE, or CREATE DATABASE), the file name is interpreted as relative to the current working directory of the database server.

Also, when a database server is started and a database file name (DatabaseFile (DBF) parameter) is supplied, the path is interpreted as relative to the current working directory.

Extensive file searching

Adaptive Server Anywhere programs, including the database server and administration utilities, carry out a more extensive search for required files, such as DLLs or shared libraries. In these cases, Adaptive Server Anywhere programs look for files in the following order:

1. **Executable directory** Holds the program executable file.
2. **Related directories** Holds directories with the following paths relative to the program executable directory:
   - Parent of the executable directory.
   - A child of the parent directory named *scripts*. The UNIX server does not search in this location.

3. **Current working directory** When a program is started, it has a current working directory (the directory from which it is started). This directory is searched for required files.

4. **Location registry entry** When installing onto Windows, Adaptive Server Anywhere adds a Location registry entry. The indicated directory is searched, followed by:
   - ♦ A child named *scripts*
   - ♦ A child with the operating system name (*win32*, *win64*, and so on)

5. **System-specific directories** This includes directories where common operating system files are held, such as the *Windows* directory and the *Windows\system32* directory on Windows operating systems.

6. **CLASSPATH directories** For Java files, directories listed in the CLASSPATH environment variable are searched to locate files.

7. **PATH directories** Directories in the system path and the user's path are searched to locate files.

8. **Library path (UNIX only)** On UNIX, the operating system requires that the library path be set. The corresponding environment variable that must be set should be one of the following:
   - ♦ LD_LIBRARY_PATH (Linux, Solaris, DEC)
   - ♦ DYLD_LIBRARY_PATH (Mac OS X)
   - ♦ SHLIB_PATH (HP-UX)
   - ♦ LIBPATH (AIX)

Environment variables

Adaptive Server Anywhere uses a set of environment variables to store various types of information. These environment variables are listed in this section. Not all variables need to be set in all circumstances.

Setting environment variables

The way you set an environment variable depends on the operating system you are using.

❖ To set an environment variable (Windows NT)

1. Right-click My Computer and choose Properties from the popup menu.
2. Click the Environment tab. If the environment variable does not already exist, type variable and its value in the spaces provided, and click Set.

   If the variable does exist, select it from the list of System Variables or User Variables, and make any modifications in the Value field. Click Set to make the setting.

❖ To set an environment variable (Windows 2000)

1. Right-click My Computer and choose Properties from the popup menu.
2. Click the Advanced tab.
3. Click the Environment Variables button.

   The Environment Variables dialog opens. If the environment variable does not already exist, click New and type the variable name and its value in the spaces provided, and then click OK.

   If the variable does exist, select it from the list of System Variables or User Variables, click Edit and make any modifications in the Variable Value field. Click OK to make the setting.

❖ To set an environment variable (Windows XP)

1. From the Start menu, choose Control Panel ➤ System.
2. Click the Advanced tab.
3. Click the Environment Variables button.

   The Environment Variables dialog opens. If the environment variable does not already exist, click New and type the variable name and its value in the spaces provided, and then click OK.

If the variable does exist, select it from the list of System Variables or User Variables, click Edit and make any modifications in the Variable Value field. Click OK to make the setting.

❖ To set an environment variable (UNIX)

1. In one of your startup files (*.cshrc*, *.shrc*, *.login*), add a line that sets the variable.

   In some shells (such as sh, bash, or ksh) the line is as follows:

   ```
   export VARIABLE=value
   ```

   In other shells (such as csh, or tsch) the line is as follows:

   ```
   setenv VARIABLE value
   ```

ASANY9 environment variable

Syntax	**ASANY9**=*directory-name*
Default	*C:\Program Files\Sybase\SQL Anywhere 9*
Description	On installation, the ASANY9 environment variable is set to your SQL Anywhere directory.

This environment variable must be set for several reasons. Among the list, samples require this environment variable in order to locate SQL Anywhere applications.

ASANYSH9 environment variable

Syntax	**ASANYSH9**=*directory-name*
Default	None.
Description	Interactive SQL, Sybase Central, and the Adaptive Server Anywhere Console utility (dbconsole) use this variable to determine the location of the shared components directory. ASANYSH9 is set to the location of the shared directory. You set the shared directory location during installation. The default shared components directory is *C:\Program Files\Sybase\Shared*.

ASCHARSET environment variable

Syntax	**ASCHARSET**=*charset*
Description	*Charset* is a character set name. For example, setting **ASCHARSET=cp1252** sets the default character set to cp1252.

The first of the following methods that returns a value determines the default character set.

- check ASCHARSET environment variable
- query of the operating system

If no character set information is specified, use iso_1 for UNIX, or cp850 otherwise.

This environment variable is not supported on Windows CE or NetWare.

ASLANG environment variable

Syntax

ASLANG=*language_code*

Description

Language_code is the two-letter combination representing a language. For example, setting **ASLANG=DE** sets the default language to German.

The first of the following methods that returns a value determines the default language.

- ASLANG environment variable check
- registry check (Windows only) as set by the installer or *dblang.exe*
- query of the operating system for language information

If no language information is set, English is the default.

This environment variable is not supported on Windows CE or NetWare.

ASLOGDIR environment variable

Syntax

ASLOGDIR=*directory-name*

Description

If the ASLOGDIR environment variable is set, it is assumed to contain the path for a directory containing the backup history file, *backup.syb*. This file is updated each time you execute a BACKUP or RESTORE statement.

Windows

On Windows, if the ASLOGDIR environment variable is not set, the server attempts to find the *backup.syb* file in the directory where the server executable normally resides. For example, on 32-bit Windows platforms, this is *C:\Program Files\Sybase\SQL Anywhere 9\win32*. The install directory is determined using the *HKLM\Software\Sybase\Adaptive Server Anywhere\9.0\Location* registry key. If this directory does not exist, an error is given. If the Location registry key does not exist, the path used for the server executable is examined, if one was specified. If no path was specified to start the server, the *backup.syb* file is placed in the root directory of the current drive.

UNIX

On UNIX, if the ASLOGDIR environment variable is not set, the server checks for a HOME environment variable. If HOME is set, the server places the *backup.syb* file in that directory. Otherwise, it attempts to place the file in the Home directory for the user who started the server. If that cannot be determined, it attempts to place the *backup.syb* file in the directory where the server was started.

See also

- "BACKUP statement" [*ASA SQL Reference*, page 307]

ASTMP environment variable

Syntax

ASTMP=*directory-name*

Default

None.

Description

The database server checks the value of the ASTMP environment variable to determine where to hold the temporary file. If the ASTMP environment variable does not exist, then the first of the TMP, TMPDIR, and TEMP environment variables to exist is used. On UNIX, if none of the above environment variables exist, */tmp* is used.

In many circumstances, ASTMP is not needed. It can be of use in security-conscious environments when running the database server as a service to enable you to hold the temporary file in a directory that cannot be accessed by other programs.

On UNIX, files that are used for shared memory access are included in the Adaptive Server Anywhere temporary files. If an application is connecting to the database server using shared memory, then the ASTMP environment variable must be set to the directory where the Adaptive Server Anywhere temporary files are located.

On Windows CE, you can specify which directory you want to use as the server's temporary directory, in the registry.

☞ For information about the temporary file location on Windows CE, see "Registry settings on Windows CE" on page 284.

See also

- "TEMP environment variable" on page 281

LD_LIBRARY_PATH environment variable [UNIX]

Syntax

LD_LIBRARY_PATH=*installation_path*/lib

Description

The LD_LIBRARY_PATH environment variable is used on UNIX only. It is modified by the installation program to include the directories where Adaptive Server Anywhere libraries are located.

The libraries are located in the *lib* subdirectory of the installation directory (for example, */opt/SYBSasa9/lib*).

On Mac OS X, the DYLD_LIBRARY_PATH environment variable is used and on AIX, the LIBPATH environment variable is used.

PATH environment variable

Syntax

PATH=*installation_path*

Description

The PATH environment variable is modified by the installation program to include the directories where Adaptive Server Anywhere executables are located.

The executables are located in a subdirectory of the installation directory.

In addition, if you are using other Sybase applications, the *SYBASE\bin* and *SYBASE\dll* directories are added to your path.

On UNIX, each user must have the directory holding the executables (*/opt/SYBSasa9/bin*) added to their path.

SQLCONNECT environment variable

Syntax

SQLCONNECT=*parameter*=*value*; . . .

Description

The SQLCONNECT environment variable is optional, and is not set by the installation program.

SQLCONNECT specifies connection parameters that are used by several of the database administration utilities when connecting to a database server. This string is a list of parameter settings, of the form **parameter**=*value*, delimited by semicolons.

The number sign (#) is an alternative to the equal sign, and should be used if you are setting the connection parameters string in the SQLCONNECT environment variable. Using = inside an environment variable setting is a syntax error.

☞ For a description of the connection parameters, see "Connection parameters" on page 176.

SQLLOCALE environment variable

The SQLLOCALE environment variable is no longer supported. It has been replaced by the ASLANG and ASCHARSET environment variables.

☞ For information about the ASLANG environment variable, see "ASLANG environment variable" on page 278.

☞ For information about the ASCHARSET environment variable, see "ASCHARSET environment variable" on page 277.

SQLPATH environment variable

Syntax

SQLPATH=*path*; . . .

Description

The SQLPATH environment variable is optional, and is not set by the installation program.

Interactive SQL searches along SQLPATH for command files and Help files before searching the system path.

SQLREMOTE environment variable

Syntax

SQLREMOTE=*path*

Description

The SQLREMOTE environment variable is optional, and is not set by the installation program.

Addresses for the FILE message link in SQL Remote are subdirectories of the SQLREMOTE environment variable. This variable should point to a shared directory.

On 32-bit Windows operating systems, an alternative to setting the SQLREMOTE environment variable is to set the *SQL Remote\Directory* registry entry to the proper root directory.

SYBASE environment variable

Syntax

SYBASE=*path*

Description

The SYBASE variable marks the home directory for installation of some Sybase applications, including Adaptive Server Enterprise and utilities such as *DSEdit.* You need this variable only if you are using Adaptive Server Anywhere together with other members of the Adaptive Server family.

TEMP environment variable

Syntax

ASTMP=*path*

TMP=*path*

TMPDIR=*path*

TEMP=*path*

Description

The database server creates a temporary file for various operations such as sorting and performing unions. Temporary files are placed in the directory specified by the ASTMP, TMP, TMPDIR, or TEMP environment variable. Adaptive Server Anywhere takes the first one of these that it finds, in that order.

If none of the environment variables is defined, temporary files are placed in the current working directory of the server. On UNIX only, if none of these environment variables are found, then */tmp* is used.

On Windows CE, you can specify which directory you want to use as the server's temporary directory, in the registry.

☞ For more information about setting the temporary directory value, see "Registry settings on Windows CE" on page 284.

ASTMP is only used by the database server (dbeng9 or dbsrv9). The command-line utilities that require a temporary directory only use TMP, TMPDIR, and TEMP on all supported platforms.

Registry and INI files

On Windows operating systems (except Windows CE), Adaptive Server Anywhere uses several registry settings. On UNIX and NetWare, these settings are held in initialization files instead.

The software makes these settings for you, and in general operation you should not need to access the registry. The settings are provided here for those people who make modifications to their operating environment.

The contents of *.ini* files used by Adaptive Server Anywhere can be obfuscated with simple encryption using the File Hiding utility.

☞ For more information, see "Hiding the contents of .ini files" on page 524.

Caution
You should not add simple encryption to the .odbc.ini system information file with the File Hiding utility (dbfhide) on UNIX unless you will only be using Adaptive Server Anywhere data sources. If you plan to use other data sources (for example, for MobiLink synchronization), then obfuscating the contents of the .odbc.ini file may prevent other drivers from functioning properly.

Current user and local machine settings

Some operating systems, such as Windows NT, hold two levels of system settings. Some settings are specific to an individual user and are used only when that user is logged on; these settings are called **current user** settings. Some settings are global to the machine, and are available no matter which user is logged on; these are called **local machine** settings. You must have administrator permissions on your machine to change local machine settings.

Adaptive Server Anywhere respects both current user and local machine settings. On Windows NT, for example, these are held in the HKEY_CURRENT_USER key and the HKEY_LOCAL_MACHINE key, respectively.

Current user takes precedence

If a setting is made in both the current user and local machine registries, the current user setting takes precedence over the local machine setting.

When local machine settings are needed

If you are running an Adaptive Server Anywhere program as a **service**, you should ensure that the settings are made at the *local machine* level.

Services can continue to run under a special account when you log off a machine as long as you do not shut the machine down entirely. They can be made independent of individual accounts, and therefore need access to local machine settings.

In addition to Adaptive Server Anywhere programs, some web servers run as services. You must set local machine settings in order for PowerDynamo to work with such a web server.

In general, the use of local machine settings is recommended.

Registry structure

On Windows (except Windows CE), you can access the registry directly with the registry editor. The Adaptive Server Anywhere registry entries are held in either the HKEY_CURRENT_USER or HKEY_LOCAL_MACHINE keys, in the following location:

```
Software
   Sybase
      Adaptive Server Anywhere
         9.0
      Sybase Central
         4.3
         Profiles
         Providers
```

Registry settings on installation

The installation program makes the following registry settings in the Sybase registry:

- **Location** In the *Adaptive Server Anywhere\9.0* registry key, this entry holds the installation directory location. For example:

  ```
  Location  "c:\Program Files\Sybase\SQL Anywhere 9"
  ```

- **Language** In the *Adaptive Server Anywhere\9.0* registry key, this entry holds a two-letter code indicating the current language for messages and errors. For example:

  ```
  Language "EN"
  ```

  The default setting is English (EN). The installation program sets this entry only if the software is installed for a language other than English.

Registry settings on Windows CE

You can specify which directory you want to use as the server's temporary directory on Windows CE by setting the

```
HKEY_CURRENT_USER\Software\Sybase\Adaptive Server Anywhere\9.0\
         TempFolder
```

value in the registry, where *TempFolder* is the name of the temporary directory you want to use. The server will either:

- use the specified directory if it exists, or
- attempt to create the specified directory if it does not already exist, as long as the parent directory already exists.

If the specified directory does not exist and cannot be created, the server will:

- use the *\Temp* directory if it exists, or
- attempt to create a *\Temp* directory if it does not already exist.

If the *\Temp* directory does not exist and cannot be created, the server uses the current directory.

CHAPTER 9

Working with Database Files

About this chapter

This chapter describes how to create, and work with database and associated files.

Contents

Overview of database files

Basic database files

Each database has the following files associated with it.

- ♦ **The database file** This file holds the database information. It typically has the extension *.db*.

  ☞ For information on creating databases, see "Working with databases" [*ASA SQL User's Guide,* page 31].

- ♦ **The transaction log** This file holds a record of the changes made to the database, and is necessary for recovery and replication. It typically has the extension *.log*.

  ☞ For information on the transaction log, see "The transaction log" on page 378.

- ♦ **The temporary file** The database server uses the temporary file to hold information needed during a database session. The database server disregards this file once the database shuts down—even if the server remains running. The file has a server-generated name with the extension *.tmp*.

  The temporary file is held in a location determined by an environment variable.

  The following environment variables are checked, in order:

  - ASTMP
  - TMP
  - TMPDIR
  - TEMP

  If none of these is defined, Adaptive Server Anywhere places its temporary file in the current directory on Windows operating systems, or in the */tmp/.SQLAnywhere* directory on UNIX.

  The server creates, maintains, and removes the temporary file. You only need to ensure that there is enough free space available for the temporary file. You can obtain information about the space available for the temporary file using the sa_disk_free_space procedure.

  ☞ For more information, see "sa_disk_free_space system procedure" [*ASA SQL Reference,* page 791].

Additional files

Other files can also become part of a database system, including:

- ♦ **Additional database files** You can spread your data over several separate files. These additional files are called dbspaces.

  ☞ For information on dbspaces, see "CREATE DBSPACE statement" [*ASA SQL Reference,* page 344].

- **Transaction log mirror files** For additional security, you can create a mirror copy of the transaction log. This file typically has the extension *.mlg*.

  ☞ For information on mirrored transaction logs, see "Transaction log mirrors" on page 379.

Using additional dbspaces

This section describes how to use additional database files, known as dbspaces.

Typically needed for large databases
For most databases, a single database file is sufficient. However, for users of large databases, additional database files are sometimes necessary. Additional database files are also convenient tools for clustering related information in separate files.

When you initialize a database, it contains one database file. This first database file is called the **main file**. All database objects and all data are placed, by default, in the main file.

Each database file has a maximum allowable size of 256M database pages. For example, a database file created with a database page size of 4 KB can grow to a maximum size of one terabyte (256M*4 KB). However, in practice, the maximum file size allowed by the physical file system in which the file is created affects the maximum allowable size significantly.

While many commonly-employed file systems restrict file size to a maximum of 2 GB, some, such as the Windows NT/2000/XP file system, allow you to exploit the full database file size. In scenarios where the amount of data placed in the database exceeds the maximum file size, it is necessary to divide the data into more than one database file. As well, you may wish to create multiple dbspaces for reasons other than size limitations, for example to cluster related objects.

Splitting existing databases

If you wish to split existing database objects among several dbspaces, you need to unload your database and modify the generated command file for rebuilding the database. To do so, add IN clauses to specify the dbspace for each table you do not wish to place in the main file.

☞ For more information, see "UNLOAD TABLE statement" [*ASA SQL Reference*, page 648].

Creating a dbspace

You create a new database file, or **dbspace**, either from Sybase Central, or using the CREATE DBSPACE statement. The database file for a new dbspace may be on the same disk drive as the main file or on another disk drive. You must have DBA authority to create dbspaces.

For each database, you can create up to twelve dbspaces in addition to the main dbspace.

Placing tables in dbspaces

A newly-created dbspace is empty. When you create a new table you can place it in a specific dbspace with an IN clause in the CREATE TABLE statement. If you don't specify an IN clause, the table appears in the main dbspace.

Each table is entirely contained in the dbspace it is created in. By default, indexes appear in the same dbspace as their table, but you can place them in a separate dbspace by supplying an IN clause.

❖ To create a dbspace (Sybase Central)

1. Open the Dbspaces folder for that database.
2. From the File menu, choose New ➤ Dbspace.

   The Dbspace Creation wizard appears.
3. Follow the instructions in the wizard.

   The new dbspace then appears in the Dbspaces folder.

❖ To create a dbspace (SQL)

1. Execute a CREATE DBSPACE statement.

Examples

The following command creates a new dbspace called **library** in the file *library.db* in the same directory as the main file:

```
CREATE DBSPACE library
AS 'library.db'
```

The following command creates a table LibraryBooks and places it in the library dbspace.

```
CREATE TABLE LibraryBooks (
title char(100),
author char(50),
isbn char(30)
) IN library
```

See also

- ♦ "CREATE DBSPACE statement" [*ASA SQL Reference,* page 344]
- ♦ "Creating tables" [*ASA SQL User's Guide,* page 40]
- ♦ "CREATE INDEX statement" [*ASA SQL Reference,* page 368]

Pre-allocating space for database files

Adaptive Server Anywhere automatically grows database files as needed. Rapidly-changing database files can lead to excessive file fragmentation on the disk, resulting in potential performance problems. Unless you are working with a database with a high rate of change, you do not need to worry about explicitly allocating space for database files. If you are working with a database with a high rate of change, you may pre-allocate disk space for dbspaces or for transaction logs using either Sybase Central or the ALTER DBSPACE statement.

You must have DBA authority to alter the properties of a database file.

Performance Tip
Running a disk defragmentation utility after pre-allocating disk space helps ensure that the database file is not fragmented over many disjoint areas of the disk drive. Performance can suffer if there is excessive fragmentation of database files.

❖ To pre-allocate space (Sybase Central)

1. Open the Dbspaces folder.
2. Right-click the desired dbspace and choose Pre-allocate Space from the popup menu.
3. Enter the amount of space to add to the dbspace. You can add space in units of pages, kilobytes (KB), megabytes (MB), gigabytes (GB), or terabytes (TB).
4. Click OK.

❖ To pre-allocate space (SQL)

1. Connect to a database.
2. Execute an ALTER DBSPACE statement.

Examples

- Increase the size of the SYSTEM dbspace by 200 pages.

```
ALTER DBSPACE system
ADD 200
```

- Increase the size of the SYSTEM dbspace by 400 megabytes.

```
ALTER DBSPACE system
ADD 400 MB
```

See also

- "Creating a dbspace" on page 290
- "ALTER DBSPACE statement" [*ASA SQL Reference*, page 270]

Deleting a dbspace

You can delete a dbspace using either Sybase Central or the DROP DBSPACE statement. Before you can delete a dbspace, you must delete all tables that use the dbspace. You must have DBA authority to delete a dbspace.

❖ To delete a dbspace (Sybase Central)

1. Open the Dbspaces folder.
2. Right-click the desired dbspace and choose Delete from the popup menu.

❖ To delete a dbspace (SQL)

1. Connect to a database.
2. Execute a DROP DBSPACE statement.

See also

- ♦ "Deleting tables" [*ASA SQL User's Guide*, page 44]
- ♦ "DROP statement" [*ASA SQL Reference*, page 454]

Working with write files (deprecated)

Deprecated feature
The use of write files is deprecated.

If you have a read-only database file (for example, if you distribute a database on a CD-ROM), you can use a write file to make local changes to the database.

You create a write file using the Create Write File utility [dbwrite] or using the CREATE WRITEFILE statement. In this section, the examples use the utility.

☞ For a description of the CREATE WRITEFILE statement, see "CREATE WRITEFILE statement (deprecated)" [*ASA SQL Reference*, page 430].

☞ For more information about opening a database as read-only to prevent local changes to the database, see "-r server option" on page 154.

❖ To use a write file

1. Create the write file for your database.

   For example, to create a write file for the sample database, execute the following command in a directory containing a copy of the sample database file *asademo.db*:

   ```
   dbwrite -c asademo.db
   ```

   This command creates a write file named *asademo.wrt*, with a transaction log named *asademo.wlg*.

2. Start a database server and load the write file. By default, the server locates files with the extension *.wrt* first, so the following command starts the personal server running the sample database write file:

   ```
   dbeng9 asademo
   ```

   Messages on the server window indicate which file starts.

3. Connect to the database using Interactive SQL. You can use the user ID **DBA** and the password **SQL**, as the sample database is the default.

4. Execute queries as usual. For example, the following query lists the contents of the department table.

   ```
   SELECT *
   FROM department
   ```

   The data for the department table is obtained from the database file *asademo.db*.

5. Try inserting a row. The following statement inserts a row into the department table:

   ```
   INSERT
   INTO department (dept_id, dept_name)
   VALUES (202, 'Eastern Sales')
   ```

   If you committed this change, it would be written to the *asademo.wlg* transaction log, and when the database checkpoints, the changes are written to the *asademo.wrt* write file.

   If you now query the department table, the results come from the write file and the database file.

6. Try deleting a row. Set the WAIT_FOR_COMMIT option to avoid referential integrity complaints here:

   ```
   SET TEMPORARY OPTION wait_for_commit = 'on' ;
   DELETE
   FROM department
   WHERE dept_id = 100
   ```

   If you committed this change, the deletion would be marked in the write file. No changes occur to the database file.

For some purposes, it is useful to use a write file with a shared database. If, for example, you have a read-only database on a network server, each user could have their own write file. In this way, they could add local information, which would be stored on their own machine, without affecting the shared database. This approach can be useful for application development also.

Deleting a write file

You can use the dberase utility to delete a write file and its associated transaction log.

Using the utility database

The **utility database** is a phantom database with no physical representation. The utility database has no database file, and therefore it cannot contain data.

The utility database feature allows you to execute database file administration statements such as CREATE DATABASE without first connecting to an existing physical database.

For example, executing the following statement after having connected to the utility database creates a database named *new.db* in the directory *C:\temp*.

```
CREATE DATABASE 'C:\\temp\\new.db'
```

For more information on the syntax of these statements, see "CREATE DATABASE statement" [*ASA SQL Reference*, page 338].

You can also retrieve values of connection properties and server properties using the utility database.

For example, executing the following statement against the utility database returns the default collation sequence, which will be used when creating a database:

```
SELECT property( 'DefaultCollation' )
```

☞ For a list of connection properties and server properties, see "Database properties" on page 713.

Allowed statements for the utility database

The following statements are the only ones allowed when connected to the utility database:

- ALTER DATABASE *dbfile* MODIFY TRANSACTION LOG
- CREATE DATABASE
- CREATE DECRYPTED FILE
- CREATE ENCRYPTED FILE
- DROP DATABASE
- RESTORE DATABASE
- START DATABASE
- STOP DATABASE
- STOP ENGINE
- SELECT (with no FROM or WHERE clauses)

Connecting to the utility database

You can start the utility database on a database server by specifying **utility_db** as the database name when connecting to the server. The user ID and password requirements are different for the personal server and the network server.

For the personal database server, there are no security restrictions for connecting to the utility database. It is assumed that anybody who can connect to the personal database server can access the file system directly so no attempt is made to screen users based on passwords.

☞ For more information, see "Utility database passwords" on page 298.

❖ To connect to the utility database on the personal server (Interactive SQL)

1. Start a database server with the following command:

   ```
   dbeng9.exe -n TestEng
   ```

2. Start Interactive SQL.

3. In the Connect dialog, enter **DBA** as the user ID, and enter any non-blank password. The password itself is not checked, but it must be non-empty.

4. On the Database tab, enter **utility_db** as the database name and **TestEng** as the server name.

5. Click OK to connect.

   Interactive SQL connects to the utility database on the personal server named **TestEng**, without loading a real database.

For the network server, there are security restrictions on connections. A password is held in the file *util_db.ini* in the same directory as the database server executable.

To protect the contents of the *util_db.ini* file from casual direct access, you can add simple encryption to the file using the File Hiding utility (dbfhide).

☞ For more information about obfuscating .ini files, see "Hiding the contents of .ini files" on page 524.

❖ **To connect to the utility database on the network server (Interactive SQL)**

1. Add a password to the file *util_db.ini* in the same directory as the database server executable. For example, the following *util_db.ini* file has the password ASA.

   ```
   [UTILITY_DB]
   PWD=ASA
   ```

2. Start a database server with the following command:

   ```
   dbsrv9.exe -n TestEng
   ```

3. Start Interactive SQL.

4. In the Connect dialog, enter **DBA** as the user ID, and enter the password held in the file *util_db.ini*.

5. On the Database tab, enter **utility_db** as the database name.

6. Click OK to connect.

   Interactive SQL connects to the utility database on the network server named **TestEng**, without loading a real database.

☞ For more information, see "Connecting to a Database" on page 37.

Utility database server security

There are two aspects to utility database server security:

♦ Who can connect to the utility database? This is controlled by the use of passwords.

♦ Who can execute file administration statements? This is controlled by database server options.

Utility database passwords

The personal server and the network server have different security models for connections.

For the personal server, you must specify the user ID **DBA**. You must also specify a password, but it can be any password. Since the personal server is for single machine use, security restrictions (for example passwords) are unnecessary.

For the network server, you must specify the user ID **DBA**, and the password that is held in the *util_db.ini* file, stored in the same directory as the database server executable file. As this directory is on the server, you can control access to the file, and thereby control who has access to the password.

The util_db.ini file

The *util_db.ini* file has the following contents:

```
[UTILITY_DB]
PWD=password
```

Use of the **utility_db** security level relies on the physical security of the computer hosting the database server, since the *util_db.ini* file can be easily read using a text editor.

To help protect the contents of the *util_db.ini* file, you can obfuscate it with simple encryption using the File Hiding utility (dbfhide).

☞ For more information about obfuscating .ini files, see "Hiding the contents of .ini files" on page 524.

Permission to execute file administration statements

A level of security is provided for the ability to execute certain administration tasks. The -gu database server option controls who can execute the file administration statements.

There are four levels of permission for the use of file administration statements. These levels include: **all**, **none**, **DBA**, and **utility_db**. The **utility_db** level permits only a person with authority to connect to the utility database to use the file administration statements.

-gu option	Effect	Applies to
All	Anyone can execute file administration statements	Any database including utility database
None	No one can execute file administration statements	Any database including utility database
Dba	Only DBA-authority users can execute file administration statements	Any database including utility database
Utility_db	Only the users who can connect to utility database can execute file administration statements	Only the utility database

☞ For more information on the database server –gu option, see "-gu server option" on page 148.

Examples

- To prevent the use of the file administration statements, start the database server using the **none** permission level of the –gu option. The following command starts a database server and names it **TestSrv**. It loads the sample database, but prevents anyone from using that server to create or delete a database, or execute any other file administration statement regardless of their resource creation rights, or whether or not they can load and connect to the utility database.

  ```
  dbsrv9.exe -n TestSrv -gu none asademo.db
  ```

- To permit only the users knowing the utility database password to execute file administration statements, start the server at the command prompt with the following command.

  ```
  dbsrv9 -n TestSrv -gu utility_db
  ```

  Assuming the utility database password has been set during installation to **asa**, the following command starts the Interactive SQL utility as a client application, connects to the server named **TestSrv**, loads the utility database, and connects the user.

  ```
  dbisql -c "uid=DBA;pwd=asa;dbn=utility_db;eng=TestSrv"
  ```

  Having executed the above command successfully, the user connects to the utility database, and can execute file administration statements.

CHAPTER 10

Automating Tasks Using Schedules and Events

About this chapter

This chapter describes how to use scheduling and event handling features of Adaptive Server Anywhere to automate database administration and other tasks.

Contents

Introduction

Many database administration tasks are best carried out systematically. For example, a regular backup procedure is an important part of proper database administration procedures.

You can automate routine tasks in Adaptive Server Anywhere by adding an **event** to a database, and providing a schedule for the event. Whenever one of the times in the schedule passes, the database server executes a sequence of actions called an **event handler**.

Database administration also requires taking action when certain conditions occur. For example, it may be appropriate to e-mail a notification to a system administrator when a disk containing the transaction log is filling up so that the administrator can handle the situation. These tasks too can be automated by defining event handlers for one of a set of **system events**.

Chapter contents

This chapter contains the following material:

- An introduction to scheduling and event handling (this section).
- Concepts and background information to help you design and use schedules and event handlers:
  - "Understanding schedules" on page 305.
  - "Understanding system events" on page 307.
- A discussion of techniques for developing event handlers:
  - "Developing event handlers" on page 311.
- Internals information:
  - "Schedule and event internals" on page 313.
- Step by step instructions for how to carry out automation tasks.
  - "Event handling tasks" on page 315.

Questions and answers

To answer the question...	**Consider reading...**
What is a schedule?	"Understanding schedules" on page 305.
What is an event?	"Understanding events" on page 304
What is a system event?	"Understanding system events" on page 307
What is an event handler?	"Understanding event handlers" on page 311

To answer the question. . .	**Consider reading. . .**
How do I debug event handlers?	"Developing event handlers" on page 311
How does the database server use schedules to trigger event handlers?	"How the database server checks for scheduled events" on page 313
How can I schedule regular backups?	For an example, see "Understanding schedules" on page 305.
What kind of system events can the database server use to trigger event handlers?	"Understanding system events" on page 307. "CREATE EVENT statement" [*ASA SQL Reference*, page 351].
What connection do event handlers get executed on?	"How event handlers are executed" on page 314.
How do event handlers get information about what triggered them?	"Developing event handlers" on page 311 "EVENT_PARAMETER function [System]" [*ASA SQL Reference*, page 150]

Understanding events

There are three types of events:

♦ **Scheduled events** have an associated schedule and execute at specified times.

♦ **System events** are associated with a particular type of condition that is tracked by the database server.

♦ **Manual events** are fired explicitly using the TRIGGER EVENT statement.

Each event type is described in more detail in the following sections.

Understanding schedules

By scheduling activities you can ensure that a set of actions is executed at a set of preset times. The scheduling information and the event handler are both stored in the database itself.

Although this is not usually necessary, you can define complex schedules by associating more than one schedule with a named event. For example, a retail outlet might want an event to occur once per hour during hours of operation, where the hours of operation vary based on the day of the week. You can achieve the same effect by defining multiple events, each with its own schedule, and by calling a common stored procedure.

The following examples give some ideas for scheduled actions that may be useful.

When scheduling events, you can use either full-length English day names (Monday, Tuesday, and so on) or the abbreviated forms of the day (Mon, Tue, and so on). Note that you must use the full-length English day names if you want the day names to be recognized by a server running in a language other than English.

☞ For more information, see "CREATE EVENT statement" [*ASA SQL Reference,* page 351].

Examples

Carry out an incremental backup daily at 1:00 am:

```
CREATE EVENT IncrementalBackup
SCHEDULE
   START TIME '1:00 AM' EVERY 24 HOURS
HANDLER
BEGIN
   BACKUP DATABASE DIRECTORY 'c:\\backup'
   TRANSACTION LOG ONLY
   TRANSACTION LOG RENAME MATCH
END
```

Summarize orders at the end of each business day:

```
CREATE EVENT Summarize
SCHEDULE
   START TIME '6:00 pm'
   ON ( 'Monday', 'Tuesday', 'Wednesday', 'Thursday',
      'Friday' )
HANDLER
BEGIN
   INSERT INTO DBA.OrderSummary
      SELECT current date,
             count( * ),
             sum( amount )
      FROM DBA.Orders
      WHERE date_ordered = current date
END
```

Defining schedules

Schedule definitions have several components to them, to permit flexibility:

- **Name** Each schedule definition has a name. You can assign more than one schedule to a particular event, which can be useful in designing complex schedules.
- **Start time** You can define a start time for the event, which is the time that it is first executed.
- **Range** As an alternative to a start time, you can specify a range of times for which the event is active. The event occurs between the start and end time specified. Frequency is determined by the specified recurrence.
- **Recurrence** Each schedule can have a recurrence. The event is triggered on a frequency that can be given in hours, minutes, or seconds on a set of days that can be specified as days of the week or days of the month. Recurring events include an **EVERY** or **ON** clause.

Understanding system events

The database server tracks several kinds of system events. Event handlers are triggered when the system event is checked by the database server, and satisfies a provided **trigger condition**.

By defining event handlers to execute when a chosen system event occurs and satisfies a trigger condition that you define, you can improve the security and safety of your data, and help ease administration.

☞ For more information on the available system events, see "Choosing a system event" on page 307.

☞ For more information on trigger conditions, see "Defining trigger conditions for events" on page 308.

Choosing a system event

Adaptive Server Anywhere tracks several system events. Each system event provides a hook on which you can hang a set of actions. The database server tracks the events for you, and executes the actions (as defined in the event handler) when needed.

The available system events include the following:

- ♦ **Backup** You can use the **BackupEnd** event type to take action at the end of a backup.
- ♦ **DatabaseStart** You can use the **DatabaseStart** event type to take action when a database is started.
- ♦ **Connection events** When a connection is made (**Connect**) or when a connection attempt fails (**ConnectFailed**). You may want to use these events for security purposes.
- ♦ **Disconnect** You can use the **Disconnect** event to take action when a user or application disconnects.
- ♦ **Free disk space** Tracks the available disk space on the device holding the database file (**DBDiskSpace**), the log file (**LogDiskSpace**), or temporary file (**TempDiskSpace**). This system event is not available on the following operating systems:
  - Windows 95 before OSR2
  - Windows CE

  You may want to use disk space events to alert administrators in case of a disk space shortage.

You can specify the -fc option when starting the database server to implement a callback function when the database server encounters a file system full condition.

☞ For more information, see "-fc server option" on page 139.

- **File size** The file reaches a specified size. This can be used for the database file (**GrowDB**), the transaction log (**GrowLog**), or the temporary file (**GrowTemp**).

  You may want to use file size events to track unusual actions on the database, or monitor bulk operations.

- **GlobalAutoIncrement** When the number of remaining values for a column defined with GLOBAL AUTOINCREMENT is less than a percentage of its range, the **GlobalAutoIncrement** event fires. This can be used to request a new value for the GLOBAL_DATABASE_ID option based on the table and number of remaining values that are supplied as parameters to this event. You can use the event_condition function with RemainingValues as an argument for this event type.

- **SQL errors** When an error is triggered, you can use the **RAISERROR** event type to take actions.

- **Idle time** The database server has been idle for a specified time (**ServerIdle**). You may want to use this event type to carry out routine maintenance operations at quiet times.

For information about creating events, see "CREATE EVENT statement" [*ASA SQL Reference,* page 351].

Defining trigger conditions for events

Each event definition has a system event associated with it. It also has one or more trigger conditions. The event handler is triggered when the trigger conditions for the system event are satisfied.

The trigger conditions are included in the WHERE clause of the CREATE EVENT statement, and can be combined using the AND keyword. Each trigger condition is of the following form:

event_condition(*condition-name*) *comparison-operator value*

The *condition-name* argument is one of a set of preset strings, which are appropriate for different event types. For example, you can use **DBSize** (the database file size in megabytes) to build a trigger condition suitable for the **GrowDB** system event. The database server does not check that the condition-name matches the event type: it is your responsibility to ensure that the condition is meaningful in the context of the event type.

Examples

- Limit the transaction log size to 10 Mb:

```
CREATE EVENT LogLimit
TYPE GrowLog
WHERE event_condition( ’LogSize’ ) > 10
HANDLER
BEGIN
  IF event_parameter( ’NumActive’ ) <= 1 THEN
   BACKUP database
     DIRECTORY ’c:\\logs’
     TRANSACTION LOG ONLY
     TRANSACTION LOG RENAME MATCH;
  END IF;
END
```

- Notify an administrator when free disk space on the device containing the database file falls below 10%, but do not execute the handler more than once every five minutes (300 seconds):

```
CREATE EVENT LowDBSpace
TYPE DBDiskSpace
WHERE event_condition( ’DBFreePercent’ ) < 10
AND event_condition( ’Interval’ ) >= 300
HANDLER
BEGIN
   CALL xp_sendmail( recipient=’DBAdmin’,
            subject=’Low disk space’,
            "message"=’Database free disk space ’
            || event_parameter( ’DBFreeSpace’ ) );
END
```

- Notify an administrator of a possible attempt to break into the database:

```
CREATE EVENT SecurityCheck
TYPE ConnectFailed
HANDLER
BEGIN
   DECLARE num_failures INT;
   DECLARE mins INT;
   INSERT INTO FailedConnections( log_time )
   VALUES ( current timestamp );

   SELECT count( * ) INTO num_failures
   FROM FailedConnections
   WHERE log_time >= dateadd( minute, -5,
      current timestamp );
   IF( num_failures >= 3 ) THEN
      SELECT datediff( minute, last_notification,
         current timestamp ) INTO mins
      FROM Notification;
```

```
         IF( mins > 30 ) THEN
            UPDATE Notification
            SET last_notification = current timestamp;
      CALL xp_sendmail( recipient='DBAdmin',
                        subject='Security Check',
                     "message"=
      'over 3 failed connections in last 5 minutes' )
         END IF
      END IF
   END
```

- Run a process when the server has been idle for ten minutes. Do not execute more frequently than once per hour:

```
CREATE EVENT Soak
TYPE ServerIdle
WHERE event_condition( 'IdleTime' ) >= 600
AND event_condition( 'Interval' ) >= 3600
HANDLER
BEGIN
   MESSAGE ' Insert your code here ... '
END
```

Understanding event handlers

Event handlers execute on a separate connection from the action that triggered the event, and so do not interact with client applications. They execute with the permissions of the creator of the event.

Developing event handlers

Event handlers, whether for scheduled events or for system event handling, contain compound statements, and are similar in many ways to stored procedures. You can add loops, conditional execution, and so on, and you can use the Adaptive Server Anywhere debugger to debug event handlers.

Context information for event handlers

One difference between event handlers and stored procedures is that event handlers do not take any arguments. Certain information about the context in which an event was triggered is available through the event_parameter function, which supplies information about the connection that caused an event to be triggered (connection ID, user ID), as well as the event name and the number of times it has been executed.

☞ For more information, see "EVENT_PARAMETER function [System]" [*ASA SQL Reference*, page 150].

Testing event handlers

During development, you want event handlers to be triggered at convenient times. You can use the TRIGGER EVENT statement to explicitly cause an event to execute, even when the trigger condition or scheduled time has not occurred. However, TRIGGER EVENT does not cause disabled event handlers to be executed.

☞ For more information, see "TRIGGER EVENT statement" [*ASA SQL Reference*, page 641].

While it is not good practice to develop event handlers on a production database, you can disable event handlers from Sybase Central or explicitly using the ALTER EVENT statement.

It can be useful to use a single set of actions to handle multiple events. For example, you may want to take a notification action if disk space is limited on any of the devices holding the database or log files. To do this, create a stored procedure and call it in the body of each event handler.

Debugging event handlers

Debugging event handlers is very similar to debugging stored procedures. The event handlers appear in the events list.

You can also determine how many instances of a particular event handler are currently active using the NumActive event parameter. This function is useful if you want to limit an event handler so that only one instance executes at any given time.

☞ For more information about the NumActive event parameter, see "EVENT_PARAMETER function [System]" [*ASA SQL Reference,* page 150].

☞ For more information about step-by-step instructions, see "Debugging an event handler" on page 317.

Schedule and event internals

This section describes how the database server processes schedules and event definitions.

How the database server checks for system events

System events are classified according to their **event type**, as specified directly in the CREATE EVENT statement or using Sybase Central. There are two kinds of event types:

- **Active event types** Some event types are the result of action by the database server itself. These active event types include growing database files, or the start and end of different database actions (**BackupEnd** and so on) or RAISERROR.

  When the database server takes the action, it checks to see whether the trigger conditions defined in the WHERE clause are satisfied, and if so, triggers any events defined for that event type.

- **Polled event types** Some event types are not triggered solely by database actions. The free disk space types (**DBDiskSpace** and so on), as well as the **IdleTime** type, are of this kind.

  For these types of events, the database server polls every thirty seconds, starting approximately thirty seconds after the database server is started.

  For the **IdleTime** event type, the database server checks whether the server has been idle for the entire thirty seconds. If no requests have started and none are currently active, it adds the idle check interval time in seconds to the idle time total; otherwise, the idle time total is reset to 0. The value for **IdleTime** is therefore always a multiple of thirty seconds. When **IdleTime** is greater than the interval specified in the trigger condition, event handlers associated with **IdleTime** are fired.

How the database server checks for scheduled events

The calculation of scheduled event times is done when the database server starts, and each time a scheduled event handler completes.

The calculation of the next scheduled time is based on the increment specified in the schedule definition, with the increment being added to the previous start time. If the event handler takes longer to execute than the specified increment, so that the next time is earlier than the current time, the database server increments until the next scheduled time is in the future.

An event handler that takes sixty-five minutes to execute and is requested to run every hour between 9:00 and 5:00 will run every two hours, at 9:00, 11:00, 1:00, and so on.

To run a process such that it operates between 9:00 and 5:00 and delays for some period before the next execution, you could define a handler to loop until its completion time has passed, with a WAITFOR statement between each iteration.

If you are running a database server intermittently, and it is not running at a scheduled time, the event handler does not run at startup. Instead, the next scheduled time is computed at startup. If, for example, you schedule a backup to take place every night at one o'clock, but regularly shut down the database server at the end of each work day, the backup never takes place.

If the next scheduled execution of an event is more than one hour away, the database server will recalculate its next scheduled time on an hourly basis. This allows events to fire when expected when the system clock is adjusted because of a change to or from Daylight Saving Time.

How event handlers are executed

When an event handler is triggered, a temporary internal connection is made on which the event handler is executed. The handler is *not* executed on the connection that caused the handler to be triggered, and consequently statements such as MESSAGE ... TO CLIENT, which interact with the client application, are not meaningful within event handlers. Similarly, statements that return result sets are not permitted.

The temporary connection on which the handler is executed does not count towards the connection limit for licensing purposes.

Event creation requires DBA authority, and events execute with the permissions of their creator. If you wish event handlers to execute with non-DBA authority, you can call a procedure from within the handler, as stored procedures run with the permissions of their creator.

Any event errors are logged to the server console.

Event handling tasks

This section collects together instructions for tasks related to automating tasks with events.

Adding an event to a database

Events are handled in a similar fashion, both from Sybase Central and in SQL.

☞ For more information, see “Understanding schedules” on page 305, and “Understanding system events” on page 307.

❖ To add an event to a database (Sybase Central)

1. Connect to the database as a user with DBA authority.
2. Click the Events folder for your database.
3. From the File menu, choose New ➤ Event.

   The Event Creation wizard appears.
4. Follow the instructions in the wizard.

   The wizard contains many options, depending on the event you wish to create. These are explained in detail in other tasks.

❖ To add an event to a database (SQL)

1. Connect to the database as a user with DBA authority.
2. Execute a CREATE EVENT statement.

   The CREATE EVENT statement contains many options, depending on the event you wish to create. These are explained in detail in other tasks.

   ☞ For more information, see “CREATE EVENT statement” [*ASA SQL Reference,* page 351].

Adding a manually-triggered event to a database

If you create an event handler without a schedule or system event to trigger it, it is executed only when manually triggered.

❖ To add a manually-triggered event to a database (Sybase Central)

1. Connect to the database as a user with DBA authority.
2. Click the Events folder for your database.
3. From the File menu, choose New ➤ Event.

   The Event Creation wizard appears.
4. Type a name for the event, and click Next.
5. Select Manually, and click Next.
6. Type the SQL statements for your event handler, and click Next.
7. Select Enable This Event, and select Execute at All Locations, and click Next.
8. Type a comment describing the event, and click Finish to add the event to the database.

If you wish to accept the default values for all remaining options, you can click Finish at an earlier stage in the wizard.

❖ To add a manually-triggered event to a database (SQL)

1. Connect to the database as a user with DBA authority.
2. Execute a CREATE EVENT statement with no schedule or WHERE clause. The restricted syntax of the CREATE EVENT is as follows:

   CREATE EVENT *event-name* **HANDLER BEGIN** . . .
   event handler **END**

If you are developing event handlers, you can add schedules or system events to control the triggering of an event later, either using Sybase Central or the ALTER EVENT statement.

☞ See also:

- For information on triggering events, see "Triggering an event handler" on page 316.
- For information on altering events, see "ALTER EVENT statement" [*ASA SQL Reference,* page 273].

Triggering an event handler

Any event handler can be triggered manually, in addition to those occasions when it executes because of a schedule or system event. Triggering events manually can be useful during development of event handlers, and also, for certain events, in production environments. For example, you may have a monthly sales report scheduled, but from time to time you may want to obtain a sales report for a reason other than the end of the month.

☞ For more information on developing event handlers, see "Developing event handlers" on page 311.

❖ To trigger an event handler (Sybase Central)

1. Connect to the database as a user with DBA authority.

2. Open the Events folder for your database.

3. Right-click the event you wish to trigger and choose Trigger from the popup menu. (The event must be enabled before you can select Trigger from the popup menu. You can enable the event on the General tab of the Event property sheet.)

   The Trigger Event dialog appears.

4. Supply any parameters the event handler requires, in a comma-separated list, as follows:

   ```
   parameter=value,parameter=value
   ```

   Click OK to trigger the event handler.

❖ To trigger an event handler (SQL)

1. Connect to the database as a user with DBA authority.

2. Execute the TRIGGER EVENT statement, supplying the name of the event. For example:

   ```
   TRIGGER EVENT sales_report_event
   ```

   ☞ For more information, see "TRIGGER EVENT statement" [*ASA SQL Reference*, page 641].

Debugging an event handler

Debugging is a regular part of any software development. Event handlers can be debugged during the development process.

☞ For more information on developing event handlers, see "Developing event handlers" on page 311.

☞ For more information on using the debugger, see "Debugging Logic in the Database" [*ASA SQL User's Guide*, page 719].

❖ To debug an event handler

1. Start Sybase Central.

   From the Start menu, choose Programs ➤ SQL Anywhere 9 ➤ Sybase Central.

2. Connect to your database.

3. Change to Debug mode.

   From the Task menu, choose Debug. The Debugger Details pane appears at the bottom of the Sybase Central main window.

4. In the Folders pane, open the Events folder.

5. Click the name of the event you wish to debug.

6. In the SQL pane, click the left margin beside the line where you want to add a breakpoint. A red stop signs appears, to indicate that a breakpoint has been set. Alternatively, you can press F9 to set a breakpoint.

7. From Interactive SQL or another application, trigger the event handler using the TRIGGER EVENT statement.

8. The execution stops at the breakpoint you have set. You can now use the debugger features to trace execution, local variables, and so on.

CHAPTER 11

International Languages and Character Sets

About this chapter

This chapter describes how to configure your Adaptive Server Anywhere installation to handle international language issues.

Contents

Introduction to international languages and character sets

This section provides an introduction to the issues you may face when working in an environment that uses more than one character set, or when using languages other than English.

When you create a database, you specify a collating sequence, or collation, for the database to use. A collation is a combination of a character set and a sort order for characters in the database.

Localized versions of Adaptive Server Anywhere

Adaptive Server Anywhere is available in the following fully localized (packages, software, documentation, books) languages:

- English
- German
- French
- Japanese
- Simplified Chinese

As well, Adaptive Server Anywhere is available in the following deployment (software only) languages:

- Italian
- Korean
- Lithuanian
- Polish
- Portuguese (Brazilian)
- Russian
- Spanish
- Traditional Chinese
- Ukrainian

Adaptive Server Anywhere international features

Adaptive Server Anywhere provides two sets of features that are of particular interest when setting up databases for languages.

♦ **Collations** You can choose from a wide selection of supplied collations when you create a database. By creating your database with the proper collation, you ensure proper sorting of data.

  Whenever the database compares strings, sorts strings, or carries out other string operations such as case conversion, it does so using the collation sequence. The database carries out sorting and string comparison when statements such as the following are executed:

  - Queries with an ORDER BY clause.
  - Expressions that use string functions, such as LOCATE, SIMILAR, and SOUNDEX.
  - Conditions using the LIKE keyword.
  - Queries that use index lookups on character data.

  The database also uses collations to identify valid or unique identifiers (column names and so on).

♦ **Character set translation** You can set up Adaptive Server Anywhere to convert data between the character set encoding on your server and client systems, thus maintaining the integrity of your data, even in mixed character set environments.

  Character set translation is provided between client and server, as well as by the Adaptive Server Anywhere ODBC driver. OLE DB and ADO.NET also provide it; all are Unicode-enabled.

Using the default collation

If you use the default actions when creating a database, the Initialization utility infers a collation from the character set and language used by the operating system on the machine on which you create the database.

☞ For more information on how to find the default collation in your environment, see "Finding the default collation" on page 352.

If all the machines in your environment share the same character set, then the default collation minimizes the requirement for character set conversion. As long as the collation provides a proper character set and sort order for your data, using this default setting is a simple way of ensuring that characters are represented consistently throughout the system.

If it is not possible to set up your system in this default manner, you need to decide which collation to use in your database, and whether to use character set translation to ensure that data is exchanged consistently between the pieces of your database system. This chapter provides the information you need to make and implement these decisions.

Character set questions and answers

The following table identifies where you can find answers to questions.

To answer the question...	Consider reading...
How do I set up my computing environment to treat character sets properly?	"Configuring your character set environment" on page 352
How do I decide which collation to use for my database?	"Understanding collations" on page 335
How are characters represented in software, and Adaptive Server Anywhere in particular?	"Understanding character sets in software" on page 323
What collations does Adaptive Server Anywhere provide?	"Choosing collations" on page 335
How do I ensure that error and informational messages sent from the database server to client applications are sent in the proper language and character set for my application?	"Character translation for database messages" on page 343
I have a different character set on client machines from that in use in the database. How can I get characters to be exchanged properly between client and server?	"Starting a database server using character set translation" on page 356
What character sets can I use for connection strings?	"Connection strings and character sets" on page 343
How do I create a collation that is different from the supplied ones?	"Creating a database with a custom collation" on page 358
How do I change the collation sequence of an existing database?	"Changing a database from one collation to another" on page 359
How do I create a database for Windows CE?	"Creating databases for Windows CE" on page 341

Understanding character sets in software

This section provides general information about software issues related to international languages and character sets.

Pieces in the character set puzzle

There are several distinct aspects to character storage and display by computer software:

- ♦ Each piece of software works with a character set. A **character set** is a set of symbols, including letters, digits, spaces, and other symbols.
- ♦ To handle these characters, each piece of software employs a character set **encoding**, in which each character is mapped onto one or more bytes of information, typically represented as hexadecimal numbers. This encoding is also called a **code page**.
- ♦ Database servers, which sort characters (for example, list names alphabetically), use a collation. A **collation** is a combination of a character encoding (a map between characters and hexadecimal numbers) and a **sort order** for the characters. There may be more than one sort order for each character set; for example, a case sensitive order and a case insensitive order, or two languages may sort characters in a different order.
- ♦ Characters are printed or displayed on a screen using a **font**, which is a mapping between characters in the character set and its appearance. Fonts are handled by the operating system.
- ♦ Operating systems also use a **keyboard mapping** to map keys or key combinations on the keyboard to characters in the character set.

Language issues in client/server computing

Database users working at client applications may see or access strings from the following sources:

- ♦ **Data in the database** Strings and other text data are stored in the database. The database server processes these strings when responding to requests.

  For example, the database server may be asked to supply all the last names beginning with a letter ordered less than N in a table. This request requires string comparisons to be carried out, and assumes a character set ordering.

The database server receives strings from client applications as streams of bytes. It associates these bytes with characters according to the database character set. If the data is held in an indexed column, the index is sorted according to the sort order of the collation.

- **Database server software messages** Applications can cause database errors to be generated. For example, an application may submit a query that references a column that does not exist. In this case, the database server returns a warning or error message. This message is held in a **language resource library**, which is a DLL or shared library called by Adaptive Server Anywhere.
- **Client application** The client application interface displays text, and internally the client application may process text.
- **Client software messages** The client library uses the same language library as the database server to provide messages to the client application.
- **Operating system** The client operating system has text displayed on its interface, and may also process text.

For a satisfactory working environment, all these sources of text must work together. Loosely speaking, they must all be working in the user's language and/or character set.

Code pages

Many languages have few enough characters to be represented in a single-byte character set. In such a character set, each character is represented by a single byte: a two-digit hexadecimal number.

At most, 256 characters can be represented in a single byte. No single-byte character set can hold all of the characters used internationally, including accented characters. This problem was addressed by the development of a set of code pages, each of which describes a set of characters appropriate for one or more national languages. For example, code page 1253 contains the Greek character set, and code page 1252 contains an international character set suitable for representing many characters in a variety of languages. There are many code pages, and many names for code pages. The above examples are code pages for Microsoft Windows.

Upper and lower pages

With few exceptions, characters 0 to 127 are the same for all the single-byte code pages. The mapping for this range of characters is called the **ASCII** character set. It includes the English language alphabet in upper and lower case, as well as common punctuation symbols and the digits. This range is often called the **seven-bit** range (because only seven bits are needed to represent the numbers up to 127) or the **lower** page. The characters from 128 to 255 are called **extended characters**, or **upper** code-page characters, and vary from code page to code page.

Problems with code page compatibility are rare if the only characters used are from the English alphabet, as these are represented in the ASCII portion of each code page (0 to 127). However, if other characters are used, as is generally the case in any non-English environment, there can be problems if the database and the application use different code pages.

Example

Suppose a database holding French language strings uses code page 850, and the client operating system uses code page 437. The character À (upper case A grave) is held in the database as character \xB7 (decimal value 183). In code page 437, character \xB7 is a graphical character. The client application receives this byte and the operating system displays it on the screen, the user sees a graphical character instead of an A grave.

ANSI and OEM code pages in Windows

For PC users, the issue is complicated because there are at least two code pages in use on most PCs. Applications using the Windows graphical user interface use the Windows code pages. These code pages are compatible with ISO character sets, and also with ANSI character sets. They are often referred to as **ANSI code pages**.

Character-mode applications (those using the console or command prompt window) in Windows 95/98/Me and Windows NT/200/XP, use code pages that were used in DOS. These are called **OEM code pages** (Original Equipment Manufacturer) for historical reasons.

Adaptive Server Anywhere supports collations based on both OEM and ANSI code pages.

Example

Consider the following situation:

- ♦ A PC is running a Windows operating system with ANSI code page 1252.
- ♦ The code page for character-mode applications is OEM code page 437.
- ♦ Text is held in a database created using the collation UTF8.

An upper case A grave in the database is stored as hex byes C380. In a Windows application, the same character is represented as hex CO. In a DOS application, it is represented as hex B7.

Adaptive Server Anywhere provides character set conversion to handle these differences in representation.

☞ For more information about choosing a collation for your database, see "Understanding collations" on page 335.

Multibyte character sets

Some languages, such as Japanese and Chinese, have many more than 256 characters. These characters cannot all be represented using a single byte, but can be represented in multibyte character sets. In addition, some character sets use the much larger number of characters available in a multibyte representation to represent characters from many languages in a single, more comprehensive, character set.

Multibyte character sets are of two types. Some are **variable width**, in which some characters are single-byte characters, others are double-byte, and so on. Other sets are **fixed width**, in which all characters in the set have the same number of bytes.

☞ For more information on the multibyte character sets, see "Using multibyte collations" on page 342.

Example

As an example, characters in the Shift-JIS character set are either one or two bytes in length. If the value of the first byte is in the range of hexadecimal values from \x81 to \x9F or from \xE0 to \xEF (decimal values 129-159 or 224-239) the character is a two-byte character and the subsequent byte (called a follow byte) completes the character. A **follow byte** is any byte(s) other than the first byte.

If the first byte is outside this range, the character is a single-byte character and the next byte is the first byte of the following character.

- ♦ The properties of any Shift-JIS character can be read from its first byte also. Characters with a first byte in the range \x09 to \x0D, or \x20 are space characters.
- ♦ Characters in the ranges \x41 to \x5A, \x61 to \x7A, \x81 to \x9F or \xE0 to \xEF are considered to be alphabetic (letters).
- ♦ Characters in the range \x30 to \x39 are digits.

Sorting characters using collations

The database collation sequence includes the notion of alphabetic ordering of letters, and extends it to include all characters in the character set, including digits and space characters.

Associating more than one character with each sort position

More than one character can be associated with each sort position. This is useful if you wish, for example, to treat an accented character the same as the character without an accent.

Two characters with the same sort position are considered identical in all ways by the database. Therefore, if a collation assigns the characters *a* and *e* to the same sort position, then a query with the following search condition:

```
WHERE col1 = 'want'
```

is satisfied by a row for which **col1** contains the entry **went**.

At each sort position, lower case and upper case forms of a character can be indicated. For case sensitive databases, the lower case and upper case characters are not treated as equivalent. For case insensitive databases, the lower case and upper case versions of the character are considered equivalent.

First-byte collation orderings for multibyte character sets

A sorting order for characters in a multibyte character set can be specified only for the first byte. Characters that have the same first byte are sorted according to the hexadecimal value of the following bytes.

International aspects of case sensitivity

Adaptive Server Anywhere is always **case preserving** and **case insensitive** for identifiers, such as table names and column names. This means that the names are stored in the case in which they are created, but any access to the identifiers is done in a case insensitive manner.

For example, the names of the system tables are stored in upper case (SYSDOMAIN, SYSTABLE, and so on), but access is case insensitive, so that the two following statements are equivalent:

```
SELECT *
FROM systable

SELECT *
FROM SYSTABLE
```

The equivalence of upper and lower case characters is defined in the collation. There are some collations where particular care is required when assuming case insensitivity of identifiers. In particular, Turkish collations have a case-conversion behavior that can cause unexpected and subtle errors. The most common error is that a system object containing a letter **I** or **I** is not found.

Turkish character sets and collations

The Turkish language has two forms of what appears to be the letter **I**. One form, referred to as **I-dot**, appears as the following:

```
i, İ
```

The second form, referred to as **I-no-dot**, appears as the following:

```
ı, I
```

Even though these letters appear as variations of the same letter, in the Turkish alphabet they are considered to be distinct letters.

Turkish rules for case conversion of these characters are incompatible with ANSI SQL standard rules for case conversion. For example, Turkish says that the lower-case equivalent of `I` is

```
ı
```

while ANSI says that it is

```
I
```

For this reason, correct case-insensitive matching is dependent on whether or not the text being matched is Turkish or English/ANSI. In many contexts, there is not enough information to make such a distinction, which leads to some non-standard behaviors in such databases.

For example, consider the statements:

```
SELECT * FROM sysinfo    // actual table name is SYSINFO
SELECT * FROM fig        // actual table name is FİG
```

The first statement references a system object and must use ANSI SQL rules to match the name, while the second statement references a user object and must use the Turkish rules.

However, there is no context that the database server can use to determine which rules to use until after the object is found. So, to find the object, the case conversion rules must be known. To know the rules, the object must be found. The situation cannot be resolved properly for both system and user objects.

In this instance, Adaptive Server Anywhere uses the Turkish rules. The first statement will fail, while the second statement will succeed.

The incompatibility of Turkish and ANSI standards requires that system object references in Turkish databases specify the object name in the *correct* case, that is, the case used to create the object. The first statement above should be written as

```
SELECT * FROM SYSINFO
```

In fact, only the letter I must be in the correct case.

As an alternative, it is acceptable, although unusual, to write the statement as

```
SELECT * FROM sysınfo // I-no-dot
```

Note that keywords, such as INSERT, are case-insensitive even in Turkish databases. Adaptive Server Anywhere knows that all keywords use only English letters, so it uses ANSI case conversion rules when matching keywords. Adaptive Server Anywhere also applies this knowledge for certain other identifiers, such as built-in functions. However, objects whose names are stored in the catalog must be specified using the correct case or letter, as described above.

Data in case-insensitive Turkish databases

Similar rules govern data in case-insensitive Turkish databases. For example if a data value is

```
FİG
```

then a lower-case reference to that data should be

```
fig
```

so that the same I-dot character is used in both forms.

Alternative Turkish collation 1254TRKALT

For some application developers, the Turkish letter I problem can cause significant problems. While the correct solution is to ensure that all object references are in the proper case or that the proper form of the letter I is used, in some cases it may be more expedient to make a decision to violate the Turkish rules in favor of the ANSI rules.

In earlier versions of Adaptive Server Anywhere, this could be done by creating a custom collation and making I-dot and I-no-dot equivalent characters. Adaptive Server Anywhere provides the collation 1254TRKALT, which is identical to 1254TRK, except that it makes I-dot and I-no-dot equivalent characters.

It is important to understand the consequences of this change. In a 1254TRKALT database, the following strings are equal:

```
fig
fıg
```

This is not correct for a Turkish user, but may be acceptable in some cases.

The second issue appears when using ORDER BY. Consider the following strings:

```
ia
ıa
ıs
is
```

In a 1254TRK database, an ORDER BY of the strings would produce the following:

```
ıa
ıs
ia
is
```

because I-no-dot is less than I-dot. In a 1254TRKALT database, the order would be

```
ia
ıa
ıs
is
```

because I-no-dot is equal to I-dot.

Understanding locales

Both the database server and the client library recognize their language and character set environment using a **locale definition**.

Introduction to locales

The application locale, or client locale, is used by the client library when making requests to the database server, to determine the character set in which results should be returned. If character set translation is enabled (the default), the database server compares its own locale with the application locale to determine whether character set translation is needed. Different databases on a server may have different locale definitions.

The locale consists of the following components:

♦ **Language** The language is a two-character string using the ISO-639 standard values: DE for German, FR for French, and so on. Both the database server and the client have language values for their locale.

  The database server uses the locale language to determine the following behavior:

  - Which language library to load.
  - The language is used together with the character set to determine which collation to use when creating databases, if no collation is explicitly specified.

  The client library uses the locale language to determine the following behavior:

  - Which language library to load.
  - Which language to request from the database.

    ☞ For more information, see "Understanding the locale language" on page 331.

♦ **Character set** The character set is the code page in use. The client and server both have character set values, and they may differ. If they differ, character set translation may be required to enable interoperability.

  For machines that use both OEM and ANSI code pages, the ANSI code page is the value used here.

  ☞ For more information, see "Understanding the locale character set" on page 333.

Understanding the locale language

The locale language is an indicator of the language being used by the user of the client application, or expected to be used by users of the database server.

☞ For more information about how to find locale settings, see "Determining locale information" on page 353.

The client library determines the language component of the locale as follows:

1. On Windows, it checks the Adaptive Server Anywhere language registry entry, as described in "Registry settings on installation" on page 284.
2. On other operating systems, or if the registry setting is not present, it checks the operating system language setting.

The database server determines the language component of the locale as follows:

1. It checks the ASLANG environment variable, if it exists.

   ☞ For more information, see "ASLANG environment variable" on page 278.
2. On Windows, it checks the Adaptive Server Anywhere language registry entry, as described in "Registry settings on installation" on page 284.
3. Queries the operating system.
4. If the language cannot be determined by the above settings, it defaults to English.

Language label values

The following table displays the valid language label values, together with the equivalent ISO 639 labels:

Language label	Alternative label	ISO_639 language code
chinese	simpchin	ZH
czech	N/A	CS
danish	N/A	DA
dutch	N/A	NL
finnish	N/A	FI
french	N/A	FR
german	N/A	DE
greek	N/A	EL
hebrew	N/A	HE

Language label	Alternative label	ISO_639 language code
hungarian	N/A	HU
italian	N/A	IT
japanese	N/A	JA
korean	N/A	KO
lithuanian	N/A	LT
norwegian	norweg	NO
polish	N/A	PL
portuguese	portugue	PT
russian	N/A	RU
spanish	N/A	ES
swedish	N/A	SV
tchinese	tradchin	TW
turkish	N/A	TR
ukrainian	N/A	UK
us_english	english	EN

Understanding the locale character set

Both application and server locale definitions have a character set. The application uses its character set when requesting character strings from the server. If character set translation is enabled (the default), the database server compares its character set with that of the application to determine whether character set translation is needed.

☞ For more information about how to find locale settings, see "Determining locale information" on page 353.

The client library determines the character set as follows:

1. If the connection string specifies a character set, it is used.

   ☞ For more information, see "CharSet connection parameter [CS]" on page 180.

2. Open Client applications check the *locales.dat* file in the Sybase *locales* directory.

3. Character set information from the operating system is used to determine the locale:
   - On Windows operating systems, Windows ANSI character set.
   - On UNIX, default to ISO8859-1.
   - On other platforms, use code page 1252.

The database server determines the character set for a connection as follows:

1. The character set specified by the client is used if it is supported.

   ☞ For more information, see "CharSet connection parameter [CS]" on page 180.

2. The database's character set is used if the client specifies a character set that is not supported.

Understanding collations

This section describes the supplied collations, and provides suggestions as to which collations to use under certain circumstances.

☞ For more information about how to create a database with a specific collation, see "Creating a database with a named collation" on page 355 and "The Initialization utility" on page 530.

☞ For information on changing a database from one collation to another, see "Changing a database from one collation to another" on page 359.

Choosing collations

When you create a database, Adaptive Server Anywhere chooses a default collation based on operating system language and character set settings. In most cases, the default collation is a suitable choice, but you can also explicitly choose a collation to match your needs from the wide selection of supplied collations.

In some cases, Adaptive Server Anywhere supports more than one collation for a particular language. Different collations vary in the characters they include and in the way they sort special characters, including ligatures (characters consisting of two or more characters joined together) and accented characters. You should choose a collation that uses a character set and sort order that are appropriate for the data in your database.

☞ For more information about sorting of data and international features, see "Adaptive Server Anywhere international features" on page 321.

How Adaptive Server Anywhere chooses the collation for a new database

When a new database is created, and the collation is not explicitly specified, then Adaptive Server Anywhere uses the language and character set to determine the collation.

- The language comes from the ASLANG environment variable (if it exists) or the operating system.
- The character set comes from the ASCHARSET environment variable (if it exists) or the operating system.

Last resort collation
If Adaptive Server Anywhere cannot locate a specific collation based on your operating system's character set and language, the Adaptive Server Anywhere registry setting, or the ASCHARSET environment variable, ISO_BINENG becomes the default, regardless of which operating system you are using.

If this is not suitable for your requirements, you can rebuild the database and choose a different collation.

☞ For more information about changing collations, see "Changing a database from one collation to another" on page 359.

☞ For information about creating a database with a specific collation, see "Creating a database with a named collation" on page 355.

Supplied and recommended collations

The following collations are supplied with Adaptive Server Anywhere. You can obtain the list of main collations by typing the following command at the command prompt:

```
dbinit -l
```

Collation label	Description
874THAIBIN	Code Page 874, Windows Thai
932JPN	Code Page 932, Japanese Shift-JIS with Microsoft extensions
936ZHO	Code Page 936, Simplified Chinese, PRC GBK 2312-80 8-bit encoding
949KOR	Code Page 949, Korean KS C 5601-1987 encoding, Wansung
950ZHO_HK	Code Page 950, Traditional Chinese, Big 5 encoding with HKSCS
950ZHO_TW	Code Page 950, Traditional Chinese, Big 5 encoding
1250LATIN2	Code Page 1250, Windows Latin 2, Central/Eastern European
1250POL	Code Page 1250, Windows Latin 2, Polish
1251CYR	Code Page 1251, Cyrillic
1252LATIN1	Code Page 1252, Windows Latin 1, Western

Collation label	Description
1252SPA	Code Page 1252, Windows Latin 1, Spanish
1252SWEFIN	Code Page 1252, Windows Latin 1, Swedish/Finnish
1253ELL	Code Page 1253, Windows Greek, ISO8859-7 with extensions
1254TRK	Code Page 1254, Windows Latin 5, Turkish, ISO 8859-9 with extensions
1255HEB	Code Page 1255, Windows Hebrew, ISO8859-8 with extensions
1256ARA	Code Page 1256, Windows Arabic, ISO8859-6 with extensions
1257LIT	Code Page 1257, Lithuanian
EUC_JAPAN	Japanese EUC JIS X 0208-1990 and JIS X 0212-1990 Encoding
EUC_CHINA	Simplified Chinese GB 2312-80 Encoding
EUC_TAIWAN	Taiwanese Big 5 Encoding
EUC_KOREA	Korean KS C 5601-1992 Encoding, Johab
ISO_1	ISO8859-1, Latin 1, Western
ISO_BINENG	Binary ordering, English ISO/ASCII 7-bit letter case mappings
ISO1LATIN1	ISO8859-1, ISO Latin 1, Western, Latin 1 Ordering
ISO9LATIN1	ISO8859-15, ISO Latin 9, Western, Latin 1 Ordering
UTF8	Unicode Transformation Format-8, 8-bit multibyte encoding for Unicode

The following collations are recommended because they provide the most likely match to the character set being used by the application: they have an appropriate sort order and support for the symbols and accented characters required for each specific language. Some languages do not have a recommended UNIX collation.

Language	Windows collations	UNIX collations
Western European (including English, French, German, Italian, Portuguese, and Spanish)	1252LATIN1	ISO9LATIN1, ISO1LATIN1
Eastern European (including Croatian, Czech, Hungarian, Romanian, Serbian, Slovakian, and Slovenian)	1250LATIN2	
Arabic	1256ARA	
Greek	1253ELL	
Hebrew	1255HEB	
Japanese	932JPN	EUC_JAPAN
Korean	949KOR	EUC_KOREA
Lithuanian	1257LIT	
Polish	1250POL	
Russian and Ukrainian	EUC_CHINA	
Simplified Chinese	936ZHO	EUC_CHINA
Spanish	1252SPA	ISOLATIN1, ISO9LATIN1
Thai	874THAIBIN	874THAIBIN
Traditional Chinese	950ZHO_TW	
Traditional Chinese + HKSCS	950ZHO_HK	
Turkish	1254TRK	

Tip
Any code that selects a collation for German should select 1252LATIN1, *not* 1252DEU. 1252DEU differentiates between characters with and without an umlaut, while 1252LATIN1 does not. 1252LATIN1 considers Muller and Müller equal, but 1252DEU does not consider them equal. Because 1252DEU views characters with umlauts as separate characters, it has the following alphabetic ordering: ob, öa.

Alternate collations

Alternate collations are available for compatibility with older versions of Adaptive Server Anywhere, or for special purposes. You can obtain a list of alternate collations by typing the following command at the command prompt:

```
dbinit -l+
```

Specifying the -l+ option lists the following collations, in addition to the list of recommended collations:

Collation label	Type	Description
437LATIN1	OEM	Code Page 437, Latin 1, Western
437ESP	OEM	Code Page 437, Spanish
437SVE	OEM	Code Page 437, Swedish/Finnish
819CYR	ANSI	Code Page 819, Cyrillic
819DAN	ANSI	Code Page 819, Danish
819ELL	ANSI	Code Page 819, Greek
819ESP	ANSI	Code Page 819, Spanish
819ISL	ANSI	Code Page 819, Icelandic
819LATIN1	ANSI	Code Page 819, Latin 1, Western
819LATIN2	ANSI	Code Page 819, Latin 2, Central/Eastern European
819NOR	ANSI	Code Page 819, Norwegian
819RUS	ANSI	Code Page 819, Russian
819SVE	ANSI	Code Page 819, Swedish/Finnish
819TRK	ANSI	Code Page 819, Turkish
850CYR	OEM	Code Page 850, Cyrillic, Western

Collation label	Type	Description
850DAN	OEM	Code Page 850, Danish
850ELL	OEM	Code Page 850, Greek
850ESP	OEM	Code Page 850, Spanish
850ISL	OEM	Code Page 850, Icelandic
850LATIN1	OEM	Code Page 850, Latin 1, Western
850LATIN2	OEM	Code Page 850, Latin 2, Central/Eastern European
850NOR	OEM	Code Page 850, Norwegian
850RUS	OEM	Code Page 850, Russian
850SVE	OEM	Code Page 850, Swedish/Finnish
850TRK	OEM	Code Page 850, Turkish
852LATIN2	OEM	Code Page 852, Latin 2, Central/Eastern European
852CYR	OEM	Code Page 852, Cyrillic
852POL	OEM	Code Page 852, Polish
855CYR	OEM	Code Page 855, Cyrillic
856HEB	OEM	Code Page 856, Hebrew
857TRK	OEM	Code Page 857, Turkish
860LATIN1	OEM	Code Page 860, Latin 1, Western
861ISL	OEM	Code Page 861, Icelandic
862HEB	OEM	Code Page 862, Hebrew
863LATIN1	OEM	Code Page 863, Latin 1, Western
865NOR	OEM	Code Page 865, Norwegian
866RUS	OEM	Code Page 866, Russian
869ELL	OEM	Code Page 869, Greek
920TRK	ANSI	Code Page 920, Turkish, ISO-8859-9
1252DEU	ANSI	Code Page 1251, Windows Specialty German, Umlaut characters not equal

Collation label	Type	Description
1254TRKALT	ANSI	Code Page 1254, Windows Specialty Turkish, I-dot and I-no-dot are equal ☞ For more information, see "Turkish character sets and collations" on page 327.

