

Copyright

Sybase® SQL Anywhere™

SQL Anywhere Release: 5.5.02

Last modified:

Copyright © 1991-1997 by Sybase, Inc. and its subsidiaries