Creating databases for Windows CE

Windows CE is a Unicode-based operating system. The Adaptive Server Anywhere ODBC driver supports either ASCII (8-bit) or Unicode strings, and carries out character set translation as needed. If you are developing embedded SQL applications, you can use Windows API functions to get the Unicode versions of strings from the database.

When creating databases for Windows CE, you should use a collation based on the same single- or multibyte character set that Windows would use for the language of interest. For example, if you are using English, French, or German, use the 1252Latin1 collation. If you are using Japanese, use the 932JPN collation, and if you are using Korean, use the 949KOR collation.

Notes on ANSI collations

The ISO_1 collation

ISO_1 is provided for compatibility with the Adaptive Server Enterprise default ISO_1 collation. The differences are as follows:

♦ The lower case letter sharp s (\xDF) sorts with the lower case **s** in Adaptive Server Anywhere, but after **ss** in Adaptive Server Enterprise.

♦ The ligatures corresponding to **AE** and **ae** (\xC6 and \xE6) sort after **A** and **a** respectively in Adaptive Server Anywhere, but after **AE** and **ae** in Adaptive Server Enterprise.

The 1252LATIN1 collation

This collation includes the euro currency symbol and several other characters (Z-with-caron and z-with-caron). This collation is recommended for Windows users using English or Western European languages.

The euro symbol sorts with the other currency symbols.

The ISO1LATIN1 collation

This collation is the same as ISO_1, but with sorting for values in the range A0-BF. For compatibility with Adaptive Server Enterprise, the ISO_1 collation has no characters for 0xA0-0xBF. However the ISO Latin 1 character set on which it is based does have characters in these positions. The ISO1LATIN1 collation reflects the characters in these positions.

If you are not concerned with Adaptive Server Enterprise compatibility, ISO1LATIN1 is generally recommended instead of ISO_1.

ISO1LATIN1 is recommended for UNIX users using English or Western European languages.

The ISO9LATIN1 collation

This collation is the same as ISO1LATIN1, but it includes the euro currency symbol and the other new characters included in the 1252LATIN1 collation.

If your machine uses the ISO Latin 9 character set, then you should use this collation.

Using multibyte collations

This section describes how multibyte character sets are handled. The description applies to the supported collations and to any multibyte custom collations you may create.

Adaptive Server Anywhere provides collations for several multibyte character sets.

☞ For more information, see "Understanding collations" on page 335.

Adaptive Server Anywhere supports variable width 8-bit character sets. In these sets, some characters are represented by one byte, and some by more than one, to a maximum of four bytes. The value of the first byte in any character indicates the number of bytes used for that character, and also indicates whether the character is a space character, a digit, or an alphabetic (alpha) character.

For the UTF8 collation, UTF-8 characters are represented by one to four bytes. For other multibyte collations, one or two bytes are used. For all provided multibyte collations, characters comprising two or more bytes are considered to be "alphabetic", such that they can be used in identifiers without requiring double quotes.

All client libraries other than embedded SQL are Unicode-enabled, using the UTF-16 encoding. Translation occurs between the client and the server.

Understanding character set translation

Adaptive Server Anywhere can carry out character set translation among character sets that represent the same characters, but at different positions in the character set or code page. There needs to be a degree of compatibility between the character sets for this to be possible. For example, character set translation is possible between EUC-JIS and Shift-JIS character sets, but not between EUC-JIS and OEM code page 850.

This section describes how Adaptive Server Anywhere carries out character set translation. This information is provided for advanced users, such as those who may be deploying applications or databases in a multi-character-set environment.

Character translation for database messages

Error and other messages from the database software are held in a language resource library. Localized versions of this library are provided with localized versions of Adaptive Server Anywhere.

Client application users may see messages from the database, as well as data from the database.

Messages are always translated into the database character set, regardless of whether character set translation is turned on or off.

A further character set translation is carried out if character set translation is turned on (the default) for the database server, and if the client character set is different from that used in the database collation.

Connection strings and character sets

Connection strings present a special case for character set translation. The connection string is parsed by the client library, in order to locate or start a database server. This parsing is done with no knowledge of the server character set or language.

The interface library parses the connection string as follows:

1. It is broken down into its *keyword=value* components. This can be done independently of the character set, as long as you do not use curly braces {} around CommLinks (LINKS) connection parameters. Instead, use the recommended parentheses (). Curly braces are valid **follow bytes** (bytes other than the first byte) in some multibyte character sets.

2. The server is located. The server name is interpreted according to the character set of the client machine. In the case of Windows operating

systems, the ANSI character set is used. Extended chars can be used unless they cause character set conversion issues between the client and server machines.

For maximum compatibility among different machines, you should use server names built from alphabetic or numeric ASCII characters 0 to 127 (or 33 to 126) and the underscore, using no punctuation characters. Server names are truncated at 40 characters.

3. The DatabaseName (DBN) or DatabaseFile (DBF) connection parameters are interpreted in the database server character set.

4. Once the database is located, the remaining connection parameters are interpreted converted to the database's character set.

Avoiding character set translation

There is a performance cost associated with character set translation. If you can set up an environment such that no character set translation is required, then you do not have to pay this cost, and your setup is simpler to maintain. Adaptive Server Anywhere disables character set conversion if the client's character set is the same as the server's character set.

If you work with a single-byte character set and are concerned only with seven-bit ASCII characters (values 0 through 127), then you do not need character set translation. Even if the code pages are different in the database and on the client operating system, they are compatible over this range of characters. Many English-language installations meet these requirements. In Adaptive Server Anywhere 9.0 and later, character set translation is turned on by default. You can turn it off using the -ct- option.

☞ For more information, see "Turning off character set translation on a database server" on page 356.

If you do require the use of extended characters, there are other steps you may be able to take:

♦ If the code page on your client machine operating system matches that used in the database, no character set translation is needed for data in the database.

  For example, in many environments it is appropriate to use the 1252LATIN1 collation in your database, which corresponds to the ANSI code page in many single-byte environments.

♦ If you are able to use a version of Adaptive Server Anywhere built for your language, and if you use the code page on your operating system, no character set translation is needed for database messages. The character set used in the Adaptive Server Anywhere message strings is as follows:

Language	Windows character set
English	cp1252
French	cp1252
German	cp1252
Italian	cp1252
Japanese	cp932 (Shift-JIS)
Korean	cp949
Lithuanian	cp1257
Polish	cp1250
Portuguese	cp1252
Russian	cp1251
Simplified Chinese	cp936
Spanish	cp1252
Traditional Chinese	cp950
Ukrainian	cp1251

Also, recall that client/server character set translation takes place by default. You can turn it off using the -ct- option.

☞ For more information, see "-ct server option" on page 131.

Collation internals

This section describes internal technical details of collations, including the file format of collation files.

This section is of particular use if you want to create a database using a custom collation. The file format described in this section is used for viewing or editing a collation. The file is then converted into a format that can be used by the database server.

☞ For information on the steps involved, see "Creating a custom collation" on page 357, and "Creating a database with a custom collation" on page 358.

You can create a database using a collation different from the supplied collations. This section describes how to build databases using such custom collations.

In building multibyte custom collations, you can specify which ranges of values for the first byte signify single- and double-byte (or more) characters, and which specify space, alpha, and digit characters.

Collation files may include the following elements:

- comment lines, which are ignored by the database.
- a title line.
- a Collation section.
- an Encodings section.
- a Properties section.

Comment lines

In the collation file, spaces are generally ignored. Comment lines start with either the percent sign (**%**) or two dashes (-).

The title line

The first non-comment line must be of the form:

Collation *collation_label (collation_name) (ase_charset) (ase_so_sensitive) (ase_so_insensitive) (java_charset)*

Argument descriptions

Argument	Description
Collation	A required keyword.
collation_label	The collation label, which appears in the system tables as SYS.SYSCOLLATION.collation_label and SYS.-SYSINFO.default_collation. The label must contain no more than 10 characters.
collation_name	A text description of the collation, usually describing the character set and ordering of the collation.
ase_charset	The Adaptive Server Enterprise name of a character set matching the character set of this collation. This name is used to make character set mappings when server character set translation is enabled (the default).
ase_so_sensitive	The name of an Open Client or Adaptive Server Enterprise case sensitive collation matching this collation.
ase_so_-insensitive	The name of an Open Client or Adaptive Server Enterprise case insensitive collation matching this collation.
java_charset	The name of a Java character set matching the character set of this collation. This name is used when converting character data between the Java virtual machine and the database.

For example, the 932JPN collation file contains the following collation line, with label 932JPN and name Code Page 932, Japanese Shift-JIS with Microsoft extensions:

```
Collation 932JPN (Code Page 932, Japanese Shift-JIS with
         Microsoft extensions) (cp932) (bin_cp932) (bin_cp932)
         (SJIS)
```

The Collation section

After the title line, each non-comment line describes one position in the collation. The ordering of the lines determines the sort ordering used by the database, and determines the result of comparisons. Characters on lines appearing higher in the file (closer to the beginning) sort before characters that appear later.

The form of each line in the sequence is:

[*sort-position*] : *character* [[, *character*] . . .]

or

[*sort-position*] : *character* [*lowercase uppercase*]

Argument descriptions

Argument	**Description**
sort-position	Optional. Specifies the position at which the characters on that line will sort. Smaller numbers represent a lesser value, so will sort closer to the beginning of the sorted set. Typically, the sort-position is omitted, and the characters sort immediately following the characters from the previous sort position.
character	The character whose sort-position is being specified.
lowercase	Optional. Specifies the lower case equivalent of the character. If not specified, the character has no lower case equivalent.
uppercase	Optional. Specifies the upper case equivalent of the character. If not specified, the character has no upper case equivalent.

Multiple characters may appear on one line, separated by commas (,). In this case, these characters are sorted and compared as if they were the same character.

Specifying character and sort position

Each character and sort position is specified in one of the following ways:

Specification	**Description**
\d*nnn*	Decimal number, using digits 0-9 (such as \d001)
\x*hh*	Hexadecimal number, using digits 0-9 and letters a-f or A-F (such as \xB4)
'*c***'**	Any character in place of c (such as ‘,’)
c	Any character other than quote (‘), back-slash (\), colon (:) or comma (,). These characters must use one of the previous forms.

The following are some sample lines for a collation:

```
% Sort some special characters at the beginning:
: ' '
: _
: \xF2
: \xEE
: \xF0
: -
: ','
: ;
: ':'
: !
% Sort some letters in alphabetical order
: A a A
: a a A
: B b B
: b b B
% Sort some E's from code page 850,
% including some accented extended characters:
: e e E, \x82 \x82 \x90, \x8A \x8A \xD4
: E e E, \x90 \x82 \x90, \xD4 \x8A \xD4
```

Other syntax notes

For databases using case insensitive sorting and comparison (no -c specified in the dbinit command), the lower case and upper case mappings are used to find the lower case and upper case characters that are sorted together.

For multibyte character sets, the first byte of a character is listed in the collation sequence, and all characters with the same first byte are sorted together, and ordered according to the value of the following bytes. For example, the following is part of the Shift-JIS collation file:

```
: \xfb
:  \xfc
:  \xfd
```

In this collation, all characters with first byte \xfc come after all characters with first byte \xfb and before all characters with first byte \xfd. The two-byte character \xfc \x01 would be ordered before the two-byte character \xfc \x02.

Any characters omitted from the collation are added to the end of the collation. The tool that processes the collation file issues a warning.

The Encodings section

The encodings section is optional, and follows the collation sequence. It is not useful for single-byte character sets.

The encodings section lists which byte values are lead-bytes, for multibyte character sets, and which byte values are valid follow-bytes.

For example, the Shift-JIS encodings section is as follows:

```
Encodings:
[\x00-\x80,\xa0-\xdf,\xf0-\xff]
[\x81-\x9f,\xe0-\xef][\x40-\xff]
```

The first line following the section title lists valid single-byte characters. The square brackets enclose a comma-separated list of ranges. Each range is listed as a hyphen-separated pair of values. In the Shift-JIS collation, values \x00 to \x80 are valid single-byte characters, but \x81 is not a valid single-byte character.

The second line following the section title lists valid multibyte characters. The brackets represent ranges. Any combination of one byte from the first range followed by one byte from the second range may be a valid character.

The Properties section

The properties section is optional, and follows the encodings section.

If a properties section is supplied, an encodings section must also be supplied.

The properties section lists values for the first byte of each character that represent alphabetic characters, digits, or spaces.

The Shift-JIS properties section is as follows:

```
Properties:
space: [\x09-\x0d,\x20]
digit: [\x30-\x39]
alpha: [\x41-\x5a,\x61-\x7a,\x81-\x9f,\xe0-\xef]
```

This indicates that characters with first bytes \x09 to \x0d, as well as \x20, are to be treated as space characters, digits are found in the range \x30 to \x39 inclusive, and alphabetic characters in the four ranges \x41-\x5a, \x61-\x7a, \x81-\x9f, and \xe0-\xef.

International language and character set tasks

This section groups together the tasks associated with international language and character set issues.

Finding the default collation

If you do not explicitly specify a collation when creating a database, a default collation is used. The default collation depends on the operating system you are working on.

❖ To find the default collation for your machine

1. Start Interactive SQL. Connect to the sample database.

2. Enter the following query:

   ```
   SELECT PROPERTY( 'DefaultCollation' )
   ```

   The default collation is returned.

   ☞ For more information about this collation, see "Choosing collations" on page 335.

Configuring your character set environment

This section describes how to set up your computing environment so that character set issues are handled properly. If you set your locale environments properly, then you do not need to explicitly choose collations for your databases, and you can disable character set translation between client and server.

☞ For more information about turning off character set translation, see "Turning off character set translation on a database server" on page 356.

❖ To configure your character set environment

1. Determine the default locale of each computing platform in your environment. The default locale is the character set and language of each computer. On Windows operating systems, the character set is the ANSI code page.

   ☞ For more information about how to find locale information, see "Determining locale information" on page 353.

2. Decide whether the locale settings are appropriate for your environment.

   ☞ For more information, see "Understanding collations" on page 335.

3. If the default settings are inappropriate, decide on a character set, language, and database collation that match your data and avoid character set translation.

   ☞ For more information, see "Avoiding character set translation" on page 344.

4. Set locales on each of the machines in the environment to these values.

   ☞ For more information, see "Setting locales" on page 354.

5. Create your database using the default collation. If the default collation does not match your needs, create a database using a named collation.

   ☞ For more information, see "Creating a database with a named collation" on page 355, and "Changing a database from one collation to another" on page 359.

When choosing the collation for your database, consider the following:

♦ Choose a collation that uses a character set and sort order appropriate for the data in the database. It is often the case that there are several alternative collations that meet this requirement.

♦ There is a performance cost, as well as extra complexity in system configuration, when you use character set translation. Choose a collation that avoids the need for character set translation. Character set translation is disabled automatically if the database server and client use the same character set.

  You can avoid character set translation by using a collation sequence in the database that matches the character set in use on your client machine operating system. In the case of Windows operating systems on the client machine, choose the ANSI character set. Character set translation is enabled by default for database servers that are version 8.0 or higher of Adaptive Server Anywhere. You can turn off character set translation using the -ct- option.

  ☞ For more information, see "Avoiding character set translation" on page 344.

Determining locale information

You can determine locale information using system functions.

☞ For more information, see "System functions" [*ASA SQL Reference*, page 102].

❖ To determine the locale of a database server

1. Start Interactive SQL, and connect to a database server.

2. Execute the following statement to determine the database server character set:

   ```
   SELECT PROPERTY( 'CharSet' )
   ```

   The query returns one of the supported character sets listed in "Character set labels" on page 369.

3. Execute the following statement to determine the database server language:

   ```
   SELECT PROPERTY( 'Language' )
   ```

   The query returns one of the supported languages listed in "Language label values" on page 332.

4. Execute the following statement to determine the database server default collation:

   ```
   SELECT PROPERTY( 'DefaultCollation' )
   ```

   The query returns one of the collations listed in "Choosing collations" on page 335.

Notes

To obtain client locale information, connect to a database server running on the client machine.

To obtain the character set for an individual database, execute the following statement:

```
SELECT DB_PROPERTY ( 'CharSet' )
```

Setting locales

You can use the default locale on your operating system, or explicitly set a locale for use by the Adaptive Server Anywhere components on your machine.

❖ To set the Adaptive Server Anywhere locale

1. If the default locale is appropriate for your needs, you do not need to take any action.

   ☞ For more information about how to find out the default locale of your operating system, see "Determining locale information" on page 353.

2. If you need to change the locale, you can set either or both of the ASLANG and ASCHARSET environment variables:

   ```
   ASCHARSET=charset;
   ASLANG=language_code
   ```

   where *charset* is a character set label from the list in "Character set labels" on page 369, and *language_code* is a language label from the list in "Language label values" on page 332.

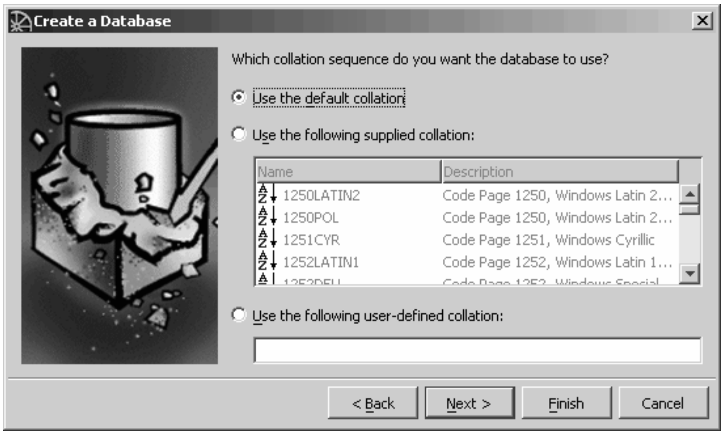
   ☞ For more information on how to set environment variables on different operating systems, see "Setting environment variables" on page 276.

Creating a database with a named collation

You may specify the collation for each database when you create the database. The default collation is inferred from the code page and language of the database server's computer's operating system.

❖ To specify a database collation when creating a database (Sybase Central)

1. You can use the Create Database wizard in Sybase Central to create a database. The wizard has a page where you choose a collation from a list.



Which collation sequence do you want the database to use?

[✓ Use the default collation]

Use the following supplied collation:

Name	Description
1250LATIN2	Code Page 1250, Windows Latin 2...
1250POL	Code Page 1250, Windows Latin 2...
1251CYR	Code Page 1251, Windows Cyrillic
1252LATIN1	Code Page 1252, Windows Latin 1...

Use the following user-defined collation:

❖ To specify a database collation when creating a database (command prompt)

1. List the recommended collation sequences by typing the following at a command prompt:

   ```
   dbinit -l
   ```

The first column of the list is the collation label, which you supply when creating the database.

```
437LATIN1  Code Page 437, Latin 1, Western
437ESP     Code Page 437, Spanish
437SVE     Code Page 437, Swedish/Finnish
819CYR     Code Page 819, Cyrillic
819DAN     Code Page 819, Danish
...
```

2. Create a database using the dbinit utility, specifying a collation sequence using the -z option. The following command creates a database with a Greek collation.

```
dbinit -z 1253ELL mydb.db
```

❖ To specify a database collation when creating a database (SQL)

1. You can use the CREATE DATABASE statement to create a database. The following statement creates a database with a Greek collation:

```
CREATE DATABASE 'mydb.db'
COLLATION '1253ELL'
```

Starting a database server using character set translation

Character set translation takes place if the client and server locales are different. Character set translation is enabled by default for database servers that are Adaptive Server Anywhere version 8.0 or higher. For database servers that are version 7.x or earlier, you must explicitly enable character set conversion in the database server command.

❖ To enable character set translation on a database server

1. Start the database server using the -ct+ option. For example:

```
dbsrv9 -ct+ asademo.db
```

Turning off character set translation on a database server

Character set translation is enabled by default. If you do not require character set translation, you can disable it in the database server command using the -ct- option.

❖ To disable character set translation on a database server

1. Start the database server using the -ct- option. For example:

   ```
   dbsrv9 -ct- asademo.db
   ```

☞ For more information, see "-ct server option" on page 131.

Creating a custom collation

If none of the supplied collations meet your needs, you can modify a supplied collation to create a **custom collation**. You can then use this custom collation when creating a database.

☞ For more information about supplied collations, see "Choosing collations" on page 335.

❖ To create a custom collation

1. **Decide on a starting collation** You should choose a collation as close as possible to the one you want to create as a starting point for your custom collation.

   ☞ For a listing of recommended collations, see "Understanding collations" on page 335.

   Alternatively, run dbinit with the -l (lower case L) option:

   ```
   dbinit -l
   ```

2. **Create a custom collation file** You do this using the Collation [dbcollat] utility. The output is a collation file.

   For example, the following statement extracts the 1252LATIN1 collation into a file named *mycustomcol.col*:

   ```
   dbcollat -z 1252LATIN1 mycustomcol.col
   ```

3. **Edit the custom collation file** Open the collation file (in this case *mycustomcol.col*) in a text editor.

4. **Change the collation label** Make the changes you wish in the custom collation file to define the collation label. For example, if you wanted to make a custom version of the 1252LATIN1 collation, you could change the Collation line from:

   ```
   Collation 1252LATIN1 (Code Page 1252, Windows Latin 1,
            Western) (cp1252) (dictionary_iso_1) (nocase_iso_1)
            (Cp1252)
   ```

   to

```
Collation MyOrdering (Code Page 1252, My Company Ordering)
        (cp1252) (dictionary_iso_1) (nocase_iso_1) (Cp1252)
```

The other entries on this line relate the Adaptive Server Anywhere collation label to the names that Java and the Sybase TDS interface give to the same collation information. If you are not using these interfaces you do not need to alter these entries.

The Collation line takes the following form:

```
Collation collation_label (collation_name) (ase_charset)
        (ase_so_sensitive) (ase_so_insensitive) (java_
        charset)
```

where character set label (*ase_charset*) and the two sort-order labels (*ase_so_sensitive* and *ase_so_insensitive*) state which Open Client character set and sort order is the closest to the current collation. The *java_charset* label is the closest character set known to Java.

You should edit this line to provide a new label. The label you need to change is the *collation_label*: in the example in step two, it is `1252LATIN1`.

5. **Change the collation definition** Make the changes you wish in the custom collation file to define your new collation.

   ☞ For more information about the collation file contents and format, see "Collation internals" on page 347.

6. **Convert the file to a SQL script** You do this using the dbcollat utility using the -d option.

   For example, the following command creates the *mycustom.SQL* file from the *mycustomcol.col* collation file:

   ```
   dbcollat -d mycustomcol.col mycustom.SQL
   ```

7. **Add the SQL script to the script in your installation** The script used when creating databases is held in the scripts subdirectory of your Adaptive Server Anywhere installation directory. Append the contents of *mycustom.SQL* to the end of *custom.SQL*.

   The new collation is now in place, and can be used when creating a database.

Creating a database with a custom collation

If none of the supplied collations meet your needs, you can create a database using a custom collation. The custom collation is used in indexes and any string comparisons.

❖ To create a database with a custom collation

1. Create a custom collation.

   You must have a custom collation in place to use when creating a database.

   ☞ For more information about how to create custom collations, see "Creating a custom collation" on page 357.

2. Create the new database.

   Use the Initialization [dbinit] utility, specifying the name of your custom collation.

   For example, the following command creates a database named *temp.db* using the custom collation sequence **newcol**.

   ```
   dbinit -z newcol temp.db
   ```

   You can also use the Initialization utility from Sybase Central.

Changing a database from one collation to another

Changing your database from one collation to another may be a good idea for any number of reasons. It can be especially useful, for example, when:

- you want to avoid character set translation across your setup.
- the characters in your database don't match the collation in your database. Using the same character set defined in your database is especially important for sorting purposes.
- you want to use a different character set. You may, for example, want to move from an OEM character set to a Windows character set.

Simply modifying the collation in an existing database is not permitted since it would invalidate all the indexes for that database. In order to change the collation for a database, you must rebuild the database. Rebuilding a database creates a new database with new settings (including collation settings), using the old database's data.

When you change the collation for a database, there are two main scenarios to consider. The difference between the two lies in whether the character set of the data needs to be changed.

Example 1

In the first example, the data in the database is in the wrong character set for the collation. Only the collation needs to be changed; the data should not change character sets. To resolve the collation issue, you need to rebuild the database with new collation settings using the old data.

In an English environment, consider an old database using the 850LATIN1 collation. If the database contains data inserted from a Windows ‘windowed’ application, it is likely that the data is actually from the CP1252 character set, which does not match CP850 used by the 850LATIN1 collation. This situation will often be discovered when an ORDER BY clause seems to sort accented characters incorrectly. To correct this problem, you would create a new database using the 1252LATIN1 collation, and move the data from the old database to the new database without translation, since the data is already in the character set (CP1252) that matches the new database’s collation.

The simplest way to ensure that translation does not occur is to start the server with the -ct- option. Then, rebuild the database normally.

☞ For more information about rebuilding a database, see “Rebuilding databases” [*ASA SQL User’s Guide*, page 580].

☞ For more information about specifying collations when creating databases, see “Creating a database with a named collation” on page 355.

Example 2

In the second situation, both the collation and the character set need to be changed. To resolve the collation and character set issues, you need to rebuild the database with the new collation settings, and change the character set of the data.

Suppose that the 850LATIN1 database had been used properly such that it contains characters from the CP850 character set. However, you want to update both the collation and the character set, perhaps to avoid character set translation. You would create a new database using 1252LATIN1, and move the data from the old database to the new database with translation, thus converting the CP850 characters to CP1252.

The translation of the database data from one character set to another occurs using the client-server translation feature of the server, which translates the character data during the communication between the client application and the server. The database’s collation determines the character set for the database server side of the communication. The locale of the operating system determines the client’s default character set, however, the client’s character set can be overridden by the CharSet (CS) connection parameter.

☞ For more information about the CharSet (CS) connection parameter, see “CharSet connection parameter [CS]” on page 180.

Since character set translation takes place during the communication between the client application and the server, an external unload or reload is necessary. An internal unload and reload does not do character set translation.

❖ To convert a database from one collation to another, and translate the data's character set (using translation on reload)

1. Unload the data from the source database.

   You can use the Unload utility to produce a *reload.SQL* file and a set of data files in the character set of the source database. Since we do not want any translation during this phase, ensure that character set translation is not enabled on the server running the source database. For servers before version 8, ensure that -ct is not specified. If you are using a server that is version 8 or higher, ensure that -ct- (no character set translation) is specified when the server is started.

   If the unload/reload is occurring on a single machine, use the -ix option to do an internal unload and an external reload. If the unload/reload occurs across machines, use the Unload utility with the -xx option to force an external unload and an external reload.

   Remember that an "external" unload or reload means that an application (the dbunload and dbisql utilities) opens a cursor on the database and either reads or writes the data to disk. Character set translation occurs. An "internal" unload or reload means that an UNLOAD TABLE or LOAD TABLE is used so that the server reads or writes the data itself. Character set translation does *not* occur.

   If you want to unload data from specific tables, use the -t option, or the Interactive SQL OUTPUT statement.

   ☞ For more information on the Unload utility, see "The Unload utility" on page 588.

2. Create a target database with the appropriate collation using the Initialization utility including the -z option to specify the collation sequence for the target database.

   The database should be created and reloaded using the version of the server and tools corresponding to the server that they will use to run it.

   ☞ For more information on specifying collations when creating databases, see "Creating a database with a named collation" on page 355.

3. Start the target database using the server running with character set translation. You can also specify the -z option.

   The –z option, while not required, allows for verification of the character sets that will be used, and that translation will occur. The server window will show character set translation settings for each connection. However, using the –z server option can cause slower performance during the reload.

   ☞ For more information, see "Starting a database server using character set translation" on page 356.

4. If you generated a *reload.SQL* file in step 1, you can run the file in a command shell using a command such as the following.

   ```
   dbisql -c "uid=dba;pwd=sql;charset=cp1252" -codepage 850
            reload.sql
   ```

   You must supply the appropriate connection parameters for your database, as well as the code page (850 in the example above) that was used to generate the *reload.SQL* file.

   If you exported only table data in step 1, you can import the data using the INPUT statement in Interactive SQL.

   ☞ For more information about the INPUT statement, see “INPUT statement [Interactive SQL]” [*ASA SQL Reference*, page 523].

Code page and character set reference

The following sections provide information about supported code pages and character sets for Adaptive Server Anywhere.

Supported code pages

The following table lists code pages supported in Interactive SQL.

☞ For more information, see "Opening Interactive SQL using the dbisql command-line utility" on page 539 or "Interactive SQL utility options" on page 541.

Code Page	Description
1252	Windows Latin-1
037	USA, Canada (Bilingual, French), Netherlands, Portugal, Brazil, Australia
273	IBM Austria, Germany
277	IBM Denmark, Norway
278	IBM Finland, Sweden
280	IBM Italy
284	IBM Catalan/Spain, Spanish Latin America
285	IBM United Kingdom, Ireland
297	IBM France
420	IBM Arabic
424	IBM Hebrew
437	MS-DOS United States, Australia, New Zealand, South Africa
500	EBCDIC 500V1
737	PC Greek
775	PC Baltic
838	IBM Thailand extended SBCS
850	MS-DOS Latin-1
852	MS-DOS Latin-2

Code Page	Description
855	IBM Cyrillic
856	IBM Hebrew
857	IBM Turkish
858	Variant of Cp850 with Euro character
860	MS-DOS Portuguese
861	MS-DOS Icelandic
862	PC Hebrew
863	MS-DOS Canadian French
864	PC Arabic
865	MS-DOS Nordic
866	MS-DOS Russian
868	MS-DOS Pakistan
869	IBM Modern Greek
870	IBM Multilingual Latin-2
871	IBM Iceland
874	IBM Thai
875	IBM Greek
918	IBM Pakistan (Urdu)
921	IBM Latvia, Lithuania (AIX, DOS)
922	IBM Estonia (AIX, DOS)
930	Japanese Katakana-Kanji mixed with 4370 UDC, superset of 5026
933	Korean Mixed with 1880 UDC, superset of 5029
935	Simplified Chinese Host mixed with 1880 UDC, superset of 5031
937	Traditional Chinese Host mixed with 6204 UDC, superset of 5033
939	Japanese Latin Kanji mixed with 4370 UDC, superset of 5035

Code Page	Description
942	IBM OS/2 Japanese, superset of Cp932
942	C Variant of Cp942
943	IBM OS/2 Japanese, superset of Cp932 and Shift-JIS
943	C Variant of Cp943
948	OS/2 Chinese (Taiwan) superset of 938
949	PC Korean
949	C Variant of Cp949
950	PC Chinese (Hong Kong, Taiwan)
964	AIX Chinese (Taiwan)
970	AIX Korean
1006	IBM AIX Pakistan (Urdu)
1025	IBM Multilingual Cyrillic: Bulgaria, Bosnia, Herzegovinia, Macedonia (FYR)
1026	IBM Latin-5, Turkey
1046	IBM Arabic - Windows
1097	IBM Iran (Farsi)/Persian
1098	IBM Iran (Farsi)/Persian (PC)
1112	IBM Latvia, Lithuania
1122	IBM Estonia
1123	IBM Ukraine
1124	IBM AIX Ukraine
1140	Variant of Cp037 with Euro character
1141	Variant of Cp273 with Euro character
1142	Variant of Cp277 with Euro character
1143	Variant of Cp278 with Euro character
1144	Variant of Cp280 with Euro character
1145	Variant of Cp284 with Euro character
1146	Variant of Cp285 with Euro character

Code Page	Description
1147	Variant of Cp297 with Euro character
1148	Variant of Cp500 with Euro character
1149	Variant of Cp871 with Euro character
1250	Windows Eastern European
1251	Windows Cyrillic
1253	Windows Greek
1254	Windows Turkish
1255	Windows Hebrew
1256	Windows Arabic
1257	Windows Baltic
1258	Windows Vietnamese
1381	IBM OS/2, DOS People’s Republic of China (PRC)
1383	IBM AIX People’s Republic of China (PRC)
33722	IBM-eucJP - Japanese (superset of 5050)
ASCII	American Standard Code for Information Interchange
ISO8859_1	ISO 8859-1, Latin alphabet No. 1
UnicodeBig	Sixteen-bit Unicode Transformation Format, big-endian byte order, with byte-order mark
UnicodeBigUnmarked	Sixteen-bit Unicode Transformation Format, big-endian byte order
UnicodeLittle	Sixteen-bit Unicode Transformation Format, little-endian byte order, with byte-order mark
UnicodeLittleUnmarked	Sixteen-bit Unicode Transformation Format, little-endian byte order
UTF8	Eight-bit Unicode Transformation Format
UTF-16	Sixteen-bit Unicode Transformation Format, byte order specified by a mandatory initial byte-order mark
Big5	Big5, Traditional Chinese
Big5_HKSCS	Big5 with Hong Kong extensions, Traditional Chinese

Code Page	Description
Big5_Solaris	Big5 with seven additional Hanzi ideograph character mappings for the Solaris zh_TW.BIG5 locale
EUC_CN	GB2312, EUC encoding, Simplified Chinese
EUC_JP	JIS X 0201, 0208, 0212, EUC encoding, Japanese
EUC_KR	KS C 5601, EUC encoding, Korean
EUC_TW	CNS11643 (Plane 1-3), EUC encoding, Traditional Chinese
GB18030	Simplified Chinese, PRC standard
GBK	GBK, Simplified Chinese
ISCII91	ISCII91 encoding of Indic scripts
ISO2022CN	ISO 2022 CN, Chinese (conversion to Unicode only)
ISO2022CN_CNS	CNS 11643 in ISO 2022 CN form, Traditional Chinese (conversion from Unicode only)
ISO2022CN_GB	GB 2312 in ISO 2022 CN form, Simplified Chinese (conversion from Unicode only)
ISO2022JP	JIS X 0201, 0208 in ISO 2022 form, Japanese
ISO2022KR	ISO 2022 KR, Korean
ISO8859_2	ISO 8859-2, Latin alphabet No. 2
ISO8859_3	ISO 8859-3, Latin alphabet No. 3
ISO8859_4	ISO 8859-4, Latin alphabet No. 4
ISO8859_5	ISO 8859-5, Latin/Cyrillic alphabet
ISO8859_6	ISO 8859-6, Latin/Arabic alphabet
ISO8859_7	ISO 8859-7, Latin/Greek alphabet
ISO8859_8	ISO 8859-8, Latin/Hebrew alphabet
ISO8859_9	ISO 8859-9, Latin alphabet No. 5
ISO8859_13	ISO 8859-13, Latin alphabet No. 7
ISO8859_15_FDIS	ISO 8859-15, Latin alphabet No. 9
JIS0201	JIS X 0201, Japanese
JIS0208	JIS X 0208, Japanese

Code Page	Description
JIS0212	JIS X 0208, Japanese
JISAutoDetect	Detects and converts from Shift-JIS, EUC-JP, ISO 2022 JP (conversion to Unicode only)
Johab	Johab, Korean
KOI8_R	KOI8-R, Russian
MS874	Windows Thai
MS932	Windows Japanese
MS936	Windows Simplified Chinese
MS949	Windows Korean
MS950	Windows Traditional Chinese
MacArabic	Macintosh Arabic
MacCentralEurope	Macintosh Latin-2
MacCroatian	Macintosh Croatian
MacCyrillic	Macintosh Cyrillic
MacDingbat	Macintosh Dingbat
MacGreek	Macintosh Greek
MacHebrew	Macintosh Hebrew
MacIceland	Macintosh Iceland
MacRoman	Macintosh Roman
MacRomania	Macintosh Romania
MacSymbol	Macintosh Symbol
MacThai	Macintosh Thai
MacTurkish	Macintosh Turkish
MacUkraine	Macintosh Ukraine
SJIS	Shift-JIS, Japanese
TIS620	TIS620, Thai

Character set labels

The following table displays the valid character set label values, together with the equivalent IANA labels and a description:

Character set label	IANA label	Description
big5	<N/A>	Traditional Chinese (cf. CP950)
cp437	<N/A>	IBM CP437 - U.S. code set
cp850	<N/A>	IBM CP850 - European code set
cp852	<N/A>	PC Eastern Europe
cp855	<N/A>	IBM PC Cyrillic
cp856	<N/A>	Alternate Hebrew
cp857	<N/A>	IBM PC Turkish
cp860	<N/A>	PC Portuguese
cp861	<N/A>	PC Icelandic
cp862	<N/A>	PC Hebrew
cp863	<N/A>	IBM PC Canadian French code page
cp864	<N/A>	PC Arabic
cp865	<N/A>	PC Nordic
cp866	<N/A>	PC Russian
cp869	<N/A>	IBM PC Greek
cp874	<N/A>	Microsoft Thai SB code page
cp932	windows-31j	Microsoft CP932 = Win31J-DBCS
cp936	</N/A>	Simplified Chinese
cp949	<N/A>	Korean
cp950	<N/A>	PC (MS) Traditional Chinese
cp1250	<N/A>	MS Windows Eastern European
cp1251	<N/A>	MS Windows Cyrillic

Character set label	IANA label	Description
cp1252	<N/A>	MS Windows US (ANSI)
cp1253	<N/A>	MS Windows Greek
cp1254	<N/A>	MS Windows Turkish
cp1255	<N/A>	MS Windows Hebrew
cp1256	<N/A>	MS Windows Arabic
cp1257	<N/A>	MS Windows Baltic
cp1258	<N/A>	MS Windows Vietnamese
deckanji	<N/A>	DEC UNIX JIS encoding
euccns	<N/A>	EUC CNS encoding: Traditional Chinese with extensions
eucgb	<N/A>	EUC GB encoding = Simplified Chinese
eucjis	euc-jp	Sun EUC JIS encoding
eucksc	<N/A>	EUC KSC Korean encoding (cf. CP949)
greek8	<N/A>	HP Greek-8
iso_1	iso_8859-1:1987	ISO 8859-1 Latin-1
iso15	<N/A>	ISO 8859-15 Latin-1 with Euro, etc.
iso88592	iso_8859-2:1987	ISO 8859-2 Latin-2 Eastern Europe
iso88595	iso_8859-5:1988	ISO 8859-5 Latin/Cyrillic
iso88596	iso_8859-6:1987	ISO 8859-6 Latin/Arabic
iso88597	iso_8859-7:1987	ISO 8859-7 Latin/Greek
iso88598	iso_8859-8:1988	ISO 8859-8 Latin/Hebrew
iso88599	iso_8859-9:1989	ISO 8859-9 Latin-5 Turkish
koi8	<N/A>	KOI-8 Cyrillic
mac	macintosh	Standard Mac coding
mac_cyr	<N/A>	Macintosh Cyrillic

Character set label	**IANA label**	**Description**
mac_ee	<N/A>	Macintosh Eastern European
macgrk2	<N/A>	Macintosh Greek
macturk	<N/A>	Macintosh Turkish
roman8	hp-rpman8	HP Roman-8
sjis	shift_jis	Shift JIS (no extensions)
tis620	<N/A>	TIS-620 Thai standard
turkish8	<N/A>	HP Turkish-8
utf8	utf-8	UTF-8 treated as a character set

CHAPTER 12

Backup and Data Recovery

About this chapter

This chapter describes how to protect your data against operating system crashes, file corruption, disk failures, and total machine failure.

The chapter describes how to make backups of your database, how to restore data from a backup, and how to run your server so that validation, performance, and data protection concerns are addressed.

Contents

Introduction to backup and recovery

A **backup** is a copy of the information in a database, held in some physically separate location from your database. If the database becomes unavailable, perhaps because of damage to a disk drive, you can **restore** it from the backup. Depending on the nature of the damage, it is often possible to restore from backups all committed changes to the database up to the time it became unavailable.

Restoring databases from a backup is one aspect of database **recovery**. The other aspect is recovery from operating system or database server crashes, and improper shutdowns. The database server checks on database startup whether the database was shut down cleanly at the end of the previous session. If it was not, the server executes an automatic recovery process to restore information. This mechanism recovers all changes up to the most recently committed transaction.

Questions and answers

To answer the question. . .	**Consider reading. . .**
What is a backup?	"Introduction to backup and recovery" on page 374
What is recovery?	"Introduction to backup and recovery" on page 374
What is a transaction log?	"The transaction log" on page 378
What are media and system failure?	"Protecting your data against failure" on page 376
From what kinds of failure do backups protect my data?	"Protecting your data against failure" on page 376
What tools are available for backups?	"Ways of making backups" on page 376
What types of backup are available?	"Types of backup" on page 381
What type of backup should I use?	"Designing backup procedures" on page 381
If my database file or transaction log becomes corrupt, what data may be lost?	"Protecting your data against media failure" on page 379
How are backups executed?	"Understanding backups" on page 378

To answer the question...	Consider reading...
How often do I carry out backups?	"Scheduling backups" on page 382
Can I schedule automatic backups?	"Scheduling backups" on page 382
My database is involved in replication. How does this affect my backup strategy?	"A backup scheme for databases involved in replication" on page 385 "Backup methods for remote databases in replication installations" on page 387
How can I backup to tape?	"Backing up a database directly to tape" on page 413
How do I plan a backup schedule?	"Designing a backup and recovery plan" on page 388
Can I automate backups?	"Automating Tasks Using Schedules and Events" on page 301
How can I be sure that my database file is not corrupt?	"Ensuring your database is valid" on page 389 "Validating a database" on page 405
How can I be sure that my transaction log is not corrupt?	"Validating the transaction log on database startup" on page 401 "Validating a transaction log" on page 406
How can I run my database for maximum protection against failures?	"Configuring your database for data protection" on page 391
How can I ensure high availability and machine redundancy?	"Protecting against total machine failure" on page 392 "Making a live backup" on page 414
How do I carry out a backup?	"Making a full backup" on page 403
How do I restore data from backups when a failure occurs?	"Recovering from media failure on the database file" on page 415 "Recovering from media failure on an unmirrored transaction log" on page 416 "Recovering from media failure on a mirrored transaction log" on page 416

Protecting your data against failure

If your database has become unusable, you have experienced a database **failure**. Adaptive Server Anywhere provides protection against the following categories of failure:

Media failure The database file and/or the transaction log become unusable. This may occur because the file system or the device storing the database file becomes unusable, or it may be because of file corruption.

For example:

- ♦ The disk drive holding the database file or the transaction log file becomes unusable.
- ♦ The database file or the transaction log file becomes corrupted. This can happen because of hardware problems or software problems.

Backups protect your data against media failure.

☞ For more information, see "Understanding backups" on page 378.

System failure A system failure occurs when the computer or operating system goes down while there are partially completed transactions. This could occur when the computer is inappropriately turned off or rebooted, when another application causes the operating system to crash, or because of a power failure.

For example:

- ♦ The computer or operating system becomes temporarily unavailable while there are partially completed transactions, perhaps because of a power failure or operating system crash, or because the computer is inappropriately rebooted.

After a system failure occurs, the database server recovers automatically when you next start the database. The results of each transaction committed before the system error are intact. All changes by transactions that were not committed before the system failure are canceled.

☞ For more information about the recovery mechanism, see "Backup and recovery internals" on page 395.

☞ It is possible to recover uncommitted changes manually. For information, see "Recovering uncommitted operations" on page 418.

Ways of making backups

There are several distinct ways of making backups. This section introduces each of the major approaches, but does not address any issues of appropriate options.

You can make backups in the following ways:

♦ **Sybase Central** You can use the Backup Database wizard in Sybase Central to make a backup. You can access the wizard by selecting a database and choosing Backup Database from the File menu (or from the popup menu).

  ☞ For more information, see "Backing up a database directly to tape" on page 413.

♦ **Backup utility** The dbbackup command-line utility makes backups. For example, executing the following command at the command prompt makes backup copies of the database and transaction log in the directory *c:\backup* on the client machine:

```
dbbackup -c "connection-string" c:\backup
```

  You can make a backup copy of the database and transaction log in the director *c:\backup* on the server machine by specifying the -s option:

```
dbbackup -c "connection-string" -s c:\backup
```

  ☞ For more information, see "The Backup utility" on page 498.

♦ **SQL Statement** You can use a SQL statement to make the database server execute a backup operation. For example, the following statement places backup copies of the database file and transaction log into the directory *c:\backup* on the server machine.

```
BACKUP DATABASE
DIRECTORY 'c:\\backup'
```

  ☞ For more information, see "BACKUP statement" [*ASA SQL Reference*, page 307].

♦ **Offline backup** The above examples are all online backups, executed against a running database. You can make offline backups by copying the database files when the database is not running.

  ☞ For more information, see "Types of backup" on page 381.

Notes

You must have DBA authority or REMOTE DBA authority to make online backups of a database.

Understanding backups

To understand what files you need to back up, and how you restore databases from backups, you need to understand how the changes made to the database are stored on disk.

The database file

When a database is shut down, the database file holds a complete and current copy of all the data in the database. When a database is running, however, the database file is generally not current or complete.

The only time a database file is guaranteed to hold a complete and current copy of all data is immediately after a checkpoint completes. Following a checkpoint, all the contents of the database cache are on disk.

The database server checkpoints a database under the following conditions:

- ♦ As part of the database shutdown operations
- ♦ When the amount of time since the last checkpoint exceeds the database option CHECKPOINT_TIME
- ♦ When the estimated time to do a recovery operation exceeds the database option RECOVERY_TIME
- ♦ When the database server is idle long enough to write all dirty pages
- ♦ When a connection issues a CHECKPOINT statement
- ♦ When the database server is running without a transaction log and a transaction is committed

Between checkpoints, you need both the database file and another file, called the transaction log, to ensure that you have a complete copy of all committed transactions.

☞ For more information about checkpoints, see "Checkpoints and the checkpoint log" on page 396, and "How the database server decides when to checkpoint" on page 399.

The transaction log

The **transaction log** is a separate file from the database file. It stores all changes to the database. Inserts, updates, deletes, commits, rollbacks, and database schema changes are all logged. The transaction log is also called the **forward log** or the **redo log**.

The transaction log is a key component of backup and recovery, and is also essential for data replication using SQL Remote or the Replication Agent.

By default, all databases use transaction logs. Using a transaction log is optional, but you should always use a transaction log unless you have a specific reason not to. Running a database with a transaction log provides much greater protection against failure, better performance, and the ability to replicate data.

☞ For more information on how to use a transaction log to protect against media failure, see "Protecting against media failure on the database file" on page 391.

When changes are forced to disk

Like the database file, the transaction log is organized into **pages**: fixed size areas of memory. When a change is recorded in the transaction log, it is made to a page in memory. The change is forced to disk when the earlier of the following happens:

- ♦ The page is full.
- ♦ A COMMIT is executed.

In this way, completed transactions are guaranteed to be stored on disk, while performance is improved by avoiding a write to the disk on every operation.

☞ Configuration options are available to allow advanced users to tune the precise behavior of the transaction log. For more information, see "COOPERATIVE_COMMITS option [database]" on page 645, and "DELAYED_COMMITS option [database]" on page 651.

Transaction log mirrors

A **transaction log mirror** is an identical copy of the transaction log, maintained at the same time as the transaction log. If a database has a mirrored transaction log, every database change is written to both the transaction log and the transaction log mirror. By default, databases do not have transaction log mirrors.

A transaction log mirror provides extra protection for critical data. It enables complete data recovery in the case of media failure on the transaction log. A mirrored transaction log also enables a database server to carry out automatic validation of the transaction log on database startup.

☞ For more information, see "Protecting against media failure on the transaction log" on page 391.

Protecting your data against media failure

Backups protect your data against media failure.

☞ For an overview of data protection mechanisms, see "Protecting your data against failure" on page 376.

The practical aspects of recovery from media failure depend on whether the media failure is on the database file or the transaction log file.

Media failure on the database file If your database file is not usable, but your transaction log is still usable, you can recover all committed changes to the database as long as you have a proper backup procedure in place. All information since the last backed up copy of the database file is held in backed up transaction logs, or in the online transaction log.

☞ For more information on how to configure your database system, see "Protecting against media failure on the database file" on page 391.

Media failure on the transaction log file Unless you use a mirrored transaction log, you cannot recover information entered between the last database checkpoint and a media failure on the transaction log. For this reason, it is recommended that you use a mirrored transaction log in setups such as SQL Remote consolidated databases, where loss of the transaction log can lead to loss of key information, or the breakdown of a replication system.

☞ For more information, see "Protecting against media failure on the transaction log" on page 391.

Designing backup procedures

When you make a backup, you have a set of choices to make about how to manage transaction logs. The choices you make depend on a set of factors including the following:

- Is the database involved in replication?

  In this chapter, replication means SQL Remote replication, or MobiLink synchronization where *dbmlsync.exe* is running, or a database using the Replication Agent. Each of these replication methods requires access to the transaction log, and potentially to old transaction logs.

- How fast is the transaction log file growing relative to your available disk space? If the transaction log is growing quickly, you may not be able to afford to keep transaction logs available.

Types of backup

This section assumes that you are familiar with basic concepts related to backups.

☞ For more information about concepts related to backups, see "Introduction to backup and recovery" on page 374, and "Understanding backups" on page 378.

Backups can be categorized in several ways:

- **Full backup and incremental backup** A **full backup** is a backup of both the database file and of the transaction log. An **incremental backup** is a backup of the transaction log only. Typically, full backups are interspersed with several incremental backups.

  ☞ For more information on making backups, see "Making a full backup" on page 403, and "Making an incremental backup" on page 404.

- **Server-side backup and client-side backup** You can execute an online backup from a client machine using the Backup utility. To execute a server side backup, you execute the BACKUP statement; the database server then carries out the backup.

  You can easily build server side backup into applications because it is a SQL statement. Also, server-side backup is generally faster because the data does not have to be transported across the client/server communications system.

  Instructions for server-side and client-side backups are given together for each backup procedure.

- **Archive backup and image backup** An **archive backup** copies the database file and the transaction log into a single archive file, typically on a tape drive. An **image backup** makes a copy of the database file and/or the transaction log, each as separate files. You can only carry out archive backups as server-side backups, and you can only make full backups.

  You should use an archive backup if you are backing up directly to tape. Otherwise, an image backup has more flexibility for transaction log file management.

  Archive backups are supported on Windows NT/2000/XP and UNIX platforms only. On Windows CE, only image backups are permitted.

  ☞ For more information on archive backups, see "Backing up a database directly to tape" on page 413.

- **Online and offline backup** Backing up a running database provides a snapshot of a consistent database, even though other users are modifying the database. An offline backup consists simply of copying the files. You should only carry out an offline backup when the database is not running, and when the database server was shut down properly.

  The information in this chapter focuses on online backups.

- **Live backup** A live backup is a *continuous* backup of the database that helps protect against total machine failure.

  ☞ For more information on when to use live backups, see "Protecting against total machine failure" on page 392.

  ☞ For more information on how to make a live backup, see "Making a live backup" on page 414.

Scheduling backups

Most backup schedules involve periodic full backups interspersed with incremental backups of the transaction log. There is no simple rule for deciding how often to make backups of your data. The frequency with which you make backups depends on the importance of your data, how often it changes, and other factors.

Most backup strategies involve occasional full backups, interspersed by several incremental backups. A common starting point for backups is to carry out a weekly full backup, with daily incremental backups of the transaction log. Both full and incremental backups can be carried out online (while the database is running) or offline, on the server side or the client side. Archive backups are always full backups.

The kinds of failure against which a backup schedule protects you depends not only on how often you make backups, but also on how you operate your database server.

☞ For more information, see "Configuring your database for data protection" on page 391.

You should always keep more than one full backup. If you make a backup on top of a previous backup, a media failure in the middle of the backup leaves you with no backup at all. You should also keep some of your full backups offsite to protect against fire, flood, earthquake, theft, or vandalism.

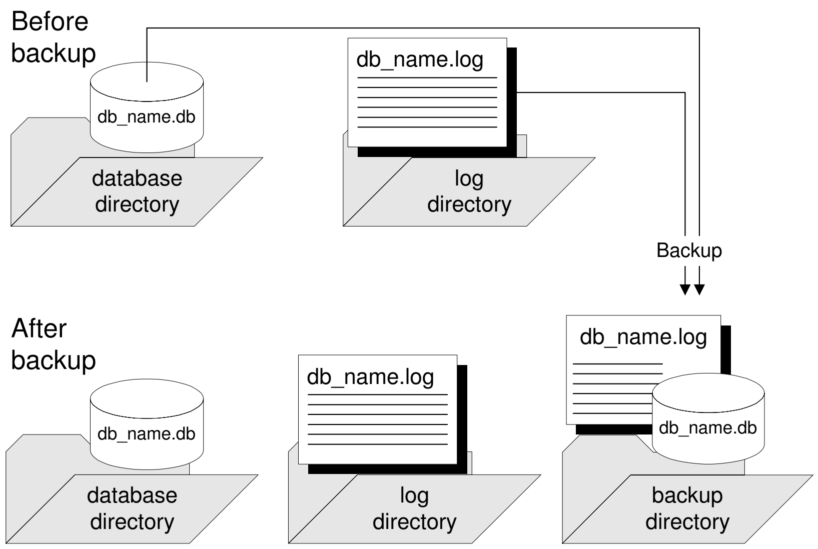
You can use the event scheduling features of Adaptive Server Anywhere to perform online backups automatically at scheduled times.

☞ For more information on scheduling operations such as backups, see "Automating Tasks Using Schedules and Events" on page 301.

A backup scheme for when disk space is plentiful

If disk space is not a problem on your production machine (where the database server is running) then you do not need to worry about choosing special options to manage the transaction log file. In this case, you can use a simple form of backup that makes copies of the database file and transaction log, and leaves the transaction log in place. All backups leave the database file in place.

A full backup of this kind is illustrated in the figure below. In an incremental backup, only the transaction log is backed up.

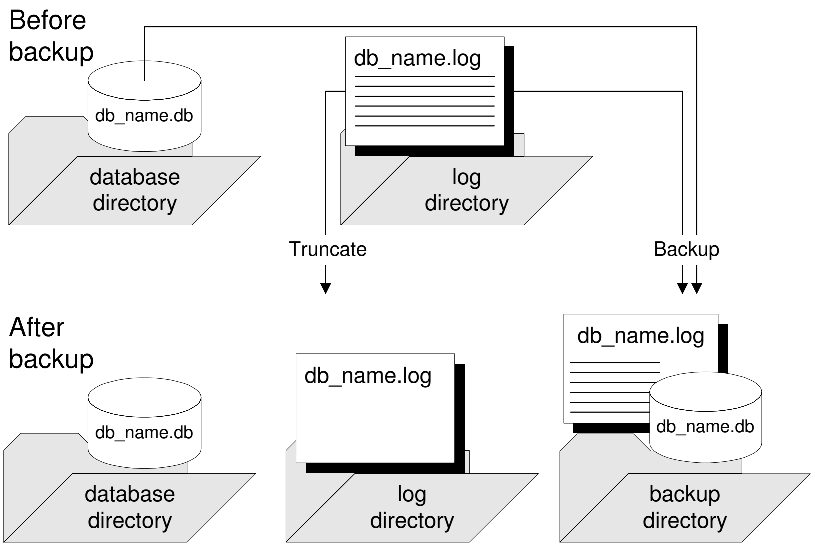


☞ For more information on how to carry out backups of this type, see "Making a backup, continuing to use the original transaction log" on page 407.

A backup scheme for databases not involved in replication

In many circumstances, disk space limitations make it impractical to let the transaction log grow indefinitely. In this case, you can choose to delete the contents of the transaction log when the backup is complete, freeing the disk space. You should not choose this option if the database is involved in replication because replication requires access to the transaction log.

A full backup, which truncates the log file, is illustrated in the figure below. In an incremental backup, only the transaction log is backed up.



Deleting the transaction log after each incremental backup makes recovery from a media failure on the database file a more complex task as there may then be several different transaction logs since the last full backup. Each transaction log needs to be applied in sequence to bring the database up to date.

You can use this kind of backup at a database that is operating as a MobiLink consolidated database because MobiLink does not rely on the transaction log. If you are running SQL Remote or the MobiLink *dbmlsync.exe* application, you must use a scheme suitable for preserving old transaction logs, as described in "A backup scheme for databases involved in replication" on page 385.

☞ For more information on how to carry out a backup of this type, see "Making a backup, deleting the original transaction log" on page 408.

A backup scheme for databases involved in replication

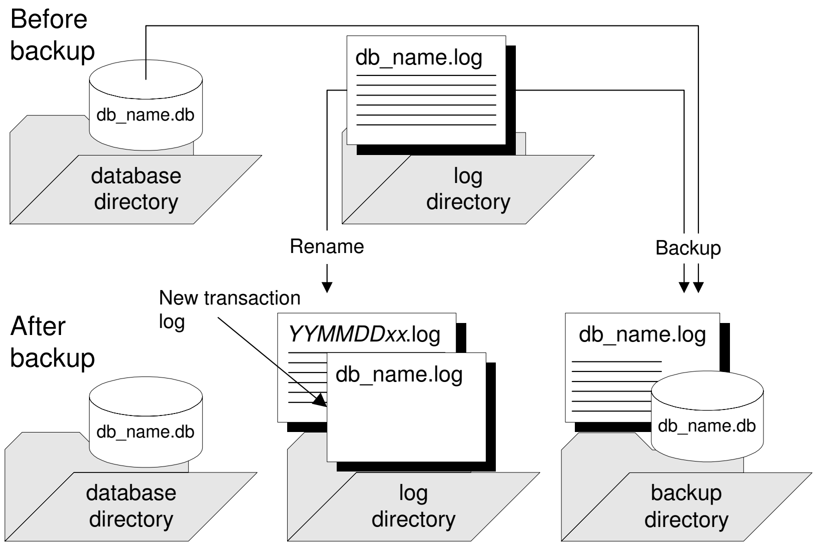
If your database is part of a SQL Remote installation, the Message Agent needs access to old transactions. If it is a consolidated database, it holds the master copy of the entire SQL Remote installation, and thorough backup procedures are essential.

If your database is a primary site in a Replication Server installation, the Replication Agent requires access to old transactions. However, disk space limitations often make it impractical to let the transaction log grow indefinitely.

If your database is participating in a MobiLink setup using the *dbmlsync.exe* executable, the same considerations apply. However, if your database is a MobiLink consolidated database, you do not need old transaction logs and can use a scheme for databases not involved in replication, as described in the previous section.

In these cases, you can choose backup options to rename and restart the transaction log. This kind of backup prevents open-ended growth of the transaction log, while maintaining information about the old transactions for the Message Agent and the Replication Agent.

This kind of backup is illustrated in the figure below.



☞ For more information on how to carry out a backup of this kind, see "Making a backup, renaming the original transaction log" on page 409.

Offline transaction logs

In addition to backing up the transaction log, the backup operation renames the online transaction log to a filename of the form *YYMMDDxx.log*. This file is no longer used by the database server, but is available for the Message Agent and the Replication Agent. It is called an **offline** transaction log. A new online transaction log is started with the same name as the old online transaction log.

There is no Year 2000 issue with the two-digit year in the *YYMMDDxx.log* filenames. The names are used for distinguishability only, not for ordering. For example, the renamed log file from the first backup on December 10, 2000, is named *001210AA.log*. The first two digits indicate the year, the second two digits indicate the month, the third two digits indicate the day of the month, and the final two characters distinguish among different backups made on the same day.

The Message Agent and the Replication Agent can use the offline copies to provide the old transactions as needed. If you set the DELETE_OLD_LOGS database option to ON, then the Message Agent and Replication Agent delete the offline files when they are no longer needed, saving disk space.

Backup methods for remote databases in replication installations

Backup procedures are not as crucial at remote databases as at the consolidated database. You may choose to rely on replication to the consolidated database as a data backup method. In the event of a media failure, the remote database would have to be re-extracted from the consolidated database, and any operations that have not been replicated would be lost. (You could use the Log Translation utility to attempt to recover lost operations.)

☞ For more information about the Log Translation utility, see "The Log Translation utility" on page 556.

Even if you do choose to rely on replication to protect remote database data, backups may still need to be done periodically at remote databases to prevent the transaction log from growing too large. You should use the same option (rename and restart the log) as at the consolidated database, running the Message Agent so that it has access to the renamed log files. If you set the DELETE_OLD_LOGS option to ON at the remote database, the old log files will be deleted automatically by the Message Agent when they are no longer needed.

Automatic transaction log renaming

You can use the -x Message Agent option to eliminate the need to rename the transaction log on the remote computer when the database server is shut down. The -x option renames the transaction log after it has been scanned for outgoing messages.

Backing up a database to a tape drive

All the types of backup described above are image backups. The backup copy of each file is also a file. To make backups to tape using an image backup, you have to take each backup copy and put it on tape using a disk backup utility.

You can carry out direct backup to a tape drive using an archive backup. Archive backups are always full backups. An archive backup makes copies of both the database file and the transaction log, but these copies are placed into a single file.

☞ You can make archive backups using the BACKUP statement. For information, see "Backing up a database directly to tape" on page 413, and "BACKUP statement" [*ASA SQL Reference*, page 307].

☞ You can restore the backup using the RESTORE statement. For information, see "Restoring an archive backup" on page 417, and "RESTORE DATABASE statement" [*ASA SQL Reference*, page 580].

Designing a backup and recovery plan

For reliable protection of your data, you should develop and implement a backup schedule. You should also ensure that you have a set of tested recovery instructions.

Typical schedules call for occasional full backups interspersed with several incremental backups. The frequency of each depends on the nature of the data that you are protecting.

If you use internal backups, you can use the scheduling features in Adaptive Server Anywhere to automate the task. Once you specify a schedule, the backups are carried out automatically by the database server.

☞ For more information on automating backups, see "Automating Tasks Using Schedules and Events" on page 301.

The length of time your organization can function without access to the data in your database imposes a maximum recovery time, and you should develop and test a backup and recovery plan that meets this requirement.

You should verify that you have the protection you need against media failure on the database file and on the transaction log file. If you are running in a replication environment, you should consider using a mirrored transaction log.

☞ For more information about media failure, see "Protecting your data against media failure" on page 379.

Factors that affect recovery time

External factors such as available hardware, the size of database files, recovery medium, disk space, and unexpected errors can affect your recovery time. When planning a backup strategy, you should allow additional recovery time for miscellaneous tasks that must be performed, such as entering recovery commands or retrieving and loading tapes.

Adding more files into the recovery scenario increases the places where recovery can fail. As the backup and recovery strategy develops, you should consider checking your recovery plan.

☞ For more information about how to implement a backup and recovery plan, see "Implementing a backup and recovery plan" on page 403.

Ensuring your database is valid

Database file corruption may not be apparent until applications try to access the affected part of the database. As part of your data protection plan, you should periodically check that your database has no errors. You can do this by **validating** the database. This task requires DBA authority.

Database validation includes a scan of every row in every table and a look-up of each row in each index on the table. Validation requires exclusive access to each table in turn. For this reason, it is best to validate when there is no other activity on the database. Database validation does not validate data, continued row references, or foreign key relationships unless you perform a full validation using the -f option.

If you created your database with checksums enabled, you can check the validity of the disk pages. For databases with checksums enabled, a checksum is calculated for each database page and this value is stored when the page is written to disk. You can use the Validation utility (dbvalid) or the Validate Database wizard in Sybase Central to perform checksum validation, which consists of reading the database pages from disk and calculating the checksum for the page. If the calculated checksum does not match the stored checksum for a page, the page has been modified or corrupted while on disk. If one or more pages has been corrupted, an error is returned and information about the invalid pages appears in the database server window.

☞ For more information about checksum validation, see "VALIDATE CHECKSUM statement" [*ASA SQL Reference*, page 660] or "The Validation utility" on page 604.

Caution
Backup copies of the database and transaction log must not be changed in any way. If there were no transactions in progress during the backup, you can check the validity of the backup database using read-only mode. However, if transactions were in progress, the database server must carry out recovery on the database when you start it. Recovery modifies the backup copy, which is not desirable.

If you can be sure that no transactions are in progress when the backup is being made, the database server does not need to carry out the recovery steps. In this case, you can carry out a validity check on the backup using the read-only database option.

Tip
Using the BACKUP statement with the WAIT BEFORE START clause ensures that no transactions are in progress when you make a backup.

If a base table in the database file is corrupt, you should treat the situation as a media failure, and recover from your previous backup. If an index is corrupt, you may want to unload the database without indexes, and reload.

☞ For more information, see "Validating a database" on page 405, and "Validating a transaction log" on page 406.

☞ For more information on read-only databases, see "-r server option" on page 154.

Configuring your database for data protection

There are several ways in which you can configure your database and the database server to provide protection against media failure while maintaining performance.

Protecting against media failure on the database file

When you create a database, by default the transaction log is put on the same device and in the same directory as the database. This arrangement does not protect against all kinds of media failure, and you should consider placing the transaction log in another location for production use.

For comprehensive protection against media failure, you should keep the transaction log on a different device from the database file. Some machines with two or more hard drives have only one physical disk drive with several logical drives or partitions: if you want reliable protection against media failure, make sure that you have a machine with at least two storage devices.

Placing the transaction log on a separate device can also result in improved performance by eliminating the need for disk head movement between the transaction log and the main database file.

You should not place the transaction log on a network directory. Reading and writing pages over a network gives poor performance and may result in file corruption.

☞ For more information on creating databases, see "Creating a database" [*ASA SQL User's Guide*, page 31].

☞ For more information on how to change the location of a transaction log, see "Changing the location of a transaction log" on page 422.

Protecting against media failure on the transaction log

It is recommended that you use a transaction log mirror when running high-volume or extremely critical applications. For example, at a consolidated database in a SQL Remote setup, replication relies on the transaction log, and if the transaction log is damaged or becomes corrupt, data replication can fail.

If you are using a mirrored transaction log, and an error occurs while trying to write to one of the logs (for example, if the disk is full), the database server stops. The purpose of a transaction log mirror is to ensure complete recoverability in the case of media failure on either log device; this purpose would be lost if the server continued with a single log.

You can specify the -fc option when starting the database server to implement a callback function when the database server encounters a file system full condition.

For more information, see "-fc server option" on page 139.

Where to store the transaction log mirror

There is a performance penalty for using a mirrored log as each database log write operation must be carried out twice. The performance penalty depends on the nature and volume of database traffic and on the physical configuration of the database and logs.

A transaction log mirror should be kept on a separate device from the transaction log. This improves performance. Also, if either device fails, the other copy of the log keeps the data safe for recovery.

Alternatives to a transaction log mirror

Alternatives to a mirrored transaction log are to use a disk controller that provides hardware mirroring, or operating-system level software mirroring, as provided by Windows NT/2000/XP and NetWare. Generally, hardware mirroring is more expensive, but provides better performance.

For more information

Live backups provide additional protection that has some similarities to transaction log mirroring.

☞ For more information, see "Differences between live backups and transaction log mirrors" on page 393.

☞ For information on creating a database with a mirrored transaction log, see "The Initialization utility" on page 530.

☞ For information on changing an existing database to use a mirrored transaction log, see "The Transaction Log utility" on page 580.

Protecting against total machine failure

You can use a **live backup** to provide a redundant copy of the transaction log that is available for restart of your system on a secondary machine in case the machine running the database server becomes unusable.

A live backup runs continuously, terminating only if the server shuts down. If you suffer a system failure, the backed up transaction log can be used for a rapid restart of the system. However, depending on the load that the server is processing, the live backup may lag behind and may not contain all committed transactions.

☞ For more information on making a live backup, see "Making a live backup" on page 414.

☞ For information on restarting database using a live backup, see "Recovering from a live backup" on page 417.

Differences between live backups and transaction log mirrors

Both a live backup and a transaction log mirror appear to provide a secondary copy of the transaction log. However, there are several differences between using a live backup and using a transaction log mirror:

- **In general, a live backup is made to a different machine** Running a transaction log mirror on a separate machine is not recommended. It can lead to performance and data corruption problems, and stops the database server if the connection between the machines goes down.

  By running the Backup utility on a separate machine, the database server does not do the writing of the backed up log file, and the data transfer is done by the Adaptive Server Anywhere client/server communications system. Therefore, performance impact is decreased and reliability is greater.

- **A live backup provides protection against a machine becoming unusable** Even if a transaction log mirror is kept on a separate device, it does not provide immediate recovery if the whole machine becomes unusable. You could consider an arrangement where two machines share access to a set of disks.

- **A live backup may lag behind the database server** A mirrored transaction log contains all the information required for complete recovery of committed transactions. Depending on the load that the server is processing, the live backup may lag behind and may not contain all the committed transactions.

Live backups and regular backups

The live backup of the transaction log is always the same length or shorter than the active transaction log. When a live backup is running, and another backup restarts the transaction log (`dbbackup -r` or `dbbackup -x`), the live backup automatically truncates the live backup log and restarts the live backup at the beginning of the new transaction log.

☞ For more information about how to make a live backup, see "Making a live backup" on page 414.

Controlling transaction log size

The size of the transaction log can determine what kind of backup is right for you, and can also affect recovery times.

You can control how fast the transaction log file grows by ensuring that all your tables have compact primary keys. If you carry out updates or deletes on tables that do not have a primary key or a unique index not allowing

NULL, the entire contents of the affected rows are entered in the transaction log. If a primary key is defined, the database server needs to store only the primary key column values to uniquely identify a row. If the table contains many columns or wide columns, the transaction log pages fill up much faster if no primary key is defined. In addition to taking up disk space, this extra writing of data affects performance.

If a primary key does not exist, the server looks for a UNIQUE NOT NULL index on the table (or a UNIQUE constraint). A UNIQUE index that allows NULL is not sufficient.

Backup and recovery internals

This section describes the internal mechanisms used during backup and during automatic recovery mechanism from system failures.

Backup internals

When you issue a backup instruction, the database may be in use by many people. If you later need to use your backup to restore your database, you need to know what information has been backed up, and what has not.

The database server carries out a backup as follows:

1. Issue a checkpoint. Further checkpoints are disallowed until the backup is complete. While the backup is taking place, any pages modified by other connections are saved before modification in the temporary file, instead of the database file, so that the backup image is made as of the checkpoint.

2. Make a backup of the database file, if the backup instruction is for a full backup.

3. Make a backup of the transaction log.

   The backup includes all operations recorded in the transaction log before the final page of the log is read. This may include instructions issued after the backup instruction was issued.

   The backup copy of the transaction log is generally smaller than the online transaction log. The database server allocates space to the online transaction logs in multiples of 64K, so the transaction log file size generally includes empty pages. However, only the non-empty pages are backed up.

4. On a database created with version 8.0 or later of Adaptive Server Anywhere, if the backup instruction requires the transaction log to be truncated or renamed, uncommitted transactions are carried forward to the new transaction log.

   On a database created with version 7.x or earlier of Adaptive Server Anywhere, if the backup instruction requires the transaction log to be truncated or renamed, then the database server waits until there are no uncommitted transactions before truncating or renaming the log file. If the database is busy, this wait may be significant.

   ☞ For more information about renaming and truncating the transaction log, see "Designing backup procedures" on page 381.

5. Mark the backup image of the database to indicate that recovery is needed. This causes any operations that happened since the start of the

backup to be applied. It also causes operations that were incomplete at the checkpoint to be undone if they were not committed.

Restrictions during backup and recovery

The database server prevents the following operations from being executed while a backup is in progress:

- ♦ Another backup, with the exception of a live backup.
- ♦ A checkpoint, other than the one issued by the backup instruction itself.
- ♦ Any statement that causes a checkpoint. This includes data definition statements, as well as the LOAD TABLE and TRUNCATE TABLE statements.

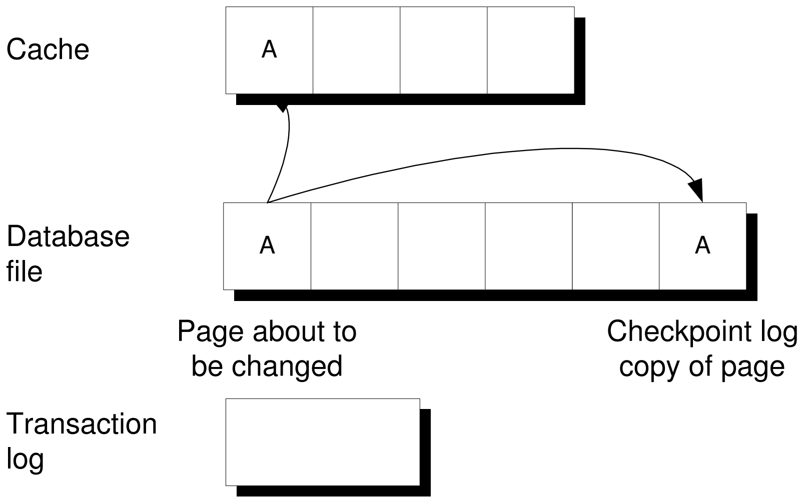
During recovery, including restoring backups, no action is permitted by other users of the database.

Checkpoints and the checkpoint log

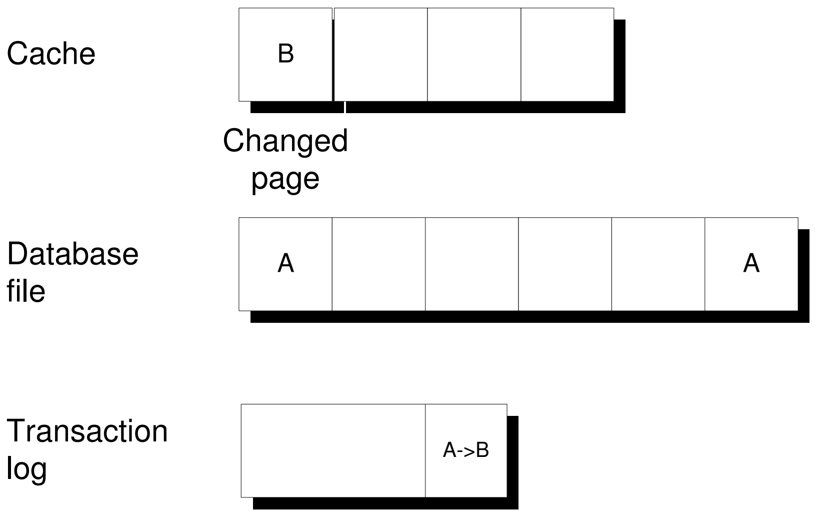
The database file is composed of pages: fixed size portions of hard disk. The checkpoint log is located at the end of the database file. Pages are added to the checkpoint log as necessary during a session, and the entire checkpoint log is deleted at the end of the session.

Before any page is updated (made **dirty**), the database server carries out the following operations:

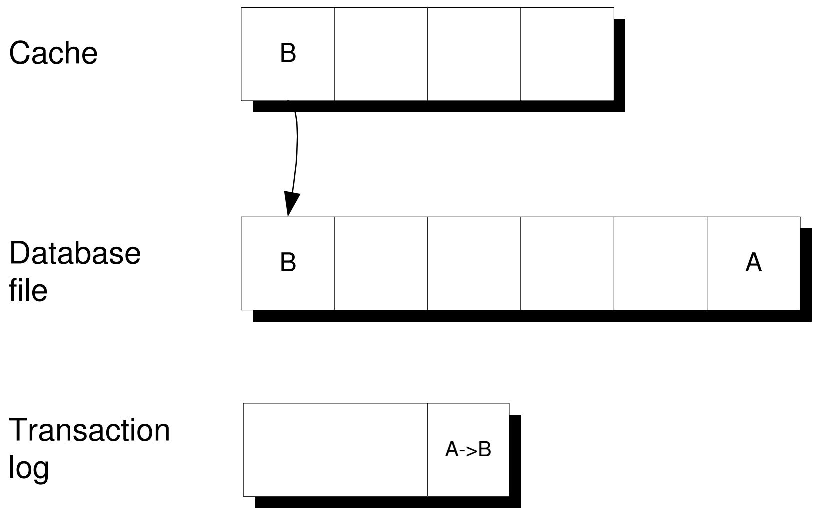
- ♦ It reads the page into memory, where it is held in the database cache.
- ♦ It makes a copy of the original page. These copied pages are the **checkpoint log**.



Changes made to the page are applied to the copy in the cache. For performance reasons they are not written immediately to the database file on disk.



When the cache is full, the changed page may get written out to disk. The copy in the checkpoint log remains unchanged.



A **checkpoint** is a point at which all dirty pages are written to disk and therefore represents a known consistent state of the database on disk. Following a checkpoint, the contents of the checkpoint log are deleted. The empty checkpoint log pages remain in the checkpoint log within a given session and can be reused for new checkpoint log data. As the checkpoint log increases in size, so does the database file.

At a checkpoint, all the data in the database is held on disk in the database file. The information in the database file matches that in the transaction log. During recovery, the database is first recovered to the most recent checkpoint, and then changes since that checkpoint are applied.

The entire checkpoint log, including all empty checkpoint log pages, is deleted at the end of each session. Deleting the checkpoint log causes the database to shrink in size.

☞ For more information, see "How the database server decides when to checkpoint" on page 399.

Transactions and the rollback log

As changes are made to the contents of a database, a **rollback log** is kept for the purpose of canceling changes if a transaction is rolled back or if a transaction is uncommitted when a system failure occurs. There is a separate rollback log for each connection. When a transaction is committed or rolled back, the rollback log contents for that connection are deleted. The rollback logs are stored in the database, and rollback log pages are copied into the checkpoint log along with other pages that are changed.

The rollback log is also called the **undo log**.

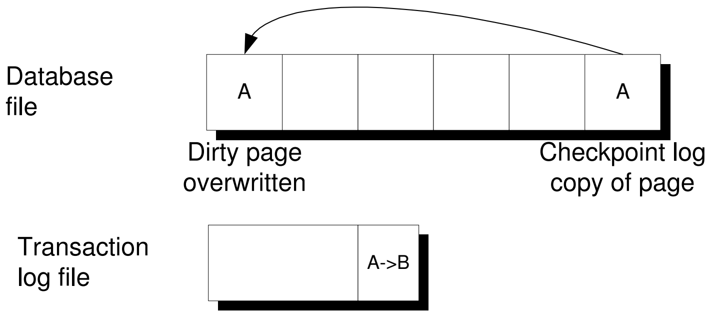
☞ For more information about transaction processing, see "Using Transactions and Isolation Levels" [*ASA SQL User's Guide*, page 101].

The automatic recovery process

When a database is shut down during normal operation, the database server carries out a checkpoint so that all the information in the database is held in the database file. This is a **clean** shutdown.

Each time you start a database, the database server checks whether the last shutdown was clean or the result of a system failure. If the database was not shut down cleanly, it automatically takes the following steps to recover from a system failure:

1. **Recover to the most recent checkpoint** All pages are restored to their state at the most recent checkpoint by copying the checkpoint log pages over the changes made since the checkpoint.



2. **Apply changes made since the checkpoint** Changes made between the checkpoint and the system failure, which are held in the transaction log, are applied.

3. **Rollback uncommitted transactions** Any uncommitted transactions are rolled back, using the rollback logs.

How the database server decides when to checkpoint

The priority of writing dirty pages to the disk increases as the time and the amount of work since the last checkpoint grows. The priority is determined by the following factors:

- **Checkpoint Urgency** The time that has elapsed since the last checkpoint, as a percentage of the checkpoint time setting of the database. The server -gc option controls the maximum desired time, in minutes, between checkpoints. You can also set the desired time using the CHECKPOINT_TIME option.

  ☞ For more information, see "-gc server option" on page 142.

- **Recovery Urgency** A heuristic to estimate the amount of time required to recover the database if it fails right now. The server -gr option controls the maximum desired time, in minutes, for recovery in the event of system failure. You can also set the desired time using the RECOVERY_TIME option.

  ☞ For more information, see "-gr server option" on page 147.

The checkpoint and recovery urgencies are important only if the server does not have enough idle time to write dirty pages.

Frequent checkpoints make recovery quicker, but also create work for the server writing out dirty pages.

There are two database options that allow you to control the frequency of checkpoints. CHECKPOINT_TIME controls the maximum desired time between checkpoints and RECOVERY_TIME controls the maximum desired time for recovery in the event of system failure.

☞ For more information, see "CHECKPOINT_TIME option [database]" on page 641 and "RECOVERY_TIME option [database]" on page 686.

The writing of dirty pages to disk is carried out by a task within the server called the **idle I/O task**. This task shares processing time with other database tasks.

There is a threshold for the number of dirty pages, below which writing of database pages does not take place.

When the database is busy, the urgency is low, and the cache only has a few dirty pages, the idle I/O task runs at a very low priority and no writing of dirty pages takes place.

Once the urgency exceeds 30%, the priority of the idle I/O task increases. At intervals, the priority is increased again. As the urgency becomes high, the server shifts its primary focus to writing dirty pages until the number gets below the threshold again. However, the server only writes out pages during the idle I/O task if the number of dirty pages is greater than the threshold.

If, because of other activity in the database, the number of dirty pages falls to zero, and if the urgency is 50% or more, then a checkpoint takes place automatically since it is a convenient time.

Both the checkpoint urgency and recovery urgency values increase in value until the checkpoint occurs, at which point they drop to zero. They do not decrease otherwise.

Validating the transaction log on database startup

When a database using a transaction log mirror starts up, the database server carries out a series of checks and automatic recovery operations to confirm that the transaction log and its mirror are not corrupt, and to correct some problems if corruption is detected.

On startup, the server checks that the transaction log and its mirror are identical by carrying out a full comparison of the two files; if they are identical, the database starts as usual. The comparison of log and mirror adds to database startup time.

If the database stopped because of a system failure, it is possible that some operations were written into the transaction log but not into the mirror. If the server finds that the transaction log and the mirror are identical up to the end of the shorter of the two files, the remainder of the longer file is copied into the shorter file. This produces an identical log and mirror. After this automatic recovery step, the server starts as usual.

If the check finds that the log and the mirror are different in the body of the shorter of the two, one of the two files is corrupt. In this case, the database does not start, and an error message is generated saying that the transaction log or its mirror is invalid.

Improving performance when validating databases

The VALIDATE TABLE statement can be slow when used on large databases running on servers with a cache size too small to contain the table and its largest index. It is often the case that all pages in the table are read at least once for each index. As well, if full compares are required for index lookups, the number of page reads can be proportional to the number of rows (not pages) in the table.

If you want to reduce the time taken to validate, you can use the WITH EXPRESS CHECK option with the VALIDATE TABLE statement, or the -fx option with the dbvalid utility. Depending on the size of your database, the size of your cache, and the type of validation you require, these two features can significantly reduce the time taken to perform validation.

Express validation causes each row of the table to be read and all columns evaluated, as is done with traditional validation using the WITH DATA CHECK clause. Each index is completely scanned once, and checks are done to ensure that the rows referenced in the index exist in the table. The express check option also does checks on the validity of individual index pages. The number of rows in the table must match the number of entries in the index. The express option saves time because it does not perform individual index lookups for each row.

Because the express check feature does not perform individual lookups, it is possible (though unlikely) for some form of index corruption to go unnoticed by the express validation feature. If index corruption should occur, data can be recovered by unloading and rebuilding the database since validation has confirmed that all of the data can be read.

Express validation is only supported for databases created with Adaptive Server Anywhere 7.0 or later.

☞ For more information about the express check option, see the "VALIDATE TABLE statement" [*ASA SQL Reference*, page 662], "Validating a database using the dbvalid command-line utility" on page 605, and the "sa_validate system procedure" [*ASA SQL Reference*, page 838].

Backup and recovery tasks

This section collects together instructions for tasks related to backup and recovery.

Implementing a backup and recovery plan

Regular backups and tested recovery commands are part of a comprehensive backup and recovery plan.

☞ For more information, see "Designing a backup and recovery plan" on page 388.

❖ To implement a backup and recovery plan

1. Create and verify your backup and recovery commands.
2. Measure the time it takes to execute backup and recovery commands.
3. Document the backup commands and create written procedures outlining where your backups are kept and identify any naming convention used, as well as the kind of backups performed.
4. Set up your backup procedures on the production server.
5. Monitor backup procedures to avoid unexpected errors. Make sure any changes in the process are reflected in your documentation.

☞ For more information about carrying out backups, see "Making a full backup" on page 403, and "Making an incremental backup" on page 404.

Making a full backup

A full backup is a backup of the database file and the transaction log file.

☞ For more information about the difference between a full backup and an incremental backup, see "Types of backup" on page 381.

❖ To make a full backup (overview)

1. Ensure that you have DBA authority on the database.
2. Perform a validity check on your database to ensure that it is not corrupt. You can use the Validation utility or the sa_validate stored procedure.

   ☞ For more information, see "Validating a database" on page 405.
3. Make a backup of your database file and transaction log.

☞ For information on how to carry out the backup operation, see the following:

♦ "Making a backup, continuing to use the original transaction log" on page 407.
♦ "Making a backup, deleting the original transaction log" on page 408.
♦ "Making a backup, renaming the original transaction log" on page 409.

Notes

Validity checking requires exclusive access to entire tables on your database. For more information and alternative approaches, see "Ensuring your database is valid" on page 389.

If you validate your backup copy of the database, make sure that you do so in read-only mode. Start the database server with the –r option to use read-only mode.

Making an incremental backup

An incremental backup is a backup of the transaction log file only. Typically, you should make several incremental backups between each full backup.

☞ For more information about the difference between a full backup and an incremental backup, see "Types of backup" on page 381.

❖ To make an incremental backup (overview)

1. Ensure that you have DBA authority on the database.
2. Make a backup of your transaction log only, not your database file.

☞ For more information about how to carry out the backup operation, see the following:

♦ "Making a backup, continuing to use the original transaction log" on page 407.
♦ "Making a backup, deleting the original transaction log" on page 408.
♦ "Making a backup, renaming the original transaction log" on page 409.

Notes

The backup copies of the database file and transaction log file have the same names as the online versions of these files. For example, if you make a backup of the sample database, the backup copies are called *asademo.db* and *asademo.log*. When you repeat the backup statement, choose a new backup directory to avoid overwriting the backup copies.

☞ For more information about how to make a repeatable incremental backup command by renaming the backup copy of the transaction log, see

"Renaming the backup copy of the transaction log during backup" on page 412.

Validating a database

Validating a database is a key part of the backup operation. For information, see "Ensuring your database is valid" on page 389.

☞ For an overview of the backup operation, see "Making a full backup" on page 403.

Caution
Validating a table or an entire database should be performed while no connections are making changes to the database; otherwise, spurious errors may be reported indicating some form of database corruption even though no corruption actually exists.

❖ To check the validity of an entire database (Sybase Central)

1. Connect to the database as a user with DBA authority.

2. In the left pane of Sybase Central, select the database.

3. From the File menu, choose Validate Database

   The Validate Database wizard appears.

4. Follow the instructions in the wizard.

   A message box indicates whether the database is valid or not.

❖ To check the validity of an entire database (SQL)

1. Connect to the database as a user with DBA authority.

2. Execute the sa_validate stored procedure:

   ```
   call sa_validate
   ```

   The procedure returns a single column, named Messages. If all tables are valid, the column contains `No errors detected`.

   ☞ For more information, see "sa_validate system procedure" [*ASA SQL Reference,* page 838].

❖ To check the validity of an entire database (Command line)

1. Connect to the database as a user with DBA authority.
2. Run the dbvalid utility:

   ```
   dbvalid -c "connection_string"
   ```

   ☞ For more information, see "The Validation utility" on page 604.

Notes

If you are checking the validity of a backup copy, you should run the database in read-only mode so that it is not modified in any way. You can only do this when there were no transactions in progress during the backup.

☞ For information on running databases in read-only mode, see "-r server option" on page 154.

Validating a single table

You can check the validity of a single table either from Sybase Central or using a SQL statement. You must have DBA authority or be the owner of a table to check its validity.

❖ To check the validity of a table (Sybase Central)

1. Open the Tables folder.
2. Right-click the table and choose Validate from the popup menu.

   A message box indicates whether the table is valid or not.

❖ To check the validity of a table (SQL)

1. Execute the VALIDATE TABLE statement:

   ```
   VALIDATE TABLE table_name
   ```

Notes

If errors are reported, you can drop all of the indexes and keys on a table and recreate them. Any foreign keys to the table will also need to be recreated. Another solution to errors reported by VALIDATE TABLE is to unload and reload your entire database. You should use the dbunload -u option so that it does not try to use a possibly corrupt index to order the data.

Validating a transaction log

The process for validating a transaction log file depends on whether it is in use on a production database (online) or is not in use (offline, or a backup copy).

❖ To validate an online transaction log

1. Run your database with a mirrored transaction log. The database server automatically validates the transaction log each time the database is started.

❖ To validate an offline or backed up transaction log

1. Run the Log Translation utility (dbtran) against the log file. If the Log Translation utility can successfully read the log file, it is valid.

Making a backup, continuing to use the original transaction log

This task describes the simplest kind of backup, which leaves the transaction log untouched.

☞ For more information about when to use this type of backup, see "A backup scheme for when disk space is plentiful" on page 383.

❖ To make a backup, continuing to use the original transaction log (Sybase Central)

1. Start Sybase Central. Connect to the database as a user with DBA authority.
2. Right-click the database and choose Create Backup Images from the popup menu.

   The Create Backup Images wizard appears.
3. Click Next on the introductory page of the wizard.
4. Select the database that you want to back up.
5. On the next page, enter the name of a directory to hold the backup copies, and choose whether to carry out a complete backup (all database files) or an incremental backup (transaction log file only).
6. On the next page, select the option Continue To Use The Same Transaction Log.
7. Click Finish to start the backup.

The procedure describes a client-side backup. There are more options available for this kind of backup.

If you choose a server-side backup, and the server is running on a different machine from Sybase Central, you cannot use the Browse button to locate a directory in which to place the backups. The Browse button browses the client machine, while the backup directory is relative to the server.

❖ To make a backup, continuing to use the original transaction log (SQL)

1. If you are using the BACKUP statement, use the following clauses only:

   ```
   BACKUP DATABASE
   DIRECTORY directory_name
   [ TRANSACTION LOG ONLY ]
   ```

   Include the TRANSACTION LOG ONLY clause if you are making an incremental backup.

❖ To make a backup, continuing to use the original transaction log (Command line)

1. If you are using the dbbackup utility, use the following syntax:

   ```
   dbbackup -c "connection_string" [ -t ] backup_directory
   ```

   Include the -t option only if you are making an incremental backup.

Making a backup, deleting the original transaction log

If your database is not involved in replication, and if you have limited disk space on your online machine, you can delete the contents of the online transaction log (**truncate** the log) when you make a backup. In this case, you need to use every backup copy made since the last full backup during recovery from media failure on the database file.

☞ For more information about when to use this type of backup, see "A backup scheme for databases not involved in replication" on page 384.

❖ To make a backup, deleting the transaction log (Sybase Central)

1. Start Sybase Central. Connect to the database as a user with DBA authority.
2. Right-click the database and choose Create Backup Images from the popup menu.

   The Create Backup Images wizard appears.
3. Click Next on the introductory page of the wizard.
4. Select the database that you want to back up.
5. On the next page, enter the name of a directory to hold the backup copies, and choose whether to carry out a complete backup (all database files) or an incremental backup (transaction log file only).
6. On the next page, select the option Truncate the Transaction Log.
7. Click Finish to start the backup.

❖ To make a backup, deleting the transaction log (SQL)

1. Use the BACKUP statement with the following clauses:

   ```
   BACKUP DATABASE
   DIRECTORY backup_directory
   [ TRANSACTION LOG ONLY ]
   TRANSACTION LOG TRUNCATE
   ```

   Include the TRANSACTION LOG ONLY clause only if you are making an incremental backup.

   The backup copies of the transaction log and database file are placed in *backup_directory*. If you enter a path, it is relative to the working directory of the database server, not your client application.

❖ To make a backup, deleting the transaction log (Command line)

1. From the command prompt, enter the following command:

   ```
   dbbackup -c "connection_string" -x [ -t ] backup_directory
   ```

   Include the -t option only if you are making an incremental backup.

   The backup copies of the transaction log and database file are placed in *backup_directory*. If you enter a path, it is relative to the directory from which you run the command.

Notes

Before the online transaction log can be erased, all outstanding transactions must terminate. If there are outstanding transactions, your backup cannot complete.

☞ For information, see "Backup internals" on page 395.

You can determine which connection has an outstanding transaction using the sa_conn_info system procedure. If necessary, you can disconnect the user with a DROP CONNECTION statement.

☞ For more information, see "Determining which connection has an outstanding transaction" on page 411.

Making a backup, renaming the original transaction log

This set of backup options is typically used for databases involved in replication. In addition to making backup copies of the database file and transaction log, the transaction log at backup time is renamed to an offline log, and a new transaction log is started with the same name as the log in use at backup time.

☞ For more information about when to use this set of backup options, see "A backup scheme for databases involved in replication" on page 385.

❖ To make a backup, renaming the transaction log (Sybase Central)

1. Start Sybase Central. Connect to the database as a user with DBA authority.
2. Right-click the database and choose Create Backup Images from the popup menu.

   The Create Backup Images wizard appears.
3. Click Next on the introductory page of the wizard.
4. Select the database that you want to back up.
5. On the next page, enter the name of a directory to hold the backup copies, and choose whether to carry out a complete backup (all database files) or an incremental backup (transaction log file only).
6. On the next page, select the option Rename the Transaction Log.
7. Click Finish to start the backup.

❖ To make a backup, renaming the transaction log (SQL)

1. Use the BACKUP statement with the following clauses:

   ```
   BACKUP DATABASE
   DIRECTORY backup_directory
   [ TRANSACTION LOG ONLY ]
   TRANSACTION LOG RENAME
   ```

   Include the TRANSACTION LOG ONLY clause only if you are making an incremental backup.

   The backup copies of the transaction log and database file are placed in *backup_directory*. If you enter a path, it is relative to the working directory of the database server, not your client application.

❖ To make a backup, renaming the transaction log (Command line)

1. From a command prompt, enter the following command. You must enter the command on a single line:

   ```
   dbbackup -c "connection_string" -r [ -t ] backup_directory
   ```

   Include the -t option if you are making an incremental backup.

   The backup copies of the transaction log and database file are placed in *backup_directory*. If you enter a path, it is relative to the directory from which you run the command.

Notes

Before the online transaction log can be renamed, all outstanding transactions must terminate. If there are outstanding transactions, your backup will not complete.

☞ For information, see "Backup internals" on page 395.

You can determine which connection has an outstanding transaction using the sa_conn_info system procedure. If necessary, you can disconnect the user with a DROP CONNECTION statement.

☞ For more information, see "Determining which connection has an outstanding transaction" on page 411.

Determining which connection has an outstanding transaction

If you are carrying out a backup that renames or deletes the transaction log on a database created with version 8.0 or later of Adaptive Server Anywhere, incomplete transactions are carried forward to the new transaction log.

If you are carrying out a backup that renames or deletes the transaction log on a database created with version 7.x or earlier of Adaptive Server Anywhere, however, and if there are outstanding transactions, the backup must wait until those transactions are complete before it can complete.

You can use a system procedure to determine which user has outstanding transactions. If there are not too many connections, you can also use the Adaptive Server Anywhere Console utility to determine which connection has outstanding connections.

❖ To determine which connection has an outstanding transaction (SQL)

1. Connect to the database from Interactive SQL or another application that can call stored procedures.

2. Execute the sa_conn_info system procedure:

```
CALL sa_conn_info
```

3. Inspect the **UncmtOps** column to see which connection has uncommitted operations.

   ☞ For more information, see "sa_conn_info system procedure" [*ASA SQL Reference*, page 782].

❖ **To determine which connection has an outstanding transaction (Adaptive Server Anywhere Console utility)**

1. Connect to the database from the Adaptive Server Anywhere Console utility.

   For example, the following command connects to the default database using user ID DBA and password SQL:

   ```
   dbconsole -c "uid=DBA;pwd=SQL"
   ```

   ☞ For more information, see "The Adaptive Server Anywhere Console utility" on page 497.

2. Double-click each connection, and inspect the Uncommitted Ops entry to see which users have uncommitted operations. If necessary, you can disconnect the user to enable the backup to finish.

Renaming the backup copy of the transaction log during backup

By default, the backup copy of the transaction log file has the same name as the online file. For each backup operation, you must assign a different name or location for the backup copy, or you must move the backup copy before the next backup is done.

You can make a repeatable incremental backup command by renaming the backup copy of the transaction log.

❖ **To rename the backup copy of the transaction log (SQL)**

1. Use the MATCH keyword in the BACKUP statement. For example, the following statement makes an incremental backup of the asademo database to the directory *c:\backup*. The backup copy of the transaction log is called *YYMMDDxx.log*, where *YYMMDD* is the date and *xx* is a counter, starting from AA.

   ```
   BACKUP DATABASE
   DIRECTORY 'c:\\backup'
   TRANSACTION LOG ONLY
   TRANSACTION LOG RENAME MATCH
   ```

❖ **To rename the backup copy of the transaction log (Command line)**

1. Supply the -n option to dbbackup. For example, the following command makes an incremental backup of the sample database, renaming the backup copy of the transaction log.

   ```
   dbbackup -c "uid=DBA;pwd=SQL;dbn=asademo" -r -t -n c:\backup
   ```

Notes

The backup copy of the transaction log is named *YYMMDDxx.log*, where *YY* is the year, *MM* is the month, *DD* is the day of the month, and *xx* runs from AA to ZZ, incrementing if there is more than one backup per day. There is no Year 2000 issue with the two-digit year in the *YYMMDDxx.log* filenames. The names are used for distinguishability only, not for ordering.

Backing up a database directly to tape

An archive backup makes a copy of the database file and transaction log file in a single archive destination. Only server-side, full backups can be made in this manner. When you make an archive backup in Sybase Central, you have the option of backing up the database directly to tape or to disk.

☞ For more information, see "Types of backup" on page 381.

❖ To make an archive backup to tape (Sybase Central)

1. Start Sybase Central. Connect to the database as a user with DBA authority.

2. Right-click the database and choose Backup Database from the popup menu.

   The Backup Database wizard appears.

3. Click Next on the introductory page of the wizard.

4. Select the database that you want to back up to tape.

5. On the second page of the wizard, select the On Tape to the Following Device option and type the tape drive in the text box below.

6. Click Finish to start the backup.

❖ To make an archive backup to tape (SQL)

1. Use the BACKUP statement, with the following clauses:

   ```
   BACKUP DATABASE
   TO archive_root
   [ ATTENDED { ON | OFF } ]
   [ WITH COMMENT comment string ]
   ```

   If you set the ATTENDED option to OFF, the backup fails if it runs out of tape or disk space. If ATTENDED is set to ON, you are prompted to take an action, such as replacing the tape, when there is no more space on the backup archive device.

Notes

The BACKUP statement makes an entry in the text file *backup.syb*, in the same directory as the server executable.

☞ For more information about restoring from an archive backup, see "Restoring an archive backup" on page 417.

Example

The following statement makes a backup to the first tape drive on a Windows NT machine:

```
BACKUP DATABASE
TO '\\\\.\\tape0'
ATTENDED OFF
WITH COMMENT 'May 6 backup'
```

The first tape drive on Windows NT is `\\.\tape0`. Because the backslash is an escape character in SQL strings, each backslash is preceded by another.

The following statement makes an archive file on disk named *c:\backup\archive.1*.

```
BACKUP DATABASE
TO 'c:\\backup\\archive'
```

Making a live backup

You carry out a live backup of the transaction log using the dbbackup utility with the -l option.

☞ For more information about live backups, see "Protecting against total machine failure" on page 392.

❖ To make a live backup

1. Set up a secondary machine from which you can run the database if the online machine fails. For example, ensure that you have Adaptive Server Anywhere installed on the secondary machine.

2. Periodically, carry out a full backup to the secondary machine.

3. Run a live backup of the transaction log to the secondary machine.

```
dbbackup -l path\filename.log -c "connection_string"
```

You should normally run the dbbackup utility from the secondary machine.

If the primary machine becomes unusable, you can restart your database using the secondary machine. The database file and the transaction log hold the information needed to restart.

Recovering from media failure on the database file

The recovery process depends on whether you leave the transaction log untouched on incremental backup in your backup process. If your backup operation deletes or renames the transaction log, you may have to apply changes from several transaction logs. If your backup operation leaves the transaction log untouched, you need to use only the online transaction log in recovery.

☞ For more information about the backup types discussed here, see "Designing backup procedures" on page 381.

❖ To recover from media failure on the database file

1. Make an extra backup copy of the current transaction log. The database file is gone, and the only record of changes since the last backup is in the transaction log.

2. Create a **recovery directory** to hold the files you use during recovery.

3. Copy the database file from the last full backup to the recovery directory.

4. Apply the transactions held in the backed up transaction logs to the recovery database.

   For each log file, in chronological order,

   - Copy the log file into the recovery directory.

   - Start the database server with the apply transaction log (-a) option, to apply the transaction log:

     ```
     dbeng9 db_name.db -a log_name.log
     ```

     The database server shuts down automatically once the transactions have been applied.

5. Copy the online transaction log into the recovery directory. Apply the transactions from the online transaction log to the recovery database.

   ```
   dbeng9 db_name.db -a db_name.log
   ```

6. Perform validity checks on the recovery database.

   ☞ For more information, see "Validating a database" on page 405.

7. Make a post-recovery backup.

8. Move the database file to the production directory.

9. Allow user access to the production database.

Recovering from media failure on an unmirrored transaction log

If your database is a primary site in a Replication Server installation, or a consolidated database in a SQL Remote installation, you should use a mirrored transaction log, or hardware equivalent.

☞ For more information, see "Protecting against media failure on the transaction log" on page 391.

❖ To recover from media failure on an unmirrored transaction log (partial recovery)

1. Make an extra backup copy of the database file immediately. With the transaction log gone, the only record of the changes between the last backup and the most recent checkpoint is in the database file.

2. Delete or rename the transaction log file.

3. Restart the database with the -f option.

   ```
   dbeng9 asademo.db -f
   ```

   > ***Caution***
   > *This command should only be used when the database is not participating in a SQL Remote or Replication Server replication system. If your database is a consolidated database in a SQL Remote replication system, you may have to re-extract the remote databases.*

   Without the -f option, the server reports the lack of a transaction log as an error. With the option, the server restores the database to the most recent checkpoint and then rolls back any transactions that were not committed at the time of the checkpoint. A new transaction log is then created.

Recovering from media failure on a mirrored transaction log

❖ To recover from media failure on a mirrored transaction log

1. Make an extra copy of the backup of your database file taken at the time the transaction log was started.

2. Identify which of the two files is corrupt. Run the Log Translation utility on the transaction log and on its mirror to see which one generates an error message. The Log Translation utility is accessible from Sybase Central or as the dbtran utility.

   The following command line translates a transaction log named *asademo.log*, placing the translated output into *asademo.SQL*:

```
dbtran asademo.log
```

The Log Translation utility properly translates the intact file, and reports an error while translating the corrupt file.

3. Copy the correct file over the corrupt file so that you have two identical files again.

4. Restart the server.

Recovering from a live backup

A live backup is made to a separate machine from the primary machine that is running your production database. To restart a database from a live backup, you must have Adaptive Server Anywhere installed on the secondary machine.

☞ For more information about live backups, see "Protecting against total machine failure" on page 392.

❖ To restart a database using a live backup

1. Start the database server on the secondary machine with the apply transaction log (-a) option to apply the transaction log and bring the database up to date:

   ```
   dbeng9 asademo.db -a filename.log
   ```

   The database server shuts down automatically once the transaction log is applied.

2. Start the database server in the normal way, allowing user access. Any new activity is appended to the current transaction log.

Restoring an archive backup

If you use an archive backup (typically to tape), you use the RESTORE statement to recover your data.

☞ For more information about making archive backups, see "Backing up a database directly to tape" on page 413.

❖ To restore a database from an archive backup (Sybase Central)

1. In Sybase Central, connect to a database with DBA authority.

2. In the left pane, select the Adaptive Server Anywhere plug-in, and click the Utilities tab in the right pane.

3. In the right pane, double-click Restore Database.

   The Restore Database wizard appears.

4. Follow the instructions in the wizard.

❖ To restore a database from an archive backup (Interactive SQL)

1. Start a personal database server. Use a command such as the following, which starts a server named restore:

   ```
   dbeng9 -n restore
   ```

2. Start Interactive SQL. On the Identification tab of the Connect dialog, enter a user ID of **DBA** and a password of **SQL**. Leave all other fields on this tab blank.

3. Click the Database tab and enter a database name of **utility_db**. Leave all other fields on this tab blank.

4. Click OK to connect.

5. Execute the RESTORE statement, specifying the archive root. At this time, you can choose to restore an archived database to its original location (default), or to a different machine with different device names using the RENAME clause.

   ☞ For more information, see "RESTORE DATABASE statement" [*ASA SQL Reference*, page 580].

Example

The following statement restores a database from a tape archive to the database file *c:\newdb\newdb.db*.

```
RESTORE DATABASE 'c:\\newdb\\newdb.db'
FROM '\\\\.\\tape0'
```

The following statement restores a database from an archive backup in file *c:\backup\archive.1* to the database file *c:\newdb\newdb.db*. The transaction log name and location are specified in the database.

```
RESTORE DATABASE 'c:\\newdb\\newdb.db'
FROM 'c:\\backup\\archive'
```

☞ For more information, see "RESTORE DATABASE statement" [*ASA SQL Reference*, page 580].

Recovering uncommitted operations

When recovering from media failure on the database file, the transaction log is intact. Recovery reapplies all committed transactions to the database. In some circumstances, you may wish to find information about transactions that were incomplete at the time of the failure.

❖ **To recover uncommitted operations from a transaction log (Sybase Central)**

1. Do one of the following:
   - ♦ If you are connected to a database, in the left pane, select the Adaptive Server Anywhere plug-in, and then click the Utilities tab in the right pane. In the right pane, double-click Translate Log File.
   - ♦ If you are not connected to a database, click Tools ➤ Adaptive Server Anywhere 9 ➤ Translate Log File.

     The Translate Log File wizard appears.
2. Follow the instructions in the wizard.
3. Edit the translated log (SQL command file) in a text editor and identify the instructions you need.

❖ **To recover uncommitted operations from a transaction log (Command line)**

1. Run dbtran to convert the transaction log into a SQL command file, using the -a option to include uncommitted transactions. For example, the following command uses dbtran to convert a transaction log:

   ```
   dbtran -a sample.log changes.SQL
   ```
2. Edit the translated log (SQL command file) in a text editor and identify the instructions you need.

   ☞ For more information on the Log Translation utility, see "The Log Translation utility" on page 556.

Note

The transaction log may or may not contain changes right up to the point where a failure occurred. It does contain any changes made before the end of the most recently committed transaction that made changes to the database.

Recovering from multiple transaction logs

Adaptive Server Anywhere 8.0.0 and later allow transactions to span multiple transaction log files. If a database is backed up part way through a transaction, the transaction may span two transaction log files. When this occurs, the first part of the transaction is contained in the offline transaction log, while the second part of the transaction is contained in the online transaction log.

If you need to recover your database and you have multiple transaction logs, you must apply the transaction log files to the backup copy of your database *in the correct order* in case there are transactions that span multiple transaction logs. If the transaction logs are not applied in the correct order, then portions of transactions that span multiple transaction logs will be rolled back.

You can determine the order in which transaction log files were generated using the dbtran utility. The SQL scripts it generates display the earliest log offset in the transaction log, which can be an effective method for determining the order in which multiple log files were generated to ensure that you apply them in the correct order. You can also use the scripts to determine if your transaction logs contain transactions that span transaction logs.

☞ For more information, see "The Transaction Log utility" on page 580.

In order to maintain the integrity of your data when recovering a database from multiple transaction logs, you can either use the -a server option to apply each log individually to the backup copy of the database, or use the Log Translation utility (dbtran) to translate one or more transaction logs into a *.SQL* file that can be applied to the backup copy of the database.

Recovering from multiple transaction logs using the -a server option

The -a server option is used recover a database by applying a single transaction log file to the backup copy of a database. When this option is specified, the database server applies the log and then terminates—it will not continue to run. If you have multiple transaction logs, you must apply them one at a time in the correct order (from oldest to most recent.

❖ To recover from multiple transaction logs using the -a server option

1. Start the database server using -a to apply the backup transaction log to the backup copy of your database.

2. Start the database server and apply the current transaction log to the backup copy of your database.

☞ For more information, see "-a recovery option" on page 170.

Example

The following example applies the offline (backup) and current transaction log to the backup copy of the sample database using the -a database server option.

1. Start the database server and apply a backup transaction log called *backupasademo.log* to the backup copy of a database called *backupasademo.db*:

```
dbeng9 -a backupasademo.log backupasademo.db
```

The database server applies the backup transaction log to the backup copy of the database and then shuts down.

2. Start the database server and apply the current transaction log called *asademo.log* to the backup copy of the database:

   ```
   dbeng9 -a asademo.log backupasademo.db
   ```

   The database server applies the current transaction log to the backup copy of the database and then shuts down.

Recovering from multiple transaction logs using the dbtran utility

To maintain the integrity of your data when you use dbtran to translate multiple transaction logs, you must specify both the -m and -n options. The -m option instructs the Log Translation utility to generate a file (named by -n) containing all the transactions from the logs in the specified directory.

You need to use -m because if you translate each log individually using dbtran, any transactions that span transaction log files could be rolled back. This is because when dbtran translates a log, it adds a ROLLBACK statement to the end of the log to undo any uncommitted transactions. In cases where a transaction spans two logs, the COMMIT for the transaction occurs in the second log file. Operations at the end of the first log file would be rolled back by dbtran because the file does not contain a COMMIT for the transaction. Translating all the transaction log files in a directory using -m ensures that all your transactions are translated.

☞ For more information, see "The Transaction Log utility" on page 580.

❖ To recover from multiple transaction logs using the dbtran utility

1. Run the Log Translation utility (dbtran) against the directory containing the transaction log files and output the resulting SQL statements into a *.SQL* file.

2. Start the backup copy of your database.

3. Apply the *.SQL* file generated by dbtran in step 1 to the backup copy of your database from Interactive SQL.

Example

The following example uses the dbtran utility to apply the backup and current transaction logs to the backup copy of the database.

1. Run the Log Translation utility against the *c:|backup* directory and output the SQL statements into a file called *recoverylog.sql*:

   ```
   dbtran -m "c:\backup" -n recoverylog.sql
   ```

2. Start the backup copy of the database called *backupasademo.db*:

   ```
   dbeng9 backupasademo.db
   ```

3. Apply the *recoverylog.sql* file to the database from Interactive SQL:

```
dbisql "uid=dba;pwd=sql;eng=backupasademo" READ
          recoverylog.sql
```

Changing the location of a transaction log

The database must not be running when you change the location of a transaction log.

☞ For more information about how to choose the location of a transaction log, see "Protecting against media failure on the database file" on page 391.

❖ To change the location of a transaction log (Sybase Central)

1. Do one of the following:
   - ♦ If you are already connected to the database associated with the log file, select the Adaptive Server Anywhere plug-in in the left pane and then open the Utilities tab in the right pane. In the right pane, double-click Change Log File Settings.
   - ♦ If you are not connected to the database, click Tools ➤ Adaptive Server Anywhere 9 ➤ Change Log File Settings.

     The Change Log File Settings wizard appears.
2. Follow the instructions in the wizard.

❖ To change the location of a transaction log mirror for an existing database (Command line)

1. Ensure that the database is not running.
2. Enter the following command at the command prompt:

   ```
   dblog -t new-log-file database-file
   ```

   ☞ For more information about dblog options, see "Transaction log utility options" on page 582.

Creating a database with a transaction log mirror

You can choose to maintain a transaction log mirror when you create a database. This option is available either from the CREATE DATABASE statement, from Sybase Central, or from the *dbinit* utility.

☞ For more information about why you may wish to use a transaction log mirror, see "Protecting against media failure on the transaction log" on page 391.

❖ To create a database that uses a transaction log mirror (Sybase Central)

1. Do one of the following:
   - ♦ If you are connected to a database, in the left pane, select the Adaptive Server Anywhere plug-in and then open the Utilities tab in the right pane. In the right pane, double-click Create Database.
   - ♦ If you are not connected to a database, click Tools ➤ Adaptive Server Anywhere 9 ➤ Create Database.

     The Create Database wizard appears.
2. Follow the instructions in the wizard.

❖ To create a database that uses a transaction log mirror (SQL)

1. Use the CREATE DATABASE statement, with the TRANSACTION LOG clause.

   ☞ For more information, see "CREATE DATABASE statement" [*ASA SQL Reference*, page 338].

❖ To create a database that uses a transaction log mirror (Command line)

1. Use the dbinit utility with the -m option. For example, the following command (which should be entered on one line) initializes a database named *company.db*, with a transaction log kept on a different device and a mirror on a third device.

   ```
   dbinit -t d:\log_dir\company.log -m
   e:\mirr_dir\company.mlg c:\db_dir\company.db
   ```

   ☞ For more information about initialization options, see "Initialization utility options" on page 532.

Starting a transaction log mirror for an existing database

Using the Transaction Log utility, you can choose to maintain a transaction log mirror for an existing database any time the database is not running. This option is available from either Sybase Central or the dblog utility.

☞ For more information about why you may wish to use a transaction log mirror, see "Protecting against media failure on the transaction log" on page 391.

❖ To start a transaction log mirror for an existing database (Sybase Central)

1. Do one of the following:
   - ♦ If you are already connected to the database associated with the log mirror, in the left pane, select the Adaptive Server Anywhere plug-in, and then open the Utilities tab in the right pane. In the right pane, double-click Change Log File Settings.
   - ♦ If you are not connected to the database, click Tools ➤ Adaptive Server Anywhere 9 ➤ Change Log File Settings.

     The Change Log File Settings wizard appears.

2. Follow the instructions in the wizard.

❖ To start a transaction log mirror for an existing database (Command line)

1. Ensure that the database is not running.

2. Enter the following command at the command prompt:

   ```
   dblog -m mirror-file database-file
   ```

   ☞ For more information about dblog options, see "Transaction log utility options" on page 582.

You can also use the dblog utility and Sybase Central to stop a database from using a transaction log mirror.

PART III

PERMISSIONS AND REPLICATION

This section describes how to use user IDs and permissions to maintain a secure database. It also describes how to replicate data with the Replication Server.

CHAPTER 13

Managing User IDs and Permissions

About this chapter

Each user of a database must have a name they type when connecting to the database, called a user ID. This chapter describes how to manage user IDs.

Contents

Database permissions overview

Proper management of user IDs and permissions lets users of a database carry out their jobs effectively, while maintaining the security and privacy of information within the database.

You use SQL statements for assigning user IDs to new users of a database, granting and revoking permissions for database users, and finding out the current permissions of users.

Database permissions are assigned to user IDs. Throughout this chapter, the term **user** is used as a synonym for user ID. Remember, however, that you grant and revoke permissions for each user ID.

Setting up individual user IDs

Even if there are no security concerns regarding a multi-user database, there are good reasons for setting up an individual user ID for each user. The administrative overhead is very low if a group with the appropriate permissions is set up. This chapter discusses groups of users.

You may want to use individual user IDs since:

- ♦ The log translation utility can selectively extract the changes made by individual users from a transaction log. This is very useful when troubleshooting or piecing together what happened if data is incorrect.
- ♦ Sybase Central displays much more useful information with individual user IDs as you can tell which connections are which users.
- ♦ Row locking messages (with the BLOCKING option set to OFF) are more informative.

DBA authority overview

When you create a database, you also create a single usable user ID. This first user ID is **DBA**, and the password is initially **SQL**. The **DBA** user ID automatically has DBA authority within the database. This level of permission enables DBA users to carry out any activity in the database. They can create tables, change table structures, create new user IDs, revoke permissions from users, and so on.

Users with DBA authority

A user with DBA authority becomes the **database administrator**. In this chapter, references made to the database administrator, or **DBA**, include *any user or users with DBA authority*.

Although DBA authority may be granted or transferred to other user IDs, this chapter assumes that the **DBA** user ID is the database administrator, and that the abbreviation DBA means both the **DBA** user ID and any user ID with DBA authority.

Adding new users

The DBA has the authority to add new users to the database. As the DBA adds users, they are also granted permissions to carry out tasks on the database. Some users may need to simply look at the database information using SQL queries, others may need to add information to the database, and others may need to modify the structure of the database itself. Although some of the responsibilities of the DBA may be handed over to other user IDs, the DBA is responsible for the overall management of the database by virtue of the DBA authority.

The DBA has authority to create database objects and assign ownership of these objects to other user IDs.

RESOURCE authority overview

RESOURCE authority is the permission to create database objects, such as tables, views, stored procedures, and triggers. RESOURCE authority may be granted only by the DBA.

To create a trigger, a user needs both RESOURCE authority and ALTER permissions on the table to which the trigger applies.

Ownership permissions overview

The creator of a database object becomes the owner of that object. Ownership of a database object carries with it permissions to carry out actions on that object. These permissions are not assigned to users in the same way that other permissions in this chapter are assigned.

Owners

A user who creates a new object within the database is called the **owner** of that object, and automatically has permission to carry out any operation on that object. The owner of a table may modify the structure of that table, for instance, or may grant permissions to other database users to update the information within the table.

The DBA has permission to modify any component within the database, and so could delete a table created by another user. The DBA has all the permissions regarding database objects that the owners of each object have. As well, the DBA can also create database objects for other users. In this case, the owner of an object is not the user ID that executed the CREATE statement. A use for this ability is discussed in “Groups without passwords” on page 448. Despite this possibility, this chapter refers interchangeably to the owner and creator of database objects as the same person.

Table and views permissions overview

There are several distinct permissions you may grant to user IDs concerning tables and views:

Permission	Description
ALTER	Permission to alter the structure of a table or create a trigger on a table
DELETE	Permission to delete rows from a table or view
INSERT	Permission to insert rows into a table or view
REFER-ENCES	Permission to create indexes on a table and to create foreign keys that reference a table
SELECT	Permission to look at information in a table or view
UPDATE	Permission to update rows in a table or view. This may be granted on a set of columns in a table only
ALL	All the above permissions

Group permissions overview

Setting permissions individually for each user of a database can be a time-consuming and error-prone process. For most databases, permission management based on groups, rather than on individual user IDs, is a much more efficient approach.

You can assign permissions to a group in exactly the same way as to an individual user. You can then assign membership in appropriate groups to each new user of the database, and they gain a set of permissions by virtue of their group membership.

Example

For example, you may create groups for different departments in a company database (sales, marketing, and so on) and assign these groups permissions. Each salesperson becomes a member of the sales group, and automatically gains access to the appropriate areas of the database.

Each user ID can be a member of multiple groups, and they inherit all permissions from each of the groups.

Setting user and group options

In Sybase Central, configurable options for users and groups are located in the User Options and Group Options dialogs (the same dialog as for setting database options). In Interactive SQL, you can specify an option in a SET OPTION statement.

❖ To set options for a user or group (Sybase Central)

1. Connect to a database and open the Users & Groups folder.
2. Right-click the desired user or group and choose Options from the popup menu.
3. Edit the desired values.
4. Click Set Permanent Now.

❖ To set the options for a user or group (SQL)

1. Specify the desired properties within a SET OPTION statement.

See also

- ♦ "Setting properties for database objects" [*ASA SQL User's Guide*, page 36]
- ♦ "Database Options" on page 613

Managing individual user IDs and permissions

This section describes how to create new users and grant permissions to them. For most databases, the bulk of permission management should be carried out using **groups**, rather than by assigning permissions to individual users one at a time. However, as a group is simply a user ID with special properties, you should read and understand this section before moving on to the discussion of managing groups.

Creating new users

You can create new users in both Sybase Central and Interactive SQL. In Sybase Central, you manage users or groups in the Users & Groups folder. In Interactive SQL, you can add a new user using the GRANT CONNECT statement. For both tools, you need DBA authority to create new users.

All new users are automatically added to the PUBLIC group. Once you have created a new user, you can:

- ♦ add it to other groups

  ☞ For more information, see "Granting group membership to existing users or groups" on page 445.

- ♦ set its permissions on tables, views, and procedures

  ☞ For more information, see "Managing individual user IDs and permissions" on page 432.

- ♦ set it as the publisher or as a remote user of the database

  ☞ For more information, see "Managing SQL Remote permissions" [*SQL Remote User's Guide,* page 201].

Initial permissions for new users

By default, the permissions assigned to new users include:

- ♦ the ability to connect to the database (assuming a password has been specified for the user)
- ♦ the ability to view system tables
- ♦ the ability to execute most system stored procedures

To access tables in the database, new users need to be assigned permissions.

The DBA can set the permissions granted automatically to new users by assigning permissions to the special PUBLIC user group, as discussed in "Special groups" on page 448.

❖ To create a new user (Sybase Central)

1. Open the Users & Groups folder.
2. From the File menu, choose New ➤ User.

   The User Creation wizard appears.
3. Follow the instructions in the wizard.

❖ To create a new user (SQL)

1. Connect to a database as a user ID with DBA authority.
2. Execute a GRANT CONNECT TO statement.

Example

Add a new user to the database with the user ID of M_Haneef and a password of welcome.

```
GRANT CONNECT TO M_Haneef
IDENTIFIED BY welcome
```

See also

- ♦ "GRANT statement" [*ASA SQL Reference,* page 503]

Setting a password

A user must have a password to be able to connect to the database. Passwords cannot be more than 255 bytes in length, and cannot contain semicolons.

It is recommended that passwords be composed of 7-bit ASCII characters as 8-bit ASCII characters above 127 may not work correctly when character set translation is turned on and the character sets do not match.

If you want to use passwords with characters above 127, you should do one of the following:

- ♦ specify **CharSet=none** in your connection string to turn off character set translation
- ♦ specify the character set using the CharSet connection parameter

  ☞ For more information about the character sets you can specify, see "Character set labels" on page 369.
- ♦ specify the -ct- option to turn off character set translation on the database server

  ☞ For more information, see "-ct server option" on page 131.

Changing a password

Using the GRANT statement, you can change your password or that of another user if you have DBA authority. For example, the following command changes the password for user ID M_Haneef to new_password:

```
GRANT CONNECT TO M_Haneef
IDENTIFIED BY new_password
```

Changing the DBA password

The default password for the **DBA** user ID for all databases is **SQL**. You should change this password to prevent unauthorized access to your database. The following command changes the password for user ID **DBA** to **new_password**:

```
GRANT CONNECT TO DBA
IDENTIFIED BY new_password
```

Granting DBA and RESOURCE authority

You can grant DBA and RESOURCE authority in the same manner.

❖ To grant RESOURCE authority to a user ID

1. Connect to the database as a user with DBA authority.

2. Type and execute the SQL statement:

   ```
   GRANT RESOURCE TO userid
   ```

For DBA authority, the appropriate SQL statement is:

```
GRANT DBA TO userid
```

Notes

- ♦ Only the DBA may grant DBA or RESOURCE authority to database users.
- ♦ DBA authority is very powerful: anyone with this authority has the ability to carry out any action on the database, as well as access to all the information in the database. It is wise to grant DBA authority to only a few people.
- ♦ Consider giving users with DBA authority two user IDs, one with DBA authority and one without, so that they connect as DBA only when necessary.
- ♦ RESOURCE authority allows the user to create new database objects, such as tables, views, indexes, procedures, or triggers.

Granting permissions on tables

You can assign a set of permissions on individual tables and grant users combinations of these permissions to define their access to a table.

You can use either Sybase Central or Interactive SQL to set permissions. In Interactive SQL, you can use the GRANT statement to grant the following permissions on tables:

- The ALTER permission allows a user to alter the structure of a table or to create triggers on a table. The REFERENCES permission allows a user to create indexes on a table and to create foreign keys. These permissions grant the authority to modify the database schema, and so will not be assigned to most users. These permissions do not apply to views.
- The DELETE, INSERT, and UPDATE permissions grant the authority to modify the data in a table. Of these, the UPDATE permission may be restricted to a set of columns in the table or view.
- The SELECT permission grants authority to look at data in a table, but does not give permission to alter it.
- ALL permission grants all the above permissions.

❖ To grant permissions on tables or columns (Sybase Central)

1. Connect to the database.
2. Open the Tables folder for that database.
3. Right-click a table and choose Properties from the popup menu.
4. On the Permissions tab of the Table property sheet, configure the permissions for the table:
   - Click Grant to select users or groups to which to grant permissions.
   - Click the fields beside the user or group to set specific permissions. Permissions are indicated by a check mark, and grant options are indicated by a check mark with two '+' signs.
   - Select a user and click the Change button beside References, Select, or Update to set that type of permission on individual columns.
   - Select a user or group in the list and click Revoke to revoke all permissions.

Tips

You can also assign permissions from the user/group property sheet. To assign permissions to many users and groups at once, use the table's property sheet. To assign permissions to many tables at once, use the user's property sheet.

❖ To grant permissions on tables or columns (SQL)

1. Connect to the database with DBA authority or as the owner of the table.
2. Execute a GRANT statement to assign the permission.

   ☞ For more information, see "GRANT statement" [*ASA SQL Reference*, page 503].

Example 1

All table permissions are granted in a very similar fashion. You can grant permission to M_Haneef to delete rows from the table named sample_table as follows:

1. Connect to the database as a user with DBA authority, or as the owner of sample_table.
2. Type and execute the following SQL statement:

```
GRANT DELETE
ON sample_table
TO M_Haneef
```

Example 2

You can grant permission to M_Haneef to update the column_1 and column_2 columns only in the table named sample_table as follows:

1. Connect to the database as a user with DBA authority, or as the owner of sample_table.
2. Type and execute the following SQL statement:

```
GRANT UPDATE (column_1, column_2)
ON sample_table
TO M_Haneef
```

Table permissions are limited in that they generally apply to all the data in a table, although SELECT and UPDATE permissions may be granted to a subset of columns. You can fine-tune user permissions by creating procedures that carry out actions on tables, and then granting users the permission to execute the procedure.

See also

♦ "GRANT statement" [*ASA SQL Reference*, page 503]

Granting permissions on views

Setting permissions on views is similar to setting them on tables.

☞ For more information about the SQL statements involved, see "Granting permissions on tables" on page 435.

A user may perform an operation through a view if one or more of the following are true:

- ♦ The appropriate permission(s) on the view for the operation has been granted to the user by a DBA.
- ♦ The user has the appropriate permission(s) on all the base table(s) for the operation.
- ♦ The user was granted appropriate permission(s) for the operation on the view by a non-DBA user. This user must be either the owner of the view or have WITH GRANT OPTION of the appropriate permission(s) on the view. The owner of the view must be either:
  - a DBA.
  - a non-DBA, but also the owner of all the base table(s) referred to by the view.
  - a non-DBA, and not the owner of some or all of the base table(s) referred to by the view. However, the view owner has SELECT permission WITH GRANT OPTION on the base table(s) not owned and any other required permission(s) WITH GRANT OPTION on the base table(s) not owned for the operation.

    Instead of the owner having permission(s) WITH GRANT OPTION on the base table(s), permission(s) may have been granted to PUBLIC. This includes SELECT permission on system tables.

UPDATE permissions can be granted only on an entire view. Unlike tables, UPDATE permissions cannot be granted on individual columns within a view.

❖ To grant permissions on views (Sybase Central)

1. Connect to the database.
2. Open the Views folder for that database.
3. Right-click a view and choose Properties from the popup menu.
4. On the Permissions tab of the View property sheet, configure the permissions for the view:

- Click Grant to select users or groups to which to grant full permissions.
- Click in the fields beside the user or group to set specific permissions. Permissions are indicated by a check mark, and grant options are indicated by a check mark with two '+' signs.
- Select a user or group in the list and click Revoke to revoke all permissions.

Tip
You can also assign permissions from the User or Group property sheet. To assign permissions to many users and groups at once, use the view's property sheet. To assign permissions to many views at once, use the User's or Group's property sheet.

See also

- "GRANT statement" [*ASA SQL Reference*, page 503]

Granting users the right to grant permissions

You can assign each of the table and view permissions described in "Granting permissions on tables" on page 435 with the WITH GRANT OPTION. This option gives the right to pass on the permission to other users. In the context of groups, you can read about this feature in section "Permissions of groups" on page 447.

In Sybase Central, you can specify a grant option by displaying the property sheet of a user, group, or table, clicking the Permissions tab, and double-clicking in the fields provided so that a check mark with two '+' signs appears.

Example

You can grant permission to M_Haneef to delete rows from the table named sample_table, and the right to pass on this permission to other users, as follows:

1. Connect to the database as a user with DBA authority, or as the owner of sample_table:
2. Type and execute the SQL statement:

```
GRANT DELETE ON sample_table
TO M_Haneef
WITH GRANT OPTION
```

☞ For more information, see "GRANT statement" [*ASA SQL Reference*, page 503]

Granting permissions on procedures

The DBA or the owner of the procedure (the user ID that created the procedure) may grant permission to execute stored procedures. The EXECUTE permission is the only permission that may be granted on a procedure.

The method for granting permissions to execute a procedure is similar to that for granting permissions on tables and views, discussed in "Granting permissions on tables" on page 435. However, the WITH GRANT OPTION clause of the GRANT statement does not apply to the granting of permissions on procedures.

You can use either Sybase Central or Interactive SQL to set permissions.

❖ To grant permissions on procedures (Sybase Central)

1. Connect to the database.
2. Open the Procedures & Functions folder for that database.
3. Right-click a procedure and choose Properties from the popup menu.
4. On the Permissions tab of the Procedure property sheet, configure the permissions for the procedure:
   - ♦ Click Grant to select users or groups to which to grant full permissions.
   - ♦ Click beside users in the Execute column to toggle between granting or not granting permission.
   - ♦ Select a user or group in the list and click Revoke to revoke all permissions.

Tip
You can also assign permissions from the User or Group property sheet. To assign permissions to many users and groups at once, use the procedure's property sheet. To assign permissions to many procedures at once, use the User's or Group's property sheet.

❖ To grant permissions on procedures (SQL)

1. Connect to the database with DBA authority or as the owner of the procedure.
2. Execute a GRANT EXECUTE ON statement.

Example

You can grant M_Haneef permission to execute a procedure named my_procedure, as follows:

1. Connect to the database as a user with DBA authority or as owner of my_procedure procedure.

2. Execute the SQL statement:

```
GRANT EXECUTE
ON my_procedure
TO M_Haneef
```

Execution permissions of procedures

Procedures execute with the permissions of their owner. Any procedure that updates information in a table will execute successfully *only* if the owner of the procedure has UPDATE permissions on the table.

As long as the procedure owner has the proper permissions, the procedure executes successfully when called by any user assigned permission to execute it, whether or not they have permissions on the underlying table. You can use procedures to allow users to carry out well-defined activities on a table, without having any general permissions on the table.

See also

- "GRANT statement" [*ASA SQL Reference,* page 503]

Execution permissions of triggers

The server executes triggers in response to a user action. Triggers do not require permissions to be executed. When a trigger executes, it does so with the permissions of the creator of the table with which it is associated.

☞ For more information on trigger permissions, see "Trigger execution permissions" [*ASA SQL User's Guide,* page 675].

Granting and revoking remote permissions

In Sybase Central, you can manage the remote permissions of both users and groups. Remote permissions allow normal users and groups to become remote users in a SQL Remote replication setup in order to exchange replication messages with the publishing database.

Granting remote permissions

You cannot grant remote permissions to a user until you define at least one message type in the database.

To grant remote permissions to a group, you must explicitly grant remote permissions to each user in the group. Remote permissions are not inherited by members of a group.

Revoking remote permissions

Revoking remote permissions reverts a remote user to a normal user. Revoking these permissions also automatically unsubscribes that user from all publications.

❖ To grant remote permissions to users (Sybase Central)

1. Connect to a database.
2. Open the Users & Groups folder.
3. Right-click the desired user and choose Change to Remote User from the popup menu.

   The Change User to Remote User dialog appears.
4. In the resulting dialog, enter the desired values.

Once you have granted remote permissions to a user, you can subscribe it to publications:

❖ To revoke remote permissions from remote users

1. Open either the Users & Groups folder or the SQL Remote Users folder.
2. Right-click the desired remote user and choose Revoke Remote from the popup menu.

☞ For more information, see "SQL Remote Concepts" [*SQL Remote User's Guide,* page 7].

Revoking user permissions

Any user's permissions are a combination of those that have been granted and those that have been revoked. By revoking and granting permissions, you can manage the pattern of user permissions on a database.

The REVOKE statement is the exact converse of the GRANT statement. To disallow M_Haneef from executing my_procedure, the command is:

```
REVOKE EXECUTE
ON my_procedure
FROM M_Haneef
```

The DBA or the owner of the procedure must issue this command.

Permission to delete rows from sample_table may be revoked by issuing the following command:

```
REVOKE DELETE
ON sample_table
FROM M_Haneef
```

See also

- "Granting permissions on tables" on page 435
- "Granting permissions on views" on page 437
- "Granting permissions on procedures" on page 438

Deleting users from the database

You can delete a user from the database using both Sybase Central and Interactive SQL. The user being removed cannot be connected to the database during this procedure.

Deleting a user also deletes all database objects (such as tables) that they own.

Only the DBA can delete a user.

❖ To delete a user from the database (Sybase Central)

1. Open the Users & Groups folder.
2. Right-click the desired user and choose Delete from the popup menu.

❖ To delete a user from the database (SQL)

1. Execute a REVOKE CONNECT FROM statement.

Example

Remove the user M_Haneef from the database.

```
REVOKE CONNECT FROM M_Haneef
```

See also

- “REVOKE statement” [*ASA SQL Reference*, page 585]
- “Revoking user permissions” on page 441
- “Deleting groups from the database” on page 449

Managing connected users

If you are working with Sybase Central, you can keep track of all users connected to the database. You can view properties of these connected users, and you can disconnect them if you want.

❖ To display a list of all users connected to a database

1. Select the database in the left pane, and click the Connected Users tab in the right pane.

   This tab displays all users currently connected to a given database (including yourself), regardless of the application that they used to connect (Sybase Central, Interactive SQL, a custom client application, etc.).

❖ To inspect the properties of a user's connection to a database

1. Select the database in the left pane, and click the Connected Users tab in the right pane.
2. Right-click the desired user and choose Properties from the popup menu.
3. Inspect the desired properties.

❖ To disconnect users from a database

1. Select the database in the left pane, and click the Connected Users tab in the right pane.
2. Right-click the desired user and choose Disconnect from the popup menu.

Managing groups

Once you understand how to manage permissions for individual users (as described in the previous section), working with groups is straightforward. A user ID identifies a group, just like it does a single user. However, a group user ID has the permission to have members.

DBA, RESOURCE, and GROUP permissions

When you grant permissions to a group or revoke permissions from a group for tables, views, and procedures, all members of the group inherit those changes. The DBA, RESOURCE, and GROUP permissions are not inherited: you must assign them individually to each of the individual user IDs requiring them.

A group is simply a user ID with special permissions. You grant permissions to a group and revoke permissions from a group in exactly the same manner as any other user, using the commands described in "Managing individual user IDs and permissions" on page 432.

You can construct a hierarchy of groups, where each group inherits permissions from its parent group. That means that a group can also be a member of a group. As well, each user ID may belong to more than one group, so the user-to-group relationship is many-to-many.

The ability to create a group without a password enables you to prevent anybody from connecting to the database using the group user ID.

☞ For more information about this security feature, see "Groups without passwords" on page 448.

☞ For more information on altering database object properties, see "Setting properties for database objects" [*ASA SQL User's Guide*, page 36].

☞ For more information about granting remote permissions for groups, see "Granting and revoking remote permissions" on page 440.

Creating groups

You can create a new group in both Sybase Central and Interactive SQL. You need DBA authority to create a new group.

❖ To create a new group (Sybase Central)

1. Click the Users & Groups folder.
2. From the File menu, choose New ➤ Group.

   The Group Creation wizard appears.
3. Follow the instructions in the wizard.

❖ To create a new group (SQL)

1. Execute a GRANT GROUP TO statement. If the user ID you cite in this statement has not already been created, you need to create it first.

Example

Create the user ID personnel.

```
GRANT CONNECT
TO personnel
IDENTIFIED BY group_password
```

Make the user ID personnel a group.

```
GRANT GROUP TO personnel
```

See also

- ♦ "GRANT statement" [*ASA SQL Reference*, page 503]
- ♦ "Creating new users" on page 432

Granting group membership to existing users or groups

You can add existing users to groups or add groups to other groups in both Sybase Central and Interactive SQL. In Sybase Central, you can control group membership on the property sheets of users or groups. In Interactive SQL, you can make a user a member of a group with the GRANT statement.

When you assign a user membership in a group, they inherit all the permissions on tables, views, and procedures associated with that group.

Only the DBA can grant membership in a group.

❖ To add a user or group to another group (Sybase Central)

1. Open the Users & Groups folder.

2. Select the user/group that you want to add to another group and from the File menu, choose New ➤ Memberships.

   The New Memberships dialog appears.

3. Select the desired group and click OK.

4. The group membership appears on the Memberships tab in the right pane for the selected user/group.

❖ To add a user or group to another group (SQL)

1. Execute a GRANT MEMBERSHIP IN GROUP statement, specifying the desired group and the users involved.

Example

Grant the user M_Haneef membership in the personnel group:

```
GRANT MEMBERSHIP
IN GROUP personnel
TO M_Haneef
```

See also

- ♦ "GRANT statement" [*ASA SQL Reference*, page 503]
- ♦ "Creating new users" on page 432

Revoking group membership

You can remove users or groups from a group in both Sybase Central and Interactive SQL.

Removing a user or group from a group does *not* delete them from the database (or from other groups). To do this, you must delete the user/group itself.

Only the DBA can revoke membership in a group.

❖ To remove a user or group from another group (Sybase Central)

1. Open the Users & Groups folder.
2. Select the desired user/group and click the Memberships tab in the right pane.
3. Right-click the desired group and choose Remove Membership from the popup menu.

   The user or group is then removed from this desired group.

Tip
You can perform this action by selecting the group, clicking the Members tab in the right pane, and then right-clicking the user/group you wish to remove and choosing Remove Member from the popup menu.

❖ To remove a user or group from another group (SQL)

1. Execute a REVOKE MEMBERSHIP IN GROUP statement, specifying the desired group and the users involved.

Example

Remove the user M_Haneef from the personnel group:

```
REVOKE MEMBERSHIP
IN GROUP personnel
FROM M_Haneef
```

See also

- ♦ "REVOKE statement" [*ASA SQL Reference*, page 585]
- ♦ "Creating new users" on page 432

- “Deleting users from the database” on page 442
- “Deleting groups from the database” on page 449

Permissions of groups

You may grant permissions to groups in exactly the same way as to any other user ID. Permissions on tables, views, and procedures are inherited by members of the group, including other groups and their members. Some complexities to group permissions exist that database administrators need to keep in mind.

Notes

Members of a group do not inherit the DBA, RESOURCE, and GROUP permissions. Even if the personnel user ID has RESOURCE authority, the members of personnel do not have RESOURCE authority.

Ownership of database objects is associated with a single user ID and is not inherited by group members. If the user ID personnel creates a table, then the personnel user ID is the owner of that table and has the authority to make any changes to the table, as well as to grant privileges concerning the table to other users. Other user IDs who are members of personnel are not the owners of this table, and do not have these rights. Only granted permissions are inherited. For example, if the DBA or the personnel user ID explicitly grants SELECT permission on a table to the personnel user ID, all group members inherit select access to the table.

Referring to tables owned by groups

Groups are used for finding tables and procedures in the database. For example, the query

```
SELECT * FROM SYSGROUPS
```

always finds the view SYS.SYSGROUPS, because all users belong to the PUBLIC group, and PUBLIC belongs to the SYS group which owns the SYSGROUPS view. (The SYSGROUPS view contains a list of *group_name*, *member_name* pairs representing the group memberships in your database.)

If a table named employees is owned by the user ID personnel, and if **M_Haneef** is a member of the personnel group, then **M_Haneef** can refer to the employees table simply as employees in SQL statements. Users who are not members of the personnel group need to use the **qualified** name personnel.employees.

Creating a group to own the tables

A good practice to follow, that allows everyone to access the tables without qualifying names, is to create a group whose only purpose is to own the tables. Do not grant any permissions to this group, but make all users members of the group. You can then create permission groups and grant users membership in these permission groups as warranted.

☞ For an example, see the section "Database object names and prefixes" on page 450.

Groups without passwords

Users connected to a group's user ID have certain permissions. This user ID can grant and revoke membership in the group. Also, this user would have ownership permissions over any tables in the database created in the name of the group's user ID.

It is possible to set up a database so that only the DBA handles groups and their database objects, rather than permitting other user IDs to make changes to group membership. You can do this by disallowing connection as the group's user ID when creating the group. To do this, type the GRANT CONNECT statement without a password. Thus:

```
GRANT CONNECT
TO personnel
```

creates a user ID personnel. This user ID can be granted group permissions, and other user IDs can be granted membership in the group, inheriting any permissions that have been given to personnel. However, nobody can connect to the database using the personnel user ID because it has no valid password.

The user ID personnel can be an owner of database objects, even though no user can connect to the database using this user ID. The CREATE TABLE statement, CREATE PROCEDURE statement, and CREATE VIEW statement all allow the owner of the object to be specified as a user other than that executing the statement. Only the DBA can carry out this assignment of ownership.

Special groups

When you create a database, the SYS, PUBLIC, and dbo groups are also automatically created. None of these groups has passwords, so it is not possible to connect to the database as SYS, PUBLIC, or dbo. However, these groups serve important functions in the database.

The SYS group

The SYS group owns the system tables and views for the database, which contain the full description of database structure, including all database objects and all user IDs.

☞ For more information about the system tables and views, together with a description of access to the tables, see the chapters "System Tables" [*ASA*

SQL Reference, page 671], and “System Views” [*ASA SQL Reference*, page 761].

The PUBLIC group

The PUBLIC group has SELECT permission on the system tables. As well, the PUBLIC group is a member of the SYS group, and has read access for some of the system tables and views, so any user of the database can find out information about the database schema. If you wish to restrict this access, you can REVOKE PUBLIC’s membership in the SYS group.

Any new user ID is automatically a member of the PUBLIC group and inherits any permissions specifically granted to that group by the DBA. You can also REVOKE membership in PUBLIC for users if you wish.

The dbo group

The dbo group owns many system stored procedures and views. The dbo group is a member of the SYS group. The PUBLIC group is a member of the dbo group. The dbo group also owns tables used for UltraLite and MobiLink.

Deleting groups from the database

You can delete a group from the database using both Sybase Central and Interactive SQL.

Deleting users or groups from the database is different from *removing* them from other groups. Deleting a group from the database does *not* delete its members from the database, although they lose membership in the deleted group.

Only the DBA can delete a group.

❖ To delete a group from the database (Sybase Central)

1. Open the Users & Groups folder.
2. Right-click the desired group and choose Delete from the popup menu.

❖ To delete a group from the database (SQL)

1. Connect to a database.
2. Execute a REVOKE CONNECT FROM statement.

Example

Remove the group personnel from the database.

```
REVOKE CONNECT FROM personnel
```

See also

- “REVOKE statement” [*ASA SQL Reference*, page 585]
- “Revoking user permissions” on page 441
- “Deleting users from the database” on page 442

Database object names and prefixes

The name of every database object is an identifier.

☞ For information about the rules for valid identifiers, see "Identifiers" [*ASA SQL Reference*, page 7].

In queries and sample SQL statements throughout this guide, database objects from the sample database are generally referred to using their simple name. For example:

```
SELECT *
FROM employee
```

Tables, procedures, and views all have an owner. The owner of the tables in the sample database is the user ID DBA. In some circumstances, you must prefix the object name with the owner user ID, as in the following statement.

```
SELECT *
FROM DBA.employee
```

The employee table reference is said to be **qualified**. In other circumstances it is sufficient to give the object name. This section describes when you need to use the owner prefix to identify tables, views and procedures, and when you do not.

When referring to a database object, you require a prefix unless:

- ♦ You are the owner of the database object.
- ♦ The database object is owned by a group ID of which you are a member.

Example

Consider the following example of a corporate database. The user ID **company** created all the tables, and since this user ID belongs to the database administrator, it therefore has DBA authority.

```
GRANT CONNECT TO company
IDENTIFIED BY secret;
GRANT DBA TO company;
```

The company user ID created the tables in the database.

```
CONNECT USER company IDENTIFIED BY secret;
CREATE TABLE company.Customers ( ... );
CREATE TABLE company.Products ( ... );
CREATE TABLE company.Orders ( ... );
CREATE TABLE company.Invoices ( ... );
CREATE TABLE company.Employees ( ... );
CREATE TABLE company.Salaries ( ... );
```

Not everybody in the company should have access to all information. Consider two user IDs in the sales department, Joe and Sally, who should have access to the Customers, Products, and Orders tables. To do this, you create a Sales group.

```
GRANT CONNECT TO Sally IDENTIFIED BY xxxxx;
GRANT CONNECT TO Joe IDENTIFIED BY xxxxx;
GRANT CONNECT TO Sales IDENTIFIED BY xxxxx;
GRANT GROUP TO Sales;
GRANT ALL ON Customers TO Sales;
GRANT ALL ON Orders TO Sales;
GRANT SELECT ON Products TO Sales;
GRANT MEMBERSHIP IN GROUP Sales TO Sally;
GRANT MEMBERSHIP IN GROUP Sales TO Joe;
```

Now Joe and Sally have permission to use these tables, but they still have to qualify their table references because the table owner is company, and Sally and Joe are not members of the company group:

```
SELECT *
FROM company.Customers
```

To rectify the situation, make the Sales group a member of the company group.

```
GRANT GROUP TO company;
GRANT MEMBERSHIP IN GROUP company TO Sales;
```

Now Joe and Sally, being members of the Sales group, are indirectly members of the company group, and can reference their tables without qualifiers. The following command now works:

```
SELECT *
FROM Customers
```

Note

Joe and Sally do not have any extra permissions because of their membership in the company group. The company group has not been explicitly granted any table permissions. (The company user ID has implicit permission to look at tables like Salaries because it created the tables and has DBA authority.) Thus, Joe and Sally still get an error executing either of these commands:

```
SELECT *
FROM Salaries;
SELECT *
FROM company.Salaries
```

In either case, Joe and Sally do not have permission to look at the Salaries table.

Using views and procedures for extra security

For databases that require a high level of security, defining permissions directly on tables has limitations. Any permission granted to a user on a table applies to the whole table. There are many cases when users' permissions need to be shaped more precisely than on a table-by-table basis. For example:

- ♦ It is not desirable to give access to personal or sensitive information stored in an employee table to users who need access to other parts of the table.
- ♦ You may wish to give sales representatives update permissions on a table containing descriptions of their sales calls, but limit such permissions to their own calls.

In these cases, you can use views and stored procedures to tailor permissions to suit the needs of your organization. This section describes some of the uses of views and procedures for permission management.

☞ For more information about how to create views, see "Working with views" [*ASA SQL User's Guide*, page 54].

☞ For more information about view permissions, see "Granting permissions on views" on page 437.

Using views for tailored security

Views are computed tables that contain a selection of rows and columns from base tables. Views are useful for security when it is appropriate to give a user access to just one portion of a table. The portion can be defined in terms of rows or in terms of columns. For example, you may wish to disallow a group of users from seeing the salary column of an employee table, or you may wish to limit a user to see only the rows of a table they have created.

Example

The Sales manager needs access to information in the database concerning employees in the department. However, there is no reason for the manager to have access to information about employees in other departments.

This example describes how to create a user ID for the sales manager, create views that provide the information she needs, and grant the appropriate permissions to the sales manager user ID.

1. Create the new user ID using the GRANT statement. While logged in as a user ID with DBA authority, enter the following:

```
CONNECT DBA
IDENTIFIED by SQL ;

GRANT CONNECT
TO SalesManager
IDENTIFIED BY sales
```

2. Define a view which only looks at sales employees as follows:

   ```
   CREATE VIEW emp_sales AS
     SELECT emp_id, emp_fname, emp_lname
     FROM DBA.employee
     WHERE dept_id = 200
   ```

   The table should therefore be identified as DBA.employee, with the owner of the table explicitly identified, for the SalesManager user ID to be able to use the view. Otherwise, when SalesManager uses the view, the SELECT statement refers to a table that user ID does not recognize.

3. Give SalesManager permission to look at the view:

   ```
   GRANT SELECT
   ON emp_sales
   TO SalesManager
   ```

   You use exactly the same command to grant permission on views and on tables.

Example 2

The next example creates a view which allows the Sales Manager to look at a summary of sales orders. This view requires information from more than one table for its definition:

1. Create the view.

   ```
   CREATE VIEW order_summary AS
     SELECT order_date, region, sales_rep, company_name
     FROM DBA.sales_order
       KEY JOIN DBA.customer
   ```

2. Grant permission for the Sales Manager to examine this view.

   ```
   GRANT SELECT
   ON order_summary
   TO SalesManager
   ```

3. To check that the process has worked properly, connect to the SalesManager user ID and look at the views you created:

   ```
   CONNECT SalesManager
   IDENTIFIED BY sales;
   SELECT *
   FROM DBA.emp_sales;
   SELECT *
   FROM DBA.order_summary;
   ```

No permissions have been granted to the Sales Manager to look at the underlying tables. The following commands produce permission errors.

```
SELECT * FROM DBA.employee;
SELECT * FROM DBA.sales_order
```

Other permissions on views

The previous example shows how to use views to tailor SELECT permissions. You can grant INSERT, DELETE, and UPDATE permissions on views in the same way.

☞ For more information about allowing data modification on views, see "Using views" [*ASA SQL User's Guide*, page 56].

Using procedures for tailored security

While views restrict access on the basis of data, procedures restrict the actions a user may take. As described in "Granting permissions on procedures" on page 438, a user may have EXECUTE permission on a procedure without having any permissions on the table or tables on which the procedure acts.

Strict security

For strict security, you can disallow all access to the underlying tables, and grant permissions to users or groups of users to execute certain stored procedures. This approach strictly defines the manner in which data in the database can be modified.

Changing ownership on nested objects

Views and procedures can access underlying objects that are owned by different users. For example, if usera, userb, userc, and userd were four different users, userd.viewd could be based on userc.viewc, which could be based on userb.viewb, which could be based on usera.tablea. Similarly for procedures, userd.procd could call userc.procc, which could call userb.procb, which could insert into usera.tablea.

The following Discretionary Access Control (DAC) rules apply to nested views and tables:

- ♦ To create a view, the user must have SELECT permission on all of the base objects (for example tables and views) in the view.
- ♦ To access a view, the view owner must have been granted the appropriate permission on the underlying tables or views with the GRANT OPTION and the user must have been granted the appropriate permission on the view.
- ♦ Updating with a WHERE clause requires both SELECT and UPDATE permission.
- ♦ If a user owns the tables in a view definition, the user can access the tables through a view, even if the user is not the owner of the view and has not been granted access on the view.

The following DAC rules apply to nested procedures:

- ♦ A user does not require any permissions on the underlying objects (for example tables, views or procedures) to create a procedure.
- ♦ For a procedure to execute, the owner of the procedure needs the appropriate permissions on the objects that the procedure references.
- ♦ Even if a user owns all the tables referenced by a procedure, the user will not be able to execute the procedure to access the tables unless the user has been granted EXECUTE permission on the procedure.

Following are some examples that describe this behavior.

Example 1: User1 creates table1, and user2 creates view2 on table1

- ♦ User1 can always access table1, since user1 is the owner.
- ♦ User1 can always access table1 through view2, since user1 is the owner of the underlying table. This is true even if user2 does not grant permission on view2 to user1.

- User2 can access table1 directly or through view2 if user1 grants permission on table1 to user2.
- User3 can access table1 if user1 grants permission on table1 to user3
- User3 can access table1 through view2 if user1 grants permission on table1 to user2 with grant option *and* user2 grants permission on view2 to user3.

Example 2: User2 creates procedure2 that accesses table1

- User1 can access table1 through procedure2 if user2 grants EXECUTE permission on procedure2 to user1. Note that this is different from the case of view2, where user1 did not need permission on view2.

Example 3: User1 creates table1, user2 creates table2, and user3 creates view3 joining table1 and table2

- User3 can access table1 and table2 through view3 if user1 grants permission on table1 to user3 *and* user2 grants permission on table2 to user3.
- If user3 has permission on table1 but not on table2, then user3 cannot use view3, even to access the subset of columns belonging to table1.
- User1 or user2 can use view3 if (a) user1 grants permission with grant option on table1 to user3, (b) user2 grants permission with grant option on table2 to user3, *and* ©) user3 grants permission on view3 to that user.

How user permissions are assessed

Groups do introduce complexities in the permissions of individual users. Suppose user M_Haneef has SELECT and UPDATE permissions on a specific table individually, but is also a member of two groups. Suppose one of these groups has no access to the table at all, and one has only SELECT access. What are the permissions in effect for this user?

Adaptive Server Anywhere decides whether a user ID has permission to carry out a specific action in the following manner:

1. If the user ID has DBA authority, the user ID can carry out any action in the database.

2. Otherwise, permission depends on the permissions assigned to the individual user. If the user ID has been granted permission to carry out the action, then the action proceeds.

3. If no individual settings have been made for that user, permission depends on the permissions of each of the groups to which the member belongs. If any of these groups has permission to carry out the action, the user ID has permission by virtue of membership in that group, and the action proceeds.

This approach minimizes problems associated with the order in which permissions are set.

Managing the resources connections use

Building a set of users and groups allows you to manage permissions on a database. Another aspect of database security and management is to limit the resources an individual user can use.

For example, you may wish to prevent a single connection from taking too much of the available memory or CPU resources, so you can avoid a having a connection slow down other users of the database.

Adaptive Server Anywhere provides a set of database options that the DBA can use to control resources. These options are called **resource governors**.

Setting options

You can set database options using the SET OPTION statement, with the following syntax:

SET [**TEMPORARY**] **OPTION** . . . [*userid*. | **PUBLIC**.]*option-name* = [*option-value*]

☞ For more information about options, see "Database Options" on page 613.

☞ For information on the SET OPTION statement, see "SET OPTION statement" [*ASA SQL Reference*, page 613].

Resources that can be managed

You can use the following options to manage resources:

♦ **JAVA_HEAP_SIZE** Sets the maximum size (in bytes) of the memory allocated to Java applications on a per connection basis.

♦ **MAX_CURSOR_COUNT** Limits the number of cursors for a connection.

♦ **MAX_STATEMENT_COUNT** Limits the number of prepared statements for a connection.

♦ **BACKGROUND_PRIORITY** Limits the impact requests on the current connection have on the performance of other connections.

Database option settings are not inherited through the group structure.

Users and permissions in the system tables

The database system tables and system views store information about the current users of a database and about their permissions.

☞ For more information about each of these tables, see "System Tables" [*ASA SQL Reference*, page 671].

The special user ID SYS owns the system tables. You cannot connect using the SYS user ID.

The DBA has SELECT access to all system tables, just as to any other tables in the database. The access of other users to some of the tables is limited. For example, only the DBA has access to the SYS.SYSUSERPERM table, which contains all information about the permissions of users of the database, as well as the encrypted passwords of each user ID. However, SYS.SYSUSERPERMS is a view containing all information in SYS.SYSUSERPERM except for the password, and by default all users have SELECT access to this view. You can fully modify all permissions and group memberships set up in a new database for SYS, PUBLIC, DBA, and dbo.

The following table summarizes the system tables containing information about user IDs, groups, and permissions. The user ID SYS owns all of the listed tables and views, and so their qualified names are SYS.SYSUSERPERM and so on.

Appropriate SELECT queries on these tables generates all the user ID and permission information stored in the database.

Table	**Default**	**Contents**
SYSUSERPERM	DBA only	Database-level permissions and password for each user ID
SYSGROUP	PUBLIC	One row for each member of each group
SYS-TABLEPERM	PUBLIC	All permissions on table given by the GRANT commands
SYSCOLPERM	PUBLIC	All columns with SELECT or UPDATE permission given by the GRANT command
DUMMY	PUBLIC	Dummy table that can be used to find the current user ID
SYSPROCPERM	PUBLIC	Each row holds one user granted permission to use one procedure

The following table summarizes the system views containing information about user IDs, groups, and permissions.

Views	**Default**	**Contents**
SYSUSERAUTH	DBA only	All information in SYSUSERPERM except for user numbers
SYSUSERPERMS	PUBLIC	All information in SYSUSERPERM except for passwords
SYSUSERLIST	PUBLIC	All information in SYSUSERAUTH except for passwords
SYSGROUPS	PUBLIC	Information from SYSGROUP in a more readable format
SYSTABAUTH	PUBLIC	Information from SYSTABLEPERM in a more readable format
SYSCOLAUTH	PUBLIC	Information from SYSCOLPERM in a more readable format
SYSPROCAUTH	PUBLIC	Information from SYSPROCPERM in a more readable format

In addition to these, there are tables and views that contain information about each object in the database.

CHAPTER 14

Replicating Data with Replication Server

About this chapter

This chapter describes how you can use Replication Server to replicate data between an Adaptive Server Anywhere database and other databases. Other databases in the replication system may be Adaptive Server Anywhere databases or other kinds of database.

Before you begin

Replication Server administrators who are setting up Adaptive Server Anywhere to take part in their Replication Server installation will find this chapter especially useful. You should have knowledge of Replication Server documentation, and familiarity with the Replication Server product. This chapter does not describe Replication Server itself.

☞ For information about Replication Server, including design, commands, and administration, see your Replication Server documentation.

Contents

Introduction to replication

Data replication is the sharing of data among physically distinct databases. Changes made to shared data at any one database are copied precisely to the other databases in the replication system.

Data replication brings some key benefits to database users.

Data availability

Replication makes data available locally, rather than through potentially expensive, less reliable, and slower connections to a single, central database. Since data is accessible locally, you are always have access to data, even in the event of a long-distance network connection failure.

Response time

Replication improves response times for data requests for two reasons. First, retrieval rates are faster since requests are processed on a local server without accessing some wide area network. Second, competition for processor time decreases since local processing offloads work from a central database server.

Sybase replication technologies

Sybase provides the following replication technologies for Adaptive Server Anywhere:

- **MobiLink** MobiLink is designed for two-way replication involving a consolidated database and large numbers of remote databases, typically including many mobile databases.
- **SQL Remote** Like MobiLink, SQL Remote is designed for two-way replication involving a consolidated database and large numbers of remote databases, typically including many mobile databases. Administration and resource requirements at the remote sites are minimal.
- **Replication Server** Replication Server is designed for replication among relatively small numbers of data servers, with a typical time lag between primary data and replicate data of a few seconds, and generally with an administrator at each site.

Each replication technology has its own documentation. This chapter describes how to use Adaptive Server Anywhere with Replication Server.

☞ For information about SQL Remote, see the book *SQL Remote User's Guide*.

☞ For information about MobiLink, see the book *MobiLink Synchronization User's Guide*.

Replicate sites and primary sites

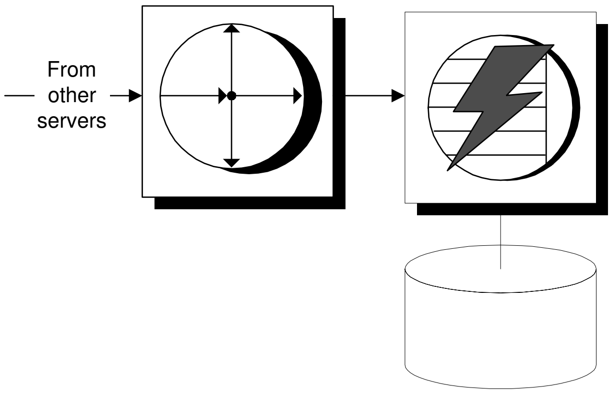
In a Replication Server installation, the data to be shared among databases is arranged in **replication subscriptions**.

For each replication definition, there is a **primary site**, where changes to the data in the replication occur. The sites that receive the data in the replication are called **replicate sites**.

Replicate site components

You can use Adaptive Server Anywhere as a replicate site with no additional components.

The following diagram illustrates the components required for Adaptive Server Anywhere to participate in a Replication Server installation as a replicate site.



- ♦ Replication Server receives data changes from primary site servers.
- ♦ Replication Server connects to Adaptive Server Anywhere to apply the changes.
- ♦ Adaptive Server Anywhere makes the changes to the database.

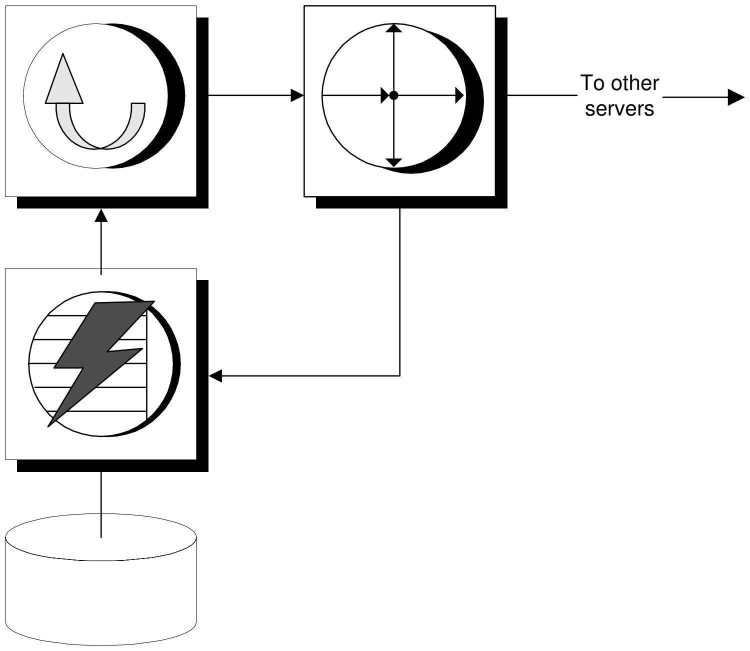
Asynchronous procedure calls

The Replication Server can use **asynchronous procedure calls** (APC) at replicate sites to alter data at a primary site database. If you are using APCs, the above diagram does not apply. Instead, the requirements are the same as for a primary site.

Primary site components

To use an Adaptive Server Anywhere database as a primary site, you need to use the Log Transfer Manager (LTM), or Replication Agent. The LTM supports Replication Server version 10.0 and greater.

The following diagram illustrates the components required for Adaptive Server Anywhere to participate in a Replication Server installation as a primary site. The arrows in the diagram represent data flow.



- The Adaptive Server Anywhere database server manages the database.
- The Adaptive Server Anywhere Log Transfer Manager connects to the database. It scans the transaction log to pick up changes to the data, and sends them to Replication Server.
- Replication Server sends the changes to replicate site databases.

Tutorial: Replicate data using Replication Server

This section provides a step-by-step tutorial describing how to replicate data from a primary database to a replicate database. Both databases in the tutorial are Adaptive Server Anywhere databases.

Replication Server assumed

This section assumes you have a running Replication Server.

☞ For more information about how to install or configure Replication Server, see the Replication Server documentation.

What is in the tutorial

This tutorial describes how to replicate only tables.

☞ For information about replicating procedures, see "Preparing procedures and functions for replication" on page 482.

The tutorial uses a simple example of a (very) primitive office news system: a single table with an ID column holding an integer, a column holding the user ID of the author of the news item, and a column holding the text of the news item. The id column and the author column make up the primary key.

Before you work through the tutorial, create a directory (for example, *c:\tutorial*) to hold the files you create in the tutorial.

Lesson 1: Set up the Adaptive Server Anywhere databases

This section describes how to create and set up the Adaptive Server Anywhere databases for replication.

You can create a database using Sybase Central or the dbinit utility. For this tutorial, we use the dbinit utility.

❖ Create the primary site database

1. Enter the following command from the tutorial directory you created (for example, *c:\tutorial*).

   ```
   dbinit primedb
   ```

   This creates a database file *primedb.db* in the current directory.

❖ Create the replicate site database

1. Enter the following command from the tutorial directory you created (for example *c:\tutorial*).

   ```
   dbinit repdb
   ```

   This creates a database file *repdb.db* in the current directory.

What's next? Next, you have to start database servers running on these databases.

Lesson 2: Start the database servers

You need to run the primary site database server, with the primary database loaded.

❖ Start the primary site database server

1. Change to the tutorial directory.
2. Enter the following command to start a network database server running the **primedb** database. You should be using the TCP/IP network communication protocol on the default communications port (2638):

```
dbsrv9 -x tcpip primedb.db
```

❖ Start the replicate site database server

1. Change to the tutorial directory.
2. Enter the following command to start a network database server running the **repdb** database, but on a different port:

```
dbsrv9 -x tcpip(port=2639) -n REPSV repdb.db
```

What's next? Next, you have to make entries for each of the Adaptive Server Anywhere servers in an interfaces file, so Replication Server can communicate with these database servers.

Lesson 3: Set up the Open Servers in your system

You need to add a set of Open Servers to the list of Open Servers in your system.

Adding Open Servers Open Servers are defined in your interfaces file (*SQL.ini*) using the *DSEdit* utility. For NetWare and UNIX users, the interfaces file is named *interfaces*, and the utility is named *sybinit*.

☞ For full instructions on how to add definitions to your interfaces file, see "Configuring Open Servers" on page 106.

Required Open Servers For each Open Server definition, you must provide a **name** and an **address**. Do not alter the other attributes of the definition. You need to add an Open Server entry for each of the following:

♦ **The primary database** Create an entry named PRIMEDB with address as follows:
  - **Protocol** NLWNSCK

- **Network address** localhost,2638

♦ **The replicate database** Create an entry named REPDB with address as follows:

- **Protocol** NLWNSCK
- **Network address** localhost,2639

♦ **The LTM at the primary database** This is necessary so you can shut down the LTM properly. Create an entry named PRIMELTM with address as follows:

- **Protocol** NLWNSCK
- **Network address** localhost,2640

♦ **Your Replication Server** This tutorial assumes you already have the Replication Server Open Server defined.

What's next? Next, confirm that the Open Servers are configured properly.

Lesson 4: Confirm that the Open Servers are configured properly

You can confirm that each Open Server is available by selecting ServerObject ➤ Ping Server from the DSEdit utility.

Alternatively, you can confirm that each Open Server is configured properly by connecting to the database using an Open Client application such as the *isql* utility.

To start *isql* running on the primary site database, type

```
isql -U DBA -P SQL -S PRIMEDB
```

The Open Client *isql* utility is not the same as the Adaptive Server Anywhere Interactive SQL utility.

Lesson 5: Add Replication Server information to the primary database

You need to add Replication Server tables and procedures to the primary site database for the database to participate in a Replication Server installation. You also need to create two user IDs for use by Replication Server. The SQL command file *rssetup.sql* comes with Adaptive Server Anywhere and carries out these tasks.

The *rssetup.sql* command file must be run on the Adaptive Server Anywhere server from the Interactive SQL utility.

❖ Run the rssetup script

1. From Interactive SQL, connect to the Adaptive Server Anywhere database as user ID **DBA** using password **SQL**.
2. Run the *rssetup* script using the following command:

   ```
   read "path\rssetup.sql"
   ```

   where *path* is your Adaptive Server Anywhere installation directory.

   You can alternatively use File ➤ Run Script, and browse to the file.

Actions carried out by rssetup.sql

The *rssetup.sql* command file carries out the following functions:

- Creates a user named **dbmaint**, with password **dbmaint** and with DBA permissions. This is the maintenance user name and password required by Replication Server to connect to the primary site database.
- Creates a user named **sa**, with password **sysadmin** and with DBA permissions. This is the user ID used by Replication Server when materializing data.
- Adds **sa** and **dbmaint** to a group named **rs_systabgroup**.

Passwords and user IDs

While the hard-wired user IDs (**dbmaint** and **sa**) and passwords are useful for test and tutorial purposes, you should change the password and perhaps also the user IDs when running databases that require security. Users with DBA permissions have full authority in an Adaptive Server Anywhere database.

The user ID **sa** and its password must match that of the system administrator account on the Replication Server. Adaptive Server Anywhere does not currently accept a NULL password.

Permissions

The *rssetup.sql* script carries out a number of operations, including some permissions management. The permissions changes made by *rssetup.sql* are outlined here. *You do not have to make these changes yourself.*

For replication, ensure that the **dbmaint** and **sa** users can access this table without explicitly specifying the owner. To do this, the table owner user ID must have group membership permissions, and the **dbmaint** and **sa** users must be members of the table owner group. To grant group permissions, you must have DBA authority.

For example, if user DBA owns the table, you should grant group permissions to the DBA user ID:

```
GRANT GROUP
TO DBA
```

You should then grant the **dbmaint** and **sa** users membership in the DBA group. To grant group membership, you must either have DBA authority or be the group ID.

```
GRANT MEMBERSHIP
IN GROUP "DBA"
TO dbmaint ;
GRANT MEMBERSHIP
IN GROUP "DBA"
TO sa ;
```

Lesson 6: Create the table for the primary database

In this section, you create a single table in the primary site database, using *isql*. First, make sure you are connected to the primary site database:

```
isql -U DBA -P SQL -S PRIMEDB
```

Next, create a table in the database:

```
CREATE TABLE news (
  ID int,
  AUTHOR char( 40 ) DEFAULT CURRENT USER,
  TEXT char( 255 ),
  PRIMARY KEY ( ID, AUTHOR )
)
go
```

Identifier case sensitivity

In Adaptive Server Anywhere, all identifiers are case insensitive. In Adaptive Server Enterprise, identifiers are case sensitive by default. Even in Adaptive Server Anywhere, ensure the case of your identifiers matches in all parts of the SQL statement to ensure compatibility with Adaptive Server Enterprise.

In Adaptive Server Anywhere, the database determines password case sensitivity unless you explicitly set the case sensitivity for passwords. You can have a case sensitive database with case insensitive passwords, and vice versa. Extended characters used in passwords are case sensitive, regardless of the password case sensitivity setting. User IDs, being identifiers, are case insensitive in all Adaptive Server Anywhere databases.

☞ For more information, see "CREATE DATABASE statement" [*ASA SQL Reference*, page 338].

For **news** to act as part of a replication primary site, you must set the REPLICATE flag to ON for the table using an ALTER TABLE statement:

```
ALTER TABLE news
REPLICATE ON
go
```

This is equivalent to running the **sp_setreplicate** or **sp_setreptable** procedure on the table in Adaptive Server Enterprise. You cannot set REPLICATE ON in a CREATE TABLE statement.

Lesson 7: Add Replication Server information to the replicate database

You should run the *rssetup.sql* command file on the replicate database in exactly the same manner as it ran on the primary database. Also ensure that the **dbmaint** and **sa** users can access this table without explicitly specifying the table owner.

☞ These tasks are the same as those carried out on the primary database. For a complete explanation, see "Lesson 5: Add Replication Server information to the primary database" on page 469.

Lesson 8: Create the tables for the replicate database

The replicate site database needs to have tables to hold the data it receives. Now is a good time to create these tables. As long as the database elements are in place, no extra statements are necessary for them to act as a replicate site in a Replication Server installation. In particular, you do not need to set the REPLICATE flag to ON, which is necessary only at the primary site.

Replication Server allows replication between tables and columns with different names. As a simple example, however, create a table in the replicate database identical in definition to that in the primary database (except for the REPLICATE flag, which is not set to ON in the replicate database). The table creation statement for this is:

```
CREATE TABLE news (
   ID int,
   AUTHOR char( 40 ) DEFAULT CURRENT USER,
   TEXT char( 255 ),
   PRIMARY KEY ( ID, AUTHOR )
)
go
```

For the tutorial, the CREATE TABLE statement must be *exactly* the same as that at the primary site.

You must ensure that the users **dbmaint** and **sa** can access this table without specifying the owner name. Also, these user IDs must have SELECT and UPDATE permissions on the table.

Lesson 9: Set up Replication Server

You need to carry out the following tasks on the Replication Server:

- Create a connection for the primary site data server.
- Create a connection for the replicate site data server.
- Create a replication definition.
- Create a subscription to the replication.

This section describes each of these tasks. It also describes starting the Adaptive Server Anywhere LTM.

Create a connection for the primary site

Using *isql*, connect to Replication Server and create a connection to the primary site Adaptive Server Anywhere database.

The following command creates a connection to the **primedb** database on the **PRIMEDB** Open Server.

```
create connection to PRIMEDB.primedb
set error class rs_sqlserver_error_class
set function string class rs_sqlserver_function_class
set username dbmaint
set password dbmaint
with log transfer on
go
```

If you have changed the **dbmaint** user ID and password in the *rssetup.sql* command file, make sure you replace the **dbmaint** username and password in this command.

Replication Server does not actually use the **primedb** database name; instead, the database name is read from the command line of the **PRIMEDB** Open Server. You must, however, include a database name in the CREATE CONNECTION statement to conform to the syntax.

☞ For a full description of the create connection statement, see the chapter "Replication Server Commands" in the *Replication Server Reference Manual*.

Create a connection for the replicate site.

Using *isql*, connect to Replication Server and create a connection to the replicate site Adaptive Server Anywhere database.

The following command creates a connection to the **repdb** database on the **REPDB** Open Server.

```
create connection to REPDB.repdb
set error class rs_sqlserver_error_class
set function string class rs_sqlserver_function_class
set username dbmaint
set password dbmaint
go
```

This statement differs from the primary site server statement in that there is no **with log transfer on** clause in this statement.

If you have changed the **dbmaint** user ID and password in the *rssetup.sql* command file, make sure you replace the **dbmaint** username and password in this command.

Create a replication definition

Using *isql*, connect to Replication Server and create a replication definition. The following statement creates a replication definition for the news table on the **primedb** database:

```
create replication definition news
with primary at PRIMEDB.primedb
( id int, author char(40), text char(255) )
primary key ( id, author )
go
```

☞ For a full description of the CREATE REPLICATION DEFINITION statement, see your *Replication Server Reference Manual.*

If you set the **qualify_table_owner** option to on in the LTM configuration file, you must specify the table owner in the statement, for all replicating tables.

Configure and start the Adaptive Server Anywhere LTM

For replication to take place, the Adaptive Server Anywhere LTM must be running against the primary site server. Before you start the Adaptive Server Anywhere LTM, make sure it is properly configured by editing an LTM configuration file.

Below is a sample configuration file for the **primedb** database. If you are following the examples, you should make a copy of this file as *primeltm.cfg*:

```
#
# Configuration file for 'PRIMELTM'
#
SQL_server=PRIMEDB
SQL_database=primedb
SQL_user=sa
SQL_pw=sysadmin
RS_source_ds=PRIMEDB
RS_source_db=primedb
RS=your_rep_server_name_here
RS_user=sa
RS_pw=sysadmin
LTM_admin_user=DBA
LTM_admin_pw=SQL
LTM_charset=cp850
scan_retry=2
APC_user=sa
APC_pw=sysadmin
SQL_log_files=C:\TUTORIAL
```

If you have changed the user ID and password in the *rssetup.sql* command file from **sa** and **sysadmin**, you should use the new user ID and password in this configuration.

To start the Adaptive Server Anywhere LTM running on the primary site server, enter the following command:

```
dbltm -S PRIMELTM -C primeltm.cfg
```

The connection information is in *primeltm.cfg*. In this command, PRIMELTM is the server name of the LTM.

You can find usage information about the Adaptive Server Anywhere LTM by typing the following statement:

```
dbltm -?
```

You can run the Adaptive Server Anywhere LTM as a Windows service.

☞ For information on running programs as services, see "Running the server outside the current session" on page 21.

Create a subscription for your replication

Using *isql*, connect to Replication Server and create a subscription for the replication.

The following statement creates a subscription for the news replication defined in "Create a replication definition" on page 474 with replicate site as the **repdb** database.

```
create subscription NEWS_SUBSCRIPTION
for news
with replicate at REPDB.repdb
go
```

You have now completed your installation. Try replicating data to confirm that the setup is working properly.

Lesson 10: Enter data at the primary site for replication

You can now replicate data from the primary database to the replicate database. As an example, connect to the primary database using the *isql* utility, and enter a row in the **news** table.

```
insert news (id, text)
values (1, 'Test news item.' )
commit
go
```

The Adaptive Server Anywhere LTM sends only committed changes to the Replication Server. The data change is replicated next time the LTM polls the transaction log.

Tutorial complete

You have now completed the tutorial.

Configuring databases for Replication Server

Each Adaptive Server Anywhere database that participates in a Replication Server installation needs to be configured before it can do so. Configuring the database involves the following tasks:

- ♦ Selecting a secure user ID for the maintenance user and the name used by Replication Server when materializing data.
- ♦ Setting up the database for Replication Server.
- ♦ Configuring the language and character set, where necessary.

Configuring the LTM

Each primary site Adaptive Server Anywhere database requires an LTM to send data to Replication Server. Each primary or replicate site Adaptive Server Anywhere database requires an Open Server definition so that Replication Server can connect to the database.

☞ For information on configuring the LTM, see "Configuring the LTM" on page 483.

Setting up the database for Replication Server

Once you have created your Adaptive Server Anywhere database and created the necessary tables and so on within the database, you can make database ready for use with Replication Server. You do this using a setup script supplied with the Adaptive Server Anywhere Replication Agent product. The script is named *rssetup.sql*.

When you need to run the setup script

You need to run the setup script at any Adaptive Server Anywhere database that is taking part in a Replication Server installation, whether as a primary or a replicate site.

What the setup script does

The setup script creates user IDs required by Replication Server when connecting to the database. It also creates a set of stored procedures and tables used by Replication Server. The tables begin with the characters **rs_**, and the procedures begin with the characters **sp_**. Procedures include some that are important for character set and language configuration.

Prepare to run the setup script

Replication Server uses a special data server **maintenance user** login name for each local database containing replicated tables. This allows Replication Server to maintain and update the replicated tables in the database.

The maintenance user

The setup script creates a maintenance user with name **dbmaint** and password **dbmaint**. The maintenance user has DBA permissions in the

Adaptive Server Anywhere database, which allows it full control over the database. For security reasons, you should change the maintenance user ID and password.

❖ To change the maintenance user ID and password

1. Open the *rssetup.sql* setup script in a text editor. The script is held in the *scripts* subdirectory of your Adaptive Server Anywhere installation directory.

2. Change all occurrences of the dbmaint user ID to the new maintenance user ID of your choice.

3. Change the **dbmaint** password to the new maintenance user password of your choice. The password occurs in the following place at the top of the setup script file:

   ```
   GRANT CONNECT TO dbmaint
   IDENTIFIED BY dbmaint
   ```

The materialization user ID

When Replication Server connects to a database to materialize the initial copy of the data in the replication, it does so using the Replication Server system administrator account.

The Adaptive Server Anywhere database must have a user ID and password that match the Replication Server system administrator user ID and password. Adaptive Server Anywhere does not accept a NULL password.

The setup script assumes a user ID of sa and a password of **sysadmin** for the Replication Server administrator. You should change this to match the actual name and password.

❖ To change the system administrator user ID and password

1. Open the *rssetup.sql* setup script in a text editor.

2. Change all occurrences of the **sa** user ID to match the Replication Server system administrator user ID.

3. Change the **sa** password to match the Replication Server system administrator password. The password has the initial setting of **sysadmin**.

Run the setup script

Once you have modified the setup script to match the user IDs and passwords appropriately, you can run the setup script to create the maintenance and system administrator users in the Adaptive Server Anywhere database.

❖ To run the setup script

1. Start the Adaptive Server Anywhere database on an Adaptive Server Anywhere database server.

2. Start the Interactive SQL utility, and connect to the database as a user with DBA authority. When you create an Adaptive Server Anywhere database, it has a user with user ID **DBA** and password **SQL**, which has DBA authority.

3. Run the script by entering the following command in the SQL Statements pane:

   ```
   read path\rssetup.sql
   ```

   where *path* is the path to the setup script.

Character set and language issues

Upon creation, each Adaptive Server Anywhere database is assigned a specific collation (character set and sort order). Replication Server uses a different set of identifiers for character sets and sort orders.

Set the character set and language parameters in the LTM configuration file. If you are unsure of the characters set label to specify, you can do the following to determine the character set of the server:

❖ To determine the character set

1. Execute the following command:

   ```
   exec sp_serverinfo csname
   ```

The language is one of the language labels listed in "Language label values" on page 332.

Using the LTM

Since the Adaptive Server Anywhere LTM relies on information in the Adaptive Server Anywhere transaction log, take care not to delete or damage the log without storing backups (for example, using a transaction log mirror).

☞ For more information about transaction log management, see the section "Transaction log and backup management" on page 487.

You cannot substitute an Adaptive Server Anywhere LTM for an Adaptive Server Enterprise LTM since the transaction logs have different formats.

The Adaptive Server Anywhere LTM supports replication of inserts, updates, and deletes, as well as replication of Transact-SQL dialect stored procedure calls.

The Adaptive Server Enterprise LTM sends data changes to the Replication Server before they are committed. The Replication Server holds the changes until a COMMIT statement arrives. By contrast, the Adaptive Server Anywhere LTM sends only committed changes to Replication Server. For long transactions, this may lead to some added delay in replication, since all changes have to go through the Replication Server before distribution.

Configuring tables for replication

Adaptive Server Anywhere does not support the **sp_setreplicate** system procedure. Instead, a table is identified as a primary data source using the ALTER TABLE statement with a single clause:

```
ALTER TABLE table-name
SET REPLICATE ON
```

The effects of setting REPLICATE ON for a table

Setting REPLICATE ON places extra information into the transaction log. Whenever an UPDATE, INSERT, or DELETE action occurs on the table. The Adaptive Server Anywhere Replication Agent uses this extra information to submit the full pre-image of the row, where required, to Replication Server for replication.

Even if only some of the data in the table needs to be replicated, all changes to the table are submitted to Replication Server. It is Replication Server's responsibility to distinguish the data to be replicated from that which is not.

When you update, insert, or delete a row, the pre-image of the row is the contents of the row before the action, and the post-image is the contents of the row after the action. For INSERTS, only the post-image is submitted (the pre-image is empty). For DELETES, the post-image is empty and only the pre-image is submitted. For UPDATES, both the pre-image and the updated values are submitted.

The following data types are supported for replication:

Data type	Description (Open Client/Open Server type)
Exact integer data types	int, smallint, tinyint
Exact decimal data types	decimal, numeric
Approximate numeric data types	float (8-byte), real
Money data types	money, smallmoney
Character data types	char(n), varchar(n), text
Date and time data types	datetime, smalldatetime
Binary data types	binary(n), varbinary(n), image
Bit data types	bit

Notes

Adaptive Server Anywhere supports data of zero length that is not NULL. However, non-null long varchar and long binary data of zero length is replicated to a replicate site as NULL.

If a primary table has columns with unsupported data types, you can replicate the data if you create a replication definition using a compatible supported data type. For example, to replicate a DOUBLE column, you could define the column as FLOAT in the replication definition.

Side effects of setting REPLICATE ON for a table

There can be a replication performance hit for heavily updated tables. You could consider using replicated procedures if you experience performance problems that may be related to replication traffic, since replicated procedures send only the call to the procedure instead of each individual action.

Since setting REPLICATE ON sends extra information to the transaction log, this log grows faster than for a non-replicating database.

Minimal column replication definitions

The Adaptive Server Anywhere LTM supports the Replication Server replicate minimal columns feature. This feature is enabled at Replication Server.

☞ For more information on replicate minimal columns, see your Replication Server documentation.

Preparing procedures and functions for replication

You can use stored procedures to modify the data in tables. Updates, inserts, and deletes execute from within the procedure.

Replication Server can replicate procedures as long as they satisfy certain conditions. The first statement in a procedure must carry out an update for the procedure to be replicated.

☞ For a full description of how Replication Server replicates procedures, see your Replication Server documentation.

Adaptive Server Anywhere supports two dialects for stored procedures: the Watcom-SQL dialect, based on the draft ISO/ANSI standard, and the Transact-SQL dialect. You must use Transact-SQL to write stored procedures for replication.

Function APC format

The Adaptive Server Anywhere LTM supports the Replication Server **function APC** format. To make use of these functions, set the configuration parameter **rep_func** to **on** (the default is **off**).

The LTM interprets all replicated APCs as either table APCs or function APCs. A single Adaptive Server Anywhere database cannot combine function APCs with other table APCs.

☞ For more information about replicate functions, see your Replication Server documentation.

SQL Statements for controlling procedure replication

A procedure can be configured to act as a replication source using the ALTER PROCEDURE statement.

The following statement makes the procedure **MyProc** act as a replication source.

```
ALTER PROCEDURE MyProc
REPLICATE ON
```

The following statement prevents the procedure **MyProc** from acting as a replication source.

```
ALTER PROCEDURE MyProc
REPLICATE OFF
```

These statements have the same effect as running **sp_setreplicate** or **sp_setrepproc 'table'** on the procedure in Adaptive Server Enterprise. The

sp_setrepproc 'function' syntax is not supported.

The effects of setting REPLICATE ON for a procedure

When a procedure is used as a replication data source, calling the procedure sends extra information to the transaction log.

Asynchronous procedures

A procedure called at a replicate site database to update data at a primary site database is an **asynchronous procedure**. The procedure carries out no action at the replicate site, but rather, the call to the procedure is replicated to the primary site, where a procedure of the same name executes. This is called an **asynchronous procedure call** (APC). The changes made by the APC are then replicated from the primary to the replicate database in the usual manner.

☞ For information about APCs, see your Replication Server documentation.

The APC_user and APC support

Support for APCs in Adaptive Server Anywhere is different from that in Adaptive Server Enterprise. In Adaptive Server Enterprise, each APC executes using the user ID and password of the user who called the procedure at the replicate site. In Adaptive Server Anywhere, however, the transaction log does not store the password, and so it is not available at the primary site. To work around this difference, the LTM configuration file holds a single user ID with associated password, and this user ID (the **APC_user**) executes the procedure at the primary site. The APC_user must, therefore, have appropriate permissions at the primary site for each APC that may be called.

Configuring the LTM

You control LTM behavior by modifying the LTM **configuration file**, which is a plain text file created and edited using a text editor. The LTM configuration file contains information the LTM needs, such as the Adaptive Server Anywhere server it transfers a log from, the Replication Server it transfers the log to. You need a valid configuration file to run the LTM.

Creating a configuration file

You must create a configuration file, using a text editor, before you can run the LTM. The -C LTM command specifies the name of the configuration file to use, and has a default of *dbltm.cfg*.

Configuration file format

The LTM configuration file shares the same format as the Replication Server configuration file described in your *Replication Server Administration Guide*. In summary:

♦ The configuration file contains one entry per line.

- An entry consists of a parameter, followed by the = character, followed by the value:

```
Entry=value
```

- Lines beginning with a # character are comments ignored by the LTM.
- The configuration file cannot contain leading blanks.
- Entries are case sensitive.

☞ For the full list of available configuration file parameters, see "The LTM configuration file" on page 552.

Example configuration file

- The following is a sample Adaptive Server Anywhere LTM configuration file.

```
# This is a comment line
# Names are case sensitive.
SQL_user=sa
SQL_pw=sysadmin
SQL_server=PRIMESV
SQL_database=primedb
RS_source_ds=PRIMESV
RS_source_db=primedb
RS=MY_REPSERVER
RS_user=sa
RS_pw=sysadmin
LTM_admin_user=DBA
LTM_admin_pw=SQL
LTM_charset=cp850
scan_retry=2
SQL_log_files=e:\logs\backup
APC_user=sa
APC_pw=sysadmin
```

Replicating transactions in batches

Effects of buffering transactions

The LTM allows buffering of replication commands to Replication Server. Buffering the replication commands and sending them in batches results in fewer messages being sent, and can significantly increase overall throughput, especially on high volume installations.

How batch mode works

By default, the LTM buffers transactions. The buffer flushes (the transactions sent to Replication Server when the buffer:

- **Reaches maximum number of commands** The **batch_ltl_sz** parameter sets the maximum number of LTL (log transfer language) commands stored in the buffer before it flushes. The default setting is 200.

- **Reaches maximum memory used** The **batch_ltl_mem** parameter sets the maximum memory that the buffer can occupy before flushes. The default setting is 256K.
- **Completes transaction log processing** If there are no more entries in the transaction log to process (that is, the LTM is up to date with all committed transactions), then the buffer flushes.

Turning off buffering

You can turn off buffering of transactions by setting the **batch_ltl_cmds** parameter to **off**:

```
batch_ltl_cmds=off
```

Language and character set issues

Language and character set issues are an important consideration in many replication sites. Each database and server in the system uses a specific collation (character set and sorting order) for storing and ordering strings. Adaptive Server Anywhere character set support is carried out in a different manner to character set support in Adaptive Server Enterprise and other Open Client/Open Server based applications.

This section describes how to configure the Adaptive Server Anywhere LTM such that data in an Adaptive Server Anywhere database can be shared with Replication Server and hence with other databases.

The LTM automatically uses the default Open Client/Open Server language, sort order, and character set. You can override these defaults by adding entries to the LTM configuration file.

Open Client/Open Server collations

Adaptive Server Enterprise, Replication Server, and other Open Client/Open Server applications share a common means of managing character sets.

☞ For information on Open Client/Open Server character set support, see the chapter “Configuring Character Sets, Sort Orders, and Languages” in the Adaptive Server Enterprise *System Administration Guide*.

☞ For more information about character set issues in Replication Server, see the chapter “International Replication Design Considerations” in the *Replication Server Design Guide*.

This section provides a brief overview of Open Client/Open Server character set support.

Internationalization files

Files that support data processing in a particular language are called **internationalization files**. Several types of internationalization files come with Adaptive Server Enterprise and other Open Client/Open Server applications.

There is a directory named *charsets* under your Sybase directory. *Charsets* has a set of subdirectories, including one for each character set available to you. Each character set contains a set of files, as described in the following table

File	Description
Charset.-loc	Character set definition files that define the lexical properties of each character such as alphanumeric, punctuation, operand, upper or lower case.
*.*srt*	Defines the sort order for alphanumeric and special characters.
*.*xlt*	Terminal-specific character translation files for use with utilities.

Character set settings in the LTM configuration file

Three settings in the LTM configuration file refer to character set issues:

- **LTM_charset** The character set for the LTM to use. You can specify any Sybase-supported character set.
- **LTM_language** The **language** used by the LTM to print its messages to the error log and to its clients. You can choose any language to which the LTM has been localized, as long as it is compatible with the LTM character set.

  The Adaptive Server Anywhere LTM has been localized to several languages.

Notes

Character set In an Open Client/Open Server environment, an LTM should use the same character set as the data server and Replication Server attached to it.

Adaptive Server Anywhere character sets are specified differently than Open Client/Open Server character sets. Consequently, the requirement is that the Adaptive Server Anywhere character set be compatible with the LTM character set.

Language The *locales.dat* file in the *locales* subdirectory of the Sybase release directory contains valid map settings. However, the LTM output messages in the user interface are currently available in those languages to which the LTM has been localized.

Sort order All sort orders in your replication system should be the same.

You can find the default entry for your platform in the *locales.dat* file in the *locales* subdirectory of the Sybase release directory.

Example

♦ The following settings are valid for a Japanese installation:

```
LTM_charset=SJIS
LTM_language=Japanese
```

Transaction log and backup management

One of the differences between the Adaptive Server Enterprise LTM and the Adaptive Server Anywhere LTM is that while the Adaptive Server Enterprise LTM depends on a temporary recovery database for access to old transactions, the Adaptive Server Anywhere LTM depends on access to old transaction logs. No temporary recovery database exists for the Adaptive Server Anywhere LTM.

Replication depends on access to operations in the transaction log, and for Adaptive Server Anywhere primary site databases, sometimes access to old transaction logs. This section describes how to set up backup procedures at an Adaptive Server Anywhere primary site to ensure proper access to old transaction logs.

Consequences of lost transaction logs

Good backup practices at Adaptive Server Anywhere primary database sites are crucial. A lost transaction log could mean rematerializing replicate site databases. At primary database sites, a transaction log mirror is recommended.

☞ For information on transaction log mirrors and other backup procedure information, see "Backup and Data Recovery" on page 373.

The LTM configuration file contains a directory entry, which points to the directory where backed up transaction logs are kept. This section describes how to set up a backup procedure to ensure that such a directory stays in proper shape.

Backup utility options

With the Backup utility, you have the option of renaming the transaction log on backup and restart. For the dbbackup utility, this is the -r option. It is recommended that you use this option when backing up the consolidated database and remote database transaction logs.

For example, consider a database named *primedb.db*, in directory *c:\prime*, with a transaction log in directory *d:\primelog\primedb.log*. Backing up this transaction log to a directory *e:\primebak* using the rename and restart option carries out the following tasks:

1. Backs up the transaction log, creating a backup file *e:\primebak\primedb.log*.

2. Renames the existing transaction log to *d:\primelog\YYMMDDxx.log*, where **xx** are sequential characters ranging from **AA** to **ZZ**.

3. Starts a new transaction log, as *d:\primelog\primedb.log*.

After several backups, the directory *d:\primelog* will contain a set of sequential transaction logs. The log directory should not contain any transaction logs other than the sequence of logs generated by this backup procedure.

Using the DELETE_OLD_LOGS option

The DELETE_OLD_LOGS Adaptive Server Anywhere database option's default is OFF. If you set the default to ON, then the LTM automatically deletes the old transaction logs when Replication Server no longer needs access to the transactions. This option can help to manage disk space in replication setups.

You set the DELETE_OLD_LOGS option for the PUBLIC group:

```
SET OPTION PUBLIC.DELETE_OLD_LOGS = 'ON'
```

☞ For more information, see "DELETE_OLD_LOGS option [replication]" on page 652.

The Unload utility and replication

If a database participates in replication, care must be taken when unloading and reloading in order to avoid needing to re-materialize the database. Replication is based on the transaction log, and unloading and reloading a database deletes the old transaction log. For this reason, good backup practices are especially important when participating in replication.

Replicating an entire database

Adaptive Server Anywhere provides a shortcut for replicating an entire database, so you don't have to set each table in the database as a replicated table.

You can set a PUBLIC database option called REPLICATE_ALL using the SET OPTION statement. You can designate a whole database for replication using the following command:

```
SET OPTION PUBLIC.Replicate_all='ON'
```

You require DBA authority to change this and other PUBLIC option settings. You must restart the database for the new setting to take effect. The REPLICATE_ALL option has no effect on procedures.

☞ For more information, see "REPLICATE_ALL option [replication]" on page 687.

Stopping the LTM

You can shut down the LTM from the user interface in Windows NT/2000/XP, or in other circumstances by issuing a command.

❖ To stop the LTM in Windows NT/2000/XP, when the LTM is not running as a service

1. Click Shutdown on the user interface.

❖ To stop the LTM by issuing a command

1. Connect to the LTM from *isql* using the LTM_admin_user login name and password in the LTM configuration file. The user ID and password are case sensitive.
2. Stop the LTM using the SHUTDOWN statement.

Example

♦ The following statements connect *isql* to the LTM PRIMELTM, and shut it down:

```
isql -SPRIMELTM -UDBA -PSQL
1> shutdown
2> go
```

Part IV

Utilities, Options, and Properties

This section describes database administration utilities, options, properties, and limitations.

CHAPTER 15

Database Administration Utilities

About this chapter

Adaptive Server Anywhere includes a set of utility programs for backing up databases and performing other database administration tasks. This chapter provides reference information for each of the database administration utilities.

Contents

Administration utilities overview

This chapter presents reference information on the programs and database administration utilities that are part of Adaptive Server Anywhere. Each of the utilities can be accessed from one or more of Sybase Central, Interactive SQL, or at a command prompt.

☞ For comprehensive documentation on Sybase Central, see the Sybase Central online help. For an introduction to the Sybase Central database administration tool, see “ Managing Databases with Sybase Central” [*Introducing SQL Anywhere Studio*, page 241].

The administration utilities use a set of system environment variables. These variables are described in “Environment variables” on page 276.

Database file administration statements

A set of SQL statements are available that carry out some of the tasks that the administration utilities carry out. These statements are listed in “SQL Statements” [*ASA SQL Reference*, page 253].

Using configuration files

Many of the utilities provided with SQL Anywhere Studio allow you to store command line options in a configuration file. If you use an extensive set of options, you may find it useful to store them in a configuration file.

The **@***data* option allows you to specify environment variables and configuration files on the command line. To specify a configuration file, replace *data* with the path and name of the configuration file.

Configuration files can contain line breaks, and can contain any set of options, including the @data option. You can use the number sign (#) to designate lines as comments.

The @data parameter can occur at any point in the command line, and parameters contained in the file are inserted at that point. You can use @data multiple times on one command line to specify multiple configuration files.

Utilities read the command line by expanding the specified configuration files and reading the entire command line from left to right. If you specify options that are overridden by other options in the command line, the option closer to the end of the line wins. In some cases, conflicting options result in an error.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information about obfuscating the contents of a configuration file, see “Hiding the contents of files using the dbfhide command-line utility” on page 524.

Example

The following configuration file holds a set of options for the Validation utility (dbvalid):

```
#Connect to the sample database as user ASA with password SQL
-c "uid=DBA;pwd=SQL;dbf=c:\Program Files\Sybase\SQL Anywhere 9\
         asademo.db"
#Perform a full check on each table
-f
#Log output messages to the specified file
-o "c:\validationlog.txt
```

If this configuration file is saved as *c:\config.txt*, it can be used in a command as follows:

```
  dbvalid @c:\config.txt
```

The Adaptive Server Anywhere Console utility

The Adaptive Server Anywhere Console provides administration and monitoring facilities for database server connections.

The Adaptive Server Anywhere Console is available on all supported platforms except Windows CE, AIX, HP-UX, HP-UX Itanium, Linux Itanium, and Compaq Tru64. On these platforms, you can use the connection-level, server-level, and database-level properties to obtain information or you can monitor your server from a machine running an operating system that supports the Adaptive Server Anywhere Console (such as Windows 95/98/Me, Windows NT/2000/XP, Mac OS X, or Linux).

☞ For more information about obtaining property values, see "Database Performance and Connection Properties" on page 703.

Monitoring connections using the dbconsole utility

Syntax

dbconsole [*options*]

Option	**Description**
-c *"keyword=value; . . . "*	Database connection parameters.

Description

The Adaptive Server Anywhere Console allows you to monitor the server from a client machine. You can use it to track who is logged on to a database server elsewhere on your network. You can also display both server and client statistics on your local client screen, disconnect users, and configure the database server. The Adaptive Server Anywhere Console can display information for multiple connections.

❖ To disconnect a user from a database

1. Connect to the database from the Adaptive Server Anywhere Console.

2. In the User ID column, right-click the user and choose Disconnect.

   You can configure the columns that appear in the Adaptive Server Anywhere Console in the Options dialog, which can be accessed by choosing File ➤ Options.

This utility does *not* accept the ***@data*** parameter to read in options from a configuration file.

Console utility options

-c "keyword=value; . . . " Specify connection parameters.

☞ For a description of supported connection parameters, see "Connection parameters" on page 176.

The Backup utility

You can use the Backup utility to back up the database files and transaction logs for running databases.

You can access the Backup utility in the following ways:

- ♦ From Sybase Central, using the Create Backup Images wizard.
- ♦ At a command prompt, using the dbbackup command. This is useful for incorporating backup procedures into batch or command files.
- ♦ From Interactive SQL, using the BACKUP DATABASE statement.

The Backup utility makes a backup copy of all the files for a single database. A simple database consists of two files: the main database file and the transaction log. More complicated databases can store tables in multiple files, with each file as a separate dbspace. All backup filenames are the same as the database filenames.

Running the Backup utility on a running database is equivalent to copying the database files when the database is not running. Thus, you can back up the database while other applications or users are using it.

Exit codes are 0 (success) or non-zero (failure).

☞ For a description of suggested backup procedures, see "Backup and Data Recovery" on page 373.

Backing up a database using the Create Backup Images wizard

❖ To back up a database file or a running database

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Utilities tab.
3. In the right pane, double-click Create Backup Images.

   The Create Backup Images wizard appears.
4. Follow the instructions in the wizard.

☞ For full information on backing up a database from Sybase Central, see "Backup and Data Recovery" on page 373.

Tip
You can also access the Create Backup Images Database wizard from within Sybase Central using any of the following methods:

- ♦ Choosing Tools ➤ Adaptive Server Anywhere 9 ➤ Create Backup Images.
- ♦ Selecting a database in the left pane, and choosing Create Backup Images from the File menu.
- ♦ Right-clicking a database, and choosing Create Backup Images from the popup menu.

Backing up a database using the dbbackup command-line utility

Syntax **dbbackup** [*options*] *target-directory*

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-c *"keyword=value; ..."*	Supply database connection parameters.
-d	Only back up the main database file.
-l *file*	Live backup of the transaction log to a file.
-n	Change the naming convention for the backup transaction log.
-o *filename*	Log output messages to a file.
-q	Quiet mode—do not print messages.
-r	Rename and start a new transaction log.
-s	Perform an image backup on the server using the BACKUP statement.
-t	Back up only the transaction log.
-w	Back up only the write file.
-x	Delete and restart the transaction log.
-xo	Delete and restart the transaction log without making a backup.
-y	Replace files without confirmation.

Option	Description
target-directory	The directory to which the backup files are copied.

Description

If neither of the options -d or -t are used, all database files are backed up.

If -s is specified, the backup is created on the server using the BACKUP statement. Otherwise, the backup is made on the client machine.

Backup utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Connection parameters (-c) If the connection parameters are not specified, connection parameters from the SQLCONNECT environment variable are used, if set. The user ID must have DBA authority or REMOTE DBA authority.

☞ For a description of the connection parameters, see "Connection parameters" on page 176.

For example, the following command backs up the **asademo** database running on the server **sample_server**, connecting as user ID **DBA** with password **SQL**, into the *asabackup* directory:

```
dbbackup -c "eng=sample_server;dbn=asademo;uid=DBA;pwd=SQL"
         asabackup
```

Back up main database only (-d) Back up the main database files only, without backing up the transaction log file, if one exists.

Live backup (-l lower-case L) This option is provided to enable a secondary system to be brought up rapidly in the event of a server crash. A live backup does not terminate, but continues running while the server runs. It runs until the primary server becomes unavailable. At that point, it is shut down, but the backed up log file is intact and can be used to bring a secondary system up quickly.

If you specify -l, then you cannot use -s to create an image back up on the server.

☞ For more information about live backups, see "Differences between live backups and transaction log mirrors" on page 393 and "Making a live backup" on page 414.

Change backup transaction log naming convention (-n) This option is used in conjunction with -r. It changes the naming convention of the backup transaction log file to *yymmddxx.log*, where *xx* are sequential letters ranging from **AA** to **ZZ** and *yymmdd* represents the current year, month, and day.

The backup copy of the transaction log file is stored in the directory specified in the command, and with the *yymmddxx.log* naming convention. This allows backups of multiple versions of the transaction log file to be kept in the same backup directory.

You can also use the both the -x option and the -n option to rename the log copy. For example

```
dbbackup -x -n
```

Log output messages to file (-o) Write output messages to the named file.

Operate quietly (-q) Do not display output messages. This option is available only when you run this utility from a command prompt.

Rename and start new transaction log (-r) This option forces a checkpoint and the following three steps to occur:

1. The current working transaction log file is copied and saved to the directory specified in the command.

2. The current transaction log remains in its current directory, but is renamed using the format *yymmddxx.log*, where *xx* are sequential characters starting at *AA* and running through to *ZZ*, and *yymmdd* represents the current year, month, and day. This file is then no longer the current transaction log.

3. A new transaction log file is generated that contains no transactions. It is given the name of the file that was previously considered the current transaction log, and is used by the database server as the current transaction log.

Perform an image backup on the server (-s) This option allows you to create an image backup on the server using the BACKUP statement. If you specify the -s option, the -l option (to create a live backup of the transaction log) cannot be used. The directory specified is a directory relative to the server's current directory, so it is recommended that you specify a full pathname. In addition, the server must have write permissions on the specified directory. When -s is specified, the Backup utility does not display progress messages and does not prompt you when it overwrites existing files. If you want to be prompted when an attempt is made to overwrite an existing file, you should not specify -s or -y.

Back up the transaction log file only (-t) This can be used as an incremental backup since the transaction log can be applied to the most recently backed up copy of the database file(s).

Back up the database write file only (-w) For a description of database write files, see "The Write File utility (deprecated)" on page 609.

Delete and restart the transaction log (-x) Back up the existing transaction log, then delete the original log and start a new transaction log.

Delete and restart the transaction log without a backup (-xo) Delete the current transaction log and start a new one. This operation does not carry out a backup; its purpose is to free up disk space in non-replication environments.

Operate without confirming actions (-y) Choosing this option creates the backup directory or replaces a previous backup file in the directory without confirmation. If you want to be prompted when an attempt is made to overwrite an existing file, you should not specify -s or -y.

Target-directory The directory to which the backup files are copied. If the directory does not exist, it is created. However, the parent directory must exist.

The Collation utility

With the Collation utility, you can extract a collation (sorting sequence) into a file suitable for creating a database using a custom collation.

The file that is produced can be modified and used with Sybase Central or the -z option of dbinit to create a new database with a custom collation.

Exit codes are 0 (success) or non-zero (failure).

You must change the label on the following line in the collation file. If you do not, the custom collation will conflict with the original collation on which it is based.

```
Collation label (name)
```

☞ For more information about custom collation sequences, see "Creating a database with a custom collation" on page 358.

You can access the Collation utility in the following ways:

- ♦ From Sybase Central, using the Create Custom Collation wizard.
- ♦ At a command prompt, using the dbcollat command.

Extracting a collation using the Create Custom Collation wizard

❖ To use the Create Custom Collation wizard

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Utilities tab.
3. In the right pane, double-click Create Custom Collation.

   The Create Custom Collation wizard appears.
4. Follow the instructions in the wizard.

Tip
You can also access the Create Custom Collation wizard from within Sybase Central using any of the following methods:

- ♦ Choosing Tools ➤ Adaptive Server Anywhere 9 ➤ Create Custom Collation.
- ♦ Selecting a database in the left pane, and choosing Create Custom Collation from the File menu.
- ♦ Right-clicking a database, and choosing Create Custom Collation from the popup menu.

Extracting a collation using the dbcollat command-line utility

Syntax

dbcollat [*options*] *output-file*

Options	Description
@*data*	Read options from the specified environment variable or configuration file.
-c *"keyword=value; ..."*	Supply database connection parameters.
-d *collation-file*	Convert the definition file to INSERT statements with collation mapping placed in *output-file*.
-e	Include empty mappings.
-o *filename*	Log output messages to a file.
-q	Quiet mode—do not print messages.
-x	Use hex for extended characters (7F-FF).
-y	Replace the file without confirmation.
-z *col-seq*	Specify a collation sequence label.

Collation utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Connection parameters (-c) For a description of the connection parameters, see "Connection parameters" on page 176. If the connection parameters are not specified, connection parameters from the SQLCONNECT environment variable are used, if set.

For example, the following command extracts a collation file from the **asademo** database that is running on the **sample_server** server, and connects as user ID **DBA** with password **SQL**:

```
dbcollat -c "eng=sample_server;dbn=asademo;uid=DBA;pwd=SQL" c:\
         sample\col
```

Convert the collation definition file to INSERT statements that describe the collation (-d) When a database is created, the collation is inserted into the SYS.SYSCOLLATION system table. A mapping from the collation to character sets and Sybase TDS collations is also inserted into the SYS.SYSCOLLATIONMAPPINGS system table. This collation is selected from the set of provided collations in the *collseqs.sql* file or from the custom collations in the *custom.sql* file in the *scripts* subdirectory of your Adaptive Server Anywhere installation directory.

☞ For more information about the SYSCOLLATIONMAPPINGS system table, see "SYSCOLLATIONMAPPINGS system table" [*ASA SQL Reference*, page 686].

Custom collations are added to the *custom.sql* script. The -d option converts the collation definition file that you provide into INSERT statements that can be copied into *custom.sql*.

For example, you can use the -d option with the dbcollat command as follows:

```
dbcollat -d coll-defn-file custom-file
```

The *coll-defn-file* is read and parsed as a collation definition. Output is written to *custom-file*. The *custom-file* contents must be added to *custom.sql*.

☞ For more information about creating a custom collation using the -d option, see "Creating a custom collation" on page 357.

Include empty mappings (-e) Normally, collations don't specify the actual value that a character is to sort to. Instead, each line of the collation sorts to one position higher than the previous line. However, older collations have gaps between some sort positions. Normally, the Collation utility skips the gaps and writes the next line with an explicit sort-position. This option causes the Collation utility to write empty mappings (consisting of just a colon (:)) for each line in the gap.

Log output messages to file (-o) Write output messages to the named file.

Operate quietly (-q) Do not display output messages. This option is available only when you run this utility from a command prompt.

Use hexadecimal for extended characters [7F to FF] (-x) Extended single-byte characters (whose value is greater than hex 7F) may or may not appear correctly on your screen, depending on whether the code page on your computer is the same as the code page of the collation that you are extracting. This option causes the Collation utility to write any character from hex 7F or above as a two-digit hexadecimal number, in the form \x*dd*. For example:

```
\x80, \xFE
```

Without the -x option, only characters from \x00 to \x1F, \x7F and \xFF are written in hexadecimal form.

Operate without confirming actions (-y) Choosing this option automatically replaces an existing collation file without prompting for confirmation.

Specify a collation sequence label (-z) Specify the label of the collation to be extracted. The names of the recommended collation sequences can be found by executing the following command:

```
dbinit -l
```

If the -z option is specified with one of the available collation labels, then dbcollat does not connect to a database. Otherwise, it connects to a database and extracts the collation of that database. If the collation label does not match the collation label of the database, an error is returned.

The Compression utility (deprecated)

With the Compression utility you can compress a database file. The Compression utility reads the given database file and creates a compressed database file. Compressed databases are usually 40 to 60 per cent of their original size. The database server cannot update compressed database files: they must be used as read-only files, in conjunction with write files.

Deprecated feature
The use of compressed databases is deprecated.

The Compression utility does not compress files other than the main database file. The database to be compressed must not be running.

Exit codes are 0 (success) or non-zero (failure).

You can access the Compression utility in the following ways:

- From Sybase Central, using the Compress Database wizard.
- At a command prompt, using the dbshrink command. This is useful for incorporating into batch or command files.

Caution
Compressing an encrypted database removes encryption from the database.

Compressing a database using the Compress Database wizard

❖ To compress a database file

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Utilities tab.
3. In the right pane, double-click Compress Database.

   The Compress Database wizard appears.
4. Follow the instructions in the wizard.

Tip
You can also access the Compress Database wizard from within Sybase Central by right-clicking a database, and choosing Compress Database from the popup menu, or by choosing any of the following options:

♦ Choosing Tools ➤ Adaptive Server Anywhere 9 ➤ Compress Database.

♦ Selecting a database in the left pane, and choosing Compress Database from the File menu.

♦ Right-clicking a database, and choosing Compress Database from the popup menu.

Compressing a database using the dbshrink command-line utility

Syntax

dbshrink [*options*] *database-file* [*compressed-database-file*]

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-ek *key*	Specify encryption key.
-ep	Prompt for encryption key.
-o *filename*	Log output messages to a file.
-q	Quiet mode—do not print messages.
-y	Replace an existing output file without confirmation.

Description

The dbshrink utility reads *database-file* and creates a compressed database file. The compressed filename defaults to the same name as the original database file, but with an extension of *.cdb*. The output filename (with extension) must not be same as the input filename (with extension).

Compression utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility” on page 524.

Specify encryption key (-ek) This option allows you to specify the encryption key for strongly encrypted databases directly in the command. If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log in any way. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify a key for a strongly encrypted database.

Prompt for encryption key (-ep) This option allows you to specify that you want to be prompted for the encryption key. This option causes a dialog box to appear, in which you enter the encryption key. It provides an extra measure of security by never allowing the encryption key to be seen in clear text. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify a key for a strongly encrypted database.

Log output messages to file (-o) Write output messages to the named file.

Operate quietly (-q) Do not display output messages. This option is available only when you run this utility from a command prompt.

Operate without confirming actions (-y) Choosing this option automatically replaces existing compressed database file(s) without requesting confirmation.

The Data Source utility

The Data Source utility is a cross-platform alternative to the ODBC Administrator for creating, deleting, describing, and listing Adaptive Server Anywhere ODBC data sources.

☞ For information about creating a data source using the ODBC Administrator, see "Working with ODBC data sources" on page 53.

Managing ODBC data sources using the dbdsn command-line utility

Syntax

dbdsn [*modifier-options*]
{ **-l**[**u** | **s**] [**-qq**]
| **-d**[**u** | **s**] *dsn*
| **-g**[**u** | **s**] *dsn*
| **-w**[**u** | **s**] *dsn* [*details-options*;. . .]
| **-cl**[**-qq**] }

Parameters

Major option	Description
@*data*	Read options from the specified environment variable or configuration file.
-l[**u** | **s**] [**-qq**]	List either all Adaptive Server Anywhere user or system data sources. By default, user data sources are listed. Using -qq with this option lists the DSNs without any banner or titles.
-d[**u** | **s**] *dsn*	Delete the named Adaptive Server Anywhere user or system data source. User data sources is the default.
-g[**u** | **s**] *dsn*	List (get) details about the named Adaptive Server Anywhere user or system data source. User data sources is the default.
-w[**u** | **s**] *dsn* [*details-options*]	Create (write) a user or system data source definition. User data sources is the default.
-cl[**-qq**]	List available connection parameters. Using -qq with this option lists the available connection parameters without any banner or titles.

Modifier-options	Description
-b	Brief. Print connection string for the data source.
-cm	Display the data source creation command.
-q	Quiet. Do not print banner.
-v	Verbose. Print connection parameters in tabular form.
-y	Delete or overwrite data source without confirmation.

Details-options	Description
-cw	Ensure that the DBF parameter (specified using -c) is an absolute filename. If the value of DBF is not an absolute filename, the Data Source utility prepends the current working directory (CWD).
-c "*keyword=value;...***"**	Supply database connection parameters.
-ec *encryption type*	Encrypt all network packets.
-o *filename*	Write client messages to *filename*.
-p *size*	Set maximum network packet size.
-r	Disable multiple record fetching.
-tl *seconds*	Client liveness timeout period.
-x *list*	List network drivers to run.
-z	Display debugging information.
server-name	Connect to named database server.

See also

- ♦ "Working with ODBC data sources" on page 53
- ♦ "Using ODBC data sources on UNIX" on page 56

Description

The Data Source utility is a cross-platform alternative to the ODBC Administrator for creating, deleting, describing, and listing Adaptive Server Anywhere ODBC data sources. The utility is useful for batch operations. On Windows operating systems, the data sources are held in the registry.

On UNIX operating systems, the data sources are held in the *.odbc.ini* file. When you use the Data Source utility to create or delete Adaptive Server Anywhere ODBC data sources on UNIX, the utility automatically updates the `[ODBC Data Sources]` section of the *.odbc.ini* file. If you do not specify the Driver connection parameter using the -c option on UNIX, the Data Source utility automatically adds a Driver entry with the full path of the Adaptive Server Anywhere ODBC driver based on the setting of the ASANY9 environment variable.

☞ For more information about the *.odbc.ini* file, see "The system information file (.odbc.ini)" [*ODBC Drivers for MobiLink and Remote Data Access*, page 4].

Caution
You should not obfuscate the odbc.ini system information file with the File Hiding utility (dbfhide) on UNIX unless you will only be using Adaptive Server Anywhere data sources. If you plan to use other data sources (for example, for MobiLink synchronization), then obfuscating the odbc.ini file may prevent other drivers from functioning properly.

The modifier options can occur before or after the major option specification.

Exit codes are 0 (success) or non-zero (failure).

Data Source utility options

When you use the Data Source utility, you can choose from major options, modifier options, and details options.

Major options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

List available connection parameters (-cl) This convenience option lists the connection parameters supported by the dbdsn utility.

☞ For descriptions of the Adaptive Server Anywhere connection parameters supported by the Data Source utility (dbdsn), see "Connection parameters" on page 176.

The Data Source utility (dbdsn) supports the following ODBC connection parameters. Boolean (true or false) arguments are either YES or 1 if true, or NO or 0 if false.

Name	**Description**
Delphi	Delphi cannot handle multiple bookmark values for a row. When you set this value to NO, one bookmark value is assigned to each row, instead of the two that are otherwise assigned. Setting this option to YES can improve scrollable cursor performance.
DescribeCursor	This parameter lets you specify how often you want a cursor to be redescribed when a procedure is executed. The default setting is If Required. ♦ **Never** Specify 0, N, or NO if you know that your cursors do not have to be redescribed. Redescribing cursors is expensive and can decrease performance. ♦ **If required** Specify 1, Y, or YES if you want the ODBC driver to determine whether a cursor must be redescribed. The presence of a RESULT clause in your procedure prevents ODBC applications from redescribing the result set after a cursor is opened. This is the default setting. ♦ **Always** If you specify 2, A, or ALWAYS, the cursor is redescribed each time it is opened. If you use Transact-SQL procedures or procedures that return multiple result sets, you must redescribe the cursor each time it is opened.
Description	This parameter allows you to provide a description of the ODBC data source.

Name	Description
Driver	This parameter allows you to specify an ODBC driver for the connection, as follows: `Driver=<`*`driver-name`*`>`. By default, the driver that is used is **Adaptive Server Anywhere 9.0**. The *driver-name* must be Adaptive Server Anywhere *X*.0, where *X* is the major version number of the software. If the *driver-name* does not begin with **Adaptive Server Anywhere**, it cannot be read by the Data Source utility (dbdsn). On UNIX, this parameter specifies the fully-qualified path to the shared object. If you do not specify the Driver connection parameter on UNIX, the Data Source utility automatically adds a Driver entry with the full path of the Adaptive Server Anywhere ODBC driver based on the setting of the ASANY9 environment variable.
GetTypeInfoChar	When this option is set to YES, CHAR columns are returned as SQL_CHAR instead of SQL_VARCHAR. By default, CHAR columns are returned as SQL_-VARCHAR.
InitString	InitString allows you to specify a command that is executed immediately after the connection is established. For example, you may with to set a database option or execute a stored procedure.

Name	Description
IsolationLevel	You can specify one of the following values to set the initial isolation level for this data source: ♦ **0** This is also called the read uncommitted isolation level. This is the default isolation level. It provides the maximum level of concurrency, but dirty reads, non-repeatable reads, and phantom rows may be observed in result sets. ♦ **1** This is also called the read committed level. This provides less concurrency than level 0, but eliminates some of the inconsistencies in result sets at level 0. Non-repeatable rows and phantom rows may occur, but dirty reads are prevented. ♦ **2** This is also called the repeatable read level. Phantom rows may occur. Dirty reads and non-repeatable rows are prevented. ♦ **3** This is also called the serializable level. This provides the least concurrency, and is the strictest isolation level. Dirty reads, non-repeatable reads, and phantom rows are prevented. ☞ For more information, see "Choosing isolation levels" [*ASA SQL User's Guide*, page 116].
KeysInSQLStatistics	Specify YES if you want the SQLStatistics function to return foreign keys. The ODBC specification states that SQLStatistics should not return primary and foreign keys; however, some Microsoft applications (such as Visual Basic and Access) assume that primary and foreign keys are returned by SQLStatistics.
LazyAutocommit	Setting this parameter to YES delays the commit operation until a statement closes.
PrefetchOnOpen	When PrefetchOnOpen is set to YES, a prefetch request is sent with a cursor open request. The prefetch eliminates a network request to fetch rows each time a cursor is opened. Columns must already be bound for the prefetch to occur on the open. This connection parameter can help reduce the number of client/server requests to help improve performance over a LAN or WAN.

Name	Description
PreventNotCapable	The Adaptive Server Anywhere ODBC driver returns a `Driver not capable` error because it does not support qualifiers. Some ODBC applications do not handle this error properly. Set this parameter to YES to prevent this error code from being returned, allowing these applications to work.
SuppressWarnings	Set this parameter to YES if you want to suppress warning messages that are returned from the database server on a fetch. Versions 8.0 and later of the database server return a wider range of fetch warnings than earlier versions of the software. For applications that are deployed with an earlier version of the software, you can select this option to ensure that fetch warnings are handled properly.
TranslationDLL	This option is provided for backwards compatibility. The use of translators is not recommended.
TranslationName	This option is provided for backwards compatibility. The use of translators is not recommended.
TranslationOption	This option is provided for backwards compatibility. The use of translators is not recommended.

Delete the named data source (-d) Deletes the named Adaptive Server Anywhere data source. You can modify the option using the **u** (user) or **s** (system) specifiers. The default specifier is **u**. If you supply -y, any existing data source is deleted without confirmation.

List (get) details of the named data source (-g) List the definition of the named Adaptive Server Anywhere data source. You can modify the format of the output using the -b or -v options. You can modify the option using the **u** (user) or **s** (system) specifiers. The default specifier is **u**.

List defined data sources (-l) Lists the available Adaptive Server Anywhere ODBC data sources. You can modify the list format using the -b or -v options. You can modify the option using the **u** (user) or **s** (system) specifiers. The default specifier is **u**.

Create (write) a data source definition (-w) Creates a new data source, or overwrites one if one of the same name exists. You can modify the option using the **u** (user) or **s** (system) specifiers. The default specifier is **u**. If you supply -y, any existing data source is overwritten without confirmation.

Modifier options

Print connection string for the data source (-b) Format the output of the list as a single line connection string.

Display the data source creation command (-cm) Displays the command used to create the data source. This option can be used to output the creation command to a file, which can be used to add the data source to another machine or can be used to restore a data source to its original state if changes have been made to it. You must specify the -g option or -l option with -cm or the command fails. Specifying -g displays the creation command for the specified data source, while specifying -l displays the creation command for all data sources.

If the specified data source does not exist, the command to delete the data source is generated. For example, if the mydsn data source does not exist on the machine, `dbdsn -cm -g mydsn` would return the following command to delete the mydsn data source:

```
dbdsn -y -du "mydsn"
```

Do not print banner (-q) Suppress the informational banner. If you specify -q when deleting or modifying a data source, you must also specify -y.

Do not print banner or titles (-qq) Suppress both the informational banner and titles. This option can only be used with the -l and the -cl options.

Print connection parameters in tabular form (-v) Format the output of the list over several lines, as a table.

Delete or overwrite data source without confirmation (-y) Automatically delete or overwrite each data source without prompting you for confirmation. If you specify -q when deleting or modifying a data source, you must also specify -y.

Details options

Connection parameters (-c) Specify connection parameters as a connection string.

☞ For more information, see “Connection parameters” on page 176.

Absolute filenames (-cw) Ensure that the DBF parameter (specified using -c) is an absolute filename. If the value of DBF is not an absolute filename, dbdsn will prepend the current working directory (CWD). This option is useful because some operating systems (Win 9x) do not have CWD information readily available in batch files.

Encrypt network packets (-ec) Encrypt packets sent between the client application and the server.

☞ For more information, see “Encryption connection parameter [ENC]” on page 191.

Log output messages to file (-o) Write output messages to the named file. By default, messages are written to the console.

☞ For more information, see "Logfile connection parameter [LOG]" on page 199.

Set maximum network packet size (-p) The maximum packet size for network communications, in bytes. The value must be greater than 300, and less than 16000. The default setting is 1460.

☞ For more information, see "CommBufferSize connection parameter [CBSIZE]" on page 180.

Disable multiple-record fetching (-r) By default, when the database server gets a simple fetch request, the application asks for extra rows. You can disable this behavior using this option.

☞ For more information, see "DisableMultiRowFetch connection parameter [DMRF]" on page 190.

Set client liveness timeout (-tl) Terminates connections when they are no longer intact. The value is in seconds.

The default is server setting, which in turn has a default value of 120 seconds.

☞ For more information, see "LivenessTimeout connection parameter [LTO]" on page 199.

Set communications links (-x) A comma separated list of network drivers to run.

☞ For more information, see "CommLinks connection parameter [LINKS]" on page 181.

Display debugging information (-z) Provide diagnostic information on communications links on startup.

server-name Connect to the named server. Only the first 40 characters are used.

☞ For more information, see "The Database Server" on page 115.

Examples

Write a definition of the data source **newdsn**. Do not prompt for confirmation if the data source already exists.

```
dbdsn -y -w newdsn -c "uid=DBA;pwd=SQL;LINKS=TCPIP" -v
```

or, with a different option order,

```
dbdsn -w newdsn -c "uid=DBA;pwd=SQL;LINKS=TCPIP" -y
```

List all known user data sources, one data source name per line:

```
dbdsn -l
```

List all known system data sources, one data source name per line:

```
dbdsn -ls
```

List all data sources along with their associated connection string:

```
dbdsn -l -b
```

Report the connection string for user data source **MyDSN**:

```
dbdsn -g MyDSN
```

Report the connection string for system data source **MyDSN**:

```
dbdsn -gs MyDSN
```

Delete the data source **BadDSN**, but first list the connection parameters for **BadDSN** and prompt for confirmation:

```
dbdsn -d BadDSN -v
```

Delete the data source **BadDSN** without prompting for confirmation.

```
dbdsn -d BadDSN -y
```

Create a data source named **NewDSN** for the database server **MyServer**:

```
dbdsn -w NewDSN -c "uid=DBA;pwd=SQL;eng=MyServer"
```

If a **NewDSN** already exists, the previous definition is overwritten.

List all connection parameter names and their aliases:

```
dbdsn -cl
```

List all user DSNs (without banner and titles):

```
dbdsn -l -qq
```

List all connection parameter names (without banner and titles):

```
dbdsn -cl -qq
```

Specify an absolute filename. When the DSN is created, it will contain *dbf=E:\asa90\my.db.*

```
E:\asa90> dbdsn -w testdsn -cw -c
         uid=dba;pwd=sql;eng=asa;dbf=my.db
```

Generate the command to create the ASA 9.0 Sample data source and output it to a file called *restoredsn.bat*:

```
dbdsn -cm -g "ASA 9.0 Sample" > restoredsn.bat
```

The *restoredsn.bat* file contains the following:

```
dbdsn -y -wu "ASA 9.0 Sample" -c "UID=dba;PWD=sql;
DBF=C:\Program Files\Sybase\SQL Anywhere 9\asademo.db;
ENG=asademo9;START=C:\Program Files\Sybase\SQL Anywhere 9\
win32\dbeng9.exe -c 8m;ASTOP=yes;
Description=Adaptive Server Anywhere Sample Database"
```

The Erase utility

With the Erase utility, you can erase a database file and its associated transaction log, or you can erase a transaction log file or transaction log mirror file. All database files and transaction log files are marked read-only to prevent accidental damage to the database and accidental deletion of the database files.

Deleting a database file that references other dbspaces does not automatically delete the dbspace files. If you want to delete the dbspace files on your own, change the files from read-only to writable, and then delete the files individually. As an alternative, you can use the DROP DATABASE statement to erase a database and its associated dbspace files.

If you erase a database file, the associated transaction log and transaction log mirror are also deleted. If you erase a transaction log for a database that also maintains a transaction log mirror, the mirror is not deleted.

The database to be erased must not be running when this utility is used.

Exit codes are 0 (success) or non-zero (failure).

You can access the Erase utility in the following ways:

♦ From Sybase Central, using the Erase Database wizard.

♦ At a command prompt, using the dberase command. This is useful for incorporating into batch or command files.

☞ For more information, see the "DROP DATABASE statement" [*ASA SQL Reference*, page 456].

Erasing a database using the Erase Database wizard

❖ **To erase a database file**

1. In the left pane, select the Adaptive Server Anywhere plug-in.

2. In the right pane, click the Utilities tab.

3. In the right pane, double-click Erase Database

   The Erase Database wizard appears.

4. Follow the instructions in the wizard.

☞ For full information on erasing a database from Sybase Central, see "Erasing databases" [*ASA SQL User's Guide*, page 34].

Tip
You can also access the Erase Database wizard from within Sybase Central using any of the following methods:

- ♦ Choosing Tools ➤ Adaptive Server Anywhere 9 ➤ Erase Database.
- ♦ Selecting a server in the left pane, and choosing Erase Database from the File menu.
- ♦ Right-clicking a server, and choosing Erase Database from the popup menu.

Erasing a database using the dberase command-line utility

Syntax

dberase [*options*] *database-file*

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-ek *key*	Specify encryption key.
-ep	Prompt for encryption key.
-o *filename*	Log output messages to a file.
-q	Operate quietly—do not print messages.
-y	Erase files without confirmation.

Description

The *database-file* may be a database file or transaction log file. The full filename must be specified, including extension. If a database file is specified, the associated transaction log file (and mirror, if one is maintained) is also erased.

Note
The Erase utility does *not* erase dbspaces. If you want to erase a dbspace, you can do so with the DROP DATABASE statement or using the Erase Database wizard in Sybase Central.

Erase utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Specify encryption key (-ek) This option allows you to specify the encryption key for strongly encrypted databases directly in the command. If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log in any way. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify the correct key for a strongly encrypted database.

Prompt for encryption key (-ep) This option allows you to specify that you want to be prompted for the encryption key. This option causes a dialog box to appear, in which you enter the encryption key. It provides an extra measure of security by never allowing the encryption key to be seen in clear text. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify the correct key for a strongly encrypted database.

Log output messages to file (-o) Write output messages to the named file.

Operate quietly (-q) Do not display output messages. If you specify this option, you must also specify -y or the operation will fail.

Operate without confirming actions (-y) Choosing this option automatically deletes each file without prompting you for confirmation. If you specify -q, you must also specify -y or the operation will fail.

The File Hiding utility

With the File Hiding utility you can add simple encryption to configuration files and initialization files to hide the contents of each file.

Hiding the contents of files using the dbfhide command-line utility

Syntax

dbfhide *original-configuration-file encrypted-configuration-file*

Option	Description
original-configuration-file	Name of the original file.
encrypted-configuration-file	Name for the new obfuscated file.

Description

Configuration files are used by some utilities to hold command-line options. These options may contain a password. You can use the File Hiding utility to add simple encryption to configuration files, as well as to .ini files used by Adaptive Server Anywhere and its utilities, and thereby obfuscate the contents of the file. The original file will not be modified. Once you add simple encryption to a file, there is no way to remove it. In order to make changes to an obfuscated file, you must keep a copy of the original file that you can modify and obfuscate again.

Hiding the contents of .ini files

In many cases, Adaptive Server Anywhere expects a .ini file to have a particular name. When you want to add simple encryption to a file whose name is important (such as *asaldap.ini*), you need to save a copy of the original file with a different name when you add simple encryption to the file. If you do not keep a copy of the original file, then you cannot modify the contents of the file once it has been obfuscated. The following steps explain how to add simple encryption to a .ini file.

❖ To hide the contents of .ini files

1. Save the file with a different name.

   ```
   rename asaldap.ini asaldap.ini.org
   ```

2. Obfuscate the file with the File Hiding utility, giving the obfuscated file the required file name:

   ```
   dbfhide asaldap.ini.org asaldap.ini
   ```

3. Protect the *asaldap.ini.org* file using file system or operating system protection, or store the file in a secure location.

   To make a change to the *asaldap.ini* file, edit the *asaldap.ini.org* file and repeat step 2.

Caution
You should not add simple encryption to the .odbc.ini system information file with the File Hiding utility (dbfhide) on UNIX unless you will only be using Adaptive Server Anywhere data sources. If you plan to use other data sources (for example, for MobiLink synchronization), then obfuscating the contents of the .odbc.ini file may prevent other drivers from functioning properly.

This utility does *not* accept the ***@data*** parameter to read in options from a configuration file.

Examples

Create a configuration file that starts the personal database server and the sample database. It should set a cache of 10 Mb, and name this instance of the personal server *Elora*. The configuration file would be written as follows:

```
# Configuration file for server Elora
-n Elora
-c 10M
path\asademo.db
```

(Note that lines beginning with # are treated as comments.)

Name the file *sample.txt*. If you wanted to start the database using this configuration file, your command line would be:

```
dbeng9 @sample.txt
```

Now, add simple encryption to the configuration.

```
dbfhide sample.txt encrypted_sample.txt
```

Use the encrypted_sample.txt file to start a database.

```
dbsrv9 @encrypted_sample.txt
```

☞ For more information about encryption, see "Keeping Your Data Secure" [*SQL Anywhere Studio Security Guide*, page 3].

☞ For more information about using configuration files, see "Using configuration files to store server startup options" on page 10.

The following command adds simple encryption to the *asaldap.ini* file:

```
dbfhide asaldap.ini encrypted_asaldap.ini
```

The Histogram utility

The Histogram utility converts a histogram into a Microsoft Excel chart containing information about the selectivity of predicates. The utility only works with Excel 97 and later, and only on Windows.

Converting a histogram using the dbhist command-line utility

Syntax

dbhist [*options*] **-t** *table-name* [*excel-output-name*]

Parameters

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-c *options*	Connection string.
-n *colname*	Column to associate with the histogram.
-t *table-name*	The name of the table to generate histograms for.
-u *owner*	The table owner.
excel-output-name:	The name of the generated Excel file.

Description

Histograms are stored in the SYSCOLSTAT system table and can also be retrieved with the sa_get_histogram stored procedure. The Histogram utility converts a histogram into a Microsoft Excel chart containing information about the selectivity of predicates. The utility only works with Excel 97 and later, and only on Windows.

To determine the selectivity of a predicate over a string column, you should use the ESTIMATE or ESTIMATE_SOURCE functions. Attempting to retrieve a histogram from string columns causes both sa_get_histogram and the Histogram utility to generate an error.

Exit codes are 0 (success) or non-zero (failure).

☞ For more information about the sa_get_histogram stored procedure, see "sa_get_histogram system procedure" [*ASA SQL Reference*, page 796].

Histogram utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Specify a connection string (-c) Specify connection parameters.

☞ For a description of the connection parameters, see "Connection parameters" on page 176.

Specify a column (-n) Specify the name of the column to associate the histogram with. If you do not specify a column, all columns that have histograms in the table are returned.

Specify a table name (-t) Specify the name of the table to generate histograms for.

Specify an owner (-u) Specify the owner of the table.

excel-output-name The name of the generated Excel file. If no name is specified, Excel prompts you to enter one with a Save As dialog.

Example

Assuming that a histogram has been created for the column, the following statement (entered all on the same line) generates an Excel chart for the column prod_id in the table sales_order_items for database *asademo.db*, and saves it as *histgram.xls*.

```
dbhist -c "uid=DBA;pwd=SQL;dbf=asademo.db"
      -n prod_id -t sales_order_items histgram.xls
```

The following statement generates charts for every column with a histogram in the table sales_order, assuming that asademo is already loaded. This statement also attempts to connect using uid=dba, pwd=sql. No output file name is specified, so Excel prompts you to enter one.

```
dbhist -t sales_order -c "UID=DBA;PWD=SQL"
```

The Information utility

The Information utility can display information about a database.

Obtaining database information using the dbinfo command-line utility

Syntax

dbinfo [*options*]

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-c *"keyword=value; . . . "*	Database connection parameters.
-o *filename*	Log output messages to a file.
-q	Operate quietly.
-u	Output page usage statistics.

Description

The dbinfo utility displays information about a database. It reports the time when the database was created, the name of any transaction log file or log mirror, the page size, the version of installed Java classes, the collation name and label, and other information. Optionally, it can also provide table usage statistics and details.

Exit codes are 0 (success) or non-zero (failure).

Information utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Connection parameters (-c) Specify connection parameters.

☞ For a description of the connection parameters, see "Connection parameters" on page 176.

Any valid user ID can run the Information utility, but to obtain page usage statistics you need DBA authority.

Log output messages to file (-o) Write output messages to the named file.

Operate quietly (-q) Do not display output messages.

Page usage statistics (-u) Display information about the usage and size of all tables, including system and user-defined tables.

You can only request page usage statistics if no other users are connected to the database and you have DBA authority.

The Initialization utility

With the Initialization utility, you can initialize (create) a database. A number of database attributes are specified at initialization and cannot be changed later except by unloading, reinitializing, and rebuilding the entire database. These database attributes include:

- ♦ Case sensitivity or insensitivity
- ♦ Password case sensitivity or insensitivity
- ♦ Storage as an encrypted file
- ♦ Treatment of trailing blanks in comparisons
- ♦ Page size
- ♦ Collation sequence

In addition, the choice of whether to use a transaction log and a transaction log mirror is made at initialization. This choice can be changed later using the Transaction Log utility.

Exit codes are 0 (success) or non-zero (failure).

You can access the Initialization utility in the following ways:

- ♦ From Sybase Central, using the Create Database wizard.
- ♦ From Interactive SQL, using the CREATE DATABASE statement.
- ♦ At a command prompt, using the dbinit command. This is useful for incorporating into batch or command files.

☞ For more information about creating a database, see "CREATE DATABASE statement" [*ASA SQL Reference*, page 338].

Creating a database using the Create Database wizard

❖ To create a database

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Utilities tab.
3. In the right pane, double-click Create Database.

   The Create Database wizard appears.
4. Follow the instructions in the wizard.

Tip
You can also access the Create Database wizard from within Sybase Central using any of the following methods:

- ♦ Choosing Tools ➤ Adaptive Server Anywhere 9 ➤ Create Database.
- ♦ Selecting a server in the left pane, and choosing Create Database from the File menu.
- ♦ Right-clicking a server, and choosing Create Database from the popup menu.

☞ For full information on creating a database in Sybase Central, see "Creating a database" [*ASA SQL User's Guide*, page 31].

Creating a database using the dbinit command-line utility

Syntax **dbinit** [*options*] *new-database-file*

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-b	Blank padding of strings for comparisons and fetching.
-c	Case sensitivity for all string comparisons.
-cp	Case sensitivity of passwords.
-e	Encrypt the database using simple encryption.
-ea *algorithm*	Specify which strong encryption algorithm to encrypt your database with: you can choose AES or AES_FIPS.
-ek *key*	Specify encryption key for strong encryption.
-ep	Prompt for encryption key for strong encryption.
-i	Do not install Sybase jConnect support.
-ja	Install default runtime Java classes.
-jdk *version*	Install entries for the named version of the Java Development Kit.
-k	Omit Watcom SQL compatibility views SYS.-SYSCOLUMNS and SYS.SYSINDEXES.
-l	List the recommended collation sequences.
-m *file-name*	Use a transaction log mirror (default is no mirror).

Option	Description
-n	No transaction log.
-o *filename*	Log output messages to a file.
-p *page-size*	Set page size.
-q	Quiet mode—do not print messages.
-s	Use checksums when writing pages to disk.
-t *log-name*	Transaction log filename (default is the database name with *.log* extension).
-z *col-seq*	Collation sequence used for comparisons.

Description

For example, the database *test.db* can be created with 4096 byte pages as follows:

```
dbinit -p 4096 test.db
```

Initialization utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Blank padding (-b) Ignore trailing blanks for comparison purposes and pad strings that are fetched into character arrays. For example, the two strings

```
'Smith'
'Smith   '
```

would be treated as equal in a database created with blank-padding.

This option is provided for compatibility with the ISO/ANSI SQL standard, which is to ignore trailing blanks in comparisons. The default is that blanks are significant for comparisons.

Case sensitivity for all string comparisons (-c) For databases created with this option, all values are considered to be case sensitive in comparisons and string operations. Identifiers in the database are case insensitive, even in case sensitive databases.

This option is provided for compatibility with the ISO/ANSI SQL standard. The default is that all comparisons are case insensitive.

Case sensitivity for passwords (-cp) By default, the case sensitivity of passwords is the same as the case sensitivity of the database. However, if you specify the -cp option, then passwords are case sensitive, regardless of the database case sensitivity. To specify that passwords are case insensitive, specify **-cp-**.

User ID and password
All databases are created with at least one user ID, **DBA**, with password **SQL**. Extended characters used in passwords are case sensitive regardless of the database sensitivity setting. User IDs, like other identifiers, are case insensitive even in case sensitive databases.

Encrypt the database using simple encryption (-e) Simple encryption makes it more difficult for someone to decipher the data in your database using a disk utility to look at the file. File compaction utilities cannot compress encrypted database files as much as unencrypted ones.

This simple encryption is equivalent to obfuscation and is intended only to keep data hidden in the event of casual direct access of the database file. For greater security, you can use strong encryption by supplying the -ea and -ek options.

Specify encryption algorithm (-ea) This option allows you to choose a strong encryption algorithm to encrypt your new database. You can choose either AES (the default) or AES_FIPS for the FIPS-approved algorithm. AES_FIPS uses a separate library and is not compatible with AES. AES_FIPS is only available on all supported 32-bit Windows platforms. Algorithm names are case insensitive. If you specify the -ea option, you must also specify -ep or -ek.

☞ For more information, see "Strong encryption" [*SQL Anywhere Studio Security Guide*, page 15].

Separately licensable option required
Strong database encryption using AES_FIPS requires that you obtain the separately-licensable SQL Anywhere Studio security option and is subject to export regulations.

☞ To order this component, see "Separately-licensable components" [*Introducing SQL Anywhere Studio*, page 5].

Specify encryption key (-ek) This option allows you to create a strongly encrypted database by specifying an encryption key directly in the command. The algorithm used to encrypt the database is AES or AES_FIPS as specified by the -ea option. If you specify the -ek option without specifying -ea, the AES algorithm is used.

Caution
Protect your key! Be sure to store a copy of your key in a safe location. A lost key will result in a completely inaccessible database, from which there is no recovery.

The following are invalid for database encryption keys:

♦ keys that begin with white space or single or double quotes

♦ keys that end with white space

♦ keys that contain semicolons

☞ For more information, see "Strong encryption" [*SQL Anywhere Studio Security Guide,* page 15].

Prompt for encryption key (-ep) This option allows you to specify that you want to create a strongly encrypted database by inputting the encryption key in a dialog box. This provides an extra measure of security by never allowing the encryption key to be seen in clear text.

You must input the encryption key twice to confirm that it was entered correctly. If the keys don't match, the initialization fails.

☞ For more information, see "Strong encryption" [*SQL Anywhere Studio Security Guide,* page 15].

Do not install Sybase jConnect support (-i) If you want to use the Sybase jConnect JDBC driver to access system catalog information, you need to install jConnect support. Use this option if you want to exclude the jConnect system objects. You can still use JDBC, as long as you do not access system information. If you want, you can add Sybase jConnect support at a later time using Sybase Central or the ALTER DATABASE statement.

☞ For more information, see "Installing jConnect system objects into a database" [*ASA Programming Guide,* page 111].

Install runtime Java classes (-ja) If you want to use Java in your database, you must install the runtime Java classes. Specifying -ja installs version 1.3 of the JDK into the database.

For example, the following command creates a database that supports JDK 1.3 applications in the database.

```
dbinit -ja java2.db
```

The runtime classes add several megabytes to the size of a database, so if you do not intend to use Java classes, you can omit the -ja option to avoid installing them.

You can add the runtime Java classes at a later time using the Upgrade Database wizard, or the ALTER DATABASE statement.

☞ For more information, see "Java-enabling a database" [*ASA Programming Guide*, page 84] and "ALTER DATABASE statement" [*ASA SQL Reference*, page 266].

Install support for the named version of the JDK (-jdk) If you want to install a version of Java other than 1.3 into your database, use the -jdk option followed by the version number. Currently, the only other version of Java supported is 1.1.8.

For example, the following command creates a database that supports JDK 1.1.8 applications in the database.

```
dbinit -jdk 1.1.8 java2.db
```

The runtime classes add several megabytes to the size of a database, so if you do not intend to use Java classes, you can omit the -jdk option to avoid installing them.

The -jdk option implies the -ja option.

You can add the runtime Java classes at a later time using the Upgrade Database wizard, or the ALTER DATABASE statement.

☞ For more information, see "Java-enabling a database" [*ASA Programming Guide*, page 84] and "ALTER DATABASE statement" [*ASA SQL Reference*, page 266].

Omit Watcom SQL compatibility views (-k) By default, database creation generates the views SYS.SYSCOLUMNS and SYS.SYSINDEXES for compatibility with system tables that were available in Watcom SQL (versions 4 and earlier of this software). These views will conflict with the Sybase Adaptive Server Enterprise compatibility views dbo.syscolumns and dbo.sysindexes.

List the available collation sequences (-l) When you specify this option, dbinit lists the recommended collation sequences and then stops. No database is created. A list of available collation sequences is automatically presented in the Sybase Central Create Database wizard.

Use a transaction log mirror (-m) A transaction log mirror is an identical copy of a transaction log, usually maintained on a separate device, for greater protection of your data. By default, Adaptive Server Anywhere does not use a mirrored transaction log.

Do not use a transaction log (-n) Creating a database without a transaction log saves disk space. The transaction log is required for data replication and provides extra security for database information in case of media failure or system failure. Databases that do not use transaction logs typically run slower than databases that use transaction logs.

Log output messages to file (-o) Write output messages to the named file.

Page size (-p) The page size for a database can be (in bytes) 1024, 2048, 4096, 8192, 16384, or 32768, with 2048 being the default. Any other value for the size will be changed to the next larger size.

Large databases usually benefit from a larger page size. For example, the number of I/O operations required to scan a table is generally lower, as a whole page is read in at a time. Also, the number of levels in an index may be reduced as more entries can be stored on each page.

However, there are additional memory requirements for large page sizes. Also, the maximum number of rows stored on a page is 255, so that tables with small rows will not fill each page. For example, with 8 kb pages the average row size must be at least 32 bytes to fully use each page. For most applications, 16 kb or 32 kb page sizes are not recommended. You should not use page sizes of 16 kb or 32 kb in production systems unless you can be sure that a large database server cache is always available, and only after you have investigated the tradeoffs of memory and disk space with its performance characteristics.

Operate quietly (-q) Do not display output messages.

Use checksums (-s) Checksums are used to determine whether a database page has been modified on disk. When you create a database with checksums enabled, a checksum is calculated for each page just before it is written to disk. The next time the page is read from disk, the page's checksum is recalculated and compared to the checksum stored on the page. If the checksums are different, then the page has been modified or corrupted on disk.

Set the transaction log filename (-t) The transaction log is a file where the database server logs all changes, made by all users, no matter what application is being used. The transaction log plays a key role in backup and recovery (see "The transaction log" on page 378), and in data replication. If the filename has no path, it is placed in the same directory as the database file. If you run dbinit without specifying -t or -n, a transaction log is created with the same filename as the database file, but with extension *.log*.

Collation sequence (-z) The collation sequence is used for all string comparisons in the database.

If -z is not specified, Adaptive Server Anywhere chooses a default collation based on operating system language and character set settings.

☞ For more information, see "Choosing collations" on page 335.

In order to change the collation that an existing database uses, it is necessary to unload the database, create a new database using the appropriate collation, and then reload the database. It may be necessary to translate the data as well.

If you want to create a custom collation, use the Collation utility to create a file containing the collation. Once you have modified the collation and inserted it into the appropriate scripts, you use the Initialization utility to create the database and specify the new collation.

You must change the collation label in the custom collation file. Otherwise, the Initialization utility prevents the insertion of the new collation, since it conflicts with an existing collation.

☞ For more information on custom collation sequences, see "International Languages and Character Sets" on page 319.

☞ For information on the Collation utility, see "The Collation utility" on page 503.

The Interactive SQL utility

Interactive SQL provides an interactive environment for database browsing and for sending SQL statements to the database server.

You can start Interactive SQL in the following ways:

- ♦ from Sybase Central, using the Open Interactive SQL menu item.
- ♦ from the Start menu by choosing Start ➤ Programs ➤ SQL Anywhere 9 ➤ Adaptive Server Anywhere ➤ Interactive SQL.
- ♦ at a command prompt, using the dbisql command.

SQL statements used only from Interactive SQL

The following is a list of the SQL statements that can be used only from Interactive SQL:

"CLEAR statement [Interactive SQL]" [*ASA SQL Reference*, page 324]

"CONFIGURE statement [Interactive SQL]" [*ASA SQL Reference*, page 331]

"CONNECT statement [ESQL] [Interactive SQL]" [*ASA SQL Reference*, page 332]

"DISCONNECT statement [ESQL] [Interactive SQL]" [*ASA SQL Reference*, page 453]

"EXIT statement [Interactive SQL]" [*ASA SQL Reference*, page 478]

"HELP statement [Interactive SQL]" [*ASA SQL Reference*, page 517]

"INPUT statement [Interactive SQL]" [*ASA SQL Reference*, page 523]

"OUTPUT statement [Interactive SQL]" [*ASA SQL Reference*, page 556]

"PARAMETERS statement [Interactive SQL]" [*ASA SQL Reference*, page 561]

"READ statement [Interactive SQL]" [*ASA SQL Reference*, page 571]

"SET CONNECTION statement [Interactive SQL] [ESQL]" [*ASA SQL Reference*, page 610]

"SET OPTION statement [Interactive SQL]" [*ASA SQL Reference*, page 616]

"START DATABASE statement [Interactive SQL]" [*ASA SQL Reference*, page 623]

"START ENGINE statement [Interactive SQL]" [*ASA SQL Reference*, page 625]

"START LOGGING statement [Interactive SQL]" [*ASA SQL Reference*, page 627]

"STOP LOGGING statement [Interactive SQL]" [*ASA SQL Reference*, page 635]

"SYSTEM statement [Interactive SQL]" [*ASA SQL Reference*, page 640]

Starting Interactive SQL from Sybase Central

❖ To open Interactive SQL from Sybase Central

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Utilities tab.
3. Double-click Open Interactive SQL in the right pane.

   The Interactive SQL window appears.

Tip
You can also access Interactive SQL from within Sybase Central as follows:

- ♦ Choosing Tools ➤ Adaptive Server Anywhere 9 ➤ Open Interactive SQL.
- ♦ Selecting a database in the left pane, and choosing Open Interactive SQL from the File menu.
- ♦ Right-clicking a database, and choosing Open Interactive SQL from the popup menu.
- ♦ Right-clicking a stored procedure, and choosing Execute from Interactive SQL from the popup menu. Interactive SQL opens with a CALL to the procedure in the SQL Statements pane and executes the stored procedure.
- ♦ Right-clicking a table and choosing View Data in Interactive SQL. Interactive SQL opens with a `SELECT * FROM` *`table-name`* and executes the query.

Opening Interactive SQL using the dbisql command-line utility

Syntax **dbisql** [*options*] [*dbisql-command* | *command-file*]

Option	Description
-c *"keyword=value; ... "*	Supply database connection parameters.
-codepage *codepage*	Specify a codepage to use when reading or writing files.

Option	Description
-d *delimiter*	Use the given string as the command delimiter.
-d1	Print statements as they are executed [command-prompt mode only].
-datasource *dsn-name*	Specify an ODBC data source to connect to.
-f *filename*	Open (without running) the file called *filename*.
-host *hostname*	Specify the *hostname* or IP address of the machine running a database server.
-jConnect	Use jConnect to connect to the database.
-nogui	Run in command-prompt mode.
-ODBC	Use the iAnywhere JDBC Driver to connect to the database.
-onerror { **continue** | **exit** }	Override the ON_ERROR option for all users.
-port *portnumber*	Look on the specified port number for the database server.
-q	Quiet mode—no windows or messages (note that this does not suppress error messages).
-x	Syntax check only—no commands executed.

Description

Interactive SQL allows you to type SQL commands or run command files. It also provides feedback about the number of rows affected, the time required for each command, the execution plan of queries, and any error messages.

If *dbisql-command* is specified, Interactive SQL executes the command. You can also specify a command file name. If no *dbisql-command* or *command-file* argument is specified, Interactive SQL enters interactive mode, where you can type a command into a command window.

Exit codes are 0 (success) or non-zero (failure).

Interactive SQL requires that the QUOTED_IDENTIFIER database option be set to ON since a number of database functions, including some statements, rely on this setting to function properly. Interactive SQL automatically sets it ON when connecting to a database.

This utility does *not* accept the ***@data*** parameter to read in options from a configuration file.

Interactive SQL utility options

Connection parameters (-c) Specify connection parameters. See "Connection parameters" on page 176 for a description of the connection parameters. If Interactive SQL cannot connect, you are presented with a dialog where you can enter the connection parameters.

Codepage (-codepage) Specify the codepage to use when reading or writing files. The default code page is the default code page for the platform you are running on.

For example, on an English Windows NT machine, Interactive SQL uses the 1252 (ANSI) code page. If you want Interactive SQL to read files created using the 297 (IBM France) code page, specify the following option.

```
-codepage 297
```

☞ For a list of supported code pages, see "Supported code pages" on page 363.

Command delimiter (-d) Specify a command delimiter. Quotation marks around the delimiter are optional, but are required when the command shell itself interprets the delimiter in some special way.

Command delimiters are used for all connections in that Interactive SQL session, regardless of the setting stored in the database (for the user, or the PUBLIC setting).

Echo statements (-d1) (The final character is a number 1, not a lower-case L). Interactive SQL echoes all statements it executes to the Command window (STDOUT). This can provide useful feedback for debugging SQL scripts, or when Interactive SQL is processing a long SQL script.

Data source (-datasource) Specify an ODBC data source to connect to. You do not need to be using the iAnywhere JDBC driver to use this option.

Open file using Interactive SQL (-f) Open (but do not run) the file called *filename*. The file name can be enclosed in quotation marks, and *must* be enclosed in quotation marks if the file name contains a blank. If the file does not exist, or if it's really a directory instead of a file, Interactive SQL prints an error message to the console and then quits. If the file name does not include a full drive and path specification, it is assumed to be relative to the current directory.

Host (-host) Specify the hostname or IP address of the computer on which the database server is running. You can use the name **localhost** to represent the current machine.

Use jConnect (-jConnect) Use the Sybase jConnect JDBC driver to connect to the database.

Run in command-prompt mode (-nogui) Run Interactive SQL in a command-prompt mode, with no windowed user interface. This is useful for batch operations. If you specify either *dbisql-command* or *command-file*, then -nogui is assumed.

In this mode, Interactive SQL sets the program exit code to indicate success or failure. On Windows operating systems, the environment variable ERRORLEVEL is set to the program exit code. The exit codes are as follows:

Program exit code	Description
0	Success.
1	General failure. At some point, a SQL or Interactive SQL statement did not execute successfully and the user chose to stop executing SQL statements. Alternatively, Interactive SQL noted an internal error.
5	The user terminated interactive SQL. When an error occurs during execution, the user is prompted to ignore it, stop, or exit Interactive SQL. If the user opts to exit, the program returns code 5. Code 5 is also returned if an error occurs and the Interactive SQL option ON_ERROR is set to EXIT.
9	Unable to connect.
255	Bad command. The command contained incomplete or invalid options.

Use iAnywhere JDBC driver (-ODBC) Connect using the iAnywhere JDBC driver. This is the default method, and is recommended in most circumstances.

Override the ON_ERROR option setting (-onerror) Controls what happens if an error is encountered while reading statements from a command file. This option overrides the ON_ERROR setting. It is useful when using Interactive SQL in batch operations.

☞ For more information, see "ON_ERROR option [Interactive SQL]" on page 674.

Database server port (-port) Specify the port number on which the database server is running. The default port number for Adaptive Server Anywhere is **2638**.

Operate quietly (-q) Do not display output messages. This is useful only if you start Interactive SQL with a command or command file. Note that specifying this option does not suppress error messages.

Syntax check only (-x) Scan commands but do not execute them. This is useful for checking long command files for syntax errors.

☞ For detailed descriptions of SQL statements and Interactive SQL commands, see "SQL Language Elements" [*ASA SQL Reference*, page 3].

Examples

The following command, entered at a command prompt, runs the command file *mycom.sql* against the current default server, using the user ID DBA and the password SQL. If there is an error in the command file, the process terminates.

```
dbisql -c "uid=DBA;pwd=SQL" -onerror exit mycom.sql
```

The following command, when entered on a single line at a command prompt, adds a user to the current default database:

```
dbisql -c "uid=DBA;pwd=SQL" grant connect to joe identified by
         passwd
```

The Language utility

The Language utility reports and changes the registry settings that control the languages used by Adaptive Server Anywhere and Sybase Central.

Managing languages using the dblang command-line utility

Syntax

dblang [*options*] [*language-code*]

Option	Description
-q	Operate quietly—do not print messages.

Language code	Language
DE	German
EN	English
ES	Spanish
FR	French
IT	Italian
JA	Japanese
KO	Korean
LT	Lithuanian
PL	Polish
PT	Portuguese
RU	Russian
TW	Traditional Chinese
UK	Ukrainian
ZH	Simplified Chinese

Description

The utility is installed only when the International Resources Deployment Kit (IRDK) is selected during installation.

Running the dblang utility without a language code reports the current settings. These settings are as follows:

- **Adaptive Server Anywhere** A key from HKEY_LOCAL_MACHINE that holds a two-letter language code from the table above.

This setting controls which language resource library is used to deliver informational and error messages from the Adaptive Server Anywhere database server. The language resource library is a DLL with a name of the form *dblgXX9.dll*, where *XX* is a two-letter language code.

Ensure that you have the appropriate language resource library on your machine when you change the settings.

♦ **Sybase Central** A key from HKEY_LOCAL_MACHINE that holds a two-letter language code from the table above.

This setting controls the resources used to display user interface elements for Sybase Central and Interactive SQL. You must have purchased the appropriate localized version of SQL Anywhere Studio for this setting to take effect.

Exit codes are 0 (success) or non-zero (failure).

This utility does *not* accept the ***@data*** parameter to read in options from a configuration file.

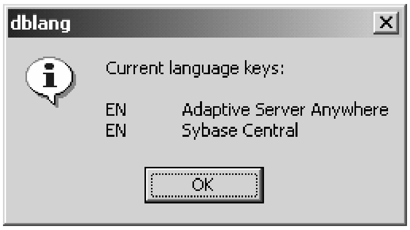
Language utility options

Operate quietly (-q) Do not display output messages.

Examples

If you selected the International Resources Deployment Kit (IRDK) during installation, the following command displays a dialog box containing the current settings:

```
dblang
```



```
Current language keys:

EN        Adaptive Server Anywhere
EN        Sybase Central
```

The following command changes the settings to French, and displays a dialog containing the previous and new settings:

```
dblang FR
```

The License utility

The License utility adds licensed users to your network database server.

Managing licenses using the dblic command-line utility

Syntax

dblic [*options*] *executable-name user-name company-name*

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-l *type*	Specify the license type. Allowed values are **perseat** or **processor**.
-o *filename*	Log output messages to a file.
-p *operating-system*	Specify the target operating system. Allowed values are **WIN32**, **WIN64**, **NetWare**, and **UNIX**.
-q	Operate quietly—do not print messages.
-u *license-number*	Specify the total number of users or processors for license.
executable-name	Specify the executable to be licensed, with path relative to the current directory. This should be either an Adaptive Server Anywhere network database server or a MobiLink synchronization server.
user-name	Specify the user name for the license.
company-name	Specify the company name for the license.

Description

The License utility adds licensed users to your network database server. You must only use this utility in accordance with your Sybase license agreement to license the number of users to which you are entitled. Running this command does *not* grant you license. You can also use this utility to view the current license information for a server without starting the server.

Repeated running of the dblic command updates the license information.

Exit codes are 0 (success) or non-zero (failure).

On UNIX, the dbsrv9 and dbmlsrv9 executables are not writable by default, and so using the license [dblic] utility with those executables will fail. Make sure the dbsrv9 and dbmlsrv9 executables are writable (for example, using `chmod +w`) before you use the License utility.

The License utility is not supported on NetWare. To license Adaptive Server Anywhere executables on NetWare, you must make sure that the Adaptive Server Anywhere database server on NetWare is not running, and then run the License utility from a Windows machine that can access the NetWare volume containing the Adaptive Server Anywhere software.

License utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

License type (-l) Enter the license type that matches the licensing model described in your software license agreement. Valid entries are **perseat** and **processor**.

Log output messages to file (-o) Write output messages to the named file.

Operating system (-p) Enter the operating system for which you are licensed. Allowed values are as follows:

- ♦ **NetWare** A Novell NetWare operating system.
- ♦ **UNIX** A UNIX operating system.
- ♦ **WIN32** A Windows 95/98/Me, Windows NT/2000/XP, or Windows CE operating system.
- ♦ **WIN64** A 64-bit Windows operating system.

Operate quietly (-q) Do not display output messages.

License number (-u) The total number of users or processors for the license. If you are adding extra licenses, this is the **total**, *not* the number of additional licenses.

Executable name Enter the path and file name of the network database server executable (*dbsrv9.exe*) or MobiLink synchronization server (*dbmlsrv9.exe*) you are licensing. The path must be either absolute or relative to the current working directory.

You can view the current license information for a server executable without starting the server by entering only the executable name.

User name A user name for the license. This name appears on the database server window on startup. If there are spaces in the name, enclose it in double quotes.

Company name The company name for the license. This name appears on the database server window on startup. If there are spaces in the name, enclose it in double quotes.

Example

The following command, executed in the same directory as the database server executable, applies a license for 50 users, in the name of **Sys Admin**, for company **My Co**, to a Windows NT network database server. The command must be entered all on one line:

```
dblic -l perseat -p WIN32 -u 50 dbsrv9.exe "Sys Admin" "My Co"
```

The following message appears on the screen to indicate the success of the license:

```
Licensed nodes: 50
User: Sys Admin
Company: My Co
```

The Log Transfer Manager

The Log Transfer Manager (LTM) is also known as a **replication agent**.

The LTM is required for any Adaptive Server Anywhere database that participates in a Replication Server installation as a primary site.

Using the dbltm command-line utility

Syntax

dbltm [*options*]

Parameters

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-A	Do not filter updates.
-C *config_file*	Use the named configuration file.
-I *interface_file*	Use the named interface file.
-M	Recovery mode.
-S *LTM_name*	Specify an LTM name.
-dl	Display log messages on screen.
-ek *key*	Specify encryption key.
-ep	Prompt for encryption key.
-o *filename*	Log output messages to a file.
-os *size*	Maximum size of output file.
-ot *file*	Truncate file, and log output messages to file.
-q	Run in minimized window.
-s	Show Log Transfer Language (LTL) commands.
-ud	Run as a daemon [UNIX].
-v	Verbose mode.

Description

The Adaptive Server Anywhere LTM reads a database transaction log and sends committed changes to Replication Server. The LTM is not required at replicate sites.

The LTM sends committed changes to Replication Server in a language named Log Transfer Language (LTL).

By default, the LTM uses a log file named *DBLTM.LOG* to hold status and other messages. You can use options to change the name of this file and to change the volume and type of messages that are sent to it.

Exit codes are 0 (success) or non-zero (failure).

Log Transfer Manager utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Do not filter updates (-A) Do not filter updates. By default, all changes made by the maintenance user are not replicated. If the -A option is set, these changes are replicated. This may be useful in non-hierarchical Replication Server installations, where a database acts as both a replicate site and as a primary site.

Use configuration file (-C) Use the configuration file *config_file* to determine the LTM settings. The default configuration file is *dbltm.cfg*.

☞ For a description of the configuration file, see "The LTM configuration file" on page 552.

Display Log Transfer Language messages (-dl) Display all messages in the LTM window or at a command prompt, as well as in the log file (if specified).

Specify encryption key (-ek) This option allows you to specify the encryption key for strongly encrypted databases directly in the command. If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log in any way, including offline transaction logs. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command fails if you do not specify a key for a strongly encrypted database.

Prompt for encryption key (-ep) This option allows you to specify that you want to be prompted for the encryption key. This option causes a dialog box to appear, in which you enter the encryption key. It provides an extra measure of security by never allowing the encryption key to be seen in clear text. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command fails if you do not specify a key for a strongly encrypted database.

Use interface file (-I) (Upper case I.) Use the named interfaces file. The interfaces file is the file created by SQLEDIT and holds the connection information for Open Servers. The default interfaces file is *SQL.ini* in the *ini* subdirectory of your *Sybase* directory.

Recover mode (-M) This is used to initiate recovery actions. The LTM starts reading logs from the earliest available position. If the offline directory is specified in the configuration file, the LTM reads from the oldest offline log file.

Log output messages to file (-o) Use a log file different from the default (*dbltm.log*). Write output messages from log transfer operations to this file.

Limit size of output file (-os) Specify the maximum size of the output file, in bytes. The minimum value is 10000 (ten thousand). If the log file grows to the point where it would exceed this limit, it is renamed to *yymmddxx.ltm*. The value of *xx* in *yymmddxx.ltm* is incremented for each file created on a given day.

Truncate log file and use named log file (-ot) Use a log file different from the default (*dbltm.log*), and truncate the log file (all existing content is deleted) when the LTM starts. Output messages from log transfer operations are sent to this file for later review.

Operate quietly (-q) Minimize the window when the LTM is started.

Specify LTM name (-S) Provides the server name for this LTM. The default LTM name is DBLTM_LTM. The LTM name must correspond to the Open Server name for the LTM that was entered in SQLEDIT.

Log all LTL commands (-s) Log all LTL commands that are generated by the LTM. This should be used only to diagnose problems, and is not recommended in a production environment. It carries a significant performance penalty.

Run as a daemon (-ud) You can run the LTM as a daemon on UNIX operating systems. If you run in this manner, output is logged to the log file.

Operate in verbose mode (-v) Displays messages, other than LTL messages, for debugging purposes.

The LTM configuration file

The Adaptive Server Anywhere and Adaptive Server Enterprise LTM configuration files are very similar. This section describes the entries in the Adaptive Server Anywhere LTM configuration file, and the differences from the Adaptive Server Enterprise LTM configuration file.

The configuration file that an LTM uses is specified using the -C option.

LTM configuration file parameters

The following table describes each of the configuration parameters that the LTM recognizes. Parameters that are used by the Adaptive Server Enterprise LTM but not by the Adaptive Server Anywhere LTM are included in this list, and marked as either ignored (in which case they may be present in the configuration file, but have no effect) or as unsupported (in which case they will cause an error if present in the configuration file).

Parameter	Description
APC_pw	The password for the APC_user login name. This entry is present only in Adaptive Server Anywhere LTM configuration files.
APC_user	A user ID that is used when executing asynchronous procedures at the primary site. This user ID must have permissions appropriate for all asynchronous procedures at the primary site. This entry is present only in Adaptive Server Anywhere LTM configuration files.
backup_only	By default, this is **off**. If set to **on**, the LTM will replicate only backed-up transactions.
batch_ltl_cmds	Set to **on** (the default) to use batch mode. Batch mode can increase overall throughput, but may lead to longer response times.
batch_ltl_sz	The number of commands that are saved in the buffer before being sent to Replication Server, when **batch_ltl_cmds** is **on**. The default is 200.
batch_ltl_mem	The amount of memory that the buffer can use before its contents are sent to Replication Server, when **batch_ltl_cmds** is **on**. The default is 256 Kb.
Continuous	By default, this is on. When set to off, the LTM automatically shuts down as soon as all committed data has been replicated.

Parameter	Description
LTM_admin_pw	The password for the LTM_admin_user login name.
LTM_admin_user	The system administrator LTM login name that is used to log in to the LTM. This parameter is required so that the LTM can check whether a user logging on to the LTM to shut it down has the correct login name.
LTM_charset	The Open Client/Open Server character set for the LTM to use.
LTM_language	The Open Client/Open Server language for the LTM to use.
LTM_sortorder	The Open Client/Open Server sort order for the LTM to use to compare user names. You can specify any Adaptive Server Enterprise-supported sort order that is compatible with the LTM's character set. All sort orders in your replication system should be the same. The default sort order is a binary sort.
maint_cmds_to_skip	[ignored]
qualify_table_owner	Set to **on** for the LTM to send LTLs with table names and columns names, as well as table owners to Replication Server. The setting applies to all replicating tables, and the create replication definition statements must match this setting. The default is **off**.
rep_func	Set to **on** to use asynchronous procedure calls (APCs). The default is **off**.
Retry	The number of seconds to wait before retrying a failed connection to an Adaptive Server Anywhere database server or Replication Server. The default is 10 seconds.
RS	The name of the Replication Server to which the LTM is transferring the log.
RS_pw	The password for the RS_user login name.

Parameter	Description
RS_source_db	The name of the database whose log the LTM transfers to the Replication Server. This name must match the name of the database as defined within the Replication Server connection definitions. Most configurations use the same setting for both RS_Source_db and SQL_database configuration options.
RS_source_ds	The name of the server whose log the LTM transfers to the Replication Server. This name must match the name of the server as defined within the Replication Server connection definitions. Most configurations use the same setting for both RS_-Source_ds and SQL_server configuration options.
RS_user	A login name for the LTM to use to log into Replication Server. The login name must have been granted **connect source** permission in the Replication Server.
scan_retry	The number of seconds that the LTM waits between scans of the transaction log. This parameter is somewhat different in meaning to that of the Adaptive Server Enterprise LTM. The Adaptive Server Anywhere server does not *wake up* and scan the log when records arrive in the log. For this reason, you may want to set the *scan_retry* value to a smaller number then that for an Adaptive Server Enterprise LTM.
skip_ltl_cmd_err	[ignored]
SQL_database	The primary site database name on the server SQL_server to which the LTM connects. For Adaptive Server Enterprise during recovery, this is the temporary database whose logs the LTM will transfer to Replication Server. The Adaptive Server Anywhere LTM uses the SQL_log_files parameter to locate offline transaction logs.
SQL_log_files	A directory that holds off-line transaction logs. The directory must exist when the LTM starts up. This entry is present only in Adaptive Server Anywhere LTM configuration files.

Parameter	Description
SQL_pw	The password for the SQL_user user ID.
SQL_server	The name of the primary site Adaptive Server Anywhere server to which the LTM connects. For Adaptive Server Enterprise during recovery, this is a data server with a temporary database whose logs the LTM will transfer to Replication Server. The LTM uses the SQL_log_files parameter to locate offline transaction logs.
SQL_user	The login name that the LTM uses to connect to the database specified by RS_source_ds and RS_source_db.

Example configuration file

♦ The following is a sample LTM configuration file.

```
# This is a comment line
# Names are case sensitive.
SQL_user=sa
SQL_pw=sysadmin
SQL_server=PRIMESV
SQL_database=primedb
RS_source_ds=PRIMEOS
RS_source_db=primedb
RS=MY_REPSERVER
RS_user=sa
RS_pw=sysadmin
LTM_admin_user=DBA
LTM_admin_pw=SQL
LTM_charset=cp850
scan_retry=2
SQL_log_files=e:\logs\backup
APC_user=sa
APC_pw=sysadmin
```

The Log Translation utility

With the Log Translation utility, you can translate a transaction log into a SQL command file.

You can access the Log Translation utility in the following ways:

- ♦ From Sybase Central, using the Translate Log File wizard.
- ♦ At a command prompt, using the dbtran command. This is useful for incorporating into batch or command files.

Translating a transaction log using the Translate Log File wizard

❖ To translate a transaction log into a command file

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Utilities tab.
3. In the right pane, double-click Translate Log File.

   The Translate Log File wizard appears.
4. Follow the instructions in the wizard.

Tip
You can also access the Translate Log File wizard from within Sybase Central using any of the following methods:

- ♦ Choosing Tools ➤ Adaptive Server Anywhere 9 ➤ Translate Log File.
- ♦ Selecting a database in the left pane, and choosing Translate Log File from the File menu.
- ♦ Right-clicking a database, and choosing Translate Log File from the popup menu.

Translating a transaction log using the dbtran command-line utility

Syntax

Running against a transaction log:

dbtran [*options*] [*transaction-log*] [*SQL-file*]

Running against a database server:

dbtran [*options*]

Option	**Description**
@*data*	Read options from the specified environment variable or configuration file.
-a	Include uncommitted transactions.
-c *"keyword=value; ... "*	Supply database connection parameters—cannot be used with a transaction log name.
-d	Display output in chronological order.
-ek *key*	Specify encryption key.
-ep	Prompt for encryption key.
-f	Output only since the last checkpoint.
-g	Include audit records in output.
-ir *offset1,offset2*	Include only the portion of the log between the two specified offsets.
-is *source,...*	Include only rows originating from the specified sources.
-it *user.table,...*	Include only operations on specified tables by specifying a comma-delimited list of *user.table*.
-j *date/time*	Output from the last checkpoint prior to the given time.
-m	Specify transaction logs directory (requires -n option).
-n *filename*	Output SQL file, when used against a database server.
-o *filename*	Log output messages to a file.
-q	Run quietly, do not print messages.
-r	Remove uncommitted transactions (default).
-rsu *username,...*	Override default Replication Server user names.
-s	Produce ANSI standard SQL UPDATE transactions.
-sr	Generate SQL Remote comments.

Option	Description
-t	Include trigger-generated transactions in output.
-u *userid,...*	Translate transactions for listed users only.
-x *userid,...*	Exclude transactions for listed users.
-y	Replace file without confirmation.
-z	Include trigger-generated transactions as comments only.
transaction-log	Log file to be translated.
SQL-file	Output file containing the translated information.

Description

The dbtran utility takes the information in a transaction log and places it as a set of SQL statements and comments into an output file. The utility can be run in the following ways:

- **Against a database server** Run in this way, the utility is a standard client application. It connects to the database server using the connection string specified following the -c option, and places output in a file specified with the -n option. DBA authority is required to run in this way.

  The following command translates log information from the server **asademo9** and places the output in a file named *asademo.SQL*.

  ```
  dbtran -c "eng=asademo9;dbn=asademo;uid=DBA;pwd=SQL" -n
           asademo.sql
  ```

- **Against a transaction log file** Run in this way, the utility acts directly against a transaction log file. You should protect your transaction log file from general access if you want to prevent users from having the capability of running this statement.

  ```
  dbtran asademo.log asademo.sql
  ```

When the dbtran utility runs, it displays the earliest log offset in the transaction log. This can be an effective method for determining the order in which multiple log files were generated.

If -c is used, dbtran attempts to translate the online transaction log file, as well as all the offline transaction log files in the same directory as the online transaction log file. If the directory contains transaction log files for more than one database, dbtran may give an error. To avoid this problem, ensure that each directory contains transaction log files for only one database.

A transaction can span multiple transaction logs. If transaction log files contain transactions that span logs, translating a single transaction log file (for example `dbtran asademo.log`) can cause the spanning transactions to be lost. In order for dbtran to generate complete transactions, use the -c or -m options with the transaction log files in the directory.

☞ For more information about recovering with transactions that span multiple log files, see "Recovering from multiple transaction logs" on page 419.

Exit codes are 0 (success) or non-zero (failure).

Log translation utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Include uncommitted transactions (-a) The transaction log contains any changes made before the most recent COMMIT by any transaction. Changes made after the most recent commit are not present in the transaction log.

If -a is not used, only committed transactions appear in the output file. If -a is used, any committed transactions found in the transaction log are output followed by a ROLLBACK statement.

Connection string (-c) When running the utility against a database server, this parameter specifies the connection string.

DBA authority is required to run dbtran.

☞ For a description of the connection parameters, see "Connection parameters" on page 176.

Output in chronological order (-d) Transactions are output in order from earliest to latest. This feature is provided primarily for use when auditing database activity: the output of this command should not be applied against a database.

Specify encryption key (-ek) This option allows you to specify the encryption key for strongly encrypted databases directly in the command. If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log in any way.

For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify the correct key for a strongly encrypted database.

If you are running against a database server (using the -c option), make sure you specify the key using a connection parameter, not using the -ek option. For example, the following command gets the transaction log information about database *enc.db* from server **sample**, and saves its output in *log.sql*.

```
dbtran -n log.sql -c
        eng=sample;dbf=enc.db;uid=dba;pwd=sql;dbkey=mykey
```

Prompt for encryption key (-ep) This option allows you to specify in the command that you want to be prompted for the encryption key. This option causes a dialog box to appear, in which you enter the encryption key. It provides an extra measure of security by never allowing the encryption key to be seen in clear text.

For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify the correct key for a strongly encrypted database.

If you are running against a database server (using the -c option), make sure you specify the key using a connection parameter, not using the -ep option. For example, the following command gets the transaction log information about database *enc.db* from server **sample**, and saves its output in *log.sql*.

```
dbtran -n log.sql -c
        eng=sample;dbf=enc.db;uid=dba;pwd=sql;dbkey=mykey
```

Output from last checkpoint only (-f) Only transactions that were completed since the last checkpoint are output.

Include audit information (-g) If the AUDITING database option is turned on, auditing information is added to the transaction log. You can include this information as comments in the output file using this option.

☞ For more information, see "AUDITING option [database]" on page 637.

The -g option implies the -a, -d, and -t options.

Include offset range (-ir) Output a portion of the transaction log between two specified offsets.

Include rows from specified sources (-is) Output operations on rows that have been modified by operations from one or more of the following sources, specified as a comma-separated list:

- **All** All rows. This is the default setting.

- ♦ **SQLRemote** Include only rows that were modified using SQL Remote. You can also use the short form **SR**.
- ♦ **RepServer** Include only rows that were modified using the Replication Agent (LTM) and Replication Server. You can also use the short form **RS**.
- ♦ **Local** Include only rows that are not replicated.

Include specified tables (-it) Output those operations on the specified, comma-separated list of tables. Each table should be specified as *owner.table*.

Output from the last checkpoint prior to a given date (-j) Only transactions from the most recent checkpoint prior to the given date and/or time are translated. The user-provided argument can be a date, time or date and time enclosed in quotes. If the time is omitted, the time is assumed to be the beginning of the day. If the date is omitted, the current day is assumed. The following is an acceptable format for the date and time: *"YYYY/MMM/DD HH*:*NN"*.

Transaction logs directory (-m) Use this option to specify a directory that contains transaction logs. This option must be used in conjunction with the -n option.

Output file (-n) When you run the dbtran utility against a database server, use this option to specify the output file that holds the SQL statements.

Log output messages to file (-o) Write output messages to the named file.

Operate quietly (-q) Do not display output messages. This option is available only when you run this utility from a command prompt. If you specify this option, you must also specify the -y option, or the operation will fail.

Do not include uncommitted transactions (-r) Remove any transactions that were not committed. This is the default behavior.

Override Replication Server user names (-rsu) By default, the -is option assumes the default Replication Server user names of dbmaint and sa. You can override this assumption using the -rsu option with a comma-separated list of user names.

Generate ANSI standard SQL UPDATE (-s) If the option is not used, and there is no primary key or unique index on a table, the Log Translation utility generates UPDATE statements with a non-standard FIRST keyword in case of duplicate rows. If the option is used, the FIRST keyword is omitted for compatibility with the SQL standard.

Generate SQL Remote comments (-sr) Place generated comments in the output file describing how SQL Remote distributes operations to remote sites.

Include transactions generated by triggers (-t) By default, actions carried out by triggers are not included in the command file. If the matching trigger is in place in the database, when the command file is run against the database, the trigger will carry out the actions automatically. Trigger actions should be included if the matching trigger does not exist in the database against which the command file is to be run.

Output transactions for listed users only (-u) This option allows you to limit the output from the transaction log to include only specified users.

Output transactions except for listed users (-x) This option allows you to limit the output from the transaction log to exclude specified users.

Operate without confirming actions (-y) Choosing this option automatically replaces existing command file(s) without prompting you for confirmation. If you specify -q, you must also specify -y or the operation will fail.

Include transactions generated by triggers as comments only (-z) Transactions that were generated by triggers will be included only as comments in the output file.

transaction-log Log file to be translated. Cannot be used together with -c or -m options.

SQL-file Output file containing the translated information. For use with *transaction-log* only.

The Ping utility

The Ping utility is provided to assist in troubleshooting connection problems.

Troubleshooting connections using the dbping command-line utility

Syntax

dbping [*options*]

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-c *"keyword=value; ..."*	Database connection parameters.
-d	Make a database connection if the server is found.
-l *library*	Load specified ODBC driver or library.
-m	Use ODBC driver manager.
-o *filename*	Log output messages to a file.
-pc *property,...*	Report specified connection properties.
-pd *property,...*	Report specified database properties.
-ps *property,...*	Report specified database server properties.
-q	Operate quietly—do not print messages.
-z	Display debugging information.

Description

The dbping utility is a tool to help debug connection problems. It takes a full or partial connection string and returns a message indicating whether the attempt to locate a server or database, or to connect, was successful.

The utility can be used for embedded SQL or ODBC connections. It cannot be used for jConnect (TDS) connections.

Exit codes are 0 (success) or non-zero (failure).

Ping utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Connection parameters (-c) For a description of the connection parameters, see "Connection parameters" on page 176. If the connection parameters are not specified, connection parameters from the SQLCONNECT environment variable are used, if set.

Make database connection (-d) Ping the database, not just the server.

If you do not supply the -d option, then dbping reports success if it finds the server specified by the -c option. If you do supply the -d option, then dbping reports success only if connects to the server and also connects to a database.

For example, if you have a server named **blair** running the database sample, the following succeeds:

```
dbping -c "ENG=blair;DBN=asademo"
```

The following command fails, with the message `Ping database failed - specified database not found`:

```
dbping -c "ENG=blair;DBN=asademo" -d
```

Load specified library (-l) Specify the library to use (without its file extension). This option avoids the use of the ODBC driver manager, and so is particularly useful on UNIX operating systems.

For example, the following command loads the ODBC driver directly:

```
dbping -m -c "dsn=ASA 9.0 Sample" -l dbodbc9
```

On UNIX, if you want to use a threaded connection library, you must use the threaded version of the Ping utility, dbping_r.

Use ODBC to connect (-m) Establish a connection using ODBC. By default, the utility connects using the embedded SQL interface.

Log output messages to file (-o) Write output messages to the named file.

Report connection properties (-pc) Upon connection, display the specified connection properties. Supply the properties in a comma-separated list. You must specify enough connection information to establish a database connection if you use this option.

☞ For a list of connection properties, see "Connection-level properties" on page 713.

For example, the following command displays the DIVIDE_BY_ZERO_ERROR option setting, which is available as a connection property.

```
dbping -c ... -pc Divide_by_zero_error
```

Report database properties (-pd) Upon connection, display the specified database properties. Supply the properties in a comma-separated list.

☞ For a list of database properties, see "Database-level properties" on page 733.

For example, the following command displays the Java version in use by the database:

```
dbping -c ... -pd JDKVersion
```

Report database server properties (-ps) Upon connection, display the specified database server properties. Supply the properties in a comma-separated list. You must specify enough connection information to establish a database connection if you use this option.

☞ For a list of database server properties, see "Server-level properties" on page 725.

For example, the following command displays the number of licensed seats or processors for the database server:

```
dbping -c ... -ps LicenseCount
```

Operate quietly (-q) If dbping fails, a message is always displayed. If dbping succeeds, no message appears if -q is specified.

Display debugging information (-z) This option is available only when an embedded SQL connection is being attempted. That is, it cannot be combined with -m or -l. It displays the network communication protocols used to attempt connection, and other diagnostic messages.

The Rebuild utility

Databases can be rebuilt using the Rebuild batch file, command file, or shell script, which invokes a series of utilities to rebuild a database, or as part of the unload process using the Unload Database wizard in Sybase Central.

☞ For more information about the Unload Database wizard, see "The Unload utility" on page 588.

Rebuilding a database using the rebuild batch or command file

Syntax

rebuild *old-database new-database* [*DBA-password*]

See also

- "Unloading a database using the dbunload command-line utility" on page 590
- "Creating a database using the dbinit command-line utility" on page 531
- "The Interactive SQL utility" on page 538

Description

This batch file, command file, or shell script uses dbunload to rebuild *old-database* into *new-database*. This is a simple script, but it helps document the rebuilding process, and provides a basis for customization. Both database names should be specified without extensions. An extension of *.db* is automatically added.

You can use dbunload with the -ar option to carry out unloading and reloading without using the rebuild batch file.

The *DBA-password* must be specified if the password to the DBA user ID in the *old-database* is not the initial password SQL.

Rebuild runs the dbunload, dbinit, and Interactive SQL commands with the default options. If you need different options, you will need to run the three commands separately.

Exit codes are 0 (success) or non-zero (failure).

This utility does *not* accept the ***@data*** parameter to read in options from a configuration file.

The Server Location utility

The Server Location utility is provided to assist in diagnosing connection problems by locating database servers on the immediate TCP/IP network.

Locating servers using the dblocate command-line utility

Syntax

dblocate [*options*] [*server-name*]

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-n	Do not resolve IP addresses into machine names.
-o *filename*	Log output messages to a file.
-q	Operate quietly—do not print messages.
server-name	A host name or IP address to restrict the list to servers running on the specified machine.

Description

The dblocate utility locates any Adaptive Server Anywhere database servers running over TCP/IP on the immediate network. It prints a list of database servers and their addresses.

Depending on your network, the utility may take several seconds before printing its results.

Exit codes are 0 (success) or non-zero (failure).

The database server can register itself with an LDAP server, which keeps track of all servers in an enterprise. This allows both clients and the Server Location utility (dblocate) to find them regardless of whether they are on a WAN or LAN, through firewalls, and without specifying an IP address. LDAP is only used with TCP/IP, and only on network servers.

☞ For more information about LDAP, see "Connecting using an LDAP server" on page 90.

If the same database server name is found more than once, the Server Location utility displays the IP address of each host, even if the -n option is not specified. The same server name could be found in cases where a server is running on a machine with multiple IP addresses (for example, if the machine has multiple network cards), or if a network server is running on a remote machine and a personal server with the same server name is running on the local machine.

Server location utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Do not resolve IP addresses into machine names (-n) List IP addresses in the output, rather than machine names. This may increase performance because looking up machine names may be slow.

Log output messages to file (-o) Write output messages to the named file.

Operate quietly (-q) Do not display output messages.

server-name List only database servers running on the machine with the specified IP address or host name. If you specify the -n option, the *server-name* value must be an IP address. If you do not specify -n, then you must supply a host name for the *server-name*. For example, the following command looks for servers on the machine **jfrancis**:

```
dblocate jfrancis
```

The Service Creation utility

The Service Creation utility is a tool used to create, delete, and modify Adaptive Server Anywhere services. You can access the Service Creation utility in the following ways:

- ♦ From Sybase Central, using the Service Creation wizard.
- ♦ At a command prompt, using the dbsvc command.

Creating services using the Service Creation wizard

❖ **To create a service**

1. In the left pane of Sybase Central, select the Adaptive Server Anywhere 9 plug-in.
2. In the right pane, click the Services tab.
3. From the File menu, choose New ➤ Service.

   The Service Creation wizard appears.

Managing services using the dbsvc command-line utility

Syntax

dbsvc [**-q**] [**-y**] **-d** <*svc*>

dbsvc [**-q**] **-g** <*svc*>

dbsvc [**-q**] **-l**

dbsvc [**-q**] [**-y**] <*creation options*> **-w** <*svc*> <*details*>

dbsvc [**-q**] **-u** <*svc*>

dbsvc [**-q**] **-x** <*svc*>

details:

<*full-executable-path*> [*options*]

Major option	Description
@*data*	Read options from the specified environment variable or configuration file.
-d *service_name*	Delete a service.
-g *service_name*	Get details of a service.
-l	List all Adaptive Server Anywhere services.
-u *service_name*	Starts the service named *service_name.*
-w *executable parameters*	Creates service.
-x *service_name*	Stops the service named *service_name.*

Creation option	Description
-a *acct*	Account name to use (used with -p).
-as	Use local system account.
-I	Allow service to interact with desktop.
-p	Password for account (used with -a).
-rg *dependency,. . .*	Specify group dependencies when creating a service.
-rs *dependency***,. . .**	Specify service dependencies when creating a service.
-s *startup*	Startup option (default manual)—you must specify automatic, manual, or disabled.
-sd *description*	Provide a description of the service.
-sn *name*	Specify a name for the service.
-t *type*	Specify the type of service.

Modifier option	Description
-cm	Display the service creation command.
-q	Do not print banner.
-y	Delete or overwrite service without confirmation.

Description

A service runs a database server or other application with a set of options.

This utility provides a comprehensive way of managing Adaptive Server Anywhere services on Windows. The Service Creation utility provides the same functionality as the Service Creation wizard in Sybase Central.

You must be a member of the Administrators group on the local machine to use the Service Creation utility.

Exit codes are 0 (success) or non-zero (failure).

☞ For more information about services, see "Understanding Windows services" on page 23.

Service creation utility options

When you use the Service Creation utility, you can choose from major options, modifier options, and details options.

Major options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Delete a service (-d) Removes the named service from the list of services. If you supply -y, any service is deleted without confirmation.

Get details of a service (-g) Lists the definition of the service, not including the password.

List all Adaptive Server Anywhere services (-l) Lists the available Adaptive Server Anywhere services.

Start a service (-u) Starts a service named *service_name*.

Create service (-w) Creates a new service, or overwrites one if one of the same name exists. If you supply -y, any existing service is overwritten without confirmation.

You must supply the full path to the executable that you want to use as a service, as the account under which the service is running may not have the appropriate SQL Anywhere directory in its path.

You must supply parameters appropriate for the service you are creating. For more information, see the following locations:

- **dbsrv9 and dbeng9** "The database server" on page 116.
- **dbmlsrv9** "MobiLink Synchronization Server Options" [*MobiLink Administration Guide,* page 189].
- **dbmlsync** "MobiLink synchronization client" [*MobiLink Clients,* page 96]
- **dbremote** "The Message Agent" [*SQL Remote User's Guide,* page 292].

Stop a service (-x) Stops a service named *service_name*.

Creation options

Account name (-a) All services run under a Windows account. If you run under an account you've created, you must name the account with the -a option and supply a password with the -p option.

Use local system account (-as) All services run under a Windows account. Using the -as option, the service will run under the Windows LocalSystem account. No password is required. On of -a or -as must be used.

Allow service to interact with desktop (-I) Displays an icon that you can double-click to display the server window.

Password for account (-p) Use this option with the -a option to specify the password for the account the service runs under.

Set group dependencies (-rg) At least one service from each of the groups in the list must be started before the service being created is allowed to start.

Set service dependencies (-rs) All the services in the list must have started before the service being created is allowed to start.

Startup option (-s) Sets startup behavior for Adaptive Server Anywhere services. You can set startup behavior to Automatic, Manual, or Disabled. The default is Manual.

Service description (-sd) Use this option to provide a description of the service. The description appears in the Windows Service Manager.

Service name (-sn) Use this option to provide a name for the service. This name appears in the Windows Service Manager. If you do not specify the -sn option, the default service name is **Adaptive Server Anywhere -** <***svc***>. For example, the following service is named Adaptive Server Anywhere - myserv by default.

```
dbsvc -as -w myserv
"C:\Program Files\Sybase\SQL Anywhere 9\win32\dbeng9.exe"
```

In order to have the service name myserv appear in the Windows Service Manager, you need to execute the following (entered all on one line):

```
dbsvc -as -sn myserv -w myserv
"C:\Program Files\Sybase\SQL Anywhere 9\win32\dbeng9.exe"
```

Type of service (-t) Specifies the type for this service. You can choose from the following types:

Type	Description
Network	Adaptive Server Anywhere network database server (dbsrv9)
Standalone	Adaptive Server Anywhere personal database server (dbeng9)
DBRemote	SQL Remote Message Agent (dbremote)
MobiLink	MobiLink synchronization server (dbmlsrv9)
DBMLSync	MobiLink synchronization client (dbmlsync)

The default setting is **Standalone**.

Modifier options

Display the service creation command (-cm) Displays the command used to create the service. This option can be used to output the creation command to a file, which can be used to add the service on another machine or can be used to restore a service to its original state if changes have been made to it. You must specify the -g option or -l option with -cm or the command fails. Specifying -g displays the creation command for the specified service, while specifying -l displays the creation command for all services.

If the specified service does not exist, the command to delete the service is generated. For example, if service_1 does not exist on the machine, `dbsvc -cm -g service_1` would return the following command to delete the service_1 service:

```
dbsvc -y -d "service_1"
```

If the service does not use the LocalSystem account, there is no way to retrieve the password, so it is not included in the command that is generated. If you created the service with `-a` *`user`* `-p` *`password`*, only `-a` *`user`* is included in the output.

Do not print banner (-q) Suppress the informational banner. If you specify this option when modifying or deleting an existing service, you must also specify -y or the operation will fail.

Delete or overwrite service without confirmation (-y) Automatically carries out the action without prompting for confirmation. This option can be used with the -w or -d options. If you specify -q when modifying or deleting an existing service, you must also specify -y or the operation will fail.

Examples

Create a personal server service called **myserv**, which starts the specified server with the specified parameters. The server runs as the **LocalSystem** user:

```
dbsvc -as -w myserv "C:\Program Files\Sybase\SQL Anywhere 9\
         win32\dbeng9.exe" -n myeng -c 8m "C:\Program Files\
         Sybase\SQL Anywhere 9\sample.db"
```

Create a network server service called **mynetworkserv**. The server runs under the local account, and starts automatically when the machine is booted:

```
dbsvc -as -s auto -t network -w mynetworkserv "C:\Program Files\
         Sybase\SQL Anywhere 9\win32\dbsrv9.exe" -x tcpip -c 8m
         "C:\Program Files\Sybase\SQL Anywhere 9\sample.db"
```

List all details about service **myserv**:

```
dbsvc -g myserv
```

Delete the service called **myserv**, without prompting for confirmation:

```
dbsvc -y -d myserv
```

Create a service dependent on the Workstation service and the TDI group:

```
dbsvc -rs Workstation -rg TDI -w ...
```

Create a service called **mysyncservice**:

```
dbsvc -as -s manual -t dbmlsync -w mysyncservice "C:\Program
         Files\Sybase\SQL Anywhere 9\win32\dbmlsync.exe" -c
         "dsn=ultralite 9.0 sample"
```

Generate the command to create the service_1 service and output it to a file called *restoreservice.bat*:

```
dbsvc -cm -g service_1 > restoreservice.bat
```

The *restoreservice.bat* file contains the following:

```
dbsvc -t Standalone -s Manual -as -y -w "service_1"
"C:\Program Files\Sybase\SQL Anywhere 9\win32\dbeng9.exe"
```

The Spawn utility

This utility is provided to start a database server in the background.

Running a server in the background using the dbspawn command-line utility

Syntax

dbspawn [*options*] *server-command*

Option	Description
@data	Read options from the specified environment variable or configuration file.
-f	Do not check for a running server.
-p	Report operating system process ID.
-q	Quiet mode—do not print messages.
server-command	Specifies the command line for starting the database server.

Description

The dbspawn utility is provided to start a server in the background. dbspawn starts the server in the background and returns with an exit code of 0 (success) or non-zero (failure). If the server specified in *server-command* is already running, dbspawn reports failure.

The dbspawn utility is useful for starting a server from a batch file, especially when subsequent commands in the batch file require a server that is accepting requests.

If the specified path includes at least one space, and if you are executing the spawn utility from a batch file, you must enclose the path in one set of double quotes. For example,

```
dbspawn dbeng9 "C:\Program Files\Sybase\SQL Anywhere 9\
         asademo.db"
```

If the specified path includes at least one space, and if you are executing the spawn utility at a command prompt, you must provide an extra set of quotes, and escape the quotes around the database file. For example,

```
dbspawn dbeng9 "\"C:\Program Files\Sybase\SQL Anywhere 9\
         asademo.db\""
```

If the specified path does not contain spaces, then quotes are not required.

Spawn utility options

@data   Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see “Using configuration files” on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see “Hiding the contents of files using the dbfhide command-line utility” on page 524.

Do not check for a running server (-f) If a database server is already running, the dbspawn command may use it. This option forces dbspawn to start a new database server each time it is run.

Report process ID (-p) The operating system process ID of the database server process. For example:

```
dbspawn -p dbeng9 -n newserver
```

reports a message of the following form to a command prompt:

```
New process id is 306
```

Operate quietly (-q) Do not display output messages.

server-command Specifies the command line for starting the database server.

☞ For a description of the server commands, see “The database server” on page 116.

The Stop utility

The Stop utility stops a database server. You can use the -d option to stop a specified database.

The Stop utility can only be run at a command prompt. In windowed environments, you can stop a database server by clicking Shutdown on the Server Messages window.

The Stop utility (dbstop) is not available on NetWare.

Stopping a database server using the dbstop command-line utility

Syntax

dbstop [*options*] [*server-name*]

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-c *"keyword=value; ... "*	Connection parameters.
-d	Stop specified database only.
-o *filename*	Log output messages to a file.
-q	Quiet mode—do not print messages.
-x	Do not stop if there are active connections.
-y	Stop without prompting even if there are active connections.
server-name	Name of a local database server to stop.

Description

Options let you control whether a server is stopped, even if there are active connections, and whether to stop a server or only a database.

Exit codes are 0 (success) or non-zero (failure).

The Stop [dbstop] utility is not available on NetWare.

Stop utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Connection parameters (-c) When stopping a network server, you must supply a connection string with a user ID that has permissions to stop the server. By default, DBA permission is required on the network server, and all users can shut down a personal server, but the -gk server option can be used to change this.

The behavior of dbstop can be controlled if there are active connections on a server. If there are active connections, dbstop provides a prompt asking if you want to shut down the server. If you specify **unconditional=YES**, the server is shut down without prompting, even if there are active connections.

If you supply connection parameters, do not supply a server name as well.

☞ For more information, see "Connection parameters" on page 176, "Unconditional connection parameter [UNC]" on page 204, and "-gk server option" on page 144.

Stop database only (-d) Do not stop the database server. Instead, only stop the database specified in the connection string.

Log output messages to file (-o) Write output messages to the named file.

Operate quietly (-q) Do not print a message if the database was not running.

Do not stop if there are active connections (-x) Do not stop the server if there are still active connections to the server.

Stop without prompting (-y) Stop the server even if there are still active connections to the server.

server-name The name of a database server running on the current machine. The database server must be started so that no permissions are required to shut it down. The personal database server starts in this mode by default. For the network database server, you must supply the **-gk all** option.

If you supply a server name, do not supply connection parameters as well.

☞ For more information, see "-gk server option" on page 144.

Examples

You are running the server named **myserver** without a database. To stop the server, specify the utility database as a **DatabaseName (DBN)** connection parameter:

```
dbstop -c "uid=DBA;pwd=SQL;eng=myserver;dbn=utility_db"
```

You are running the server named **myserver** with the database **asademo.db** started. To stop the server and database:

```
dbstop -c "uid=DBA;pwd=SQL;eng=myserver"
```

The Transaction Log utility

With the Transaction Log utility, you can display or change the name of the transaction log or transaction log mirror associated with a database. You can also stop a database from maintaining a transaction log or mirror, or start maintaining a transaction log or mirror.

A transaction log mirror is a duplicate copy of a transaction log, maintained by the database in tandem.

The name of the transaction log is first set when the database is initialized. The Transaction Log utility works with database files. The database server must not be running on that database when the transaction log filename is changed (or an error message appears).

You can access the Transaction Log utility in the following ways:

- From Sybase Central, using the Change Log File Settings wizard.
- From Interactive SQL, using the ALTER DATABASE *dbfile* MODIFY LOG statement.

  ☞ For more information, see "ALTER DATABASE statement" [*ASA SQL Reference*, page 266].
- At a command prompt, using the dblog command.

Managing log files using the Change Log File Settings wizard

❖ **To change a transaction log file name**

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Utilities tab.
3. In the right pane, double-click Change Log File Settings.

   The Change Log File Settings wizard appears.
4. Follow the instructions in the wizard.

Tip
You can also access the Change Log File Settings wizard by choosing Tools ➤ Adaptive Server Anywhere 9 ➤ Change Log File Settings.

Managing log files using the dblog command-line utility

Syntax **dblog** [*options*] *database-file*

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-ek *key*	Specify encryption key.
-ep	Prompt for encryption key.
-g *n*	Sets the LTM generation number to *n*.
-il	Ignores the LTM truncation offset stored in the database.
-ir	Ignores the SQL Remote truncation offset stored in the database.
-is	Ignores the dbmlsync truncation offset stored in the database.
-m *mirror-name*	Set transaction log mirror name.
-n	No longer use a transaction log or mirror log.
-o *file-name*	Log output messages to a file.
-q	Quiet mode—do not print messages.
-r	No longer use a transaction log mirror.
-t *log-name*	Set the transaction log name.
-x *n*	Set the transaction log current relative offset to *n*.
-z *n*	Set the transaction log starting offset to *n*.

Description

The dblog utility allows you to display or change the name of the transaction log or transaction log mirror associated with a database. You can also stop a database from maintaining a transaction log or mirror, or start maintaining a transaction log or mirror.

The utility displays additional information about the transaction log, including the following:

- ♦ Version number
- ♦ Starting offset, for use in replication
- ♦ Ending offset, for use in replication
- ♦ Page size
- ♦ Total number of pages

- Number of empty pages
- Percentage of the log file in use

Exit codes are 0 (success) or non-zero (failure).

Transaction log utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Specify encryption key (-ek) This option allows you to specify the encryption key for strongly encrypted databases directly in the command. If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log in any way. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify the correct key for a strongly encrypted database.

Prompt for encryption key (-ep) This option allows you to specify that you want to be prompted for the encryption key. This option causes a dialog box to appear, in which you enter the encryption key. It provides an extra measure of security by never allowing the encryption key to be seen in clear text. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify the correct key for a strongly encrypted database.

Set the generation number (-g) Use this option if you are using the Log Transfer Manager to participate in a Replication Server installation. It can be used after a backup is restored, to set the generation number. It performs the same function as the following Replication Server function:

```
dbcc settrunc( 'ltm', 'gen_id', n )
```

☞ For information about generation numbers and dbcc, see your Replication Server documentation.

Ignore the LTM truncation offset (-il) This option can be used if you have stopped using the Log Transfer Manager to participate in a Replication Server installation on this database, but continue to use SQL Remote or

MobiLink synchronization. It resets the Log Transfer Manager log offset that is kept for the DELETE_OLD_LOGS option, allowing transaction logs to be deleted when they are no longer needed.

It performs the same function as the following Replication Server function:

```
dbcc settrunc( 'ltm', 'ignore' )
```

☞ For information about dbcc, see your Replication Server documentation.

Ignore the SQL Remote truncation offset (-ir) This option can be used if you have stopped using SQL Remote on this database, but continue to use the Log Transfer Manager or MobiLink synchronization. It resets the SQL Remote log offset that is kept for the DELETE_OLD_LOGS option, allowing transaction logs to be deleted when they are no longer needed.

Ignore the dbmlsync truncation offset (-is) This option can be used if you have stopped using MobiLink synchronization on this database, but continue to use the Log Transfer Manager or SQL Remote. It resets the MobiLink log offset that is kept for the DELETE_OLD_LOGS option, allowing transaction logs to be deleted when they are no longer needed.

Set the name of the transaction log mirror file (-m) This option sets a filename for a new transaction log mirror. If the database is not currently using a transaction log mirror, it starts using one. If the database is already using a transaction log mirror, it changes to using the new file as its transaction log mirror.

No longer use a transaction log (-n) Stop using a transaction log, and stop using a mirror log. Without a transaction log, the database can no longer participate in data replication or use the transaction log in data recovery. If a SQL Remote, Log Transfer Manager, or dbmlsync truncation offset exists, the transaction log cannot be removed unless the corresponding ignore option (-il for the Log Transfer Manager, -ir for SQL Remote, or -is for dbmlsync) is also specified.

Log output messages to file (-o) Write output messages to the named file.

Operate quietly (-q) Do not display output messages.

No longer use a transaction log mirror (-r) For databases that maintain a mirrored transaction log, this option changes their behavior to maintain only a single transaction log.

Set the name of the transaction log file (-t) This option sets a filename for a new transaction log. If the database is not currently using a transaction log, it starts using one. If the database is already using a transaction log, it changes to using the new file as its transaction log.

Set the current log offset (-x) For use when reloading a SQL Remote consolidated database. This option resets the current log offset so that the database can take part in replication.

☞ For information on how to use this option, see "Unloading and reloading a database participating in replication" [*SQL Remote User's Guide*, page 258].

Set the starting log offset (-z) For use when reloading a SQL Remote consolidated database. This option resets the starting log offset so that the database can take part in replication.

☞ For information on how to use this option, see "Unloading and reloading a database participating in replication" [*SQL Remote User's Guide*, page 258].

The Uncompression utility (deprecated)

With the Uncompression utility, you can expand a compressed database file created by the Compression utility. The Uncompression utility reads the compressed file and restores the database file to its uncompressed state.

You can access the Uncompression utility in the following ways:

- ♦ From Sybase Central, using the Uncompress Database wizard.
- ♦ At a command prompt, using the dbexpand command. This is useful for incorporating into batch or command files.

Deprecated feature
The use of compressed databases is deprecated.

Uncompressing a database using the Uncompress Database wizard

❖ **To uncompress a compressed database file**

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Utilities tab.
3. In the right pane, double-click Uncompress Database.

   The Uncompress Database wizard appears.
4. Follow the instructions in the wizard.

Tip
You can also access the Uncompress Database wizard by choosing Tools ➤ Adaptive Server Anywhere 9 ➤ Uncompress Database.

Uncompressing a database using the dbexpand command-line utility

Syntax **dbexpand** [*options*] *compressed-database-file* [*database-file*]

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-ek *key*	Specify encryption key.
-ep	Prompt for encryption key.
-o *filename*	Log output messages to a file.
-q	Operate quietly—do not print messages.
-y	Erase an existing output file without confirmation.

Description

The input filename extension defaults to *cdb*. The output filename (with extension) must not have the same name as the input filename (with extension).

Exit codes are 0 (success) or non-zero (failure).

Uncompression utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Specify encryption key (-ek) This option allows you to specify the encryption key for strongly encrypted databases directly in the command. If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log in any way. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify a key for a strongly encrypted database.

Prompt for encryption key (-ep) This option allows you to specify that you want to be prompted for the encryption key. This option causes a dialog box to appear, in which you enter the encryption key. It provides an extra measure of security by never allowing the encryption key to be seen in clear text. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify a key for a strongly encrypted database.

Log output messages to file (-o) Write output messages to the named file.

Operate quietly (-q) Do not display output messages. This option is available only when you run this utility from a command prompt.

Operate without confirming actions (-y) Choosing this option automatically replaces an existing database file without prompting you for confirmation.

The Unload utility

With the Unload utility, you can unload a database and put a set of data files in a named directory. The Unload utility creates an Interactive SQL command file to rebuild your database. It also unloads all of the data in each of your tables into files in the specified directory in comma-delimited format. Binary data is properly represented with escape sequences.

You can also use the Unload utility to directly create a new database from an existing one. This avoids potential security problems with the database contents being written to ordinary disk files.

If you only want to unload table data, you can do so in one step using the Unload Data dialog in Sybase Central.

☞ For more information, see "Using the Unload Data dialog" [*ASA SQL User's Guide*, page 573].

Accessing the Unload utility

You can access the Unload utility in the following ways:

- From Sybase Central, using the Unload Database wizard.
- At a command prompt, using the dbunload command. This is useful for incorporating into batch or command files.

The Unload utility should be run from a user ID with DBA authority. This is the only way you can be sure of having the necessary privileges to unload all the data. In addition, the *reload.sql* file should be run from the DBA user ID. (Usually, it is run on a new database where the only user ID is **DBA** with password **SQL**.)

The database server -gl option controls the permissions required to unload data from the database.

☞ For information, see "-gl server option" on page 144.

Objects owned by dbo

The **dbo** user ID owns a set of Adaptive Server Enterprise-compatible system objects in a database.

The Unload utility does not unload the objects that were created for the **dbo** user ID during database creation. Changes made to these objects, such as redefining a system procedure, are lost when the data is unloaded. Any objects that were created by the **dbo** user ID since the initialization of the database are unloaded by the Unload utility, and so these objects are preserved.

Unloading and replication

There are special considerations for unloading databases involved in replication.

☞ For information, see "Unloading and reloading a database participating in replication" [*SQL Remote User's Guide,* page 258] and "Upgrading the database file format" [*What's New in SQL Anywhere Studio,* page 230].

Unloading a database using the Unload Database wizard

The Unload Database wizard walks you through the steps for unloading a database in Sybase Central. When using this wizard to unload your database, you can choose to unload all the tables in a database, or only a subset of tables from the database. Only tables for users selected in the Filter Objects by Owner dialog appear in the Unload Database wizard. If you want to view tables belonging to a particular database user, right-click the database you are unloading, choose Filter Objects by Owner from the popup menu, and then select the desired user in the resulting dialog.

The Unload Database wizard also gives you the option to reload into an existing database, rather than into a reload file. In order to do this, you must be connected to both the database you are unloading and the database you are reloading into in Sybase Central.

❖ To unload a database file or a running database

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Utilities tab.
3. In the right pane, double-click Unload Database.

   The Unload Database wizard appears.
4. Follow the instructions in the wizard.

Tip
You can also access the Unload Database wizard from within Sybase Central using any of the following methods:

♦ Choosing Tools ➤ Adaptive Server Anywhere 9 ➤ Unload Database.

♦ Selecting a database in the left pane, and choosing Unload Database from the File menu.

♦ Right-clicking a database, and choosing Unload Database from the popup menu.

☞ For full information on unloading a database from Sybase Central, see "Exporting databases" [*ASA SQL User's Guide,* page 576].

Unloading a database using the dbunload command-line utility

Syntax **dbunload** [*options*] [*directory*]

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-ac *"keyword=value; . . . "*	Supply connection parameters for the reload.
-an *database*	Create a database file with the same settings as the database being unloaded, and automatically reloads it.
-ap *size*	Specify the page size of the new database (-an or -ar must also be specified).
-ar [*directory*]	Rebuild and replace database.
-c *"keyword=value; . . . "*	Supply database connection parameters for unload.
-d	Unload data only.
-e *table*, . . .	Do not unload listed tables.
-ea *algorithm*	Specify which strong encryption algorithm to encrypt your database with: you can choose AES or AES_FIPS.
-ek *key*	Specify encryption key for new database.
-ep	Prompt for encryption key for new database.
-ii	Internal unload, internal reload (default).
-ix	Internal unload, external reload.
-jr	Ignore Java when unloading or reloading.
-m	Do not preserve user ids for a replicating database.
-n	No data—schema definition only.
-o *filename*	Log output messages to a file.

Option	Description
-p *char*	Specify the escape character for external unloads (default “\”).
-q	Quiet mode—no windows or messages.
-r *reload-file*	Specify the name and directory of generated reload Interactive SQL command file (default *reload.sql*).
-t *table,...*	Unload only the listed tables.
-u	Unordered data. Do not use indexes to unload data.
-v	Verbose messages.
-xi	External unload, internal reload.
-xx	External unload, external reload.
-y	Replace the command file without confirmation.

Description

The *directory* is the destination directory where the unloaded data is to be placed. The *reload.sql* command file is always relative to the current directory of the user.

In the default mode, or if -ii or -ix is used, the directory used by dbunload to hold the data is relative to the database server, not to the current directory of the user.

☞ For details of how to supply a filename and path in this mode, see “UNLOAD TABLE statement” [*ASA SQL Reference*, page 648].

If -xi or -xx is used, the directory is relative to the current directory of the user.

If no list of tables is supplied, the whole database is unloaded. If a list of tables is supplied, only those tables are unloaded.

Unloaded data includes the column list for the LOAD TABLE statements generated in the *reload.sql* file. Unloading the column list facilitates reordering of the columns in a table. Tables can be dropped or recreated, and then repopulated using *reload.sql*.

Exit codes are 0 (success) or non-zero (failure).

There are special considerations for unloading databases involved in replication. For information, see "Unloading and reloading a database participating in replication" [*SQL Remote User's Guide,* page 258].

Password case sensitivity

When a database is unloaded using -an or -ar, the password case sensitivity setting is preserved. For example, if you unloaded a database with case sensitive passwords and specified -an or -ar, the newly-created database would also have case-sensitive passwords. When you do not specify -an or -ar, password case sensitivity is determined as follows:

- ♦ if the password was originally entered into a database that used case sensitive passwords, then the password remains case sensitive, even when reloaded into a case-insensitive database.
- ♦ if the password was originally entered into a database with case insensitive passwords, then the lowercase version of the password must be used if the password is reloaded into a case sensitive database.

Encrypted databases

If you want to unload a strongly encrypted database, you must provide the encryption key. You can use the DatabaseKey (DBKEY) connection parameter to provide the key in the command. Alternatively, if you want to be prompted for the encryption key rather than entering it in plain view, you can use the -ep server option as follows:

```
dbunload -c "dbf=enc.db;start=dbeng9 -ep"
```

If you are using **dbunload -an** to unload a database and reload into a new one, and you want to use the -ek or -ep options to set the encryption key for the new database, keep the following in mind:

- ♦ If the original database is strongly encrypted, you need to specify the key for the original database using the DatabaseKey (DBKEY) connection parameter in the -c option, not using -ek or -ep.
- ♦ Using the -ek and -ep options, it is possible to unload an unencrypted database and reload into a new, strongly encrypted database. When you use -ep and -an, you must confirm the key correctly or the unload fails.
- ♦ If the original database is strongly encrypted, but the -ek and -ep options are not used, then the new database will be encrypted with simple encryption.
- ♦ The -ek and -ep options are ignored if -an is not specified.

☞ For more information about encryption, see "-ep server option" on page 138 or the "DatabaseKey connection parameter [DBKEY]" on page 187.

Rebuilding a database

To unload a database, start the database server with your database, and run the Unload utility with the DBA user ID and password.

To reload a database, create a new database and then run the generated *reload.sql* command file through Interactive SQL.

On Windows 95/98/Me, Windows NT/2000/XP, and UNIX, there is a file (*rebuild.bat*, *rebuild.cmd*, or *rebuild*) that automates the unload and reload process.

☞ For more information, see "The Rebuild utility" on page 566.

Unload utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Connection parameters for reload database (-ac) This option causes the Unload utility to connect to an existing database and reload the data directly into it. You can combine the operation of unloading a database and reloading the results into an existing database using this option.

Typically, you would create a new database using the Initialization utility, and then reload it using this option. This method is useful when you want to change initialization options, such as page size or collation. If you are changing collations, the -xx option should also be used to ensure that character set translation occurs properly for both the old and new databases.

For example, the following command (which should be entered all on one line) loads a copy of the *asademo.db* database into an existing database file named *newdemo.db*:

```
dbunload -c "uid=DBA;pwd=SQL;dbf=asademo.db" -ac
         "uid=DBA;pwd=SQL;dbf=newdemo.db"
```

If you use this option, no interim copy of the data is created on disk, so you do not specify an unload directory in the command. This provides greater security for your data.

Create a database for reloading (-an) You can combine the operations of unloading a database, creating a new database, and loading the data using this option. This option applies to personal server connections, and network server connections over shared memory. When you specify -an, the case sensitivity setting for passwords is preserved.

Typically, you would use this option when you do not want to change the initialization options of your database. The options specified when you created the source database are used to create the new database.

For example, the following command (which should be entered all on one line) creates a new database file named *asacopy.db* and copies the schema and data of *asademo.db* into it:

```
dbunload -c "uid=DBA;pwd=SQL;dbf=asademo.db" -an asacopy.db
```

If you use this option, no interim copy of the data is created on disk, so you do not specify an unload directory in the command. This provides greater security for your data, but at some cost for performance.

You do not need to specify a directory for this option.

When the new database is created, the dbspace file names have an **R** appended to the file name to prevent file name conflicts if the dbspace file for the new database is created in the same directory as the dbspace for the original database. For example, if an unloaded database has a dbspace called library in the file *library.db*, then the library dbspace for the new database is *library.dbR*.

Set page size for the new database (-ap) This option allows you to set the page size of the new database. The page size for a database can be (in bytes) 1024, 2048, 4096, 8192, 16384, or 32768, with 2048 being the default. You must specify either -an or -ar with this option. If there are already databases running on the database server, the server's page size (set with the -gp option) must be large enough to handle the new page size.

☞ For more information, see "-gp server option" on page 146.

Rebuild and replace database (-ar) This option creates a new database with the same settings as the old database, reloads it, and replaces the old database. If you use this option, there can be no other connections to the database, and the database connection must be local, not over a network. When you specify -an, the case sensitivity setting for passwords is preserved.

If you specify an optional *directory*, the transaction log offsets are reset for replication purposes, and the transaction log from the old database is moved to the specified directory. The named directory should be the directory that holds the old transaction logs used by the Message Agent and the Replication Agent. The transaction log management is handled only if the database is used in replication: if there is no SQL Remote publisher or LTM check, then the old transaction log is not needed and is deleted instead of being copied to the specified directory.

☞ For more information on transaction log management, see "Backup methods for remote databases in replication installations" on page 387.

Connection parameters for source database (-c) For a description of the connection parameters, see "Connection parameters" on page 176. The user ID should have DBA authority, to ensure that the user has permissions on all the tables in the database.

For example, the following statement unloads the **asademo** database running on the **sample_server** server, connecting as user ID **DBA** with password **SQL**. The data is unloaded into the *c:\unload* directory.

```
dbunload -c "eng=sample_server;dbn=asademo;uid=DBA;pwd=SQL" c:\
         unload
```

Unload data only (-d) With this option, none of the database definition commands are generated (CREATE TABLE, CREATE INDEX, and so on); *reload.sql* contains statements to reload the data only.

No data output for listed tables (-e) This option is accessible only when you run this utility at a command prompt. If you want to unload almost all of the tables in the database, the -e option unloads all tables except the specified tables. A *reload.sql* file created with the -e option should not be used to rebuild a database because the file will not include all the database tables.

Specify encryption algorithm (-ea) This option allows you to choose which strong encryption algorithm used to encrypt your new database. You can choose either AES (the default) or AES_FIPS for the FIPS-approved algorithm. AES_FIPS uses a separate library and is not compatible with AES. Algorithm names are case insensitive. If you specify the -ea option, you must also specify -ep or -ek. If you specify -ea without specifying -an, the -ea option is ignored.

☞ For more information about strong database encryption, see "Strong encryption" [*SQL Anywhere Studio Security Guide*, page 15].

Separately licensable option required
Strong database encryption using AES_FIPS requires that you obtain the separately-licensable SQL Anywhere Studio security option and is subject to export regulations.

☞ To order this component, see "Separately-licensable components" [*Introducing SQL Anywhere Studio*, page 5].

Specify encryption key (-ek) This option allows you to specify an encryption key for the new database created if you unload and reload a database (using the -an option). If you create a strongly encrypted database, you must provide the encryption key to use the database or transaction log in any way. The algorithm used to encrypt the database is AES or AES_FIPS as specified by the -ea option. If you specify the -ek option without specifying -ea, the algorithm used will be AES. If you specify -ek without specifying -an, the -ek option is ignored.

Caution
Protect your key! Be sure to store a copy of your key in a safe location. A lost key will result in a completely inaccessible database, from which there is no recovery.

☞ For more information about strong database encryption, see "Strong encryption" [*SQL Anywhere Studio Security Guide,* page 15].

Prompt for encryption key (-ep) This option prompts you to specify an encryption key for the new database created if you unload and reload your database (using the -an option). It provides an extra measure of security by never allowing the encryption key to be seen in clear text. If you specify -ep without specifying -an, the -ep option is ignored. If you specify -ep and -an, you must input the encryption key twice to confirm that it was entered correctly. If the keys don't match, the unload fails.

☞ For more information about strong database encryption, see "Strong encryption" [*SQL Anywhere Studio Security Guide,* page 15].

Internal versus external unloads and reloads
The following options offer combinations of internal and external unloads and reloads: -ii, -ix, -xi, and -xx. A significant performance gain can be realized using internal commands (UNLOAD/LOAD) versus external commands (Interactive SQL's INPUT and OUTPUT statements). However, internal commands are executed by the server so that file and directory paths are relative to the location of the database server. Using external commands, file and directory paths are relative to the current directory of the user.

In Sybase Central, you can specify whether to unload relative to the server or client.

☞ For more information on filenames and paths for the Unload utility, see "UNLOAD TABLE statement" [*ASA SQL Reference,* page 648].

Use internal unload, internal reload (-ii) This option uses the UNLOAD statement to extract data from the database, and uses the LOAD statement in the *reload.sql* file to repopulate the database with data. This is the default.

Use internal unload, external reload (-ix) This option uses the UNLOAD statement to extract data from the database, and uses the Interactive SQL

INPUT statement in the *reload.sql* file to repopulate the database with data.

Ignore Java (-jr) With this option, Java is ignored when unloading or reloading the database.

-m With this option, user IDs are not preserved for a replicating database.

Unload schema definition only (-n) With this option, none of the data in the database is unloaded; *reload.sql* contains SQL statements to build the structure of the database only.

Log output messages to file (-o) Write output messages to the named file.

Escape character (-p) The default escape character (\) for external unloads (dbunload -x option) can be replaced by another character using this option. This option is available only when you run this utility from a command prompt.

Operate quietly (-q) Do not display output messages. This option is available only when you run this utility from a command prompt. If you specify -q, you must also specify -y or the unload will fail.

Specify reload filename (-r) Modify the name and directory of the generated reload Interactive SQL command file. The default is *reload.sql* in the current directory. The directory is relative to the current directory of the client application, not the server.

Unload only listed tables (-t) Provide a list of tables to be unloaded. By default, all tables are unloaded. Together with the -n option, this allows you to unload a set of table definitions only.

Output unordered data (-u) Normally, the data in each table is ordered by the primary key. Use this option if you are unloading a database with a corrupt index, so that the corrupt index is not used to order the data.

Enable verbose mode (-v) The table name of the table currently being unloaded, and how many rows have been unloaded, appears. This option is available only when you run this utility from a command prompt.

Use external unloading, internal reload (-xi) This option unloads data to the dbunload client, and uses the LOAD statement in the generated reload command file, *reload.sql*, to repopulate the database with data.

Use external unloading, external reload (-xx) This option unloads data to the dbunload client, and uses the Interactive SQL INPUT statement in the generated reload command file, *reload.sql*, to repopulate the database with data.

Operate without confirming actions (-y) Choosing this option replaces existing command files without prompting you for confirmation. If you specify -q, you must also specify -y or the unload will fail.

The Upgrade utility

While you can run a database created with an older version of Adaptive Server Anywhere on a newer release of Adaptive Server Anywhere, the new features are only available if the database is upgraded.

The Upgrade utility upgrades a database from an older version of Adaptive Server Anywhere to a newer version of Adaptive Server Anywhere, and allows you to take advantage of the newer release's list of features.

Upgrading a database does not require you to unload and reload your database.

If you want to use replication on an upgraded database, you must also archive your transaction log and start a new one on the upgraded database.

You can access the Upgrade utility in the following ways:

- ♦ From Sybase Central, using the Upgrade Database wizard.
- ♦ From Interactive SQL, using the ALTER DATABASE UPGRADE statement.

  ☞ For more information, see "ALTER DATABASE statement" [*ASA SQL Reference*, page 266].
- ♦ At a command prompt, using the dbupgrad command.

Back up before upgrading
As with any software, it is recommended that you make a backup of your database before upgrading.

Upgrading a database using the Upgrade Database wizard

❖ To upgrade a database

1. Connect to the database.
2. In the left pane, select the Adaptive Server Anywhere plug-in.
3. In the right pane, click the Utilities tab.
4. In the right pane, double-click Upgrade Database.

   The Upgrade Database wizard appears.
5. Follow the instructions in the wizard.

Tip

You can also access the Upgrade Database wizard by:

- ♦ Choosing Tools ➤ Adaptive Server Anywhere 9 ➤ Upgrade Database.
- ♦ Right-clicking the database, and choosing Upgrade Database from the popup menu.
- ♦ Selecting a database in the left pane, and choosing Upgrade Database from the File menu.

Upgrading databases that are too old for the Upgrade utility

❖ To upgrade a database created with a version of Adaptive Server Anywhere that is too old to be upgraded

1. Unload the database using the Unload utility.
2. Create a database with the name you want to use for the upgraded version using the Initialization utility.
3. Connect to the new database from Interactive SQL as the DBA user ID, and read the *reload.sql* command file to build the new database.

You can also use the dbunload utility to directly rebuild.

☞ For more information about using the dbunload utility, see "Rebuilding a database" on page 592.

Upgrading a database using the dbupgrad command-line utility

Syntax **dbupgrad** [*options*]

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-c *"keyword=value; . . . "*	Supply database connection parameters.
-I	Do not install Sybase jConnect support.
-j	Do not upgrade runtime Java classes.
-ja	Add runtime Java classes.
-jdk *version*	Install entries for the named version of the Java Development Kit.
-jr	Remove runtime Java classes.

Option	Description
-o *filename*	Log output messages to a file.
-q	Quiet mode—no windows or messages.

Description

The dbupgrad utility upgrades a database created with earlier versions of the software to enable features from the current version of the software. The earliest version that can be upgraded is Watcom SQL 3.2. While later versions of the database server do run against databases were created with earlier releases of the software, some of the features introduced since the version that created the database are unavailable unless the database is upgraded.

Exit codes are 0 (success) or non-zero (failure).

Not all features made available
Features that require a physical reorganization of the database file are not made available by dbupgrad. Such features include index enhancements and changes in data storage. To obtain the benefits of these enhancements, you must unload and reload your database.

☞ For more information, see "Upgrading the database file format" [*What's New in SQL Anywhere Studio,* page 230].

Java in the database

By default, Java in the database is not included in new databases. Java in the database is removed from the database during an upgrade if all of the following are true:

- the database is Java-enabled, but Java in the database is not being used
- the Java VM cannot be loaded by the database server
- you did not specify any Java-related options to the upgrade

The runtime classes add several megabytes to the size of a database. If you do not intend to use Java classes, you can specify the -j option to avoid including these classes in the upgraded database. If you want, you can add Sybase runtime Java classes later from Sybase Central or using the ALTER DATABASE statement.

☞ For more information, see "Java-enabling a database" [*ASA Programming Guide,* page 84].

Java in the database is a separately-licensable component. These classes are installed during upgrade only if you have installed the Java-in-the-database option or if the Java classes were in use in the database being upgraded.

Upgrade utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Connection parameters (-c) For a description of the connection parameters, see "Connection parameters" on page 176. The user ID must have DBA authority.

For example, the following command upgrades a database called sample80 to a version 9 format, connecting as user **DBA** with password **SQL**:

```
dbupgrad -c "uid=DBA;pwd=SQL;dbf=c:\asa80\sample80.db"
```

The dbupgrad utility must be run by a user with DBA authority.

Do not install Sybase jConnect support (-i) If you want to use the Sybase jConnect JDBC driver to access system catalog information, you need to install jConnect support. Use this option if you want to exclude the jConnect system objects. You can still use JDBC, as long as you do not access system information. If you want, you can add Sybase jConnect support later using Sybase Central or the ALTER DATABASE UPGRADE statement.

☞ For more information, see "Installing jConnect system objects into a database" [*ASA Programming Guide*, page 111].

Do not install runtime Java classes (-j) If you do not intend to use Java classes, you can specify either no Java option, or the -j option to avoid including these classes in the database.

Install runtime Java classes (-ja) Add Java in the database to the upgraded database. Specifying -ja installs version 1.3 of the JDK into the database.

Install support for the named version of the JDK (-jdk) If you want to install a version of Java, use the -jdk option followed by the version number. Valid values are 1.1.8, and 1.3.

For example, the following command upgrades a database to support JDK 1.1.8 applications in the database.

```
dbupgrad -jdk 1.1.8 java2.db
```

The -jdk option implies the -ja option.

Remove runtime Java classes (-jr) Remove Java in the database from the upgraded database.

Log output messages to file (-o) Write output messages to the named file.

Operate quietly (-q) Do not display output messages.

The Validation utility

With the Validation utility, you can validate the indexes and keys on some or all of the tables in the database. The Validation utility scans the entire table, and looks up each record in every index and key defined on the table.

This utility can be used in combination with regular backups (see "Backup and Data Recovery" on page 373) to give you confidence in the integrity of the data in your database.

You can access the Validation utility in the following ways:

♦ From Sybase Central, using the Validate Database wizard

♦ From Interactive SQL, using the sa_validate stored procedure or the VALIDATE TABLE statement.

♦ At a command prompt, using the dbvalid command. This is useful for incorporating into batch or command files.

☞ For more information on validating tables, see "VALIDATE TABLE statement" [*ASA SQL Reference*, page 662].

Caution
Validating a table or an entire database should be performed while no connections are making changes to the database; otherwise, spurious errors may be reported indicating some form of database corruption even though no corruption actually exists.

Validating a database using the Validate Database wizard

You can validate a database or individual tables from Sybase Central.

❖ To validate a database

1. In the left pane, select the Adaptive Server Anywhere plug-in.

2. In the right pane, click the Utilities tab.

3. In the right pane, double-click Validate Database.

   The Validate Database wizard appears.

4. Follow the instructions in the wizard.

❖ To validate a running database

1. Connect to the database.
2. Right-click the database and choose Validate Database from the popup menu.

   The Validate Database wizard appears.
3. Follow the instructions in the wizard.

❖ To validate an individual table

1. Connect to the database.
2. Locate the table you want to validate.
3. Right-click the table and choose Validate from the popup menu.

Tip
You can also access the Validate Database wizard from within Sybase Central using any of the following methods:

- ♦ Choosing Tools ➤ Adaptive Server Anywhere 9 ➤ Validate Database.
- ♦ Right-clicking the database, and choosing Validate Database from the popup menu.
- ♦ Selecting a database in the left pane, and choosing Validate Database from the File menu.

Validating a database using the dbvalid command-line utility

Syntax **dbvalid** [*options*] [*object-name*, . . .]

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-c *"keyword=value; . . . "*	Supply database connection parameters.
-f	Validate tables with full check.
-fd	Validate tables with data check.
-fi	Validate tables with index check.
-fn	Validate tables with the historical validation algorithm (used in versions 9.0.0 and earlier of the software).

Option	Description
-fx	Validate tables with express check.
-I	Each *object-name* is an index.
-o *filename*	Log output messages to a file.
-q	Quiet mode—do not print messages.
-s	Validate database pages using checksums.
-t	Each *object-name* is a table.
object-name	The name of a table or an index to validate.

Description

With the Validation utility, you can validate the indexes and keys on some or all of the tables in the database. This utility scans the entire table, and confirms that each row in the table exists in the appropriate indexes. It is equivalent to running the VALIDATE TABLE statement on each table.

By default, the Validation utility uses the express check option. However, the express check option is *not* used if you specify -f, -fd, -fi, -fn, or -I, or if the database was created with a version of the software earlier than version 7.0.

Exit codes are:

Program Exit Code	Description
0	Database validated successfully.
1	General failure in utility.
2	Error validating database.
7	Unable to find database to connect to (database name is wrong).
8	Unable to connect to database (user ID/password is wrong).
11	Unable to find server to connect to (server name is wrong).
12	Incorrect encryption key for starting database.

☞ For information on specific checks made during validation, see "VALIDATE TABLE statement" [*ASA SQL Reference,* page 662].

Validation utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Connection parameters (-c) For a description of the connection parameters, see "Connection parameters" on page 176. The user ID must have DBA authority or REMOTE DBA authority.

For example, the following validates the sample database, connecting as user **DBA** with password **SQL**:

```
dbvalid -c "uid=DBA;pwd=SQL;dbf=c:\asa\asademo.db"
```

Full check for each table (-f) In addition to the default validation checks, carry out both data checks (-fd) and index checks (-fi). This corresponds to the WITH FULL CHECK option on the VALIDATE TABLE statement. Depending on the contents of your database, this option may significantly extend the time required to validate it.

Data check for each table (-fd) In addition to the default validation checks, check that all of each LONG BINARY, LONG VARCHAR, TEXT, or IMAGE data type can be read. This corresponds to the WITH DATA CHECK option of the VALIDATE TABLE statement. Depending on the contents of your database, this option may significantly extend the time required to validate it.

Index check for each table (-fi) In addition to the default validation checks, validate each index on the table. This corresponds to the WITH INDEX CHECK option on the VALIDATE TABLE statement. Depending on the contents of your database, this option may significantly extend the time required to validate it.

Use historical validation algorithm to check each table (-fn) Use the validation algorithm that was used in versions 9.0.0 and earlier of the software instead of the express check algorithm.

Express check for each table (-fx) In addition to the default and data checks (-fd), check that the number of rows in the table matches the number of entries in the index. This corresponds to the WITH EXPRESS CHECK option on the VALIDATE TABLE STATEMENT. This option does not perform individual index lookups for each row. Using this option can significantly improve performance when validating large databases with a small cache. The express check option is on by default.

Validate specified indexes (-I) Instead of validating tables, validate indexes. In this case, for dbvalid, each of the *object-name* values supplied represents an index rather than a table, and has a name of the following form:

[[*owner*.]*table-name*.]*index-name*

Log output messages to file (-o) Write output messages to the named file.

Operate quietly (-q) Do not display output messages.

Validate database using page checksums (-s) Checksums are used to determine whether a database page has been modified on disk. If you created a database with checksums enabled, you can validate the database using checksums. Checksum validation reads each page of the database from disk and calculates its checksum. If the calculated checksum is different from the checksum stored on the page, the page has been modified on disk and an error is returned. The page numbers of any invalid pages appear in the server messages window. The -s option cannot be used in conjunction with -I, -t, or any of the -f options.

Validate tables (-t) The list of object-name values represents a list of tables. This is also the default behavior.

object-name The name of a table to validate.

If -I is used, *object-name* refers to an index to validate instead.

The Write File utility (deprecated)

The Write File utility is used to manage database write files. A **write file** is a file associated with a particular database. All changes are written into the write file, leaving the database file unchanged.

Deprecated feature
The use of write files is deprecated.

Using write files for development

Write files can be used effectively for testing when you do not want to modify the production database. They can also be used in network environments where read-only access to a database is desired, or when you distribute on CD-ROM a database that you want users to be able to modify.

Compressed databases

If you are using a compressed database, you must use a write file; compressed database files cannot be accessed directly. The write file name is then used in place of the database name when connecting to the database, or when loading a database on the database server command line.

You can access the Write File utility in the following ways:

- From Sybase Central, using the Create Write File wizard.
- At a command prompt, using the dbwrite command. This is useful for incorporating into batch or command files.

The Write File utility runs against a database file. The database must not be running on a server when you run the Write File utility.

The database file cannot be modified after the write file is created, otherwise the write file becomes invalid.

Creating a write file using the Create Write File wizard

❖ To create a write file for a database

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Utilities tab.
3. In the right pane, double-click Create Write File.

   The Create Write File wizard appears.
4. Follow the instructions in the wizard.

Tip
You can also access the Create Write File wizard from within Sybase Central using any of the following methods:

- ♦ Choosing Tools ➤ Adaptive Server Anywhere 9 ➤ Create Write File.
- ♦ Right-clicking the server, and choosing Create Write File from the popup menu.
- ♦ Selecting a server in the left pane, and choosing Create Write File from the File menu.

Creating a write file using the dbwrite command-line utility

Syntax

dbwrite [*options*] *database-file* [*write-name*]

Option	Description
@*data*	Read options from the specified environment variable or configuration file.
-c	Create a new write file.
-d *database-file*	Point a write file to a different database.
-ek *key*	Specify encryption key.
-ep	Prompt for encryption key.
-f *database-file*	Force the write file to point at a file.
-m *mirror-name*	Set the transaction log mirror name.
-o *filename*	Log output messages to a file.
-q	Quiet mode—do not print messages.
-s	Report the write file status only.
-t *log-name*	Set the transaction log name.
-y	Erase old files without confirmation.

Description

If any changes are made to the original database (not using the write file), the write file will no longer be valid. This happens if you start the server using the original database file, so you should create your write file from an archived copy of a database.

If a write file becomes invalid, you can discard all of your changes and create a new write file with the following command.

```
dbwrite -c db-name write-name
```

The *log-name* and *mirror-name* parameters are used only when creating a new write file. The *write-name* parameter is used only with the -c and -d parameters. Note that the *database-file* parameter must be specified before the *write-name* parameter.

Exit codes are 0 (success) or non-zero (failure).

Write file utility options

@data Use this option to read in options from the specified environment variable or configuration file. If both exist with the same name, the environment variable is used.

☞ For more information about configuration files, see "Using configuration files" on page 495.

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ For more information, see "Hiding the contents of files using the dbfhide command-line utility" on page 524.

Create a new write file (-c) If an existing write already exists, any information in the old write file will be lost. If no write file name is specified in the command, the write filename defaults to the database name with the extension *wrt*. If no transaction log name is specified, the log file name will default to the database name with the extension *wlg*.

Change the database file to which an existing write file points (-d) If a database file is moved to another directory, or renamed, this option allows you to maintain the link between the write file and the database file. This option is available only when you run this utility from a command prompt.

Specify encryption key (-ek) This option allows you to specify the encryption key for strongly encrypted databases directly in the command. If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log in any way. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify a key for a strongly encrypted database.

Prompt for encryption key (-ep) This option allows you to specify that you want to be prompted for the encryption key. This option causes a dialog box to appear, in which you enter the encryption key. It provides an extra measure of security by never allowing the encryption key to be seen in clear text. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify a key for a strongly encrypted database.

Force a write file to point to a file (-f) This option is available only when you run this utility from a command prompt. This option is for use when a write file is being created and the database file is held on a Novell NetWare or other network path, for operating systems on which they cannot be entered directly. By providing the full Novell path name for the database file (for example: *SYS:\asademo.db*), you can avoid dependencies on local mappings of the NetWare path. Unlike with the -d option, no checking is done on the specified path.

Set mirror name (-m) Set the name of the transaction log mirror file. This can only be used in conjunction with -c.

Log output messages to file (-o) Write output messages to the named file.

Operate quietly (-q) Do not display output messages. This option is available only when you run this utility from a command prompt.

Report the write file status only (-s) This displays the name of the database to which the write file points. This option is available only when you run this utility from a command prompt.

Set the transaction log filename (-t) If you are creating a new write file (-c), you can use this option to set the name of the transaction log file.

Operate without confirming actions (-y) Choosing this option replaces the existing write file without prompting you for confirmation.

CHAPTER 16

Database Options

About this chapter

This chapter describes the database, replication, compatibility and Interactive SQL options that can be set to customize and modify database behavior.

Contents

Introduction to database options

Database options control many aspects of database behavior. For example, you can use database options for the following purposes:

- **Compatibility** You can control how much like Adaptive Server Enterprise your Adaptive Server Anywhere database operates, and whether SQL that does not conform to SQL/92 generates errors.
- **Error handling** You can control what happens when errors such as dividing by zero, or overflow errors, occur.
- **Concurrency and transactions** You can control the degree of concurrency, and details of COMMIT behavior.

Setting options

You set options with the SET OPTION statement. It has the following general syntax:

SET [**EXISTING**] [**TEMPORARY**] **OPTION**
[*userid*. | **PUBLIC**.]*option-name* = [*option-value*]

Specify a user ID or group name to set the option for that user or group only. Every user belongs to the PUBLIC group. If no user ID or group is specified, the option change is applied to the currently logged on user ID that issued the SET OPTION statement.

For example, the following statement applies an option change to the user DBA, if DBA is the user that issues it:

```
SET OPTION login_mode = mixed
```

The following statement applies a change to the **PUBLIC** user ID, a user group to which all users belong.

```
SET OPTION Public.login_mode = standard
```

If *option-value* is omitted, the specified option setting will be deleted from the database. If it was a personal option setting, the value will revert back to the PUBLIC setting. If a TEMPORARY option is deleted, the option setting will revert back to the permanent setting.

Caution
Changing option settings while fetching rows from a cursor is not supported because it can lead to ill-defined behavior. For example, changing the DATE_FORMAT setting while fetching from a cursor would lead to different date formats among the rows in the result set. Do not change option settings while fetching rows.

☞ For more information, see "SET OPTION statement" [*ASA SQL Reference,* page 613].

Finding option settings

You can obtain a list of option settings, or the values of individual options, in a variety of ways.

Getting a list of option values

- Current option settings for your connection are available as a subset of **connection properties**. You can list all connection properties using the sa_conn_properties system procedure.

  ```
  CALL sa_conn_properties
  ```

  To order this list, you can call sa_conn_properties_by_name.

  If you want to filter the result or order by anything other than name, you could also use a SELECT statement. For example:

  ```
  SELECT *
  FROM sa_conn_properties()
  WHERE PropDescription like '%cache%'
  ORDER BY PropNum
  ```

  ☞ For more information, see "sa_conn_properties_by_name system procedure" [*ASA SQL Reference,* page 787], and "sa_conn_properties system procedure" [*ASA SQL Reference,* page 784].

- In Interactive SQL, the SET statement with no arguments lists the current setting of options.

  ```
  SET
  ```

- In Sybase Central, right-click a database, and choose Options from the popup menu.

- Use the following query on the SYSOPTIONS system view:

  ```
  SELECT *
  FROM SYSOPTIONS
  ```

  This displays all PUBLIC values, and those USER values that have been explicitly set.

Getting individual option values

You can obtain a single setting using the connection_property system function. For example, the following statement reports the value of the ANSI_INTEGER_OVERFLOW option:

```
SELECT CONNECTION_PROPERTY ('ANSI_INTEGER_OVERFLOW')
```

☞ For more information, see “CONNECTION_PROPERTY function [System]” [*ASA SQL Reference*, page 117].

Scope and duration of database options

You can set options at 3 levels of scope: public, user, and temporary.

Temporary options take precedence over user and public settings. User level options take precedence over public settings. If you set a user level option for the current user, the corresponding temporary option is set as well.

Some options (such as COMMIT behavior) are database-wide in scope. Setting these options requires DBA permissions. Other options (such as ISOLATION_LEVEL) can also be applied to just the current connection, and need no special permissions.

Changes to option settings take place at different times, depending on the option. Changing a global option such as RECOVERY_TIME takes place the next time the server is started.

Generally, only options that affect the current connection take place immediately. You can change option settings in the middle of a transaction, for example. One exception to this is that *changing options when a cursor is open can lead to unreliable results*. For example, changing DATE_FORMAT may not change the format for the next row when a cursor is opened. Depending on the way the cursor is being retrieved, it may take several rows before the change works its way to the user.

Setting temporary options

Adding the TEMPORARY keyword to the SET OPTION statement changes the duration of the change. Ordinarily an option change is permanent. It does not change until it is explicitly changed using the SET OPTION statement.

When the SET TEMPORARY OPTION statement is executed, the new option value takes effect only for the current connection, and only for the duration of the connection.

When the SET TEMPORARY OPTION is used to set a PUBLIC option, the change is in place for as long as the database is running. When the database is shut down, temporary options for the PUBLIC user ID revert back to their permanent value.

Setting an option for the PUBLIC user ID temporarily offers a security advantage. For example, when the LOGIN_MODE option is enabled, the database relies on the login security of the system on which it is running. Enabling it temporarily means that a database relying on the security of a Windows NT/2000/XP domain will not be compromised if the database is shut down and copied to a local machine. In this case, the LOGIN_MODE option will revert to its permanent value, which could be Standard, a mode where integrated logins are not permitted.

Setting public options

DBA authority is required to set an option for the PUBLIC user ID.

Changing the value of an option for the PUBLIC user ID sets the value of the option for all users who have not SET their own value. An option value cannot be set for an individual user ID unless there is already a PUBLIC user ID setting for that option.

Some options which can only be set for the PUBLIC user take effect immediately for existing connections, even though the changed setting will not be visible to users via the connection_property() function. An example of this is the GLOBAL_DATABASE_ID option. For this reason, PUBLIC-only options should not be changed while other users are connected to the database.

Deleting option settings

If *option-value* is omitted, the specified option setting will be deleted from the database. If it was a personal option setting, the value reverts back to the PUBLIC setting. If a TEMPORARY option is deleted, the option setting reverts back to the permanent setting.

For example, the following statement resets the ANSI_INTEGER_OVERFLOW option to its default value:

```
SET OPTION ANSI_INTEGER_OVERFLOW =
```

☞ For more information, see "SET OPTION statement" [*ASA SQL Reference*, page 613].

Option classification

Adaptive Server Anywhere provides many options. It is convenient to divide them into a few general classes. The classes of options are:

- "Database options" on page 619
- "Compatibility options" on page 624

- “Replication options” on page 628
- “Interactive SQL options” on page 630

Initial option settings

Connections to Adaptive Server Anywhere can be made through the TDS protocol (Open Client and jConnect JDBC connections) or through the Adaptive Server Anywhere protocol (ODBC and embedded SQL).

If you have users who use both TDS and the Adaptive Server Anywhere-specific protocol, you can configure their initial settings using stored procedures. As it is shipped, Adaptive Server Anywhere uses this method to set Open Client connections and jConnect connections to reflect default Adaptive Server Enterprise behavior.

The initial settings are controlled using the LOGIN_PROCEDURE option. This option names a stored procedure to run when users connect. The default setting is to use the sp_login_environment system procedure. If you wish to change this behavior, you can do so.

In turn, sp_login_environment checks to see if the connection is being made over TDS. If it is, it calls the sp_tsql_environment procedure, which sets several options to new “default” values for the current connection.

☞ For more information, see “LOGIN_PROCEDURE option [database]” on page 666, “sp_login_environment system procedure” [*ASA SQL Reference*, page 840], and “sp_tsql_environment system procedure” [*ASA SQL Reference*, page 849].

Database options

This section lists all database options.

OPTION	VALUES	DEFAULT
“AUDITING option [database]” on page 637	ON, OFF	OFF
“BACKGROUND_PRIORITY option [database]” on page 639	ON, OFF	OFF
“BLOCKING option [database]” on page 640	ON, OFF	ON
“BLOCKING_TIMEOUT option [database]” on page 640	Integer	0
“CHECKPOINT_TIME option [database]” on page 641	Number of minutes	60
“CIS_OPTION option [database]” on page 641	0, 7	0
“CIS_ROWSET_SIZE option [database]” on page 642	Integer	50
“COOPERATIVE_COMMIT_TIMEOUT option [database]” on page 645	Integer	250
“COOPERATIVE_COMMITS option [database]” on page 645	ON, OFF	ON
“DEBUG_MESSAGES option [database]” on page 648	ON, OFF	OFF
“DEDICATED_TASK option [database]” on page 649	ON, OFF	OFF
“DEFAULT_TIMESTAMP_INCREMENT option [database]” on page 650	Integer	1
“DELAYED_COMMIT_TIMEOUT option [database]” on page 650	Integer	500
“DELAYED_COMMITS option [database]” on page 651	ON, OFF	OFF

OPTION	VALUES	DEFAULT
"EXCLUDE_OPERATORS option [database]" on page 653	reserved	reserved
"EXTENDED_JOIN_SYNTAX option [database]" on page 653	ON, OFF	ON
"FIRST_DAY_OF_WEEK option [database]" on page 654	1, 2, 3, 4, 5, 6, 7	7
"FOR_XML_NULL_TREATMENT option [database]" on page 656	EMPTY, OMIT	OMIT
"FORCE_VIEW_CREATION option [database]" on page 656	reserved	reserved
"GLOBAL_DATABASE_ID option [database]" on page 656	Integer	2147483647
"INTEGRATED_SERVER_NAME option [database]" on page 658	String	NULL
"JAVA_HEAP_SIZE option [database]" on page 663	Number of bytes	1 000 000
"JAVA_INPUT_OUTPUT option [database]" on page 664	ON, OFF	OFF
"JAVA_NAMESPACE_SIZE option [database]" on page 664	Number of bytes	4 000 000
"LOG_DEADLOCKS option [database]" on page 665	ON, OFF	OFF
"LOGIN_MODE option [database]" on page 665	STANDARD, MIXED, INTEGRATED	STANDARD
"LOGIN_PROCEDURE option [database]" on page 666	String	sp_login_environment
"MAX_CURSOR_COUNT option [database]" on page 668	Integer	50
"MAX_HASH_SIZE option [database] (deprecated)" on page 668	Integer	10
"MAX_PLANS_CACHED option [database]" on page 669	Integer	20

OPTION	VALUES	DEFAULT
"MAX_RECURSIVE_ITERATIONS option [database]" on page 669	Integer	100
"MAX_STATEMENT_COUNT option [database]" on page 670	Integer >=0	50
"MAX_WORK_TABLE_HASH_SIZE option [database] (deprecated)" on page 670	Integer	20
"MIN_PASSWORD_LENGTH option [database]" on page 671	Integer	0
"ODBC_DESCRIBE_BINARY_AS_VARBINARY [database]" on page 673	ON, OFF	OFF
"ODBC_DISTINGUISH_CHAR_AND_VARCHAR option [database]" on page 673	ON, OFF	OFF
"ON_CHARSET_CONVERSION_FAILURE option [database]" on page 674	IGNORE, WARNING, ERROR	IGNORE
"OPTIMIZATION_GOAL option [database]" on page 676	First-row or all-rows	all-rows
"OPTIMIZATION_LEVEL option [database]" on page 677	0–15	9
"OPTIMIZATION_WORKLOAD option [database]" on page 678	Mixed, OLAP	Mixed
"PINNED_CURSOR_PERCENT_OF_CACHE option [database]" on page 681	0–100	10
"PRECISION option [database]" on page 682	Number of digits	30
"PREFETCH option [database]" on page 682	OFF, CONDITIONAL, ALWAYS	CONDITIONAL

OPTION	VALUES	DEFAULT
"PRESERVE_SOURCE_FORMAT option [database]" on page 683	ON, OFF	ON
"PRE-VENT_ARTICLE_PKEY_UPDATE option [database]" on page 684	ON, OFF	ON
"READ_PAST_DELETED option [database]" on page 685	ON, OFF	ON
"RECOVERY_TIME option [database]" on page 686	Number of minutes	2
"REMOTE_IDLE_TIMEOUT option [database]" on page 686	Number of seconds	15
"RE-TURN_DATE_TIME_AS_STRING option [database]" on page 688	ON, OFF	OFF
"RETURN_JAVA_AS_STRING option [database]" on page 688	ON, OFF	OFF
"ROLLBACK_ON_DEADLOCK [database]" on page 689	ON, OFF	ON
"ROW_COUNTS option [database]" on page 689	ON, OFF	OFF
"SCALE option [database]" on page 690	Number of digits	6
"SORT_COLLATION option [database]" on page 690	any valid collation_-name or collation_ID	Internal
"SUBSUME_ROW_LOCKS option [database]" on page 693	ON, OFF	ON
"SUPPRESS_TDS_DEBUGGING option [database]" on page 693	ON, OFF	OFF
"TDS_EMPTY_STRING_IS_NULL option [database]" on page 693	ON, OFF	OFF
"TEMP_SPACE_LIMIT_CHECK option [database]" on page 694	ON, OFF	OFF

OPTION	VALUES	DEFAULT
"TIME_ZONE_ADJUSTMENT option [database]" on page 695	Integer, or Negative integer enclosed in quotation marks, or String representing time in hours and minutes, preceded by + or -	Set by either the client's or the server's time zone, depending on the client's connection type
"TRUNCATE_DATE_VALUES option [database] (deprecated)" on page 697	ON, OFF	ON (post version 7 databases) OFF (pre version 7 databases)
"TRUNCATE_TIMESTAMP_VALUES option [database]" on page 697	ON, OFF	OFF
"TRUNCATE_WITH_AUTO_COMMIT option [database]" on page 699	ON, OFF	OFF
"UPDATE_STATISTICS option [database]" on page 700	ON, OF	ON
"USER_ESTIMATES option [database]" on page 700	ENABLED, OVERRIDE-MAGIC, DISABLED	OVERRIDE-MAGIC
"WAIT_FOR_COMMIT option [database]" on page 702	ON, OFF	OFF
"WEBSERVICE_NAMESPACE_HOST option [database]" on page 702	String, NULL	NULL

Compatibility options

The following options allow Adaptive Server Anywhere behavior to be made compatible with Adaptive Server Enterprise, or to support both old behavior and allow ISO SQL/92 behavior.

For further compatibility with Adaptive Server Enterprise, some of these options can be set for the duration of the current connection using the Transact-SQL SET statement instead of the Adaptive Server Anywhere SET OPTION statement.

☞ For a listing, see "SET statement [T-SQL]" [*ASA SQL Reference*, page 606].

Default settings

The default setting for some of these options differs from the Adaptive Server Enterprise default setting. For compatibility, you should explicitly set the options to ensure compatible behavior.

When a connection is made using the Open Client or JDBC interfaces, some option settings are explicitly set for the current connection to be compatible with Adaptive Server Enterprise. These options are listed in the following table.

☞ For information on how the settings are made, see "System and catalog stored procedures" [*ASA SQL Reference*, page 777].

Options for Open Client and JDBC connection compatibility with Adaptive Server Enterprise

Option	**Setting**
ALLOW_NULLS_BY_DEFAULT	OFF
ANSI_BLANKS	ON
ANSINULL	OFF
CHAINED	OFF
CONTINUE_AFTER_RAISERROR	ON
DATE_FORMAT	YYYY-MM-DD
DATE_ORDER	MDY
ESCAPE_CHARACTER	OFF
FLOAT_AS_DOUBLE	ON
ISOLATION_LEVEL	1
ON_TSQL_ERROR	CONDITIONAL
QUOTED_IDENTIFIER	OFF

Option	Setting
TIME_FORMAT	HH:NN:SS.SSS
TIMESTAMP_FORMAT	YYYY-MM-DD HH:NN:SS.SSS
TSQL_HEX_CONSTANT	ON
TSQL_VARIABLES	ON

Transact-SQL and SQL/92 compatibility options

The following table lists the compatibility options, their allowed values, and their default settings.

OPTION	VALUES	DEFAULT
"ALLOW_NULLS_BY_DEFAULT option [compatibility]" on page 633	ON, OFF	ON
"ANSI_BLANKS option [compatibility]" on page 633	ON, OFF	OFF
"ANSI_CLOSE_CURSORS_ON_ROLLBACK option [compatibility]" on page 633	ON, OFF	OFF
"ANSI_INTEGER_OVERFLOW option [compatibility]" on page 634	ON, OFF	OFF
"ANSI_PERMISSIONS option [compatibility]" on page 634	ON, OFF	ON
"ANSI_UPDATE_CONSTRAINTS option [compatibility]" on page 635	OFF, CURSORS, STRICT	CURSORS
"ANSINULL option [compatibility]" on page 636	ON, OFF	ON
"AUTOMATIC_TIMESTAMP option [compatibility]" on page 638	ON, OFF	OFF
"CHAINED option [compatibility]" on page 640	ON, OFF	ON
"CLOSE_ON_ENDTRANS option [compatibility]" on page 642	ON, OFF	ON

OPTION	VALUES	DEFAULT
"CONTINUE_AFTER_RAISERROR option [compatibility]" on page 644	ON, OFF	ON
"CONVERSION_ERROR option [compatibility]" on page 645	ON, OFF	ON
"DATE_FORMAT option [compatibility]" on page 646	*String*	YYYY-MM-DD
"DATE_ORDER option [compatibility]" on page 648	MDY, YMD, DMY	YMD
"DIVIDE_BY_ZERO_ERROR option [compatibility]" on page 652	ON, OFF	ON
"ESCAPE_CHARACTER option [compatibility]" on page 653	System use	OFF
"FIRE_TRIGGERS option [compatibility]" on page 654	ON, OFF	ON
"FLOAT_AS_DOUBLE option [compatibility]" on page 655	ON, OFF	OFF
"ISOLATION_LEVEL option [compatibility]" on page 659	0, 1, 2, 3	0 1 for Open Client and JDBC connections
"NEAREST_CENTURY option [compatibility]" on page 671	Integer	50
"NON_KEYWORDS option [compatibility]" on page 672	Comma-separated keywords list	No keywords turned off
"ON_TSQL_ERROR option [compatibility]" on page 675	STOP, CONDITIONAL, CONTINUE	CONDITIONAL
"OPTIMISTIC_WAIT_FOR_COMMIT option [compatibility]" on page 675	ON, OFF	OFF
"PERCENT_AS_COMMENT option [compatibility]" on page 681	ON, OFF	ON

OPTION	VALUES	DEFAULT
"QUERY_PLAN_ON_OPEN option [compatibility]" on page 684	ON, OFF	OFF
"QUOTED_IDENTIFIER option [compatibility]" on page 685	ON, OFF	ON
"RI_TRIGGER_TIME option [compatibility]" on page 689	BEFORE, AFTER	AFTER
"SQL_FLAGGER_ERROR_LEVEL option [compatibility]" on page 691	E, I, F, W	W
"SQL_FLAGGER_WARNING_LEVEL option [compatibility]" on page 691	E, I, F, W	W
"STRING_RTRUNCATION option [compatibility]" on page 692	ON, OFF	OFF
"TIME_FORMAT option [compatibility]" on page 694	String	HH:NN:SS.SSS
"TIMESTAMP_FORMAT option [compatibility]" on page 695	String	YYYY-MM-DD HH:NN:ss.SSS
"TSQL_HEX_CONSTANT option [compatibility]" on page 699	ON, OFF	ON
"TSQL_VARIABLES option [compatibility]" on page 700	ON, OFF	OFF

Replication options

The following options are included to provide control over replication behavior. Some replication options are useful for SQL Remote replication, and some are relevant for use with Sybase Replication Server.

OPTION	VALUES	DEFAULT
"BLOB_THRESHOLD option [replication]" on page 639	Integer	256
"COMPRESSION option [replication]" on page 643	Integer, from -1 to 9	6
"DELETE_OLD_LOGS option [replication]" on page 652	ON, OFF	OFF
"QUALIFY_OWNERS option [replication]" on page 684	ON, OFF	ON
"QUOTE_ALL_IDENTIFIERS option [replication]" on page 685	ON, OFF	OFF
"REPLICATE_ALL option [replication]" on page 687	ON, OFF	OFF
"REPLICATION_ERROR option [replication]" on page 687	Procedure-name	(no procedure)
"REPLICATION_ERROR_PIECE option [replication]" on page 687	Procedure-name	(no procedure)
"SUBSCRIBE_BY_REMOTE option [replication]" on page 692	ON, OFF	ON

OPTION	VALUES	DEFAULT
"VERIFY_ALL_COLUMNS option [replication]" on page 701	ON, OFF	OFF
"VERIFY_THRESHOLD option [replication]" on page 701	Integer	1 000

Interactive SQL options

Syntax 1

SET [**TEMPORARY**] **OPTION**

[*userid*. | **PUBLIC**.]*option-name* = [*option-value*]

Syntax 2

SET PERMANENT

Syntax 3

SET

Parameters

userid: *identifier*, *string*, or *host-variable*

option-name: *identifier*, *string*, or *host-variable*

option-value: *host-variable* (indicator allowed), *string*, *identifier*, or *number*

Description

SET PERMANENT (syntax 2) stores all current Interactive SQL options in the SYSOPTION system table. These settings are automatically established every time Interactive SQL is started for the current user ID.

Syntax 3 is used to display all of the current option settings. If any options have temporary settings for Interactive SQL or the database server, these are displayed; otherwise, the permanent option settings are displayed.

OPTION	VALUES	DEFAULT
"AUTO_COMMIT option [Interactive SQL]" on page 638	ON, OFF	OFF
"AUTO_REFETCH option [Interactive SQL]" on page 638	ON, OFF	ON
"BELL option [Interactive SQL]" on page 639	ON, OFF	ON
"COMMAND_DELIMITER option [Interactive SQL]" on page 642	String	' ; '
"COMMIT_ON_EXIT option [Interactive SQL]" on page 643	ON, OFF	ON
"DEFAULT_ISQL_ENCODING option [Interactive SQL]" on page 649	String	(empty string)

OPTION	VALUES	DEFAULT
“DESCRIBE_JAVA_FORMAT option [Interactive SQL]” on page 652	Varchar, binary	Varchar
“ECHO option [Interactive SQL]” on page 653	ON, OFF	ON
“INPUT_FORMAT option [Interactive SQL]” on page 657	ASCII, DBASE, DBASEII, DBASEIII, EXCEL, FIXED, FOXPRO, LOTUS	ASCII
“ISQL_COMMAND_TIMING option [Interactive SQL]” on page 659	ON, OFF	ON
“ISQL_ESCAPE_CHARACTER option [Interactive SQL]” on page 660	Character	‘ \‘
“ISQL_FIELD_SEPARATOR option [Interactive SQL]” on page 660	String	‘ , ‘
“ISQL_LOG option [Interactive SQL]” on page 661	File-name	(empty string)
“ISQL_PLAN option [Interactive SQL]” on page 661	NONE, SHORT, LONG, GRAPHICAL, GraphicalWithStatistics	SHORT
“ISQL_PRINT_RESULT_SET option [Interactive SQL]” on page 662	LAST, ALL, NONE	LAST
“ISQL_QUOTE option [Interactive SQL]” on page 663	String	‘
“NULLS option [Interactive SQL]” on page 672	String	‘(NULL)’

OPTION	VALUES	DEFAULT
"ON_ERROR option [Interactive SQL]" on page 674	STOP, CONTINUE, PROMPT, EXIT, NOTIFY_CONTINUE, NOTIFY_STOP, NOTIFY_EXIT	PROMPT
"OUTPUT_FORMAT option [Interactive SQL]" on page 679	ASCII, DBASEII, DBASEIII, EXCEL, FIXED, FOXPRO, HTML, LOTUS, SQL, XML,	ASCII
"OUTPUT_LENGTH option [Interactive SQL]" on page 680	Integer	0
"OUTPUT_NULLS option [Interactive SQL] " on page 680	String	'NULL'
"STATISTICS option [Interactive SQL]" on page 692	0,3,4,5,6	3
"TRUNCATION_LENGTH option [Interactive SQL]" on page 699	Integer	256

Alphabetical list of options

This section lists options alphabetically.

ALLOW_NULLS_BY_DEFAULT option [compatibility]

Function	Controls whether new columns that are created without specifying either **NULL** or **NOT NULL** are allowed to contain NULL values.
Allowed values	**ON**, **OFF**
Default	**ON** **OFF** for Open Client and JDBC connections
Description	The ALLOW_NULLS_BY_DEFAULT option is included for Transact-SQL compatibility. ☞ For more information, see "Setting options for Transact-SQL compatibility" [*ASA SQL User's Guide*, page 485].

ANSI_BLANKS option [compatibility]

Function	Controls behavior when character data is truncated at the client side.
Allowed values	**ON**, **OFF**
Default	**OFF** **ON** for Open Client and JDBC connections
Description	The ANSI_BLANKS option has no effect unless the database was created with the -b command-line option. It forces a truncation error whenever a value of data type CHAR(*N*) is read into a C char(*M*) variable for values of *N* greater than or equal to *M*. With ANSI_BLANKS set to **OFF**, a truncation error occurs only when at least one non-blank character is truncated. For embedded SQL, if ANSI_BLANKS is ON when you supply a value of data type DT_STRING, you must set the **sqllen** field to the length of the value, including space for the terminating null character. With ANSI_BLANKS off, the length is determined solely by the position of the null character. The ANSI_BLANKS setting persists for the life of a connection. Changing it after a connection has been established does not affect that connection.

ANSI_CLOSE_CURSORS_ON_ROLLBACK option [compatibility]

Function	Controls whether cursors that were opened WITH HOLD are closed when a

ROLLBACK is performed.

Allowed values	**ON**, **OFF**
Default	**OFF**
Description	The draft SQL/3 standard requires all cursors be closed when a transaction is rolled back. By default, on a rollback Adaptive Server Anywhere closes only those cursors that were opened without a WITH HOLD clause. This option allows you to force closure of all cursors.

The CLOSE_ON_ENDTRANS option overrides the ANSI_CLOSE_CURSORS_ON_ROLLBACK option.

ANSI_INTEGER_OVERFLOW option [compatibility]

Function	Controls what happens when an arithmetic expression causes an integer overflow error.
Allowed values	**ON, OFF**
Default	**OFF**
Description	The ISO SQL/92 standard requires integer overflow should result in a `SQLSTATE = 22003 - overflow` error. Adaptive Server Anywhere behavior was previously different from this. The ANSI_INTEGER_OVERFLOW option can be set to **OFF** to maintain compatibility with previous releases of the software.

ANSI_PERMISSIONS option [compatibility]

Function	Controls permissions checking for DELETE and UPDATE statements.
Allowed values	**ON**, **OFF**
Scope	Can be set for the PUBLIC group only. Takes effect immediately. DBA authority is required to set this option.
Default	**ON**
Description	With ANSI_PERMISSIONS ON, the SQL/92 permissions requirements for DELETE and UPDATE statements are checked. The default value is OFF in Adaptive Server Enterprise. The following table outlines the differences.

SQL statement	Permissions required with ANSI_PERMISSIONS off	Permissions required with ANSI_PERMISSIONS on
UPDATE	UPDATE permission on the columns where values are being set	UPDATE permission on the columns where values are being set SELECT permission on all columns appearing in the WHERE clause SELECT permission on all columns on the right side of the set clause
DELETE	DELETE permission on the table	DELETE permission on the table SELECT permission on all columns appearing in the WHERE clause

The ANSI_PERMISSIONS option can be set only for the PUBLIC group. No private settings are allowed.

ANSI_UPDATE_CONSTRAINTS option [compatibility]

Function	Controls the range of updates that are permitted.
Allowed values	**OFF**, **CURSORS**, **STRICT**
Default	**CURSORS** in new databases
	OFF in databases created before version 7.0

Description

Adaptive Server Anywhere provides several extensions that allow updates which are not permitted by the ANSI SQL standard. These extensions provide powerful, efficient mechanisms for performing updates. However, in some cases, they cause behavior that is not intuitive. This behavior can produce anomalies such as lost updates if the user application is not designed to expect the behavior of these extensions.

The ANSI_UPDATE_CONSTRAINTS option controls whether updates are restricted to those permitted by the SQL/92 standard.

If the option is set to STRICT, the following updates are prevented:

- Updates of cursors containing JOINS
- Updates of columns that appear in an ORDER BY clause
- The FROM clauses is not allowed in UPDATE statements

If the option is set to CURSORS, these same restrictions are in place, but only for cursors. If a cursor is not opened with FOR UPDATE or FOR READ ONLY, the database server chooses updatability based on the SQL/92 standard. If the ANSI_UPDATE_CONSTRAINTS option is CURSORS or STRICT, cursors containing an ORDER BY clause default to FOR READ ONLY; otherwise, they continue to default to FOR UPDATE.

See also

"UPDATE statement" [*ASA SQL Reference,* page 650]

ANSINULL option [compatibility]

Function	Controls the interpretation of NULL values.
Allowed values	**ON**, **OFF**
Default	**ON**

Description

This option is implemented primarily for Transact-SQL (Adaptive Server Enterprise) compatibility. ANSINULL affects the results of comparison predicates with NULL constants, and also affects warnings issued for grouped queries over NULL values.

With ANSINULL **ON**, ANSI three-valued logic is used for all comparison predicates in a WHERE or HAVING clause, or in an ON condition. Any comparisons with NULL using **=** or **!=** evaluate to unknown.

Setting ANSINULL to **OFF** means that Adaptive Server Anywhere uses two-valued logic for the following four conditions:

<*expr*> = NULL

<*expr* > != NULL

<*expr*> = @var // @var is a procedure variable, or a host variable

<*expr*> != @var

In each case, the predicate evaluates to either true or false—never unknown. In such comparisons, the NULL value is treated as a special value in each domain, and an equality (=) comparison of two NULL values will yield true. Note that the expression *expr* must be a relatively simple expression, referencing only columns, variables, and literals; subqueries and functions are not permitted.

With ANSINULL **ON**, the evaluation of any aggregate function, except COUNT(*), on an expression that contains at least one NULL value, may generate the warning `null value eliminated in aggregate function` (SQLSTATE=01003). With ANSINULL OFF, this warning does not appear.

Limitations

- Setting ANSINULL to **OFF** affects only WHERE, HAVING, or ON predicates in SELECT, UPDATE, DELETE, and INSERT statements. The semantics of comparisons in a CASE or IF statement, or in IF expressions, are unaffected.
- Adaptive Server Enterprise 12.5 introduced a change in behavior for LIKE predicates with a NULL pattern string when ANSINULL is **OFF**. In Adaptive Server Anywhere, LIKE predicates remain unaffected by the setting of ANSINULL.

AUDITING option [database]

Function

Enables and disables auditing in the database.

Allowed values

ON, **OFF**

Scope

Can be set for the PUBLIC group only. Takes effect immediately. DBA permissions are required to set this option.

Default

OFF

Description

This option turns auditing on and off.

Auditing is the recording of detailed information about many events in the database in the transaction log. Auditing provides some security features, at the cost of some performance.

For the AUDITING option to work, you must set the AUDITING option to ON, and also specify which types of information you want to audit using the sa_enable_auditing_type system procedure. Auditing will not take place if either of the following are true:

- AUDITING is set to OFF
- Auditing options have been disabled

If you set AUDITING ON, and do not specify auditing options, all types of auditing information are recorded. Alternatively, you can choose to record any combination of the following: audit permission checks, connection attempts, DDL statements, public options, and triggers.

See also

"sa_enable_auditing_type" [*ASA SQL Reference,* page 792]

"sa_disable_auditing_type" [*ASA SQL Reference,* page 790]

Example

- Turn on auditing

```
SET OPTION PUBLIC.AUDITING = 'ON'
```

AUTO_COMMIT option [Interactive SQL]

Function	Controls whether a COMMIT is performed after each statement.
Allowed values	**ON**, **OFF**
Default	**OFF**
Description	If AUTO_COMMIT is **ON**, a database COMMIT is performed after each successful statement. By default, a COMMIT or ROLLBACK is performed only when the user issues a COMMIT or ROLLBACK statement or a SQL statement that causes an automatic commit (such as the CREATE TABLE statement).

AUTO_REFETCH option [Interactive SQL]

Function	Controls whether query results are fetched again after deletes, updates, and inserts.
Allowed values	**ON**, **OFF**
Default	**ON**
Description	If AUTO_REFETCH is on, the current query results that appear on the Results tab in the Interactive SQL Results pane will be refetched from the database after any INSERT, UPDATE, or DELETE statement. Depending on how complicated the query is, this may take some time. For this reason, it can be turned off.

AUTOMATIC_TIMESTAMP option [compatibility]

Function	Controls interpretation of new columns with the TIMESTAMP data type.
Allowed values	**ON**, **OFF**
Default	**OFF**
Description	Controls whether any new columns with the TIMESTAMP data type that do not have an explicit default value defined are given a default value of the Transact-SQL **timestamp** value as a default. The AUTOMATIC_TIMESTAMP option is included for Transact-SQL compatibility. The default is OFF. ☞ For more information, see "Setting options for Transact-SQL compatibility" [*ASA SQL User's Guide*, page 485].

BACKGROUND_PRIORITY option [database]

Function	To limit impact on the performance of connections other than the current connection.
Allowed values	**ON**, **OFF**
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately. If you set this option temporarily, that setting applies to the current connection only. Different connections under the same user ID can have different settings for this option.
Default	**OFF**
Description	When set to **ON**, it requests that the current connection have minimal impact on the performance of other connections. This option allows tasks for which responsiveness is critical to coexist with other tasks for which performance is not as important.

BELL option [Interactive SQL]

Function	Controls whether the bell sounds when an error occurs.
Allowed values	**ON**, **OFF**
Default	**ON**
Description	Set this option according to your preference.

BLOB_THRESHOLD option [replication]

Function	Controls the size of value that the Message Agent treats as a long object (BLOB).
Allowed values	*Integer*, in kilobytes
Default	**256**
Description	Any value longer than the BLOB_THRESHOLD option is replicated as a BLOB. That is, it is broken into pieces and replicated in chunks before being reconstituted using a SQL variable and concatenating the pieces at the recipient site. If you set BLOB_THRESHOLD to a high value in remote Adaptive Server Anywhere databases, BLOBs are not broken into pieces, and operations can be applied to Adaptive Server Enterprise by the Message Agent. Each SQL

statement must fit within a message, so this only allows replication of small BLOBs.

BLOCKING option [database]

Function	Controls the behavior in response to locking conflicts.
Allowed values	**ON**, **OFF**
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.
Default	**ON**
Description	If BLOCKING is ON, any transaction attempting to obtain a lock that conflicts with an existing lock held by another transaction waits until every conflicting lock is released. At that time, the write goes through. If BLOCKING is OFF, the transaction that attempts to obtain a conflicting lock receives an error.

☞ For more information, see "Two-phase locking" [*ASA SQL User's Guide*, page 144].

BLOCKING_TIMEOUT option [database]

Function	To control how long a transaction waits to obtain a lock.
Allowed values	*Integer*, in milliseconds
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.
Default	**0**
Description	When BLOCKING is ON, any transaction attempting to obtain a lock that conflicts with an existing lock waits for BLOCKING_TIMEOUT milliseconds for the conflicting lock to be released. If the lock is not released within BLOCKING_TIMEOUT milliseconds, then an error is returned for the waiting transaction. Setting this option to 0 forces all transactions attempting to obtain a lock to wait until all conflicting transactions release their locks.
See also	"BLOCKING option [database]" on page 640

CHAINED option [compatibility]

Function	Controls transaction mode in the absence of a BEGIN TRANSACTION statement.

Allowed values	**ON**, **OFF**
Default	**ON**
	OFF for Open Client and JDBC connections
Description	Controls the Transact-SQL transaction mode. In Unchained mode (CHAINED = OFF), each statement is committed individually unless an explicit BEGIN TRANSACTION statement is executed to start a transaction. In chained mode (CHAINED = ON) a transaction is implicitly started before any data retrieval or modification statement.

CHECKPOINT_TIME option [database]

Function	Set the maximum number of minutes that the database server will run without doing a checkpoint.
Allowed values	*Integer*
Scope	Can be set for the PUBLIC group only. DBA authority is required to set the option. You must shut down and restart the database server for the change to take effect.
Default	**60**
Description	This option is used with the RECOVERY_TIME option to decide when checkpoints should be done.
	☞ For information on checkpoints, see "Checkpoints and the checkpoint log" on page 396.
See also	"RECOVERY_TIME option [database]" on page 686

CIS_OPTION option [database]

Function	Controls whether debugging information for remote data access appears in the server window.
Allowed values	**0**, **7**
Scope	Can be set for an individual connection or for the PUBLIC group.
Default	**0**
Description	This option controls whether information about how queries are executed on a remote database appears in the server window when using remote data access. Set this option to **7** to see debugging information in the server window. When this option is set to **0** (the default), debugging information for remote data access does not appear in the server window.
	Once you have turned on remote tracing, the tracing information appears in

the database server window. You can log this output to a file by specifying the - o *filename* server option when you start the database server.

☞ For more information about the -o server option, see "-o server option" on page 151.

CIS_ROWSET_SIZE option [database]

Function	Set the number of rows that are returned from remote servers for each fetch.
Allowed values	*Integer*
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect when a new connection is made to a remote server.
Default	**50**
Description	This option sets the ODBC **FetchArraySize** value when using ODBC to connect to a remote database server.

CLOSE_ON_ENDTRANS option [compatibility]

Function	Controls the closing of cursors at the end of a transaction.
Allowed values	**ON**, **OFF**
Default	**ON**
Description	When CLOSE_ON_ENDTRANS is set to **ON**, cursors are closed whenever a transaction is committed unless the cursor was opened WITH HOLD. The behavior when a transaction is rolled back is governed by the ANSI_CLOSE_CURSORS_AT_ROLLBACK option.

When CLOSE_ON_ENDTRANS is set to **OFF**, cursors are not closed at either a commit or a rollback, regardless of the ANSI_CLOSE_CURSORS_AT_ROLLBACK option setting or whether the cursor was opened WITH HOLD or not.

Setting this to **OFF** provides Adaptive Server Enterprise compatible behavior.

COMMAND_DELIMITER option [Interactive SQL]

Function	Sets the string that indicates the end of a statement in Interactive SQL.
Allowed values	*String*
Default	Semi-colon (**;**)
Description	You can specify a command delimiter to indicate the end of a SQL statement. This command delimiter can be any sequence of characters

(including numbers, letters, and punctuation), but it cannot contain embedded blanks. As well, it can contain a semicolon, but only as the first character.

If the command delimiter is set to a string beginning with a character that is valid in identifiers, the command delimiter must be preceded by a space. This command delimiter is case sensitive. You must enclose the new command delimiter in single quotation marks.

Note that if the command delimiter is a semicolon (the default), no space is required before the semicolon.

See also

"The Interactive SQL utility" on page 538

Examples

```
SET OPTION COMMAND_DELIMITER='~'
```

```
SET TEMPORARY OPTION command_delimiter = 'select';
message 'hello' select
```

You can also use Interactive SQL's -d command line option to set the command delimiter without including a SET TEMPORARY OPTION COMMAND_DELIMITER statement in a *.sql* file. For example, if you have a script file named *test.sql* which uses tildes (~) as command delimiters, you could run:

```
dbisql -d "~" test.sql
```

COMMIT_ON_EXIT option [Interactive SQL]

Function

Controls behavior when Interactive SQL disconnects or terminates.

Allowed values

ON, **OFF**

Default

ON

Description

Controls whether a COMMIT or ROLLBACK is done when you leave Interactive SQL. When COMMIT_ON_EXIT is set to **ON**, a COMMIT is done.

COMPRESSION option [replication]

Function

Set the level of compression for SQL Remote messages.

Allowed values

Integer, from -1 to 9

Default

6

Description

The values have the following meanings:

- **-1** Send messages in Version 5 format. Message Agents (both dbremote and ssremote) from previous versions of SQL Remote cannot read messages sent in Version 6 format. You should ensure that COMPRESSION is set to -1 until all Message Agents in your system are upgraded to Version 6.

- **0** No compression.
- **1 to 9** Increasing degrees of compression. Creating messages with high compression can take longer than creating messages with low compression.

CONTINUE_AFTER_RAISERROR option [compatibility]

Function	Controls behavior following a RAISERROR statement.
Allowed values	**ON**, **OFF**
Default	**OFF** for databases created with Version 5.x.
	ON for databases created with Version 6 or later.

Description

The RAISERROR statement is used within procedures and triggers to generate an error. When this option is set to **OFF**, the execution of the procedure or trigger is stopped whenever the RAISERROR statement is encountered.

If you set the CONTINUE_AFTER_RAISERROR option to **ON**, the RAISERROR statement no longer signals an execution-ending error. Instead, the RAISERROR status code and message are stored and the most recent RAISERROR is returned when the procedure completes. If the procedure that caused the RAISERROR was called from another procedure, the RAISERROR is not returned until the outermost calling procedure terminates.

Intermediate RAISERROR statuses and codes are lost after the procedure terminates. If, at return time, an error occurs along with the RAISERROR, then the information for the new error is returned and the RAISERROR information is lost. The application can query intermediate RAISERROR statuses by examining the @@error global variable at different execution points.

The setting of the CONTINUE_AFTER_RAISERROR option is used to control behavior following a RAISERROR statement *only* if the ON_TSQL_ERROR option is set to CONDITIONAL (the default). If you set the ON_TSQL_ERROR option to STOP or CONTINUE, the ON_TSQL_ERROR setting takes precedence over the CONTINUE_AFTER_RAISERROR setting.

See also	"ON_TSQL_ERROR option [compatibility]" on page 675

CONVERSION_ERROR option [compatibility]

Function	Controls the reporting of data type conversion failures on fetching information from the database.
Allowed values	**ON**, **OFF**
Default	**ON**
Description	This option controls whether data type conversion failures, when data is fetched from the database or inserted into the database, are reported by the database as errors (CONVERSION_ERROR set to **ON**) or as a warning (CONVERSION_ERROR set to **OFF**).

When CONVERSION_ERROR is set to **ON**, the SQLE_CONVERSION_ERROR error is generated. If the option is set to **OFF**, the warning SQLE_CANNOT_CONVERT is produced.

If conversion errors are reported as warnings only, the NULL value is used in place of the value that could not be converted. In embedded SQL, an indicator variable is set to -2 for the column or columns that cause the error.

COOPERATIVE_COMMIT_TIMEOUT option [database]

Function	Governs when a COMMIT entry in the transaction log is written to disk.
Allowed values	*Integer*, in milliseconds
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.
Default	**250**
Description	This option has meaning only when COOPERATIVE_COMMITS is set to **ON**. The database server waits for the specified number of milliseconds for other connections to fill a page of the log before writing to disk. The default setting is 250 milliseconds.

COOPERATIVE_COMMITS option [database]

Function	Controls when commits are written to disk.
Allowed values	**ON or OFF**
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.
Default	**ON**

Description

If COOPERATIVE_COMMITS is set to **OFF**, a COMMIT is written to disk as soon as the database server receives it, and the application is then allowed to continue.

If COOPERATIVE_COMMITS is set to **ON** (the default) and if there are other active connections, the database server does not immediately write the COMMIT to the disk. Instead, the application waits for up to the maximum length set by the COOPERATIVE_COMMIT_TIMEOUT option for something else to put on the pages before they are written to disk.

Setting COOPERATIVE_COMMITS to **ON**, and increasing the COOPERATIVE_COMMIT_TIMEOUT setting, increases overall database server throughput by cutting down the number of disk I/Os, but at the expense of a longer turnaround time for each individual connection.

If both COOPERATIVE_COMMITS and DELAYED_COMMITS are set to **ON**, and the COOPERATIVE_COMMIT_TIMEOUT interval passes without the pages getting written, the application is resumed (as if the commit had worked), and the remaining interval (DELAYED_COMMIT_TIMEOUT - COOPERATIVE_COMMIT_TIMEOUT) is used as a DELAYED_COMMIT interval. The pages will then be written, even if they are not full.

DATE_FORMAT option [compatibility]

Function

Sets the format for dates retrieved from the database.

Allowed values

String

Scope

Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.

Default

'YYYY-MM-DD'. This corresponds to ISO date format specifications.

Description

The format is a string using the following symbols:

Symbol	**Description**
yy	Two digit year
yyyy	Four digit year
mm	Two digit month, or two digit minutes if following a colon (as in hh:mm)
mmm[*m*...]	Character short form for months—as many characters as there are "m"s
d	Single digit day of week, (0 = Sunday, 6 = Saturday)

Symbol	Description
dd	Two digit day of month
ddd[*d*...]	Character short form for day of the week
hh	Two digit hours
nn	Two digit minutes
ss[.*ss*..]	Seconds and parts of a second
aa	AM or PM (12 hour clock)
pp	PM if needed (12 hour clock)
jjj	Day of the year, from 1 to 366

Each symbol is substituted with the appropriate data for the date that is being formatted.

For symbols that represent character data (such as *mmm*), you can control the case of the output as follows:

♦ Type the symbol in all upper case to have the format appear in all upper case. For example, `MMM` produces `JAN`.

♦ Type the symbol in all lower case to have the format appear in all lower case. For example, `mmm` produces `jan`.

♦ Type the symbol in mixed case to have Adaptive Server Anywhere choose the appropriate case for the language that is being used. For example, in English, typing `Mmm` produces `May`, while in French it produces `mai`.

For symbols that represent numeric data, you can control zero-padding with the case of the symbols:

♦ Type the symbol in same-case (such as `MM` or `mm`) to allow zero padding. For example, `yyyy/mm/dd` could produce `2002/01/01`.

♦ Type the symbol in mixed case (such as `Mm`) to suppress zero padding. For example, `yyyy/Mm/Dd` could produce `2002/1/1`.

Examples

♦ The following table illustrates DATE_FORMAT settings, together with the output from the following statement, executed on Thursday May 21, 2001.

```
SELECT CURRENT DATE
```

DATE_FORMAT	SELECT CURRENT DATE
yyyy/mm/dd/ddd	2001/05/21/thu
jjj	141
mmm yyyy	May 1998
mm-yyyy	05-1998

DATE_ORDER option [compatibility]

Function	Controls the interpretation of date formats.
Allowed values	**'MDY'**, **'YMD'**, **'DMY'**
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.
Default	**YMD**. This corresponds to ISO date format specifications. For Open Client and JDBC connections, the default is set to **MDY**.
Description	The database option DATE_ORDER is used to determine whether 10/11/12 is Oct 11 1912, Nov 12 1910, or Nov 10 1912. The option can have the value 'MDY', 'YMD', or 'DMY'.

DEBUG_MESSAGES option [database]

Function	Controls whether or not MESSAGE statements that include a DEBUG ONLY clause are executed.
Allowed values	**ON**, **OFF**
Default	**OFF**
Description	This option allows you to control the behavior of debugging messages in stored procedures and triggers that contain a MESSAGE statement with the DEBUG ONLY clause specified. By default, this option is set to OFF and debugging messages do not appear when the MESSAGE statement is executed. By setting DEBUG_MESSAGES to ON, you can enable the debugging messages in all stored procedures and triggers.

Note
DEBUG ONLY messages are inexpensive when the DEBUG_MESSAGES option is set to OFF, so these statements can usually be left in stored procedures on a production system. However, they should be used sparingly in locations where they would be executed frequently; otherwise, they may result in a small performance penalty.

See also	♦ "MESSAGE statement" [*ASA SQL Reference*, page 549]

DEDICATED_TASK option [database]

Function	Dedicates a request handling task to handling requests from a single connection.
Allowed values	**ON**, **OFF**
Scope	Can be set as a temporary option only, for the duration of the current connection. DBA permissions are required to set this option.
Default	**OFF**
Description	When the DEDICATED_TASK connection option is set to ON, a request handling task is dedicated exclusively to handling requests for the connection. By pre-establishing a connection with this option enabled, you will be able to gather information about the state of the database server if it becomes otherwise unresponsive.

DEFAULT_ISQL_ENCODING option [Interactive SQL]

Function	Specifies the code page that should be used by READ, INPUT, and OUTPUT statements.
Allowed values	*identifier* or *string*
Scope	Can be set as a temporary option only, for the duration of the current connection.
Default	Use system code page (empty string)
Description	This option is used to specify the code page to use when reading or writing files. It cannot be set permanently. The default code page is the default code page for the platform you are running on. On English Windows machines, the default code page is 1252.

Interactive SQL determines the code page that is used for a particular INPUT, OUTPUT, or READ statement as follows, where code page values occurring earlier in the list take precedence over those occurring later in the list:

- the code page specified in the ENCODING clause of the INPUT, OUTPUT, or READ statement
- the code page specified with the DEFAULT_ISQL_ENCODING option (if this option is set)
- the code page specified with the -codepage command-line option when Interactive SQL was started

- the default code page for the computer Interactive SQL is running on

☞ For a complete list of supported code pages, see "Supported code pages" on page 363.

See also

- "READ statement [Interactive SQL]" [*ASA SQL Reference*, page 571]
- "INPUT statement [Interactive SQL]" [*ASA SQL Reference*, page 523]
- "OUTPUT statement [Interactive SQL]" [*ASA SQL Reference*, page 556]
- "Pieces in the character set puzzle" on page 323

Example

- Set the encoding to **UTF-16** (for reading Unicode files):

```
SET TEMPORARY OPTION DEFAULT_ISQL_ENCODING = 'UTF-16'
```

DEFAULT_TIMESTAMP_INCREMENT option [database]

Function	Specifies the number of microseconds to add to a column of type TIMESTAMP in order to keep values in the column unique.
Allowed values	*Integer*, between 1 and 60 000 000 inclusive.
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.
Default	**1**

Description

Since a TIMESTAMP value is precise to six decimal places in Adaptive Server Anywhere, by default 1 microsecond (0.000001 of a second) is added to differentiate between two identical TIMESTAMP values.

Some software, such as Microsoft Access, truncates TIMESTAMP values to three decimal places, making valid comparisons a problem. You can set the TRUNCATE_TIMESTAMP_VALUES option to ON to specify the number of decimal place values Adaptive Server Anywhere stores to maintain compatibility.

For MobiLink synchronization, this option must be set prior to performing the first synchronization.

See also

"TRUNCATE_TIMESTAMP_VALUES option [database]" on page 697

DELAYED_COMMIT_TIMEOUT option [database]

Function	Determines when the server returns control to an application following a COMMIT.
Allowed values	*Integer*, in milliseconds.
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.

Default	**500**
Description	This option has meaning only when DELAYED_COMMITS is set to **ON**. it governs when a COMMIT entry in the transaction log is written to disk. With DELAYED_COMMITS set to **ON**, the database server waits for the number of milliseconds set in the DELAYED_COMMIT_TIMEOUT option for other connections to fill a page of the log before writing the current page contents to disk. ☞ For more information, see "DELAYED_COMMITS option [database]" on page 651.

DELAYED_COMMITS option [database]

Function	Determines when the server returns control to an application following a COMMIT.
Allowed values	**ON**, **OFF**
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.
Default	**OFF**. This corresponds to ISO COMMIT behavior.
Description	When set to **ON**, the database server replies to a COMMIT statement immediately instead of waiting until the transaction log entry for the COMMIT has been written to disk. When set to **OFF**, the application must wait until the COMMIT is written to disk. When this option is **ON**, the log is written to disk when the log page is full or according to the DELAYED_COMMIT_TIMEOUT option setting, whichever is first. There is a slight chance that a transaction may be lost even though committed if a system failure occurs after the server replies to a COMMIT, but before the page is written to disk. Setting DELAYED_COMMITS to **ON**, and the DELAYED_COMMIT_TIMEOUT option to a high value, promotes a quick response time at the cost of security. If both COOPERATIVE_COMMITS and DELAYED_COMMITS are set to **ON**, and if the COOPERATIVE_COMMIT_TIMEOUT interval passes without the pages getting written, the application is resumed (as if the commit had worked), and the remaining interval (DELAYED_COMMIT_TIMEOUT - COOPERATIVE_COMMIT_TIMEOUT) is used as a DELAYED_COMMIT interval after which the pages will be written, even if they are not full.

DELETE_OLD_LOGS option [replication]

Function	Controls whether transaction logs are deleted when their messages have been replicated or synchronized.
Allowed values	**ON**, **OFF**, **DELAY**
Default	**OFF**
Description	This option is used at MobiLink clients, by SQL Remote, and by the Adaptive Server Anywhere Replication Agent. The default setting is **OFF**. When it is set to **ON**, each old transaction log is deleted when all the changes it contains have been sent and confirmed as received. When it is set to **DELAY**, each old transaction log with a file name indicating that it was created on the current day is not deleted. ☞ For more information about how to use the DELETE_OLD_LOGS option in conjunction with the Backup statement to delete old copies of transaction logs, see the "BACKUP statement" [*ASA SQL Reference*, page 307].

DESCRIBE_JAVA_FORMAT option [Interactive SQL]

Function	Controls whether Java objects are interpreted as strings (for display) or as binary (for loading and unloading). This option is of use only if you are connecting to a version 8 or earlier database server, as Java objects are not supported in later versions of the software.
Allowed values	**varchar**, **binary**
Default	**varchar**
Description	When set to **varchar**, Interactive SQL converts all data fetched from the database to strings. The database server calls the `toString()` method on Java columns to provide formatted output for the Results tab in the Results pane. When set to **binary**, Interactive SQL does no data conversion.

DIVIDE_BY_ZERO_ERROR option [compatibility]

Function	Controls the reporting of division by zero.
Allowed values	**ON**, **OFF**
Default	**ON**
Description	This option indicates whether division by zero is reported as an error. If the

option is set ON, then division by zero results in an error with SQLSTATE 22012.

If the option is set OFF, division by zero is not an error. Instead, a NULL is returned.

ECHO option [Interactive SQL]

Function	Controls whether statements are echoed to the log file before they are executed.
Allowed values	**ON**, **OFF**
Default	**ON**
Description	This option is most useful when you use the READ statement to execute a Interactive SQL command file. Logging must be turned on in order for this option to take effect.

☞ For information about turning on logging, see "START LOGGING statement [Interactive SQL]" [*ASA SQL Reference*, page 627].

ESCAPE_CHARACTER option [compatibility]

This option is reserved for system use. Do not change the setting of this option.

EXCLUDE_OPERATORS option [database]

This option is reserved for system use. Do not change the setting of this option.

EXTENDED_JOIN_SYNTAX option [database]

Function	Controls whether queries with duplicate correlation names syntax for multi-table joins are allowed, or reported as an error.
Allowed values	**ON**, **OFF**
Default	**ON**
Description	If this option is set to **ON** then Adaptive Server Anywhere allows duplicate correlation names to be used in the null-supplying side of the outer joins. The tables or views specified with the same correlation name are interpreted as the same instance of the table or view.

The following FROM clause illustrates the Adaptive Server Anywhere interpretation of a join using duplicate correlation names:

```
( R left outer join T on (C1), T  join S on ( C2 ) )
```

where C1 and C2 are search conditions.

If the option is set to **ON**, this join is interpreted as follows:

```
( R left outer join T on ( C1 ) ) join S on ( C2 )
```

If the option is set to **OFF**, the following error is generated:

```
ASA Error -137:  Table 'T' requires a unique correlation
name.
```

Note
To see the result of eliminating duplicate correlation names, you can view the rewritten statement using the REWRITE function with the second argument set to ANSI.

See also
“REWRITE function [Miscellaneous]” [*ASA SQL Reference*, page 214]

FIRE_TRIGGERS option [compatibility]

Function
Controls whether triggers are fired in the database.

Allowed values
ON, **OFF**

Default
ON

Description
When set to **ON**, triggers are fired. When set to **OFF**, no triggers are fired, including referential integrity triggers (such as cascading updates and deletes). Only a user with DBA authority can set this option. The option is overridden by the -gf command-line option, which turns off all trigger firing regardless of the FIRE_TRIGGERS setting.

This option is relevant when replicating data from Adaptive Server Enterprise to Adaptive Server Anywhere because all actions from Adaptive Server Enterprise transaction logs are replicated to Adaptive Server Anywhere, including actions carried out by triggers.

FIRST_DAY_OF_WEEK option [database]

Function
Sets the numbering of the days of the week.

Allowed values
1 | 2 | 3 | 4 | 5 | 6 | 7

Default
7 (Sunday is the first day of the week)

Description
The values have the following meaning:

Value	Meaning
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday
7	Sunday

FLOAT_AS_DOUBLE option [compatibility]

Function	Controls the interpretation of the FLOAT keyword.
Allowed values	**ON**, **OFF**
Default	**OFF** **ON** for Open Client and JDBC connections

Description

The FLOAT_AS_DOUBLE option makes the FLOAT keyword behave like Adaptive Server Enterprise's FLOAT keyword when a precision is not specified.

When enabled (set to **ON**), all occurrences of the keyword FLOAT are interpreted as equivalent to the keyword DOUBLE within SQL statements.

By default, Adaptive Server Anywhere FLOAT values are interpreted by Adaptive Server Enterprise as REAL values. Since Adaptive Server Enterprise treats its own FLOAT values as DOUBLE, enabling this option makes Adaptive Server Anywhere treat FLOAT values in the same way Enterprise treats FLOAT values.

REAL values are four bytes; DOUBLE values are eight bytes. According to the ANSI SQL/92 specification, FLOAT can be interpreted based on the platform. It is up to the database to decide what size the value is, so long as it can handle the necessary precision. Adaptive Server Enterprise and Adaptive Server Anywhere exhibit different default behavior.

The FLOAT_AS_DOUBLE option takes effect only when no precision is specified. For example, the following statement is not affected by the option setting:

```
CREATE TABLE t1(
   c1 float(5)
)
```

The following statement is affected by the option setting:

```
CREATE TABLE t2(
   c1 float)
// affected by option setting
```

FOR_XML_NULL_TREATMENT option [database]

Function	Controls the treatment of NULL values in queries that use the FOR XML clause.
Allowed values	**EMPTY**, **OMIT**
Default	**OMIT**
Description	If you execute a query that includes the FOR XML clause, the FOR_XML_NULL_TREATMENT option determines how NULL values are treated. By default, elements and attributes that contain NULL values are omitted from the result. Setting this option to EMPTY generates empty elements or attributes if the value is NULL.
See also	"Using the FOR XML clause to retrieve query results as XML" [*ASA SQL User's Guide*, page 525] "SELECT statement" [*ASA SQL Reference*, page 597]

FORCE_VIEW_CREATION option [database]

This option is reserved for system use. Do not change the setting of this option.

Caution
The FORCE_VIEW_CREATION option should only be used within a reload.sql script. This option is used by the Unload utility (dbunload) and should not be set explicitly.

GLOBAL_DATABASE_ID option [database]

Function	For use in generating unique primary keys in a replication environment. Controls the beginning of the range of values for columns created with DEFAULT GLOBAL AUTOINCREMENT.
Allowed values	*Integer*
Default	**2147483647**
Scope	Can be set for the PUBLIC group only. DBA authority required.

Description	If you create a column with DEFAULT GLOBAL AUTOINCREMENT, its value is incremented. The initial value is determined by the GLOBAL_DATABASE_ID value and the partition size.

☞ For more information on the partition size, see "CREATE TABLE statement" [*ASA SQL Reference*, page 407].

Setting GLOBAL_DATABASE_ID to the default value indicates that GLOBAL DEFAULT AUTOINCREMENT is disabled. In this case NULL is generated as a default.

You can find the value of the option in the current database using the following statement:

```
SELECT db_property( 'GlobalDBId' )
```

This feature is of particular use in replication environments to ensure unique primary keys.

☞ For more information, see "Ensuring unique primary keys" [*SQL Remote User's Guide*, page 129] and "Maintaining unique primary keys using global autoincrement" [*MobiLink Administration Guide*, page 57].

See also

"CREATE TABLE statement" [*ASA SQL Reference*, page 407]

"Database-level properties" on page 733

"Ensuring unique primary keys" [*SQL Remote User's Guide*, page 129]

INPUT_FORMAT option [Interactive SQL]

Function: Sets the default data format expected by the INPUT statement.

Allowed values: *String*. See below.

Default: **ASCII**

Description: Certain file formats contain information about column names and types. Using this information, the INPUT statement will create the database table if it does not already exist. This is a very easy way to load data into the database. The formats that have enough information to create the table are: DBASEII, DBASEIII, FOXPRO, and LOTUS.

Allowable input formats are:

- **ASCII** Input lines are assumed to be ASCII characters, one row per line, with values separated by commas. Alphabetic strings may be enclosed in apostrophes (single quotes) or quotation marks (double quotes). Strings containing commas must be enclosed in either single or double quotes. If single or double quotes are used, double the quote character to use it within the string. Optionally, you can use the DELIMITED BY clause to specify a different delimiter string than the default, which is a comma (,).

Three other special sequences are also recognized. The two characters **\n** represent a newline character, \\ represents a single backslash character, and the sequence **\x*DD*** represents the character with hexadecimal code DD.

- **DBASE** The file is in dBASE II or dBASE III format. Interactive SQL will attempt to determine which format, based on information in the file. If the table doesn't exist, it will be created.
- **DBASEII** The file is in dBASE II format. If the table doesn't exist, it will be created.
- **DBASEIII** The file is in dBASE III format. If the table doesn't exist, it will be created.
- **EXCEL** Input file is in the format of Microsoft Excel 2.1. If the table doesn't exist, it will be created.
- **FIXED** Input lines are in fixed format. The width of the columns can be specified using the COLUMN WIDTHS clause. If they are not specified, column widths in the file must be the same as the maximum number of characters required by any value of the corresponding database column's type.
- **FOXPRO** The file is in FoxPro format. The FoxPro memo field is different than the dBASE memo field. If the table doesn't exist, it will be created.
- **LOTUS** The file is a Lotus WKS format worksheet. INPUT assumes that the first row in the Lotus WKS format worksheet consists of column names. If the table doesn't exist, it will be created. In this case, the types and sizes of the columns created may not be correct because the information in the file pertains to a cell, not to a column.

See also "INPUT statement [Interactive SQL]" [*ASA SQL Reference*, page 523]

INTEGRATED_SERVER_NAME option [database]

Function Specifies the name of the Domain Controller server used for looking up Windows user group membership for integrated logins.

Allowed values *String*

Scope Can be set for the PUBLIC group only. DBA authority is required to set this option.

Default	NULL
Description	This option allows the DBA to specify the name of the Domain Controller server that is used to look up group membership when using Windows user groups for integrated logins. By default, the machine that Adaptive Server Anywhere is running on is used for verifying group membership.
See also	"Creating integrated logins for Windows user groups" on page 75
	"GRANT statement" [*ASA SQL Reference,* page 503]
Example	The following example specifies that group membership is verified on the machine server-1.

```
SET OPTION PUBLIC.INTEGRATED_SERVER_NAME = '\\server-1'
```

ISOLATION_LEVEL option [compatibility]

Function	Controls the locking isolation level.
Allowed values	**0**, **1**, **2**, **3**
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.
Default	**0**
	1 for Open Client and JDBC connections
Description	This option controls the locking isolation level as follows.

- **0** Allow dirty reads, non-repeatable reads, and phantom rows.
- **1** Prevent dirty reads. Allow non-repeatable reads and phantom rows.
- **2** Prevent dirty reads and guarantee repeatable reads. Allow phantom rows.
- **3** Serializable. Do not allow dirty reads, guarantee repeatable reads, and do not allow phantom rows.

☞ For more information, see "Isolation levels and consistency" [*ASA SQL User's Guide,* page 106].

ISQL_COMMAND_TIMING option [Interactive SQL]

Function	Controls whether SQL statements are timed or not.
Allowed values	**ON**, **OFF**
Default	**ON**

Description

This boolean option controls whether SQL statements are timed or not. If you set the option to **ON**, the time of execution appears in the Messages pane after you execute a statement. If you set the option to **OFF**, the time does not appear.

You can also set this option on the Messages tab of the Options dialog.

ISQL_ESCAPE_CHARACTER option [Interactive SQL]

Function

Controls the escape character used in place of unprintable characters in data exported to ASCII files.

Allowed values

Any single character

Default

A backslash (\ )

Description

When Interactive SQL exports strings that contain unprintable characters (such as a carriage return), it converts each unprintable character into a hexadecimal format and precedes it with an escape character. The character you specify for this setting is used in the output if your OUTPUT statement does not contain an ESCAPE CHARACTER clause. This setting is used only if you are exporting to an ASCII file.

Example

- Create a table that contains one string value with an embedded carriage return (denoted by the “\n” in the INSERT statement). Then export the data to *c:\escape.txt* with a # sign as the escape character.

  ```
  CREATE TABLE escape_test( TEXT varchar(10 ) );
  INSERT INTO escape_test VALUES( 'one\ntwo' );
  SET TEMPORARY OPTION ISQL_ESCAPE_CHARACTER='#';
  SELECT * FROM escape_test;
  OUTPUT TO c:\escape.txt FORMAT ASCII
  ```

  This code places the following data in *escape.txt*:

  ‘one#x0Atwo’

  where # is the escape character and *x0A* is the hexadecimal equivalent of the “\n” character.

  The start and end characters (in this case, single quotation marks) depend on the ISQL_QUOTE setting.

ISQL_FIELD_SEPARATOR option [Interactive SQL]

Function

Controls the default string used for separating values in data exported to ASCII files.

Allowed values

String

Default

A comma (**,**)

Description	Controls the default string used for separating (or delimiting) values in data exported to ASCII files. If an OUTPUT statement does not contain a DELIMITED BY clause, the value of this setting is used.

Example

♦ Set the field separator to a colon in the data exported to *c:\employee.txt*.

```
SET TEMPORARY OPTION ISQL_FIELD_SEPARATOR=':';
SELECT emp_lname, emp_fname FROM employee WHERE emp_id <
         150;
OUTPUT TO c:\employee.txt FORMAT ASCII
```

This code places the following data in *employee.txt*:

'Whitney': 'Fran'

'Cobb':'Matthew'

'Chin':'Philip'

'Jordan':'Julie'

The start and end characters (in this case, single quotation marks) depend on the ISQL_QUOTE setting.

ISQL_LOG option [Interactive SQL]

Function	Controls logging behavior.
Allowed values	*String*, containing a file name
Default	*Empty string*
Description	If ISQL_LOG is set to a non-empty string, all Interactive SQL statements are added to the end of the named file. Otherwise, if ISQL_LOG is set to the empty string, Interactive SQL statements are not logged.

Individual session only
This option logs an individual Interactive SQL session only.

☞ For a description of the transaction log that logs all changes to the database by all users, see "Backup and Data Recovery" on page 373.

ISQL_PLAN option [Interactive SQL]

Function	Controls the type of access plan that appears on the Plan tab and the UltraLite Plan tab in the Interactive SQL Results pane after you execute statements.
Allowed values	**SHORT**, **LONG**, **GRAPHICAL**, **GRAPHICALWITHSTATISTICS**, **NONE**
Default	**SHORT**

Description	This option allows you to specify the type of access plan that appears for SELECT, INSERT, UPDATE, and DELETE statements. To set the level of detail for the plan the optimizer provides, you can choose one of the following values:

SHORT provides basic information about the access plan.

LONG provides detailed information about the access plan.

GRAPHICAL provides a tree diagram of the query. You can click a node in the plan diagram to see details about that part of the query.

GRAPHICALWITHSTATISTICS provides a tree diagram of the query, as well as statistics which indicate the resources used by the part of the query that is selected. The UltraLite plan treats this value the same as the GRAPHICAL value.

NONE means no optimizer information can be accessed. This parameter is deprecated.

The plan is computed only when you click the Plan tab.

You can also specify the plan type on the Plan tab of the Interactive SQL Options dialog.

Note that the access plan that appears may not be the same plan that was used when the statement was executed. The plan is fetched after the SQL statement is executed, at which point the optimizer may have new information available. This means that the optimizer may return a different plan than the one it used to execute the statement.

ISQL_PRINT_RESULT_SET option [Interactive SQL]

Function

This option specifies which result set(s) are printed when a *.SQL* file is run.

The ISQL_PRINT_RESULT_SET option takes effect only when you run the Interactive SQL [dbisql] utility within a command window (for example, when running a *.SQL* file).

Allowed values

LAST, **ALL**, **NONE**

Default

LAST

Description

This option allows you to specify which result set(s) are printed when a *.SQL* file is run. It is remembered on a per-machine basis (not per-user), and is case-insensitive.

You can choose one of the following print options:

LAST prints the result set from the last statement in the file.

ALL prints result sets from each statement in the file which returns a result set.

NONE does not print any result sets.

Although this option has no affect when Interactive SQL is running in windowed mode, you can still view and set the option in windowed mode. Choose Tools ➤ Options, then select the appropriate action in the Console Mode box on the Results page. Remember to click Make Permanent if you want to set this option for all Interactive SQL sessions. Otherwise, the change will be discarded when you exit Interactive SQL.

ISQL_QUOTE option [Interactive SQL]

Function	Controls the default string that begins and ends all strings in data exported to ASCII files.
Allowed values	*String*
Default	A single apostrophe (**’**)
Description	Controls the default string that begins and ends all strings in data exported to ASCII files. If an OUTPUT statement does not contain a QUOTE clause, this value is used by default.

Example

♦ To change the default string that begins and ends all strings to a double quote character.

```
SET TEMPORARY OPTION ISQL_QUOTE='"';
SELECT emp_lname, emp_fname FROM employee WHERE emp_id <
         150;
OUTPUT TO c:\employee.txt FORMAT ASCII
```

This code places the following data in *employee.txt*:

”Whitney”, “Fran”

”Cobb”,”Matthew”

”Chin”,”Philip”

”Jordan”,”Julie”

The separator characters (in this case, commas) depend on the ISQL_FIELD_SEPARATOR setting.

JAVA_HEAP_SIZE option [database]

Function	To limit the memory used by Java applications for a connection.
Allowed values	*Integer*

Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately. DBA permissions are required to set this option for *any* connection.
Default	**1 000 000**
Description	This option sets the maximum size (in bytes) of the memory that is allocated to Java applications on a per-connection basis. Per-connection memory allocations typically consist of the user's working set of allocated Java variables and Java application stack space.
	While a Java application is executing on a connection, the per-connection allocations come out of the fixed cache of the database server, so it is important that a run-away Java application is prevented from using up too much memory.

JAVA_INPUT_OUTPUT option [database]

Function	Enables file access from Java in the database
Allowed values	**ON**, **OFF**
	Supported on Windows NT only.
Scope	DBA authority required.
Default	**OFF**
Description	By default, the **java.io** package is only partially supported. In particular, classes that handle file access are disabled. If JAVA_INPUT_OUTPUT is set to **ON**, Java file access classes in **java.io** are enabled.
See also	"Security management for Java" [*ASA Programming Guide,* page 96]

JAVA_NAMESPACE_SIZE option [database]

Function	Limits the memory used by Java applications for a connection.
Allowed values	*Integer*
Scope	Can be set for the PUBLIC group only. DBA authority required. Takes effect immediately.
Default	**4 000 000**
Description	This option sets the maximum size (in bytes) of the memory that is allocated to Java applications on a per-database basis.
	Per-database memory allocations include Java class definitions. Because class definitions are effectively read-only, they are shared between

connections. Consequently, their allocations come right out of the fixed cache, and this option sets a limit on the size of these allocations.

LOG_DEADLOCKS option [database]

Function	Controls whether deadlock reporting is turned on or off.
Allowed values	**ON**, **OFF**
Scope	Can be set for the PUBLIC group only. DBA authority required. Takes effect immediately.
Default	**ON**
Description	When this option is set to ON, the database server logs information about deadlocks in an internal buffer. The size of the buffer is fixed at 10,000 bytes. You can view the deadlock information using the sa_report_deadlocks stored procedure. The contents of the buffer are cleared when this option is set to OFF. When deadlock occurs, information is reported for only those connections involved in the deadlock. The order in which connections are reported is based on which connection is waiting for which row. For thread deadlocks, information is reported about all connections.
See also	"sa_report_deadlocks system procedure" [*ASA SQL Reference*, page 825] "Determining who is blocked" [*ASA SQL User's Guide*, page 114]

LOGIN_MODE option [database]

Function	Controls the use of integrated logins for the database.
Allowed values	**Standard**, **Mixed**, **Integrated**
Scope	Can be set for the PUBLIC group only. DBA authority required. Takes effect immediately.
Default	**Standard**
Description	This option specifies whether integrated logins are permitted. The following values are accepted (the values are case insensitive):

- **Standard** This is the default setting, which does not permit integrated logins. An error occurs if an integrated login connection is attempted.
- **Mixed** With this setting, both integrated logins and standard logins are allowed.
- **Integrated** With this setting, all logins to the database must be made using integrated logins.

Caution

Setting the LOGIN_MODE database option to ***Integrated*** *restricts connections to only those Windows users or groups who have been granted an integrated login mapping. Attempting to connect with a user ID and password generates an error. The only exceptions to this are users with DBA authority (full administrative rights).*

☞ For more information on integrated logins see "Using integrated logins" on page 74.

LOGIN_PROCEDURE option [database]

Function

A login procedure that sets connection compatibility options at startup. By default the procedure calls the sp_login_environment procedure to determine which options to set.

Allowed values

String

Scope

DBA authority required.

Default

sp_login_environment

Description

This login procedure calls the sp_login_environment procedure at run time to determine the database connection settings.

You can customize the default database option settings by creating a new procedure and setting LOGIN_PROCEDURE to call the new procedure. You should not edit either sp_login_environment or sp_tsql_environment.

Examples

♦ The following example shows how you can disallow a connection by signaling the INVALID_LOGON error.

```
create procedure DBA.login_check()
   begin
      declare INVALID_LOGON exception for sqlstate '28000';
      // Allow a maximum of 3 concurrent connections
      if( db_property('ConnCount') > 3 ) then
      signal INVALID_LOGON;
      else
     call sp_login_environment;
      end if;
   end
   go
   grant execute on DBA.login_check to PUBLIC
   go
   set option PUBLIC.Login_procedure='DBA.login_check'
   go
```

☞ For an alternate way to disallow connections, see "RAISERROR statement [T-SQL]" [*ASA SQL Reference*, page 569].

- The following example shows how you can block connection attempts if the number of failed connections for a user exceeds 3 within a 30 minute period. All blocked attempts during the block out period receive an invalid password error and are logged as failures. The log is kept long enough for a DBA to analyze it.

```
create table DBA.ConnectionFailure(
    pk int primary key default autoincrement,
    user_name char(128) not null,
    tm timestamp not null default current timestamp
)
go
create index ConnFailTime on DBA.ConnectionFailure(
    user_name, tm )
go
create event ConnFail type ConnectFailed
handler
begin
    declare usr char(128);
    set usr = event_parameter( 'User' );
    // Put a limit on the number of failures logged.
    if (select count(*) from DBA.ConnectionFailure
        where user_name = usr
        and tm >= dateadd( minute, -30,
            current timestamp )) < 20 then
        insert into DBA.ConnectionFailure( user_name )
            values( usr );
        commit;
        // Delete failures older than 7 days
        delete DBA.ConnectionFailure
        where user_name = usr
        and tm < dateadd( day, -7, current timestamp );
        commit;
    end if;
end
go

create procedure DBA.login_check()
begin
    declare usr char(128);
    declare INVALID_LOGON exception for sqlstate '28000';
    set usr = connection_property( 'Userid' );
    // Block connection attempts from this user
    // if 3 or more failed connection attempts have occurred
    // within the past 30 minutes.
```

```
        if (select count(*) from DBA.ConnectionFailure
            where user_name = usr
            and tm >= dateadd( minute, -30,
                current timestamp ) ) >= 3 then
            signal INVALID_LOGON;
        else
            call sp_login_environment;
        end if;
    end
    go
    grant execute on DBA.login_check to PUBLIC
    go

    set option PUBLIC.Login_procedure='DBA.login_check'
    go
```

MAX_CURSOR_COUNT option [database]

Function	A resource governor to limit the maximum number of cursors that a connection can use at once.
Allowed values	*Integer*
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately. DBA permissions are required to set this option for *any* connection.
Default	**50**
Description	This resource governor allows a DBA to limit the number of cursors per connection that a user can use. If an operation would exceed the limit for a connection, an error is generated, indicating that the governor for the resource has been exceeded.

If a connection executes a stored procedure, that procedure is executed under the permissions of the procedure owner. However, the resources used by the procedure are assigned to the current connection.

You can remove resource limits by setting the option to 0 (zero).

MAX_HASH_SIZE option [database] (deprecated)

Function	Specify the default maximum hash size for new indexes.
Allowed values	*Integer*, from 2 to 64 inclusive.
Scope	Can be set for the PUBLIC group only. Takes effect immediately. DBA permissions are required to set this option.
Default	**10**

Description	This option controls the default hash size for indexes. This option is deprecated.
See also	"CREATE INDEX statement" [*ASA SQL Reference,* page 368]
	"CREATE TABLE statement" [*ASA SQL Reference,* page 407]

MAX_PLANS_CACHED option [database]

Function	Specify the maximum number of execution plans to be stored in a cache.
Allowed values	*Integer*
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately. DBA permissions are required to set this option for the PUBLIC group.
Default	**20**

Description

This option specifies the maximum number of plans cached for each connection. The optimizer caches the execution plan for queries, INSERT, UPDATE, and DELETE statements that are performed inside stored procedures, functions, and triggers. After a statement in a stored procedure, stored function, or trigger is executed several times by a connection, the optimizer builds a reusable plan for the statement.

Reusable plans do not use the values of host variables for selectivity estimation or rewrite optimizations. As a result of this, the reusable plan can have a higher cost than if the statement was re-optimized. When the cost of the reusable plan is close to the best observed cost for a statement, the optimizer adds the plan to the plan cache.

The cache is cleared when you execute statements, such as CREATE TABLE and DROP TABLE, that modify the table schema. Statements that reference declared temporary tables are not cached.

Setting this option to 0 disables plan caching.

See also	"Access plan caching" [*ASA SQL User's Guide,* page 404]
	"Connection-level properties" on page 713

MAX_RECURSIVE_ITERATIONS option [database]

Function	Limits the maximum number of iterations a recursive common table expression can make.
Allowed values	*Integer*
Scope	Can be set for an individual connection or for the PUBLIC group. Takes

	effect immediately. DBA permissions are required to set this option for the PUBLIC group.
Default	**100**
Description	Computation of a recursive common table expression aborts and an error is generated if they fail to terminate within the specified number of iterations. Recursive subqueries often increase geometrically in the amount of resources required for each additional iteration. Set this option so as to limit the amount of time and resources that will be consumed before infinite recursion is detected, yet permit your recursive common table expressions to work as intended. Setting this option to 0 disables recursive common table expressions.
See also	"Common Table Expressions" [*ASA SQL User's Guide*, page 307]

MAX_STATEMENT_COUNT option [database]

Function	A resource governor to limit the maximum number of prepared statements that a connection can use at once.
Allowed values	*Integer >=0*
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately. DBA permissions are required to set this option for *any* connection.
Default	**50**
Description	This resource governor allows a DBA to limit the number of prepared statements per connection a user can use. If an operation would exceed the limit for a connection, an error is generated, indicating that the governor for the resource has been exceeded. If a connection executes a stored procedure, that procedure is executed under the permissions of the procedure owner. However, the resources used by the procedure are assigned to the current connection. You can remove resource limits by setting the option to 0 (zero).

MAX_WORK_TABLE_HASH_SIZE option [database] (deprecated)

Function	Specify the maximum hash size used during query optimization for internal temporary tables.
Allowed values	*Integer*, from 2 to 64 inclusive
Scope	Can be set for the PUBLIC group only. Takes effect immediately. DBA permissions are required to set this option.

Default	**20**
Description	In general, you should not need to use this option as the query optimizer allocates hash sizes for the internal temporary tables based on the data distribution within the table. The option allows you to override the optimizer behavior to either increase the maximum hash size it can use beyond 20, or to restrict it further. This option is deprecated.
See also	"Use of work tables in query processing" [*ASA SQL User's Guide*, page 190].

MIN_PASSWORD_LENGTH option [database]

Function	Sets the minimum length for new passwords in the database.
Allowed values	*Integer*, greater than or equal to zero. The value is in bytes. For single-byte character sets, this is the same as the number of characters.
Scope	Can be set for the PUBLIC group. Takes effect immediately. DBA permissions are required to set this option.
Default	**0** characters
Description	This option allows the database administrator to impose a minimum length on all new passwords for greater security. Existing passwords are not affected. Passwords have a maximum length of 255 bytes.

Example

♦ Set the minimum length for new passwords to 6 bytes.

```
SET OPTION PUBLIC.MIN_PASSWORD_LENGTH = 6
```

NEAREST_CENTURY option [compatibility]

Function	Controls the interpretation of two-digit years in string-to-date conversions.
Allowed values	*Integer*, between 0 and 100 inclusive.
Default	**50** for databases created with Version 6 or later. **0** for databases created with Version 5.5 or earlier.
Description	This option controls the handling of two-digit years when converting from strings to dates or timestamps. The NEAREST_CENTURY setting is a numeric value that acts as a rollover point. Two digit years less than the value are converted to 20yy, while years greater than or equal to the value are converted to 19yy. The historical Adaptive Server Anywhere behavior is to add 1900 to the year. Adaptive Server Enterprise behavior is to use the nearest century, so

any year where value *yy* is less than 50, the year is set to 20yy.

NON_KEYWORDS option [compatibility]

Function	Turns off individual keywords, allowing their use as identifiers.
Allowed values	*String*
Default	Empty string

Description

This option turns off individual keywords or all keywords introduced since a specific release of the product. This provides a way of ensuring that applications created with older versions of the product are not broken by new keywords. If you have an identifier in your database that is now a keyword, you can either add double quotes around the identifier in all applications or scripts, or turn off the keyword using the NON_KEYWORDS option.

In addition to specifying individual keywords, you can turn off all keywords since a specified release using one of the following special values in the list of keywords:

```
keywords_4_0_d, keywords_4_0_c, keywords_4_0_b, keywords_4_0_a,
         keywords_4_0, keywords_5_0_01, keywords_5_0
```

The following statement prevents TRUNCATE and SYNCHRONIZE from being recognized as keywords:

```
  SET OPTION NON_KEYWORDS = 'TRUNCATE, SYNCHRONIZE'
```

The following statement prevents all keywords introduced since release 4.0d from being recognized as keywords:

```
  SET OPTION NON_KEYWORDS = 'keywords_4_0_d'
```

Each new setting of this option replaces the previous setting. The following statement clears all previous settings.

```
  SET OPTION NON_KEYWORDS =
```

A side-effect of this options is that SQL statements that use a turned off keyword cannot be used: they produce a syntax error.

See also

"Keywords" [*ASA SQL Reference*, page 4]

NULLS option [Interactive SQL]

Function	Specifies how NULL values in the database appear.
Allowed values	*String*
Default	**(NULL)**

Description	Set this according to your preference.

ODBC_DESCRIBE_BINARY_AS_VARBINARY [database]

Function	Controls how the Adaptive Server Anywhere ODBC driver describes BINARY columns.
Allowed values	**ON**, **OFF**
Default	**OFF**
Description	This option allows you to choose whether you want all BINARY and VARBINARY columns to be described to your application as BINARY or VARBINARY. By default, the Adaptive Server Anywhere ODBC driver describes both BINARY and VARBINARY columns as SQL_BINARY. When this option is set to ON, the ODBC driver describes BINARY and VARBINARY columns as SQL_VARBINARY. Regardless of the setting of this option, it is not possible to distinguish between BINARY and VARBINARY columns. It may be useful to turn this option ON if you are using Delphi applications where BINARY columns are always zero-padded, but VARBINARY columns are not. You can improve performance in Delphi by setting this option to ON so that all columns are treated as variable length data types.
See also	♦ "BINARY data type [Binary]" [*ASA SQL Reference*, page 74] ♦ "VARBINARY data type [BINARY]" [*ASA SQL Reference*, page 76]

ODBC_DISTINGUISH_CHAR_AND_VARCHAR option [database]

Function	Controls how the Adaptive Server Anywhere ODBC driver describes CHAR columns.
Allowed values	**ON**, **OFF**
Default	**OFF**
Description	When a connection is opened, the Adaptive Server Anywhere ODBC driver uses the setting of this option to determine how CHAR columns are described. If this option is set to **OFF** (the default), then CHAR columns are described as SQL_VARCHAR. If this option is set to **ON**, then CHAR columns are described as SQL_CHAR. VARCHAR columns are always described as SQL_VARCHAR.
See also	♦ "CHAR data type [Character]" [*ASA SQL Reference*, page 55] ♦ "CHARACTER VARYING (VARCHAR) data type [Character]" [*ASA SQL Reference*, page 55]

ON_CHARSET_CONVERSION_FAILURE option [database]

Function	Controls what happens if an error is encountered during character conversion.
Allowed values	**String**. See below for allowed values.
Default	**IGNORE**
Description	Controls what happens if an error is encountered during character conversion, as follows:

- **IGNORE** Errors and warnings do not appear.
- **WARNING** Reports substitutions and illegal characters as warnings. Illegal characters are not translated.
- **ERROR** Reports substitutions and illegal characters as errors.

Single-byte to single-byte converters are not able to report substitutions and illegal characters, and must be set to IGNORE.

ON_ERROR option [Interactive SQL]

Function	Controls what happens if an error is encountered while executing statements in Interactive SQL.
Allowed values	*String*. See below for allowed values.
Default	**PROMPT**
Description	Controls what happens if an error is encountered while executing statements, as follows:

- **STOP** Interactive SQL stops executing statements.
- **PROMPT** Interactive SQL prompts the user to see if the user wishes to continue.
- **CONTINUE** The error is ignored and Interactive SQL continues executing statements.
- **EXIT** Interactive SQL terminates.
- **NOTIFY_CONTINUE** The error is reporting, and the user is prompted to press ENTER or click OK to continue.
- **NOTIFY_STOP** The error is reported, and the user is prompted to press ENTER or click OK to stop executing statements.

- ♦ **NOTIFY_EXIT** The error is reported and the user is prompted to press ENTER or click OK to terminate Interactive SQL.

When you are executing a *.SQL* file, the values STOP and EXIT are equivalent.

ON_TSQL_ERROR option [compatibility]

Function	Controls error-handling in stored procedures.
Allowed values	*String*. See below for allowed values.
Default	**CONDITIONAL**

Description

This option controls error handling in stored procedures.

- ♦ **STOP** Stop execution immediately upon finding an error.
- ♦ **CONDITIONAL** If the procedure uses ON EXCEPTION RESUME, and the statement following the error handles the error, continue, otherwise exit.
- ♦ **CONTINUE** Continue execution, regardless of the following statement. If there are multiple errors, the first error encountered in the stored procedure is returned.

Both CONDITIONAL and CONTINUE settings for ON_TSQL_ERROR are used for Adaptive Server Enterprise compatibility, with CONTINUE most closely simulating Adaptive Server Enterprise behavior. The CONDITIONAL setting is recommended, particularly when developing new Transact-SQL stored procedures, as it allows errors to be reported earlier.

When this option is set to STOP or CONTINUE, it supercedes the setting of the CONTINUE_AFTER_RAISERROR option. However, when this option is set to CONDITIONAL (the default), behavior following a RAISERROR statement is determined by the setting of the CONTINUE_AFTER_RAISERROR option.

See also

"CREATE PROCEDURE statement" [*ASA SQL Reference*, page 373]

"CREATE PROCEDURE statement [T-SQL]" [*ASA SQL Reference*, page 383]

"Transact-SQL procedure language overview" [*ASA SQL User's Guide*, page 497]

"CONTINUE_AFTER_RAISERROR option [compatibility]" on page 644

OPTIMISTIC_WAIT_FOR_COMMIT option [compatibility]

Function	Controls locking behavior for WAIT_FOR_COMMIT option.

Allowed values **ON**, **OFF**

Default **OFF**

Description By default, OPTIMISTIC_WAIT_FOR_COMMIT is **OFF**.

Locking behavior is changed as follows when both OPTIMISTIC_WAIT_FOR_COMMIT and WAIT_FOR_COMMIT are **ON**:

- No locks are placed on primary key rows when adding a foreign key row without a matching primary key row.
- If a transaction adds a primary row, it will be allowed to commit only if the transaction has exclusive locks on all foreign rows that reference the primary row at the time the primary row is added.

This feature is for use in migrating 5.x applications to version 8.x or later by mimicking 5.x locking behavior when transactions add foreign rows before primary rows (as long as no two transactions concurrently add foreign rows with the same key value).

Note that this option is not recommended for general use as there are a number of situations in which transactions will not be allowed to commit, including:

- If transactions concurrently add foreign rows with the same key value when the primary row does not exist, at most 1 of the transactions will be allowed to commit (the one that adds the corresponding primary row).
- If a transaction deletes a primary row and then adds it back, it likely will not be allowed to commit (unless it has somehow obtained exclusive locks on all of the matching foreign rows).

OPTIMIZATION_GOAL option [database]

Function Determines whether query processing is optimized towards returning the first row quickly, or minimizing the cost of returning the complete result set.

Allowed values **first-row**, **all-rows**

Default **all-rows**

Description The OPTIMIZATION_GOAL option controls whether Adaptive Server Anywhere optimizes SQL data manipulation language (DML) statements for response time or total resource consumption.

If the option is set to **all-rows** (the default), then Adaptive Server Anywhere optimizes a query so as to choose an access plan with the minimal estimated total retrieval time. Setting OPTIMIZATION_GOAL to **all-rows** may be appropriate for applications that intend to process the entire result set, such as PowerBuilder DataWindow applications. A setting of all-rows is also appropriate for insensitive (ODBC static) cursors since the entire result is materialized when the cursor is opened. It may also be appropriate for scroll (ODBC keyset-driven) cursors, since the intent of such a cursor is to permit scrolling through the result set.

If the option is set to **first-row**, Adaptive Server Anywhere chooses an access plan that is intended to reduce the time to fetch the first row of the query's result, possibly at the expense of total retrieval time. In particular, the Adaptive Server Anywhere optimizer will typically avoid, if possible, access plans that require the materialization of results in order to reduce the time to return the first row. With this setting, the optimizer favors access plans that utilize an index to satisfy a query's ORDER BY clause, rather than plans that require an explicit sorting operation.

You can use the FASTFIRSTROW table hint in a query's FROM clause to set the optimization goal for a specific query to **first-row**, without having to change the OPTIMIZATION_GOAL setting.

☞ For more information about using the FASTFIRSTROW table hint, see "FROM clause" [*ASA SQL Reference*, page 491].

OPTIMIZATION_LEVEL option [database]

Function	Controls the amount of effort made by the Adaptive Server Anywhere query optimizer to find an access plan for an SQL statement.
Allowed values	**0–15**
Default	**9**
Description	The OPTIMIZATION_LEVEL option controls the amount of effort that the Adaptive Server Anywhere optimizer spends on optimizing SQL data manipulation language (DML) statements. This option controls the maximum number of alternative join strategies that the optimizer will consider for any SELECT block. The higher the setting of OPTIMIZATION_LEVEL, the greater the maximum number of join strategies that the optimizer will consider.

If the option is set to 0, then Adaptive Server Anywhere optimizer chooses the first access plan it considers for execution, in effect avoiding any cost-based comparison of alternative plans. In addition, with level 0 some semantic optimizations of nested queries are disabled. If this option is set to a value higher than 0, the optimizer evaluates alternative strategies and chooses the one with the lowest expected cost. If this option is set to a value greater than the default (9), the optimizer is more aggressive in its search for alternative strategies, possibly resulting in much higher elapsed time spent in the optimization phase.

In typical scenarios, this option is temporarily set to lower levels (0, 1, or 2) when the application desires faster OPEN times for a DML statement, and it is known that though the statement may be complex the query's execution time is very small, and hence the specific access plan chosen by the optimizer is less consequential. It is not recommended that the PUBLIC setting of OPTIMIZATION_LEVEL be changed from its default.

The effect of setting the OPTIMIZATION_LEVEL option is independent of the settings of the OPTIMIZATION_GOAL and OPTIMIZATION_WORKLOAD options.

Simple DML statements (single-block, single-table queries that contain equality conditions in the WHERE clause that uniquely identify a specific row) are optimized heuristically and bypass the cost-based optimizer altogether. The optimization of simple DML statements is not affected by the setting of the OPTIMIZATION_LEVEL option. The count of the number of requests optimized through the optimizer bypass mechanism is available as the QueryBypassed connection property.

☞ For more information about the QueryBypassed connection property, see "Connection-level properties" on page 713.

OPTIMIZATION_WORKLOAD option [database]

Function	Determines whether query processing is optimized towards a workload that is a mix of updates and reads or a workload that is predominantly read-based.
Allowed values	**Mixed**, **OLAP**
Scope	Can be set for the PUBLIC group only. DBA authority required.
Default	**Mixed**
Description	The OPTIMIZATION_WORKLOAD option controls whether Adaptive Server Anywhere optimizes queries for a workload that is a mix of updates and reads or predominantly read only.

If the option is set to Mixed (the default), Adaptive Server Anywhere chooses query optimization algorithms appropriate for a workload that is a mixture of short inserts, updates, and deletes and longer running read-only queries.

If the option is set to OLAP, Adaptive Server Anywhere chooses algorithms appropriate for a workload that consists for the most part of long-running queries, combined with batch updates. In particular, the optimizer may choose to use the Clustered Hash Group By query execution algorithm.

When the option is set to OLAP, the Clustered Hash Group By algorithm is enabled. If the option is set to Mixed (the default), it is disabled.

See also

"Clustered Hash Group By algorithm" [*ASA SQL User's Guide*, page 418]

OUTPUT_FORMAT option [Interactive SQL]

Function

Sets the default output format for data retrieved by a SELECT statement that is redirected to a file or output using the OUTPUT statement.

Allowed values

String. See below for allowed values.

Default

ASCII

Description

The valid output formats are:

- **ASCII** The output is an ASCII format file with one row per line in the file. All values are separated by commas, and strings are enclosed in apostrophes (single quotes). The delimiter and quote strings can be changed using the DELIMITED BY and QUOTE clauses. If ALL is specified in the QUOTE clause, then all values (not just strings) will be quoted.

  Three other special sequences are also used. The two characters **\n** represent a newline character; \\ represents a single backslash character, and the sequence **\xDD** represents the character with hexadecimal code DD.

- **DBASEII** The output is a dBASE II format file with the column definitions at the top of the file. Note that a maximum of 32 columns can be output. Column names are truncated to 11 characters, and each row of data in each column is truncated to 255 characters.

- **DBASEIII** The output is a dBASE III format file with the column definitions at the top of the file. Note that a maximum of 128 columns can be output. Column names are truncated to 11 characters, and each row of data in each column is truncated to 255 characters.

- **EXCEL** The output is an Excel 2.1 worksheet. The first row of the worksheet contains column labels (or names if there are no labels defined). Subsequent worksheet rows contain the actual table data.

- **FIXED** The output is fixed format, with each column having a fixed width. The width for each column can be specified using the COLUMN WIDTH clause. If this clause is omitted, the width for each column is computed from the data type for the column, and is large enough to hold any value of that data type. No column headings are output in this format.

- **FOXPRO** The output is a FoxPro format file (the FoxPro memo field is different than the dBASE memo field) with the column definitions at the top of the file. Note that a maximum of 128 columns can be output. Column names are truncated to 11 characters, and each row of data in each column is truncated to 255 characters.
- **HTML** The output is in the Hyper Text Markup Language (HTML) format.
- **LOTUS** The output is a Lotus WKS format worksheet. Column names will be put as the first row in the worksheet. Note that there are certain restrictions on the maximum size of Lotus WKS format worksheets that other software (such as Lotus 1-2-3) can load. There is no limit to the size of file Interactive SQL can produce.
- **SQL** The output is an Interactive SQL INPUT statement required to recreate the information in the table.
- **XML** The output is an XML file encoded in UTF-8 and containing an embedded DTD. Binary values are encoded in CDATA blocks with the binary data rendered as 2-hex-digit strings.

OUTPUT_LENGTH option [Interactive SQL]

Function	Controls the length of column values when Interactive SQL exports information to an external file.
Allowed values	*Integer*
Default	**0** (no truncation)
Description	This option controls the maximum length of column values when Interactive SQL exports data to an external file (using output redirection with the OUTPUT statement). This option affects only ASCII, HTML, and SQL output formats.

OUTPUT_NULLS option [Interactive SQL]

Function	Controls the way NULL values are exported.
Allowed values	*String*
Default	(empty string)
Description	This option controls the way NULL values are written by the OUTPUT statement. Every time a NULL value is found in the result set, the string from this option is returned instead. This option affects only ASCII, HTML, FIXED, EXCEL, and SQL output formats.

PERCENT_AS_COMMENT option [compatibility]

Function	Controls the interpretation of the percent character.
Allowed values	**ON**, **OFF**
Default	**ON**
Description	It is recommended that you do not use % as a comment marker.

Versions of this product before Version 6 treated the percent character (%) in SQL statements exclusively as a comment delimiter. Since Version 5, alternative comment markers such as //, **/* */**, and **–** (double dash) have been available. The double-dash style is the SQL/92 comment delimiter.

Adaptive Server Enterprise treats **%** as a modulo operator and does not support the Adaptive Server Anywhere **mod** function. Writing a statement that works in both environments and performs a modulo operation was previously impossible.

The PERCENT_AS_COMMENT option controls the meaning of %. The default setting is ON for backwards compatibility. You can set the option to OFF for compatibility with Adaptive Server Enterprise.

Procedures, triggers and views that were created with %-style comments are converted to double-dash comments when they are stored in the catalog.

Existing procedures must be recreated before changing option
Any existing procedures that contain %-style comments must be recreated before you change the option setting; otherwise, the procedures will fail to load.

PINNED_CURSOR_PERCENT_OF_CACHE option [database]

Function	Specifies how much of the cache can be used for pinning cursors.
Allowed values	*Integer*, between 0–100
Scope	Can be set for the PUBLIC group only. DBA authority required.
Default	**10**
Description	The server uses pages of virtual memory for the data structures needed to implement cursors. These pages are kept locked in memory between fetch requests so they are readily available when the next fetch request arrives.

To prevent these pages from occupying too much of the cache in low memory environments, a limit is placed on the percentage of the cache allowed to be used for pinning cursors. You can use the PINNED_CURSOR_PERCENT_OF_CACHE option to adjust this limit.

The option value is specified as a percentage from 0 to 100, with a default of 10. Setting the option to 0 means that cursor pages will not be pinned between fetch requests.

PRECISION option [database]

Function	Specifies the maximum number of digits in the result of any decimal arithmetic.
Allowed values	*Integer*, between 0 and 127 inclusive.
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.
Default	**30**
Description	Precision is the total number of digits to the left and right of the decimal point. The SCALE option specifies the minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum PRECISION.

Multiplication, division, addition, subtraction, and aggregate functions can all have results that exceed the maximum precision.

For example, when a DECIMAL(8,2) is multiplied with a DECIMAL(9,2), the result could require a DECIMAL(17,4). If PRECISION is 15, only 15 digits will be kept in the result. If SCALE is 4, the result will be a DECIMAL(15,4). If SCALE is 2, the result will be a DECIMAL(15,2). In both cases, there is a possibility of overflow.

PREFETCH option [database]

Function	The PREFETCH option controls whether or not rows are fetched to the client side before being made available to the client application.
Allowed values	**OFF**, **CONDITIONAL**, **ALWAYS**
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.
Default	**CONDITIONAL**
Description	This option controls whether rows are fetched to the client side in advance of being made available to the client application. Fetching a number of rows at a time, even when the client application requests rows one at a time (for example, when looping over the rows of a cursor) can cut down on response

time and improve overall throughput by cutting down the number of requests to the database.

- ♦ OFF means no prefetching is done.
- ♦ CONDITIONAL (the default) causes prefetching to occur unless the cursor type is SENSITIVE or the query includes a proxy table.
- ♦ ALWAYS means prefetching is done even for SENSITIVE cursor types and cursors that involve a proxy table.

The ALWAYS value must be used with caution, as it affects some cursor semantics. For example, it causes the sensitive cursor to become asensitive. Old values may be fetched if the value was updated between the prefetch and the application's fetch request. In addition, using prefetch on a cursor that involves a proxy table can cause the error –668, Cursor is restricted to FETCH NEXT operations, if the client attempts to re-fetch prefetch rows. A client may attempt to re-fetch prefetch rows after a rollback or on a fetch relative 0, if a fetch column is re-bound or bound for the first time after the first fetch, or in some cases when GET DATA is used.

The setting of PREFETCH is ignored by Open Client and jConnect connections.

If the DisableMultiRowFetch connection parameter is set to YES, the PREFETCH database option is ignored and no prefetching is done.

This option previously accepted the value ON. This value is now an alias for CONDITIONAL.

See also

- ♦ "Prefetching rows" [*ASA Programming Guide*, page 42]
- ♦ "DisableMultiRowFetch connection parameter [DMRF]" on page 190

PRESERVE_SOURCE_FORMAT option [database]

Function

Controls whether the original source definition of procedures, triggers, views, and event handlers is saved in system files. If saved, it is saved in the column source in SYSTABLE, SYSPROCEDURE, SYSTRIGGER, and SYSEVENT.

Allowed values

ON, **OFF**

Scope

Can be set for the PUBLIC group only. DBA authority required.

Default

ON

Description

When PRESERVE_SOURCE_FORMAT is ON, the server saves the formatted source from CREATE and ALTER statements on procedures, views, triggers, and events, and puts it in the appropriate system table's source column.

Unformatted source text is stored in the same system tables, in the columns proc_defn, trigger_defn, and view_defn. However, these definitions are not easy to read in Sybase Central. The formatted source column allows you to view the definitions with the spacing, comments, and case that you want.

This option can be turned off to reduce space used to save object definitions in the database. The option can be set only for the user PUBLIC.

PREVENT_ARTICLE_PKEY_UPDATE option [database]

Function	Controls updates to the primary key columns of tables involved in publications.
Allowed values	**ON**, **OFF**
Default	**ON**
Description	Setting this option to ON disallows updates to the primary key columns of tables that are part of a publication. This option helps ensure data integrity, especially in a replication and synchronization environment

QUALIFY_OWNERS option [replication]

Function	Controls whether SQL statements being replicated by SQL Remote should use qualified object names.
Allowed values	**ON**, **OFF**
Default	The default in Adaptive Server Anywhere is **ON**. The default in Adaptive Server Enterprise is **OFF**.
Description	Qualifying owners in Adaptive Server Enterprise setups is rarely needed because it is common for objects to be owned by **dbo**. When qualification is not needed in Adaptive Server Anywhere setups, messages will be slightly smaller with the option OFF.

QUERY_PLAN_ON_OPEN option [compatibility]

Function	Controls whether a plan is returned when a cursor is opened.
Allowed values	**ON**, **OFF**
Default	**OFF**
Description	In early versions of the software, each time an OPEN was done on a cursor, the server would return in the SQLCA **sqlerrmc** field a string representing the query plan (limited to 70 bytes). A more complete description can be obtained using the EXPLAIN statement or the PLAN function. For this

reason, computing and returning the query plan on an OPEN is needed only for compatibility with old applications. The QUERY_PLAN_ON_OPEN option controls whether the plan is returned on an OPEN. By default, the setting is **OFF**.

QUOTE_ALL_IDENTIFIERS option [replication]

Function	Controls whether SQL statements being replicated by SQL Remote should use quoted identifiers.
Allowed values	**ON**, **OFF**
Default	**OFF**
Description	When this option is **OFF**, dbremote quotes identifiers that require quotes by Adaptive Server Anywhere (as it has always done) and *ssremote* does not quote any identifiers.

When the option is **ON**, all identifiers are quoted.

QUOTED_IDENTIFIER option [compatibility]

Function	Controls the interpretation of strings that are enclosed in double quotes.
Allowed values	**ON**, **OFF**
Default	**ON**
	OFF for Open Client and JDBC connections
Description	This option controls whether strings that are enclosed in double quotes are interpreted as identifiers (**ON**) or as literal strings (**OFF**). The QUOTED_IDENTIFIER option is included for Transact-SQL compatibility.

Note that Interactive SQL automatically executes the equivalent of

```
SET TEMPORARY OPTION QUOTED_IDENTIFIER = ON
```

when it connects to a database. TEMPORARY options apply only to the current connection, and last only until the connection is closed.

☞ For more information, see "Setting options for Transact-SQL compatibility" [*ASA SQL User's Guide*, page 485].

READ_PAST_DELETED option [database]

Function	Controls server behavior on uncommitted deletes at isolation levels 1 and 2.
Allowed values	**ON**, **OFF**
Default	**ON**

Description	If READ_PAST_DELETED is **ON** (the default), sequential scans at isolation levels 1 and 2 will skip uncommitted deleted rows. If **OFF**, sequential scans will block on uncommitted deleted rows at isolation levels 1 and 2 (until the deleting transaction commits or rolls back). This option changes server behavior at isolation levels 1 and 2. For most purposes, this option should be left **ON**. If set to **OFF**, the blocking behavior depends on the plan chosen by the optimizer (if there is an index that could possibly be used).

RECOVERY_TIME option [database]

Function	Sets the maximum length of time, in minutes, that the database server will take to recover from system failure.
Allowed values	*Integer*, in minutes
Scope	Can be set for the PUBLIC group only. DBA authority required. Takes effect when server is restarted.
Default	**2**
Description	This option is used with the CHECKPOINT_TIME option to decide when checkpoints should be done. Adaptive Server Anywhere uses a heuristic to estimate the recovery time based on the operations that have been performed since the last checkpoint. Thus, the recovery time is not exact. ☞ For more information, see "The automatic recovery process" on page 399.

REMOTE_IDLE_TIMEOUT option [database]

Function	Controls how many seconds of inactivity web service client procedures and functions will tolerate.
Allowed values	*Integer*, in seconds
Default	**15**
Description	This option affects web service client procedures and functions. If more time than the specified number of seconds passes without activity, the procedure or function times out.

REPLICATE_ALL option [replication]

REPLICATION_ERROR option [replication]

Function

For SQL Remote, allows you to specify a stored procedure to be called by the Message Agent when a SQL error occurs.

Allowed values

Stored procedure name

Default

No procedure

Description

For SQL Remote, the REPLICATION_ERROR option allows you to specify a stored procedure to be called by the Message Agent when a SQL error occurs. By default no procedure is called.

The procedure must have a single argument of type CHAR, VARCHAR, or LONG VARCHAR. The procedure is called once with the SQL error message and once with the SQL statement that causes the error.

Although the option allows you to track and monitor SQL errors in replication, you must still design them out of your setup; this option is not intended to resolve such errors.

REPLICATION_ERROR_PIECE option [replication]

Function

For SQL Remote, the REPLICATION_ERROR_PIECE option works in conjunction with the REPLICATION_ERROR option to allow you to specify a long varchar stored procedure to be called by the Message Agent when a SQL error occurs.

This is a compatibility feature most useful for databases created with older versions of Adaptive Server Enterprise that do not support LONG VARCHAR.

Allowed values

Stored procedure name

Default

No procedure

Description

If an error occurs and REPLICATION_ERROR is defined then the REPLICATION_ERROR procedure is called with the full error string.

If REPLICATION_ERROR and REPLICATION_ERROR _PIECE are both defined then the error is broken up into VARCHAR pieces. REPLICATION_ERROR is called with the first piece and REPLICATION_ERROR_PIECE is called repeatedly with the remaining pieces.

RETURN_DATE_TIME_AS_STRING option [database]

Function	To control how a date, time, or timestamp value is passed to the client application when queried.
Allowed values	**ON**, **OFF**
Scope	Can be set as a temporary option only, for the duration of the current connection.
Default	**OFF**
Description	This option indicates whether date, time, and timestamp values are returned to applications as a date or time datatype or as a string.
	When this option is set to ON, the server converts the date, time, or timestamp value to a string before it is sent to the client in order to preserve the TIMESTAMP_FORMAT, DATE_FORMAT, or TIME_FORMAT option setting.
	Sybase Central and Interactive SQL automatically turn the RETURN_DATE_TIME_AS_STRING option ON.
See also	"DATE_FORMAT option [compatibility]" on page 646
	"TIME_FORMAT option [compatibility]" on page 694
	"TIMESTAMP_FORMAT option [compatibility]" on page 695

RETURN_JAVA_AS_STRING option [database]

Function	To control how a Java object is passed to the client application when queried.
Allowed values	**ON**, **OFF**
Scope	Can be set as a temporary option only, for the duration of the current connection.
Default	**OFF**
Description	The option indicates how a Java object is returned to applications connecting over Open Client or jConnect.
	By default, a Java object is returned over TDS as a Sun serialization of the object. The client, or receiver of the object serialization, is responsible for deserializing the object into an instance.
	If RETURN_JAVA_AS_STRING is set to **ON**, the object is first converted to an instance of **java.lang.String** using the **toString()** method, and the String object is returned to the client.

RI_TRIGGER_TIME option [compatibility]

Function	Controls the relative timing of referential integrity checks and trigger actions.
Allowed values	**BEFORE**, **AFTER**
Scope	Can be set for the PUBLIC group only. DBA authority required.
Default	**AFTER**
Description	The option can be set to either **BEFORE** or **AFTER**. When it's set to **AFTER**, referential integrity actions are executed after the UPDATE or DELETE.
	Only the PUBLIC setting can be used; any other setting is ignored.

ROLLBACK_ON_DEADLOCK [database]

Function	Controls how transactions are treated when a deadlock occurs.
Allowed values	**ON**, **OFF**
Scope	Can be set by any user and can be set for the PUBLIC group, as well as individual connections. Takes effect immediately.
Default	**ON**
Description	When this option is set to ON, a transaction is automatically rolled back if it encounters a deadlock. The rollback happens after the current request completes. If this option is set to OFF, Adaptive Server Anywhere automatically rolls back the statement that encountered the deadlock, and returns an error to that transaction indicating which form of deadlock occurred. Note that rolling back the statement likely will not release any of the locks acquired by the statement.
	☞ For more information about deadlocks, see "Deadlock" [*ASA SQL User's Guide*, page 114].

ROW_COUNTS option [database]

Function	Specifies whether the database will always count the number of rows in a query when it is opened.
Allowed values	**ON**, **OFF**
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.

Default

OFF

Description

If this option is set to **OFF**, the row count is usually only an estimate. If this option is set to **ON**, the row count is always accurate.

> **Warning**
> When ROW_COUNTS is set to **ON**, it may take significantly longer to execute queries. In fact, it will usually cause Adaptive Server Anywhere to execute the query twice, doubling the execution time.

SCALE option [database]

Function

Specifies the minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum PRECISION.

Allowed values

Integer, between 0 and 127 inclusive.

Scope

Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.

Default

6

Description

Multiplication, division, addition, subtraction, and aggregate functions can all have results that exceed the maximum precision.

☞ For an example, see "PRECISION option [database]" on page 682.

SORT_COLLATION option [database]

Function

Allows implicit use of the SORTKEY function on ORDER BY expressions.

Allowed values

INTERNAL, *collation_name*, or *collation_id*

Default

INTERNAL

Description

When the value of this option is **INTERNAL**, the ORDER BY clause remains unchanged.

When the value of this option is set to a valid *collation name* or *collation ID*, any string expression in the ORDER BY clause is treated as if the SORTKEY function had been invoked.

See also

"SORTKEY function [String]" [*ASA SQL Reference*, page 222]

Example

Set the sort collation to binary:

```
set temporary option sort_collation='binary'
```

Having the sort collation set to binary transforms

```
SELECT name, id
FROM product
ORDER BY name, id
```

and

```
SELECT name, id
FROM product
ORDER BY 1, 2
```

into

```
SELECT name, id
FROM product
ORDER BY SORTKEY(name, 'binary'), id
```

SQL_FLAGGER_ERROR_LEVEL option [compatibility]

Function	Controls the response to any SQL that is not part of a specified set of SQL/92.
Allowed values	**E**, **I**, **F**, **W**
Default	**W**
Description	This option flags any SQL that is not part of a specified set of SQL/92 as an error.

The allowed values are as follows:

- **E** Flag syntax that is not entry-level SQL/92 syntax
- **I** Flag syntax that is not intermediate-level SQL/92 syntax
- **F** Flag syntax that is not full-SQL/92 syntax
- **W** Allow all supported syntax

SQL_FLAGGER_WARNING_LEVEL option [compatibility]

Function	Controls the response to any SQL that is not part of a specified set of SQL/92.
Allowed values	**E**, **I**, **F**, **W**
Default	**W**
Description	This option flags any SQL that is not part of a specified set of SQL/92 as a warning.

The allowed values are as follows:

- **E** Flag syntax that is not entry-level SQL/92 syntax
- **I** Flag syntax that is not intermediate-level SQL/92 syntax
- **F** Flag syntax that is not full-SQL/92 syntax
- **W** Allow all supported syntax

STATISTICS option [Interactive SQL]

Function	Controls whether the execution time is enabled.
Allowed values	*Any non-negative integer*
Default	**7**
Description	This option is an older version of the ISQL_PLAN option, and is used only by `dbisqlc`. When you set the STATISTICS option to 0, no information appears in the Messages pane after you execute a SQL statement. When you set the option to any other number, the execution time appears in the Messages pane after you execute a statement. The number you specify also becomes the default height (in lines) of the Messages pane.
See also	"ISQL_COMMAND_TIMING option [Interactive SQL]" on page 659 "ISQL_PLAN option [Interactive SQL]" on page 661

STRING_RTRUNCATION option [compatibility]

Function	Determines whether an error is raised when an INSERT or UPDATE truncates a CHAR or VARCHAR string.
Allowed values	**ON**, **OFF**
Default	**OFF**
Description	If the truncated characters consist only of spaces, no exception is raised. The setting of ON corresponds to ANSI/ISO SQL/92 behavior. When it is set to **OFF**, the exception is not raised and the character string is silently truncated.

SUBSCRIBE_BY_REMOTE option [replication]

Function	Controls interpretation of NULL or empty-string SUBSCRIBE BY values.
Allowed values	**ON**, **OFF**
Default	**ON**

Description	When the option is set to **ON**, operations from remote databases on rows with a SUBSCRIBE BY value that is NULL or an empty string assume that the remote user is subscribed to the row. When it is set to **OFF**, the remote user is assumed not to be subscribed to the row.

SUBSUME_ROW_LOCKS option [database]

Function	Controls when the server acquires individual row locks for a table.
Allowed values	**ON**, **OFF**
Default	**ON**
Description	If the SUBSUME_ROW_LOCKS option is **ON** (the default) then whenever a table *t* is locked exclusively with LOCK TABLE *t* IN EXCLUSIVE MODE, the server will no longer acquire individual row locks for *t*.

This can result in a significant performance improvement if extensive updates are made to *t* in a single transaction, especially if *t* is large relative to cache size. It also allows for atomic update operations that are larger than the lock table can currently handle (> ~2-4m rows).

When this option is *ON*, keyset cursors over a table locked in this fashion will return row changed warnings for every row in the cursor, if any row in the database has been modified. Note that the server could turn an updateable cursor with an ORDER BY into a keyset cursor as a result.

SUPPRESS_TDS_DEBUGGING option [database]

Function	Determines whether TDS debugging information appears in the server window.
Allowed values	**ON**, **OFF**
Default	**OFF**
Description	When the server is started with the -z option, debugging information appears in the server window, including debugging information about the TDS protocol.

The SUPPRESS_TDS_DEBUGGING option restricts the debugging information about TDS that appears in the server window. When this option is set to **OFF** (the default) TDS debugging information appears in the server window.

TDS_EMPTY_STRING_IS_NULL option [database]

Function	Controls whether empty strings are returned as NULL or a string containing one blank character for TDS connections.

Allowed values	**ON**, **OFF**
Default	**OFF**
Description	By default, this option is set to **OFF** and empty strings are returned as a string containing one blank character for TDS connections. When this option is set to **ON**, empty strings are returned as NULL strings for TDS connections. Non-TDS connections distinguish empty strings from NULL strings.

TEMP_SPACE_LIMIT_CHECK option [database]

Function	Checks the amount of temporary file space used by a connection and fails the request if the amount of space requested is greater than the connection's allowable quota.
Allowed values	**ON**, **OFF**
Scope	Can be set for the PUBLIC group only. DBA authority required.
Default	**OFF**
Description	When TEMP_SPACE_LIMIT_CHECK is set to OFF (the default), the database server does not check the amount of temporary file space used by a connection. If a connection requests more than its quota of temporary space when this option is set to off, a fatal error can occur. When this option is set to ON, if a connection requests more than its quota of temporary file space, then the request fails and the error SQLSTATE_TEMP_SPACE_LIMIT is returned. Two factors are used to determine the temporary file quota for a connection: the maximum size of the temporary file and the number of active database connections. The maximum size of the temporary file is the sum of the file's current size and the amount of disk space available on the partition containing the file. When limit checking is turned on, a connection is checked for exceeding its quota when the temporary file has grown to 80% or more of its maximum size *and* the connection requests more temporary file space. Once this happens, any connection fails that uses more than the maximum temporary file space divided by the number of active connections. You can obtain information about the space available for the temporary file using the sa_disk_free_space system procedure.
See also	"sa_disk_free_space system procedure" [*ASA SQL Reference*, page 791]

TIME_FORMAT option [compatibility]

Function	Sets the format for times retrieved from the database.

Allowed values	*String*, composed of the symbols listed below.
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.
Default	*HH*:*NN*:*SS*.*SSS*

Description

The format is a string using the following symbols:

- **hh** Two digit hours (24 hour clock)
- **nn** Two digit minutes
- **mm** Two digit minutes if following a colon (as in hh:mm)
- **ss[.s...]** Two digit seconds plus optional fraction

Each symbol is substituted with the appropriate data for the time that is being formatted. Any format symbol that represents character rather than digit output can be put in upper case, which causes the substituted characters to also be in upper case. For numbers, using mixed case in the format string suppresses leading zeros.

TIME_ZONE_ADJUSTMENT option [database]

Function	Allows a connection's time zone adjustment to be modified.
Allowed values	*Integer*, (for example, 300), *Negative integer*, enclosed in quotation marks (for example, '-300'), *String*, representing a time in hours and minutes, preceded by + or -, enclosed in quotation marks (for example, '+5:00', or '-5:00').
Default	If the client is connecting via embedded SQL, ODBC, OLE DB, or ADO, the default value is set according to the client's time zone. If the client is connecting via jConnect or Open Client, the default is based on the server's time zone.
Description	The TIME_ZONE_ADJUSTMENT option value is the same value as that returned by `connection_property('TimeZoneAdjustment')`. The value represents the number of minutes that must be added to the Coordinated Universal Time (UTC) to display time local to the connection.
See also	"Connection-level properties" on page 713

TIMESTAMP_FORMAT option [compatibility]

Function	Sets the format for timestamps that are retrieved from the database.
Allowed values	*String*, composed of the symbols listed below.

Scope Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.

Default *YYYY-MM-DD HH*:*NN*:*ss.SSS*

For Open Client and JDBC connections the default is set to *YYYY-MM-DD HH*:*NN*:*SS.SSS*.

Description The format is a string using the following symbols:

Symbol	Description
yy	Two digit year
yyyy	Four digit year
mm	Two digit month, or two digit minutes if following a colon (as in ‘hh:mm’)
mmm[*m*...]	Character short form for months—as many characters as there are “m”s
dd	Two digit day of month
ddd[*d*...]	Character short form for day of the week
hh	Two digit hours
nn	Two digit minutes
ss.ssssss	Seconds and fractions of a second, up to six decimal places. Not all platforms support timestamps to a precision of six places.
aa	am or pm (12 hour clock)
pp	pm if needed (12 hour clock)
f	Use French days and months (deprecated)

Each symbol is substituted with the appropriate data for the date that is being formatted.

For symbols that represent character data (such as *mmm*), you can control the case of the output as follows:

- ♦ Type the symbol in all upper case to have the format appear in all upper case. For example, `MMM` produces `JAN`.
- ♦ Type the symbol in all lower case to have the format appear in all lower case. For example, `mmm` produces `jan`.

- Type the symbol in mixed case to have Adaptive Server Anywhere choose the appropriate case for the language that is being used. For example, in English, typing `Mmm` produces `May`, while in French it produces `mai`.

For symbols that represent numeric data, you can control zero-padding with the case of the symbols:

- Type the symbol in same-case (such as `MM` or `mm`) to allow zero padding. For example, `yyyy/mm/dd` could produce `2002/01/01`.
- Type the symbol in mixed case (such as `Mm`) to suppress zero padding. For example, `yyyy/Mm/Dd` could produce `2002/1/1`.

TRUNCATE_DATE_VALUES option [database] (deprecated)

Function	To change the storage of time values within DATE data type values.
Allowed values	**ON**, **OFF**
Scope	Can be set for the PUBLIC group only. DBA authority required.
Default	**ON** for databases created with Version 7 or later software, or upgraded to Version 7 or later. **OFF** for older databases

Description

Note
This option is deprecated.

When the option is set to **ON**, columns or variables declared as DATE data types have no hours or minutes stored.

When the option is set to **OFF**, DATE columns or variables also have hours and minutes stored. These time components are typically not displayed because of the DATE_FORMAT setting. However, exact comparisons between DATE values may unexpectedly fail because of non-matching time components.

See also

"DATE_FORMAT option [compatibility]" on page 646

"DATE data type [Date and Time]" [*ASA SQL Reference*, page 71]

TRUNCATE_TIMESTAMP_VALUES option [database]

Function	Limits the resolution of timestamp values.
Allowed values	**ON**, **OFF**

Scope	Can be set for the PUBLIC group only. DBA authority required. This option should not be enabled for databases already containing timestamp data.
Default	**OFF**

Description

A TIMESTAMP value is precise to six decimal places in Adaptive Server Anywhere. However, to maintain compatibility with other software, which may truncate the TIMESTAMP value to three decimal places, you can set the TRUNCATE_TIMESTAMP_VALUES option to **ON** to limit the number of decimal places Adaptive Server Anywhere stores. The DEFAULT_TIMESTAMP_INCREMENT option determines the number of decimal places to which the TIMESTAMP value is truncated.

For MobiLink synchronization, this option must be set prior to performing the first synchronization.

If the database server finds TIMESTAMP values with a higher resolution than that specified by the combination of TRUNCATE_TIMESTAMP_VALUES and DEFAULT_TIMESTAMP_INCREMENT, an error is reported.

In most cases, unloading the database and then reloading it into a new database in which the TRUNCATE_TIMESTAMP_VALUES and DEFAULT_TIMESTAMP_INCREMENT values have been set is the easiest solution to ensure the proper TIMESTAMP values are used. However, depending on the type of TIMESTAMP columns in your table, you can also do the following:

- If the TIMESTAMP columns are defined with DEFAULT TIMESTAMP or DEFAULT UTC TIMESTAMP (so that the value is automatically updated by the server when the row is modified), you must delete all the rows in the table before the TRUNCATE_TIMESTAMP_VALUES option is changed. You can delete the rows using the DELETE or TRUNCATE TABLE statement.

- If the TIMESTAMP column is defined with a value other than DEFAULT TIMESTAMP or DEFAULT UTC TIMESTAMP, you can execute an UPDATE statement that casts the values to a string and then back to a TIMESTAMP. For example,

  ```
  UPDATE T
     SET ts = CAST( DATEFORMAT( ts, 'yyyy/mm/dd hh:nn:ss.ss')
   AS TIMESTAMP)
  ```

  Note that this process may lose more precision than is necessary. The format string to use depends on the number of digits of precision to be kept.

Example

Setting the DEFAULT_TIMESTAMP_INCREMENT option to 100 000 causes truncation after the first decimal place in the seconds component, allowing a value such as '2000/12/05 10:50:53:700' to be stored.

TRUNCATE_WITH_AUTO_COMMIT option [database]

Function	To speed up TRUNCATE TABLE statements.
Allowed values	**ON**, **OFF**
Scope	Can be set for the PUBLIC group only. DBA authority required.
Default	**ON**

Description

If TRUNCATE_WITH_AUTO_COMMIT is set to **ON**, then a COMMIT is executed both before and after the TRUNCATE TABLE statement is executed. The primary purpose of the option is to enable faster table truncation (delete of all rows).

There are some cases where a fast TRUNCATE cannot be done:

- ♦ If there are foreign keys either to or from the table
- ♦ If the TRUNCATE TABLE statement is executed within a trigger
- ♦ If the TRUNCATE TABLE statement is executed within an atomic statement

See also

"TRUNCATE TABLE statement" [*ASA SQL Reference*, page 642]

TRUNCATION_LENGTH option [Interactive SQL]

Function	Controls the truncation of wide columns for displays to fit on a screen.
Allowed values	*Integer*
Default	**256**
Description	The TRUNCATION_LENGTH option limits the length of displayed column values. The unit is in characters. A value of 0 means that column values are not truncated. The default truncation length is 256.

TSQL_HEX_CONSTANT option [compatibility]

Function	Controls whether hexadecimal constants are treated as binary typed constants.
Allowed values	**ON**, **OFF**
Default	**ON**

Description	When this option is set to **ON**, hexadecimal constants are treated as binary typed constants. To get the historical behavior, set the option to OFF.

TSQL_VARIABLES option [compatibility]

Function	Controls whether the @ sign can be used as a prefix for embedded SQL host variable names.
Allowed values	**ON**, **OFF**
Default	**OFF**
	ON for Open Client and JDBC connections
Description	When this option is set to **ON**, you can use the @ sign instead of the colon as a prefix for host variable names in embedded SQL. This is implemented primarily for Transact-SQL compatibility.

UPDATE_STATISTICS option [database]

Function	Controls the gathering of statistics during query execution
Allowed values	**ON**, **OFF**
Default	**ON**
Description	The server collects statistics during normal query execution and uses the gathered statistics to self-tune the column histograms. You can set the UPDATE_STATISTICS option **OFF** to disable the gathering of statistics during query execution.
	The UPDATE_STATISTICS option does not affect changes to statistics as a result of updates to data (LOAD/INSERT/UPDATE/DELETE).
	Under normal circumstances, it should not be necessary to turn this option off.

USER_ESTIMATES option [database]

Function	Controls whether or not user selectivity estimates in query predicates are respected or ignored by the query optimizer.
Allowed values	**ENABLED**, **DISABLED**, **OVERRIDE-MAGIC**
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.
Default	**OVERRIDE-MAGIC**
Description	Adaptive Server Anywhere allows you to specify user selectivity estimates

can improve the optimizer's performance when the server is unable to accurately predict the selectivity of a predicate. However, user selectivity estimates should be used only in appropriate circumstances. For example, it may be useful to supply a selectivity estimate for a predicate that involves one or more functions if the **OVERRIDE-MAGIC** selectivity estimate used by the optimizer is significantly different from the actual selectivity.

If you have used selectivity estimates that are inaccurate as a workaround to performance problems where the software-selected access plan was poor, it is recommended that you set this option to DISABLED. The server may not select an optimal plan if you use inaccurate estimates.

☞ For more information about user selectivity estimates, see "Explicit selectivity estimates" [*ASA SQL Reference*, page 31].

When a user selectivity estimate is supplied with a predicate, the estimate is respected or ignored based on the setting of this option. The following values are accepted:

- **ENABLED** All user-supplied selectivity estimates are respected. You can also use ON to turn on this option.
- **OVERRIDE-MAGIC** A user selectivity estimate is respected and used only if the optimizer would otherwise choose to use its last-resort, heuristic value (also called the magic value).
- **DISABLED** All user estimates are ignored and magic values are used when no other estimate data is available. You can also use OFF to turn off this option.

VERIFY_ALL_COLUMNS option [replication]

Function	Controls whether messages that contain updates published by the local database are sent with all column values included.
Allowed values	**ON**, **OFF**
Default	**OFF**
Description	This option is used by SQL Remote only. When it is set to **ON**, messages that contain updates published by the local database are sent with all column values included, and a conflict in any column triggers a RESOLVE UPDATE trigger at the subscriber database.

VERIFY_THRESHOLD option [replication]

Function	Controls which columns are verified when updates are replicated.
Allowed values	*Integer*, in bytes

Default	**1 000**
Description	This option is used by SQL Remote only. If the data type of a column is longer than the threshold, old values for the column are not verified when an UPDATE is replicated. This keeps the size of SQL Remote messages down, but has the disadvantage that conflicting updates of long values are not detected.

WAIT_FOR_COMMIT option [database]

Function	Determines when foreign key integrity is checked, as data is manipulated.
Allowed values	**ON**,+ **OFF**
Scope	Can be set for an individual connection or for the PUBLIC group. Takes effect immediately.
Default	**OFF**
Description	If this option is set to **ON**, the database does not check foreign key integrity until the next COMMIT statement. Otherwise, all foreign keys that are not created with the CHECK_ON_COMMIT option are checked as they are inserted, updated or deleted.

WEBSERVICE_NAMESPACE_HOST option [database]

Function	Specifies the hostname to be used as the XML namespace within generated WSDL documents. Requires DBA authority.
Allowed values	*hostname-string* or **NULL**
Scope	Can be set for the PUBLIC group only. Takes effect immediately. DBA authority is required to set this option.
Default	**NULL**
Description	Webservices Description Language Documents (WSDLs) are exported by DISH services. These are XML documents that contain descriptions of the available SOAP services. The URL of the targetNameSpace and the soapAction operations within this XML document contain a hostname. When this option is set to NULL, the default value, the hostname is that of the computer on which the database server is running. If this option is set to a string value, the string is used as the hostname instead. This option is intended for use when developing Web Service client applications that will, when deployed, target a host other than the one used for development.

CHAPTER 17

Database Performance and Connection Properties

About this chapter

This chapter contains information and tables relating to database performance monitoring and database connection parameters.

Contents

Database performance statistics

Adaptive Server Anywhere provides a set of statistics that can be used to monitor database performance. These are accessible from Sybase Central, and client applications can access the statistics as functions. In addition, these statistics are made available by the server to the Windows Performance Monitor.

Note
The Windows Performance Monitor is available in Windows NT/2000/XP.

This section describes how to access performance and related statistics from client applications, how to monitor database performance using Sybase Central, and how to monitor database performance using the Windows Performance Monitor.

Performance Monitor statistics

The Windows Performance Monitor is an application for viewing the behavior of objects such as processors, memory, and applications. Adaptive Server Anywhere provides many statistics for the Performance Monitor to display.

☞ For information on how to use the Performance Monitor, see "Monitoring database statistics from Windows Performance Monitor" [*ASA SQL User's Guide*, page 195].

Adaptive Server Anywhere makes statistics available for the Performance Monitor. Rates are reported per second. The statistics are grouped into the following areas:

- "Cache statistics" on page 705
- "Checkpoint and recovery statistics" on page 705
- "Communications statistics" on page 706
- "Disk I/O statistics" on page 707
- "Disk read statistics" on page 708
- "Disk write statistics" on page 708
- "Index statistics" on page 709
- "Java VM statistics" on page 709
- "Memory pages statistics" on page 710

- ♦ "Request statistics" on page 711
- ♦ "Miscellaneous statistics" on page 711

Cache statistics

These statistics describe the use of the cache.

Statistic	Scope	Description
Cache Hits/sec	Connection and Database	The rate at which database page lookups are satisfied by finding the page in the cache.
Cache Reads: Index Interior/sec	Connection and Database	The rate at which index internal-node pages are read from the cache.
Cache Reads: Index Leaf/sec	Connection and Database	The rate at which index leaf pages are read from the cache.
Cache Reads: Table/sec	Connection and Database	The rate at which table pages are read from the cache.
Cache Reads: Total Pages/sec	Connection and Database	The rate at which database pages are looked up in the cache.
Cache Size: Current	Server	The current size of the database server cache, in kilobytes.
Cache Size: Maximum	Server	The maximum allowed size of the database server cache, in kilobytes.
Cache Size: Minimum	Server	The minimum allowed size of the database server cache, in kilobytes.
Cache Size: Peak	Server	The peak size of the database server cache, in kilobytes.

Checkpoint and recovery statistics

These statistics isolate the checkpoint and recovery actions performed when the database is in an idle state.

Statistic	Scope	Description
Checkpoint Flushes/sec	Database	The rate at which ranges of adjacent pages are written out during a checkpoint.
Checkpoint Log	Database	The rate at which the transaction log is checkpointed.

Statistic	Scope	Description
Checkpoint Urgency	Database	Checkpoint Urgency, expressed as a percentage.
Checkpoints/sec	Database	The rate at which checkpoints are performed.
Idle Actives/sec	Database	The rate at which the server's idle thread becomes active to do idle writes, idle checkpoints, etc.
Idle Checkpoint Time	Database	The total time spent doing idle checkpoints, in seconds.
Idle Checkpoints/sec	Database	The rate at which checkpoints are completed by the server's idle thread. An idle checkpoint occurs whenever the idle thread writes out the last dirty page in the cache.
Idle Writes/sec	Database	The rate at which disk writes are issued by the server's idle thread.
Recovery I/O Estimate	Database	The estimated number of I/O operations required to recover the database.
Recovery Urgency	Database	Recovery Urgency expressed as a percentage.

Communications statistics

These statistics describe client/server communications activity.

Statistic	Scope	Description
Comm: Buffer Misses	Server	The total number of network buffer allocations exceeding the connection buffer pool.
Comm: Bytes Received/sec	Connection and Server	The rate at which network data (in bytes) are received.
Comm: Bytes Received Uncompressed/sec	Connection and Server	The rate at which bytes would have been received if compression was disabled.
Comm: Bytes Sent/sec	Connection and Server	The rate at which bytes are transmitted over the network.

Statistic	Scope	Description
Comm: Bytes Sent Uncompressed/sec	Connection and Server	The rate at which bytes would have been sent if compression was disabled.
Comm: Free Buffers	Server	Number of free network buffers.
Comm: Licenses in Use	Server	The number of unique client network addresses connected.
Comm: Multi-packets Received/sec	Server	The rate at which multi-packet deliveries are received.
Comm: Multi-packets Sent/sec	Server	The rate at which multi-packet deliveries are transmitted.
Comm: Packets Received/sec	Connection and Server	The rate at which network packets are received.
Comm: Packets Received Uncompressed/sec	Connection and Server	The rate at which network packets would have been received if compression was disabled.
Comm: Packets Sent/sec	Connection and Server	The rate at which network packets are transmitted.
Comm: Packets Sent Uncompressed/sec	Connection and Server	The rate at which network packets would have been transmitted if compression was disabled.
Comm: Send Fails/sec	Server	The rate at which the underlying protocol(s) failed to send a packet.
Comm: TotalBuffers	Server	The total number of network buffers.

Disk I/O statistics

These statistics combine disk reads and disk writes to give overall information about the amount of activity devoted to disk I/O.

Statistic	Scope	Description
Disk: Active I/O	Database	The current number of file I/Os issued by the server which have not yet completed.
Disk: Maximum I/O	Database	The maximum value “Disk Reads: Active I/O” has reached.

Disk read statistics

These statistics describe the amount and type of activity devoted to reading information from disk.

Statistic	Scope	Description
Disk Reads: Total Pages/sec	Connection and Database	The rate at which pages are read from a file.
Disk Reads: Active	Database	The current number of file reads issued by the server which have not yet completed.
Disk Reads: Index interior/sec	Connection and Database	The rate at which index internal-node pages are being read from disk.
Disk Reads: Index leaf/sec	Connection and Database	The rate at which index leaf pages are being read from disk.
Disk Reads: Table/sec	Connection and Database	The rate at which table pages are being read from disk.
Disk Reads: Maximum Active	Database	The maximum value “Disk Reads: Active” has reached.

Disk write statistics

These statistics describe the amount and type of activity devoted to writing information to disk.

Statistic	Scope	Description
Disk Writes: Active	Database	The current number of file writes issued by the server which are not yet completed.
Disk Writes: Maximum Active	Database	The maximum value “Disk Writes: Active” has reached.

Statistic	Scope	Description
Disk Writes: Commit Files/sec	Database	The rate at which the server forces a flush of the disk cache. Windows NT/2000/XP and NetWare platforms use unbuffered (direct) I/O, so the disk cache does not need to be flushed.
Disk Writes: Database Extends/sec	Database	The rate at which the database file is extended, in pages/sec.
Disk Writes: Temp Extends/sec	Database	The rate at which temporary files are extended, in pages/sec.
Disk Writes: Pages/sec	Connection and Database	The rate at which modified pages are being written to disk.
Disk Writes: Transaction Log/sec	Connection and Database	The rate at which pages are written to the transaction log.
Disk Writes: Transaction Log Group Commits	Connection and Database	Occurs when a commit of the transaction log is requested but the log has already been written (so the commit was done for “free”).

Index statistics

These statistics describe the use of the index.

Statistic	Scope	Description
Index: Adds/sec	Connection and Database	The rate at which entries are added to indexes.
Index: Lookups/sec	Connection and Database	The rate at which entries are looked up in indexes.
Index: Full Compares/sec	Connection and Database	The rate at which comparisons beyond the hash value in an index must be performed.

Java VM statistics

These statistics describe the memory used by the Java VM.

Statistic	Scope	Description
JVM: Global Fixed Size	Server	The total bytes allocated to the Java VM fixed heap.
JVM: Heap Size	Connection and Database	The total bytes allocated to connection Java VMs.
JVM: Namespace Size	Database	The total number of bytes allocated to the Java VM namespace.

Memory pages statistics

These statistics describe the amount and purpose of memory used by the database server.

Statistic	Scope	Description
Mem Pages: Locked Heap	Server	The number of heap pages locked in the cache.
Mem Pages: Lock Table	Database	The number of pages used to store lock information.
Mem Pages: Rollback Log	Connection and Database	The number of pages in the rollback log.
Mem Pages: Main Heap	Server	The number of pages used for global server data structures.
Mem Pages: Map Pages	Database	The number of map pages used for accessing the lock table, frequency table, and table layout.
Mem Pages: Procedure Definitions	Database	The number of relocatable heap pages used for procedures.
Mem Pages: Relocatable	Database	The number of pages used for relocatable heaps (cursors, statements, procedures, triggers, views, etc.).
Mem Pages: Relocations/sec	Database	The rate at which relocatable heap pages are read from the temporary file.
Mem Pages: Trigger Definitions	Database	The number of relocatable heap pages used for triggers.
Mem Pages: View Definitions	Database	The number of relocatable heap pages used for views.

Request statistics

These statistics describe the database server activity devoted to responding to requests from client applications.

Statistic	Scope	Description
Requests	Server	The rate at which the server is entered to allow it to handle a new request or continue processing an existing request.
Requests: Active	Server	The number of server threads currently handling a request.
Requests: Unscheduled	Server	The number of requests that are currently queued up waiting for an available server thread.
Cursors	Connection	The number of declared cursors currently maintained by the server.
Cursors Open	Connection	The number of open cursors currently maintained by the server.
Statements	Connection	The number of prepared statements currently maintained by the server.
Statement Prepares	Connection	The rate at which statement prepares are being handled by the server.
Transaction Commits	Connection	The rate at which Commit requests are handled.
Transaction Rollbacks	Connection	The rate at which Rollback requests are handled.

Miscellaneous statistics

Statistic	Scope	Description
Avail IO	Server	Expressed as a count.
Connection Count	Database	The number of connections to this database.
Main Heap Bytes	Server	The number of bytes used for global server data structures.
Query Low Memory Strategy	Connection and Database	The number of times the server changed its execution plan during execution because of low memory conditions.
Temporary Table Pages	Connection and Database	The number of pages in the temporary file used for temporary tables.

Database properties

Adaptive Server Anywhere provides a set of properties that are made available to client applications. These properties describe aspects of connection, database, and database server behavior.

Accessing properties

Each type of property can be accessed by supplying its name as an argument to a system function.

❖ To access connection properties

1. Use the connection_property system function: the following statement returns the number of pages that have been read from file by the current connection.

   ```
   SELECT connection_property ( 'DiskRead' )
   ```

❖ To access database properties

1. Use the db_property system function. For example, the following statement returns the page size of the current database:

   ```
   SELECT db_property ( 'PageSize' )
   ```

❖ To access database server properties

1. Use the property system function: the following statement returns the number of pages that have been read from file by the current connection.

   ```
   SELECT property ( 'MainHeapPages' )
   ```

Connection-level properties

The following table lists properties available for each connection.

Examples

❖ To retrieve the value of a connection property

1. Use the connection_property system function. The following statement returns the number of pages that have been read from file by the current connection.

   ```
   SELECT connection_property ( 'DiskRead' )
   ```

❖ To retrieve the values of all connection properties

1. Use the sa_conn_properties system procedure:

   ```
   CALL sa_conn_properties
   ```

   A separate row appears for each connection.

Descriptions

Property	Description
Allow_nulls_by_default	"ALLOW_NULLS_BY_DEFAULT option [compatibility]" on page 633
Ansi_blanks	"ANSI_BLANKS option [compatibility]" on page 633
Ansi_close_cursors_on_-rollback	"ANSI_CLOSE_CURSORS_ON_ROLLBACK option [compatibility]" on page 633
Ansi_integer_overflow	"ANSI_INTEGER_OVERFLOW option [compatibility]" on page 634
Ansi_permissions	"ANSI_PERMISSIONS option [compatibility]" on page 634
Ansi_update_constraints	"ANSI_UPDATE_CONSTRAINTS option [compatibility]" on page 635
Ansinull	"ANSINULL option [compatibility]" on page 636
AppInfo	Returns information about the client that made the connection. For HTTP connections, this includes information about the browser. For connections using older versions of jConnect or Open Client, the information may be incomplete. The API value can be DBLIB, ODBC, OLEDB, or ADO.NET. ☞ For more information about the values returned for other types of connections, see "AppInfo connection parameter [APP]" on page 177.
Auditing	"AUDITING option [database]" on page 637
AuditingTypes	The types of auditing currently enabled. "AUDITING option [database]" on page 637

Property	Description
Automatic_timestamp	"AUTOMATIC_TIMESTAMP option [compatibility]" on page 638
Background_priority	"BACKGROUND_PRIORITY option [database]" on page 639
BlockedOn	If the current connection is not blocked, this is zero. If it is blocked, the connection number on which the connection is blocked due to a locking conflict.
Blocking	"BLOCKING option [database]" on page 640
Blocking_timeout	"BLOCKING_TIMEOUT option [database]" on page 640
BytesReceived	The number of bytes received during client/server communications.
BytesReceivedUncomp	The number of bytes that would have been received during client/server communications if compression was disabled. (This value is the same as the value for BytesReceived if compression is disabled.)
BytesSent	The number of bytes sent during client/server communications.
BytesSentUncomp	The number of bytes that would have been sent during client/server communications if compression was disabled. (This value is the same as the value for BytesSent if compression is disabled.)
CacheHits	The number of successful reads of the cache.
CacheRead	The number of database pages that have been looked up in the cache.
CacheReadIndInt	The number of index internal-node pages that have been read from the cache.
CacheReadIndLeaf	The number of index leaf pages that have been read from the cache.
CacheReadTable	The number of table pages that have been read from the cache.

Property	Description
Chained	"CHAINED option [compatibility]" on page 640
CharSet	The character set used by the connection.
Checkpoint_time	"CHECKPOINT_TIME option [database]" on page 641
Cis_option	"CIS_OPTION option [database]" on page 641
Cis_rowset_size	Reserved
ClientLibrary	Returns**jConnect** for jConnect connections; **CT_Library** for Open Client connections; **None** for HTTP connections, and **CmdSeq** for ODBC, embedded SQL, OLE DB, ADO.NET, and iAnywhere JDBC driver connections.
ClientPort	Returns the client's TCP/IP port number or **0** if the connection is not a TCP/IP connection.
Close_on_EndTrans	"CLOSE_ON_ENDTRANS option [compatibility]" on page 642
Commit	The number of Commit requests that have been handled.
CommLink	The communication link for the connection. This is one of the network protocols supported by Adaptive Server Anywhere, or **local** for a same-machine connection.
CommNetworkLink	The communication link for the connection. This is one of the network protocols supported by Adaptive Server Anywhere. Values can include **SharedMemory**, **TCPIP**, **SPX**, or **NamedPipes**. The CommLinkNetwork property always returns the name of the link, regardless of whether it is same-machine or not.
CommProtocol	Returns **TDS** for Open Client and jConnect connections, **HTTP** for HTTP connections, and **CmdSeq** for OLE DB, ADO.NET and iAnywhere JDBC driver connections.

Property	Description
Compression	Returns ON or OFF to indicate whether communication compression is enabled on the connection.
Connection_-authentication	A string used to authenticate the client. Authentication is required before the database can be modified.
Conversion_error	"CONVERSION_ERROR option [compatibility]" on page 645
Cooperative_commit_-timeout	"COOPERATIVE_COMMIT_TIMEOUT option [database]" on page 645
Cooperative_commits	"COOPERATIVE_COMMITS option [database]" on page 645
Cursor	The number of declared cursors that are currently being maintained by the server.
CursorOpen	The number of open cursors that are currently being maintained by the server.
Database_authentication	A string used to authenticate the database. Authentication is required before the database can be modified.
Date_format	"DATE_FORMAT option [compatibility]" on page 646
Date_order	"DATE_ORDER option [compatibility]" on page 648
DBNumber	The ID number of the database.
Debug_messages	"DEBUG_MESSAGES option [database]" on page 648
Dedicated_task	"DEDICATED_TASK option [database]" on page 649
Default_timestamp_-increment	"DEFAULT_TIMESTAMP_INCREMENT option [database]" on page 650
Delayed_commit_timeout	"DELAYED_COMMIT_TIMEOUT option [database]" on page 650
Delayed_commits	"DELAYED_COMMITS option [database]" on page 651

Property	Description
DiskRead	The number of pages that have been read from disk.
DiskReadIndInt	The number of index internal-node pages that have been read from disk.
DiskReadIndLeaf	The number of index leaf pages that have been read from disk.
DiskReadTable	The number of table pages that have been read from disk.
DiskWrite	The number of modified pages that have been written to disk.
Divide_by_zero_error	"DIVIDE_BY_ZERO_ERROR option [compatibility]" on page 652
Encryption	"Encryption connection parameter [ENC]" on page 191
Escape_character	"ESCAPE_CHARACTER option [compatibility]" on page 653
EventName	The name of the associated event if the connection is running an event handler. Otherwise, the result is NULL.
Exclude_operators	"EXCLUDE_OPERATORS option [database]" on page 653
Extended_join_syntax	"EXTENDED_JOIN_SYNTAX option [database]" on page 653
Fire_triggers	"FIRE_TRIGGERS option [compatibility]" on page 654
First_day_of_week	"FIRST_DAY_OF_WEEK option [database]" on page 654
Float_as_double	"FLOAT_AS_DOUBLE option [compatibility]" on page 655
For_xml_null_treatment	"FOR_XML_NULL_TREATMENT option [database]" on page 656
Force_view_creation	"FORCE_VIEW_CREATION option [database]" on page 656

Property	Description
FullCompare	The number of comparisons that have been performed beyond the hash value in an index.
Global_database_id	"GLOBAL_DATABASE_ID option [database]" on page 656
IdleTimeout	The idle timeout value of the connection. ☞ For more information, see "Idle connection parameter [IDLE]" on page 196.
IndAdd	The number of entries that have been added to indexes.
IndLookup	The number of entries that have been looked up in indexes.
Integrated_server_name	"INTEGRATED_SERVER_NAME option [database]" on page 658
Isolation_level	"ISOLATION_LEVEL option [compatibility]" on page 659
Java_heap_size	"JAVA_HEAP_SIZE option [database]" on page 663
Java_input_output	"JAVA_INPUT_OUTPUT option [database]" on page 664
Java_namespace_size	"JAVA_NAMESPACE_SIZE option [database]" on page 664
Java_page_buffer_size	The page buffer size used by the Java VM.
JavaHeapSize	The heap size per Java VM.
Language	The locale language.
LastIdle	The number of ticks between requests.
LastReqTime	The time at which the last request for the specified connection started.
LastStatement	The most recently prepared SQL statement for the current connection. ☞ For more information, see "-zl server option" on page 166.

Property	Description
LivenessTimeout	The liveness timeout period for the current connection. ☞ For more information, see "LivenessTimeout connection parameter [LTO]" on page 199.
Lock_rejected_rows	Reserved
LockName	A 64-bit unsigned integer value representing the lock for which a connection is waiting.
Log_deadlocks	"LOG_DEADLOCKS option [database]" on page 665
LogFreeCommit	The number of Redo Free Commits. A Redo Free Commit occurs when a commit of the transaction log is requested but the log has already been written (so the commit was done for free.)
Login_mode	"LOGIN_MODE option [database]" on page 665
Login_procedure	"LOGIN_PROCEDURE option [database]" on page 666
LoginTime	The date and time the connection was established.
LogWrite	The number of pages that have been written to the transaction log.
Max_cursor_count	"MAX_CURSOR_COUNT option [database]" on page 668
Max_plans_cached	"MAX_PLANS_CACHED option [database]" on page 669
Max_recursive_iterations	"MAX_RECURSIVE_ITERATIONS option [database]" on page 669
Max_statement_count	"MAX_STATEMENT_COUNT option [database]" on page 670
MessageReceived	The string that was generated by the MESSAGE statement that caused the WAITFOR statement to be interrupted. Otherwise, an empty string is returned.

Property	Description
Min_password_length	"MIN_PASSWORD_LENGTH option [database]" on page 671
Name	The name of the current connection.
Nearest_century	"NEAREST_CENTURY option [compatibility]" on page 671
NodeAddress	The node for the client in a client/server connection. When the client and server are both on the same machine, an empty string is returned.
Non_keywords	"NON_KEYWORDS option [compatibility]" on page 672
Number	The ID number of the connection.
ODBC_describe_binary_-as_varbinary	"ODBC_DESCRIBE_BINARY_AS_VARBINARY [database]" on page 673
ODBC_distinguish_-char_and_varchar	"ODBC_DISTINGUISH_CHAR_AND_VARCHAR option [database]" on page 673
On_charset_conversion_-failure	"ON_CHARSET_CONVERSION_FAILURE option [database]" on page 674
On_tsql_error	"ON_TSQL_ERROR option [compatibility]" on page 675
Optimistic_wait_for_-commit	"OPTIMISTIC_WAIT_FOR_COMMIT option [compatibility]" on page 675
Optimization_goal	"OPTIMIZATION_GOAL option [database]" on page 676
Optimization_level	Reserved
Optimization_workload	"OPTIMIZATION_WORKLOAD option [database]" on page 678
PacketSize	The packet size used by the connection, in bytes.
PacketsReceived	The number of client/server communication packets received.

Property	Description
PacketsReceivedUncomp	The number of packets that would have been received during client/server communications if compression was disabled. (This value is the same as the value for PacketsReceived if compression is disabled.)
PacketsSent	The number of client/server communication packets sent.
PacketsSentUncomp	The number of packets that would have been sent during client/server communications if compression was disabled. (This value is the same as the value for PacketsSent if compression is disabled.)
Percent_as_comment	"PERCENT_AS_COMMENT option [compatibility]" on page 681
Pinned_cursor_percent_of_cache	"PINNED_CURSOR_PERCENT_OF_CACHE option [database]" on page 681
Precision	"PRECISION option [database]" on page 682
Prefetch	"PREFETCH option [database]" on page 682
Prepares	The number of statement preparations carried out.
PrepStmt	The number of prepared statements currently being maintained by the server.
Preserve_source_format	"PRESERVE_SOURCE_FORMAT option [database]" on page 683
Prevent_article_pkey_update	"PREVENT_ARTICLE_PKEY_UPDATE option [database]" on page 684
Query_plan_on_open	"QUERY_PLAN_ON_OPEN option [compatibility]" on page 684
QueryBypassed	The number of requests optimized by the optimizer bypass.
QueryCachedPlans	The number of query execution plans currently cached for the connection.
QueryCachePages	The number of pages used to cache execution plans.

Property	Description
QueryLowMemoryStrategy	The number of times the server changed its execution plan during execution as a result of low memory conditions. The strategy can change because less memory is available than the optimizer estimated, or because the execution plan required more memory than the optimizer estimated.
QueryOptimized	The number of requests that have been fully optimized.
QueryReused	The number of requests that have been reused from the plan cache.
Quoted_identifier	"QUOTED_IDENTIFIER option [compatibility]" on page 685
Read_past_deleted	"READ_PAST_DELETED option [database]" on page 685
Recovery_time	"RECOVERY_TIME option [database]" on page 686
Remote_idle_timeout	"REMOTE_IDLE_TIMEOUT option [database]" on page 686
Replicate_all	"REPLICATE_ALL option [replication]" on page 687
ReqType	A string for the type of the last request.
Return_date_time_as_string	"RETURN_DATE_TIME_AS_STRING option [database]" on page 688
RI_trigger_time	"RI_TRIGGER_TIME option [compatibility]" on page 689
Rlbk	The number of Rollback requests that have been handled.
Rollback_on_deadlock	"ROLLBACK_ON_DEADLOCK [database]" on page 689
RollbackLogPages	The number of pages in the rollback log.
Row_counts	"ROW_COUNTS option [database]" on page 689
Scale	"SCALE option [database]" on page 690

Property	Description
ServerPort	Returns the server's TCP/IP port number or **0**.
Sort_collation	"SORT_COLLATION option [database]" on page 690
SQL_flagger_error_level	"SQL_FLAGGER_ERROR_LEVEL option [compatibility]" on page 691
SQL_flagger_warning_-level	"SQL_FLAGGER_WARNING_LEVEL option [compatibility]" on page 691
String_rtruncation	"STRING_RTRUNCATION option [compatibility]" on page 692
Subsume_row_locks	"SUBSUME_ROW_LOCKS option [database]" on page 693
Suppress_TDS_-debugging	"SUPPRESS_TDS_DEBUGGING option [database]" on page 693
TDS_empty_string_is_-null	"TDS_EMPTY_STRING_IS_NULL option [database]" on page 693
Temp_space_limit_check	"TEMP_SPACE_LIMIT_CHECK option [database]" on page 694
TempTablePages	The number of pages in the temporary file used for temporary tables.
Time_format	"TIME_FORMAT option [compatibility]" on page 694
Timestamp_format	"TIMESTAMP_FORMAT option [compatibility]" on page 695
TimeZoneAdjustment	The number of minutes that must be added to the Coordinated Universal Time (UTC) to display time local to the connection. By default, the value is set according to the client's time zone.
TransactionStartTime	A string containing the time the database was first modified after a COMMIT or ROLLBACK, or an empty string if no modifications have been made to the database since the last COMMIT or ROLLBACK.
Truncate_date_values	"TRUNCATE_DATE_VALUES option [database] (deprecated)" on page 697

Property	Description
Truncate_timestamp_-values	"TRUNCATE_TIMESTAMP_VALUES option [database]" on page 697
Truncate_with_-autocommit	"TRUNCATE_WITH_AUTO_COMMIT option [database]" on page 699
Tsql_hex_constant	"TSQL_HEX_CONSTANT option [compatibility]" on page 699
Tsql_variables	"TSQL_VARIABLES option [compatibility]" on page 700
UncommitOp	The number of uncommitted operations.
User_estimates	"USER_ESTIMATES option [database]" on page 700
UserAppInfo	The string specified by the AppInfo connection parameter in a connection string. ☞ For more information, see "AppInfo connection parameter [APP]" on page 177.
Userid	The user ID for the connection.
UtilCmdsPermitted	Returns ON or OFF to indicate whether utility commands such as CREATE DATABASE, DROP DATABASE, and RESTORE DATABASE are permitted for the connection. ☞ For more information, see "-gu server option" on page 148.
Wait_for_commit	"WAIT_FOR_COMMIT option [database]" on page 702

Server-level properties

The following table lists properties that apply across the server as a whole.

Examples

❖ To retrieve the value of a server property

1. Use the property system function. For example, the following statement returns the number of cache pages being used to hold the main heap:

```
SELECT property ( 'MainHeapPages' )
```

❖ To retrieve the values of all server properties

1. Use the sa_eng_properties system procedure:

```
CALL sa_eng_properties
```

Descriptions

Property	Description
ActiveReq	The number of server threads that are currently handling a request.
AvailIO	Reserved
BuildChange	Reserved
BuildClient	Reserved
BuildReproducible	Reserved
BytesReceived	The number of bytes received during client/server communications.
BytesReceivedUncomp	The number of bytes that would have been received during client/server communications if compression was disabled. (This value is the same as the value for BytesReceived if compression is disabled.)
BytesSent	The number of bytes sent during client/server communications.
BytesSentUncomp	The number of bytes that would have been sent during client/server communications if compression was disabled. (This value is the same as the value for BytesSent if compression is disabled.)
C2	Returns YES if the -sc option was used when the server was started. Otherwise, returns NO. ☞ For more information, see "-sc server option" on page 157.
CacheHitsEng	The number of database page lookups.
CacheReplacements	The number of pages in the cache that have been replaced.
CharSet	The character set in use by the database server.

Property	Description
CommandLine	The command line that was used to start the server. If the encryption key for a database was specified using the -ek option, the key is replaced with a constant string of asterisks in the value returned by this property. If you need to specify the encryption key, you can start the database server with the -ep option to be prompted for the key, or use the START DATABASE statement. As well, if the database can be autostarted, the key can be provided in the DBKEY connection parameter.
CompactPlatformVer	A condensed version of the PlatformVer property.
CompanyName	The name of the company owning this software.
ConnsDisabled	Returns ON or OFF to indicate the current setting of the server option to disable new connections. ☞ For information, see “sa_server_option system procedure” [*ASA SQL Reference*, page 830].
ConsoleLogFile	Returns the name of the file where messages from the database server window are logged if the -o option was specified, otherwise returns an empty string.
CurrentCacheSize	The current cache size, in kilobytes.
DefaultCollation	The collation that would be used for new databases, if none is explicitly specified.
FipsMode	Returns YES if the -fips option was specified when the database server was started, and NO otherwise.
FreeBuffers	The number of available network buffers.
IdleTimeout	The default idle timeout. ☞ For more information, see “-ti server option” on page 157.
IsFipsAvailable	Returns YES if the FIPS DLL is installed, and NO otherwise.
IsIQ	Returns YES if the server is an IQ server, and NO otherwise.

Property	Description
IsJavaAvailable	Returns YES if the JavaVM is installed, and NO if the JavaVM is not installed. This property only indicates if the Java VM is available, not whether it is currently being used.
IsNetworkServer	Returns YES if connected to a network database server, and NO if connected to a personal database server.
IsRuntimeServer	Returns YES if connected to the limited desktop runtime database server, and NO otherwise.
JavaGlobFix	Java VM global fixed size.
Language	The locale language for the server.
LegalCopyright	The copyright string for the software.
LegalTrademarks	Trademark information for the software.
LicenseCount	The number of licensed seats or processors.
LicensedCompany	The name of the licensed company.
LicensedUser	The name of the licensed user.
LicensesInUse	The number of concurrent users currently connected to the network server, as determined by the number of unique client network addresses connected to the server.
LicenseType	The license type. Can be networked seat (per-seat) or cpu-based.
LivenessTimeout	The client liveness timeout default.
LockedHeapPages	The number of heap pages locked in the cache.
MachineName	The name or IP address of the computer running a database server.
MainHeapBytes	The number of bytes used for global server data structures.
MainHeapPages	The number of pages used for global server data structures.
MaxCacheSize	The maximum allowed cache size, in kilobytes.

Property	Description
MaxMessage	The current maximum line number that can be retrieved from the server's message window. This represents the most recent message displayed in the server's message window.
Message, *linenumber*	A line from the server's message window, prefixed by the date and time the message appeared. The second parameter specifies the line number. The value returned by `PROPERTY( "message" )` is the first line of output that was written to the Server Messages window. Calling `PROPERTY( "message", i )` returns the i-th line of server output (with zero being the first line). The buffer is finite, so as messages are generated, the first lines are dropped and may no longer be available in memory. In this case, NULL is returned.
MessageText, *linenumber*	The text associated with the specified line number in the server's message window, without a date and time prefix. The second parameter specifies the line number.
MessageTime, *linenumber*	The date and time associated with the specified line number in the server's message window. The second parameter specifies the line number.
MessageWindowSize	The maximum number of lines that can be retrieved from the server's message window.
MinCacheSize	The minimum allowed cache size, in kilobytes.
MultiPacketsReceived	The number of multi-packet deliveries received during client/server communications.
MultiPacketsSent	The number of multi-packet deliveries sent during client/server communications.
Name	The name of the server.

Property	Description
NativeProcessorArchitecture	On platforms where a processor can be emulated (such as X86 on Win64), returns a string that identifies the native processor type. In all other cases, it returns the same value as property('ProcessorArchitecture'). Values can include: 32-bit Windows (not CE) - X86 NetWare - X86 Intel Solaris - X86 CE - SH3, SH4, MIPS or ARM 64-bit Windows - IA64 or AMD64 64-bit UNIX - IA64 or AMD64 Solaris - SPARC AIX - PPC MAC OS - PPC HP - PA_RISC DEC UNIX - ALPHA Linux - X86, SPARC, IA64
NumProcessorsAvail	The number of processors on the server.
NumProcessorsMax	The maximum number of processors used. Normally this should be 2 for *dbeng.exe* and 0 for *dbsrv.exe*.
PacketsReceived	The number of client/server communication packets received.
PacketsReceivedUncomp	The number of packets that would have been received during client/server communications if compression was disabled. (This value is the same as the value for PacketsReceived if compression is disabled.)
PacketsSent	The number of client/server communication packets sent.
PacketsSentUncomp	The number of packets that would have been sent during client/server communications if compression was disabled. (This value is the same as the value for PacketsSent if compression is disabled.)

Property	Description
PageSize	The size of the database server cache pages. This can be set using the -gp option, otherwise, it is the maximum database page size of the databases specified on the command line.
PeakCacheSize	The largest value the cache has reached in the current session, in kilobytes.
Platform	The operating system on which the software is running. For example, if you are running on Windows 2000, this property returns `Windows2000`.
PlatformVer	The operating system on which the software is running, including build numbers, service packs, etc. For example, it could return `Windows 2000 Build 2195 Service Pack 3 .`
ProcessCPU	CPU usage statistics for the server process. Values are in seconds. This property is supported on Windows NT/2000/XP, Windows 95/98/Me, and UNIX. This property is not supported on Windows CE or NetWare.
ProcessCPUSystem	Process CPU system usage. Values are in seconds. This property is supported on Windows NT/2000/XP, Windows 95/98/Me, and UNIX. This property is not supported on Windows CE or NetWare.
ProcessCPUUser	Process CPU user usage. Values are in seconds. This property is supported on Windows NT/2000/XP, Windows 95/98/Me, and UNIX. This property is not supported on Windows CE or NetWare.

Property	Description
ProcessorArchitecture	A string that identifies the processor type. Values can include: 32-bit Windows (not CE) - X86 NetWare - X86 Intel Solaris - X86 CE - SH3, SH4, or ARM 64-bit Windows - IA64 or AMD64 64-bit UNIX - IA64 or AMD64 Solaris - SPARC AIX - PPC MAC OS - PPC HP - PA_RISC DEC UNIX - ALPHA
ProductName	The name of the software.
ProfileFilterConn	Returns the ID of the connection being monitored if procedure profiling for a specific connection is turned on. Otherwise, returns an empty string. You control procedure profiling by user with the sa_server_option procedure. ☞ For more information, see "sa_server_option system procedure" [*ASA SQL Reference*, page 830].
ProfileFilterUser	Returns the name of the user being monitored if procedure profiling for a specific user is turned on. Otherwise, returns an empty string. You control procedure profiling by user with the sa_server_-option procedure. ☞
ProductVersion	The version of the software being run.
QuittingTime	Shutdown time for the server. If none is specified, the value is **none**.
RememberLastStatement	Returns ON if the server is recording the last statement prepared by each connection, and OFF otherwise.
Req	The number of times the server has been entered to allow it to handle a new request or continue processing an existing request.

Property	Description
RequestLogFile	The name of the request logging file. An empty string is returned if there is no level logging. ☞ For information, see "sa_server_option system procedure" [*ASA SQL Reference*, page 830].
RequestLogging	ALL, SQL, or NONE. ☞ For information, see "sa_server_option system procedure" [*ASA SQL Reference*, page 830].
RequestLogNumFiles	The number of request log files being kept. ☞ For more information, see "sa_server_option system procedure" [*ASA SQL Reference*, page 830].
SendFail	The number of times that the underlying communications protocols have failed to send a packet.
StartTime	The date/time that the server started
Tempdir	The directory in which temporary files are stored by the server.
TimeZoneAdjustment	The number of minutes that must be added to the Coordinated Universal Time (UTC) to display time local to the server.
TotalBuffers	The total number of network buffers.
UnschReq	The number of requests that are currently queued up waiting for an available server thread.

Database-level properties

The following table lists properties available for each database on the server.

Examples

❖ To retrieve the value of a database property

1. Use the db_property system function. For example, the following statement returns the page size of the current database:

```
SELECT db_property ( 'PageSize' )
```

❖ To retrieve the values of all database properties

1. Use the sa_db_properties system procedure:

```
CALL sa_db_properties
```

Descriptions

Property	Description
Alias	The database name.
BlankPadding	The status of the blank padding feature. Returns ON if the database has blank padding enabled. Otherwise, it returns OFF.
BlobArenas	The status of the BlobArenas feature. Returns ON if the database stores extension (BLOB) pages separately from table pages for the database. Otherwise, returns OFF.
CacheHits	The number of database page lookups satisfied by finding the page in the cache.
CacheRead	The number of database pages that have been looked up in the cache.
CacheReadIndInt	The number of index internal-node pages that have been read from the cache.
CacheReadIndLeaf	The number of index leaf pages that have been read from the cache.
CacheReadTable	The number of table pages that have been read from the cache.
Capabilities	The capability bits enabled for the database. This property is primarily for use by technical support.
CaseSensitive	The status of the case sensitivity feature. Returns ON if the database is case sensitive. Otherwise, it returns OFF.
CaseSensitivePass-words	The status of password case sensitivity. In versions 9.0.0 and later, password case sensitivity is independent of database case sensitivity. Returns ON if database passwords are case sensitive. Otherwise, it returns OFF.
CharSet	The character set of the database.

Property	Description
CheckpointUrgency	The time that has elapsed since the last checkpoint as a percentage of the checkpoint time setting of the database.
Checksum	Returns ON if database page checksums are enabled for the database. Otherwise, returns OFF.
Chkpt	The number of checkpoints that have been performed.
ChkptFlush	The number of ranges of adjacent pages written out during a checkpoint.
ChkptPage	The number of transaction log checkpoints.
CommitFile	The number of times the server has forced a flush of the disk cache. On Windows NT/2000/XP and NetWare platforms, the disk cache does not need to be flushed if unbuffered (direct) I/O is used.
CompressedBTrees	Returns ON if Compressed B-tree indexes are supported. Otherwise, returns OFF.
Compression	The compression status of the database. Returns either ON (meaning the database is compressed) or OFF. If a write file is created on a compressed database, the write file is NOT compressed. Starting a write file created on a compressed database and selecting db_property('compression'), returns OFF.
ConnCount	The number of connections to the database.
CurrentRedoPos	The current offset in the transaction log file where the next database operation is to be logged.
CurrIO	The current number of file I/Os that were issued by the server but have not yet completed.
CurrRead	The current number of file reads that were issued by the server but have not yet completed.
CurrWrite	The current number of file writes that were issued by the server but have not yet completed.
DBFileFragments	The number of database file fragments. This property is supported on Windows NT/2000/XP.

Property	Description
DiskRead	The number of pages that have been read from disk.
DiskReadIndInt	The number of index internal-node pages that have been read from disk.
DiskReadIndLeaf	The number of index leaf pages that have been read from disk.
DiskReadTable	The number of table pages that have been read from disk.
DiskWrite	The number of modified pages that have been written to disk.
DriveType *dbspace*	The drive on which the database file is located. Returns CD, FIXED, RAMDISK, REMOTE, REMOVABLE, and UNKNOWN. On UNIX, depending on the version of UNIX and the type of drive, it may not be possible to determining the drive type. In these cases “UNKNOWN” is returned. When used with **db_extended_property**, you can specify which dbspace you want the size for. *Dbspace* can be either the *name* of the dbspace or the *file_id* of the dbspace. Leaving *dbspace* unspecified or using *system* both refer to the system dbspace. If the specified dbspace does not exist, the property function returns NULL. If the name of a dbspace is specified and the ID of a database that is not the database of the current connection is also specified, the function also returns NULL.
Encryption	The type of encryption applied to the database. Returns None, Simple, or AES.
ExtendDB	The number of pages by which the database file has been extended.
ExtendTempWrite	The number of pages by which temporary files have been extended.
File	The file name of the database root file, including path.

Property	Description
FileSize *dbspace*	When used with **db_property**, this property returns the file size of the system dbspace, in pages. When used with **db_extended_property**, you can specify which dbspace you want the size for. *Dbspace* can be either the *name* of the dbspace, the *file_id* of the dbspace, or *temporary* to refer to the temporary dbspace. You can also specify *translog* to return the size of the log file. Finally, you can specify *writefile* to refer to the write file. When using a write file, FileSize on a dbspace returns the amount of space in the virtual dbspace, represented by the underlying dbspace plus the modifications to that dbspace that have been stored in the write file. Leaving the *dbspace* unspecified, or using *system*, both refer to the system dbspace. If the specified dbspace does not exist, the property function returns NULL. If the name of a dbspace is specified and an id or name of a database which is not the database of the current connection is also specified, the function also returns NULL.
FileVersion	The version of the database file. This does not correspond to a software release version.
FreePageBitMaps	Returns ON if free database pages are managed via bitmaps. Otherwise, returns OFF.

Property	Description
FreePages *dbspace*	FreePages is only supported on databases created with version 8.0.0 or later. When used with **db_property**, this property returns the number of free pages in the system dbspace. When used with **db_extended_property**, you can specify which dbspace you want the number of free pages for. *Dbspace* can be either the *name* of the dbspace, the *file_id* of the dbspace, or *temporary* to refer to the temporary dbspace. You can also specify *translog* to return the number of free pages in the log file. Finally, you can specify *writefile* to refer to the write file. When using a write file, FreePages on a dbspace returns the number of free pages in the virtual dbspace, represented by the underlying dbspace plus the modifications to that dbspace that have been stored in the write file. Leaving the *dbspace* unspecified, or using *system* both refer to the system dbspace. If the specified dbspace does not exist, the property function returns null. If the name of a dbspace is specified and an id or name of a database which is not the database of the current connection is also specified, the function also returns null.
FullCompare	The number of comparisons that have been performed beyond the hash value in an index.
GlobalDBId	The value of the GLOBAL_DATABASE_ID option used to generate unique primary key values in a replication environment.
Histograms	Returns ON if optimizer statistics are maintained as histograms. Otherwise, returns OFF.
IdleCheck	The number of times that the server's idle thread has become active to do idle writes, idle checkpoints, and so on.
IdleChkpt	The number of checkpoints completed by the server's idle thread. An idle checkpoint occurs whenever the idle thread writes out the last dirty page in the cache.

Property	Description
IdleChkTime	The number of 100ths of a second spent checkpointing during idle I/O.
IdleWrite	The number of disk writes that have been issued by the server's idle thread.
IndAdd	The number of entries that have been added to indexes.
IndLookup	The number of entries that have been looked up in indexes.
IOToRecover	The estimated number of I/O operations required to recover the database.
IQStore	Reserved.
JavaHeapSize	Heap size per Java VM.
JavaNSSize	Java VM Namespace size.
JDKVersion	The Java runtime library version used by this database.
Language	Returns a comma-separated list of languages known to be supported by the database collation. The languages are in two-letter ISO format. If the language is not known (usually a custom collation), the return value is NULL. ☞ For a list of the two-letter ISO format language names and the language they correspond to, see "Understanding the locale language" on page 331.
LargeProcedureIDs	Returns ON if 32-bit stored procedure IDs are supported for the database. Otherwise, returns OFF.
LockTablePages	The number of pages used to store lock information.
LogFileFragments	The number of log file fragments. This property is supported on Windows NT/2000/XP.
LogFreeCommit	The number of Redo Free Commits. A Redo Free Commit occurs when a commit of the transaction log is requested but the log has already been written (so the commit was done for free).

Property	Description
LogName	The file name of the transaction log, including path.
LogWrite	The number of pages that have been written to the transaction log.
LTMGeneration	The generation number of the LTM or Replication Agent. This property is primarily for use by technical support.
LTMTrunc	The minimal confirmed log offset for the Replication Agent.
MapPages	The number of map pages used for accessing the lock table, frequency table, and table layout.
MaxIO	The maximum value that CurrIO has reached.
MaxRead	The maximum value that CurrRead has reached.
MaxWrite	The maximum value that CurrWrite has reached.
MultiByteCharSet	Returns ON if the database uses a multi-byte character set. Otherwise, returns OFF.
Name	The database name (identical to alias).
PageRelocations	The number of relocatable heap pages that have been read from the temporary file.
PageSize	The page size of the database, in bytes.
PreserveSource	Returns ON if the database preserves the source for procedures and views. Otherwise, returns OFF.
ProcedurePages	The number of relocatable heap pages that have been used for procedures.
ProcedureProfiling	Returns ON if procedure profiling is turned on for the database. Otherwise, returns OFF.
QueryBypassed	The number of requests optimized by the optimizer bypass.
QueryCachedPlans	The number of cached execution plans across all connections.
QueryCachePages	The number of pages used to cache execution plans.

Property	Description
QueryLowMemoryStrategy	The number of times the server changed its execution plan during execution as a result of low memory conditions. The strategy can change because less memory is available than the optimizer estimated, or because the execution plan required more memory than the optimizer estimated.
QueryOptimized	The number of requests fully optimized.
QueryBypassed	The number of requests reused from the plan cache.
ReadOnly	Returns ON if the database is being run in read-only mode. Otherwise, returns OFF.
RecoveryUrgency	An estimate of the amount of time required to recover the database.
RelocatableHeapPages	The number of pages used for relocatable heaps (cursors, statements, procedures, triggers, views, etc.).
RemoteTrunc	The minimal confirmed log offset for the SQL Remote Message Agent.
RollbackLogPages	The number of pages in the rollback log.
SeparateCheckpointLog	Returns ON if the checkpoint log for the database is maintained at the end of the SYSTEM dbspace. Otherwise, returns OFF.
SeparateForeignKeys	Returns ON if primary and foreign keys are stored separately. Otherwise, returns OFF.
SyncTrunc	The minimal confirmed log offset for the MobiLink client dbmlsync executable.
TableBitMaps	Returns ON if the database supports table bitmaps. Otherwise, returns OFF.
TempFileName	The file name of the database temporary file, including path.
TempTablePages	The number of pages in the temporary file used for temporary tables.
TransactionsSpanLogs	Returns ON if transactions can span multiple log files. Otherwise, returns OFF.

Property	Description
TriggerPages	The number of relocatable heap pages used for triggers.
VariableHashSize	Returns ON if the hash size can be specified for B-tree indexes. Otherwise, returns OFF.
ViewPages	The number of relocatable heap pages used for views.

CHAPTER 18

Physical Limitations

About this chapter

This chapter describes the limitations on size and number of objects in Adaptive Server Anywhere databases.

Contents

Size and number limitations

The following table lists the physical limitations on size and number of objects in an Adaptive Server Anywhere database. The memory, CPU, and disk drive of the computer are more limiting factors in most cases.

Item	Limitation
Database size	13 files per database. For each file, the largest file allowed by operating system and file system
Field size	2 GB
File size (FAT 12)	16 MB
File size (FAT 16)	2 GB
File size (FAT 32)	4 GB
File size for NTFS, NetWare (NSS volumes), HP-UX 11.0 and later, Solaris 2.6 and later, Linux 2.4 and later)	♦ 256 GB for 1 KB pages ♦ 512 GB for 2KB pages ♦ 1 TB for 4 KB pages ♦ 2 TB for 8 KB pages
NetWare (traditional volumes)	4 GB
File size (all other platforms and file systems)	2 GB
Maximum cache size (non-AWE cache) (Windows 95/98, Windows NT, Windows 2000 Professional, Windows 2000 Server, Windows XP Home Edition, Windows XP Professional, Windows Server 2003 Web Edition, Windows Server 2003 Standard Edition)	1.8 GB
Maximum cache size (non-AWE cache) (Windows NT Advanced Server, Windows 2000 Advanced Server, Windows 2000 Enterprise Server, Windows 2000 Datacenter Server, Windows Server 2003 Enterprise Edition, Windows Server 2003 Datacenter Edition)	2.7 GB

Item	Limitation
Maximum cache size (AWE cache) (Windows 2000 Professional, Windows 2000 Server, Windows 2000 Advanced Server, Windows 2000 Datacenter Server, Windows XP Home Edition, Windows XP Professional, Windows Server 2003 Web Edition, Windows Server 2003 Standard Edition, Windows Server 2003 Enterprise Edition, Windows Server 2003 Datacenter Edition)	100% of all available memory - 128 MB
Maximum cache size (Windows CE)	Limited by available memory on the device
Maximum cache size (UNIX—Solaris, x86 Linux, AIX, HP)	2 GB for 32-bit servers
Maximum cache size (Win 64)	Limited by physical memory on 64-bit servers
Maximum cache size (UNIX—Tru64 UNIX, Itanium Linux, Itanium HP-UX)	Limited by physical memory on 64-bit servers
Maximum cache size (NetWare)	2 GB
Maximum index entry size	No limit
Number of databases per server	255
Number of columns per table	♦ ((page size)/4)^2 for 32-bit servers ♦ ((page size)/8)^2 for 64-bit servers Note: An excessive number of columns, although allowed, will affect performance.
Number of indexes per table	2^32
Number of rows per database	Limited by file size
Number of rows per table	Limited by file size
Number of tables per database	2^32 – 2^20 - 1 = 4 293 918 719
Number of temporary tables per connection	2^20 = 1 048 576

Item	Limitation
Number of tables referenced per transaction	No limit
Number of stored procedures per database	2^32 – 1 = 4 294 967 295
Number of events per database	2^31 – 1 = 2 147 483 647
Number of triggers per database	2^32 – 1 = 4 294 967 295
Row size	Limited by file size
Table size	Maximum file size. User-created indexes for the table can be stored separately from the table

Index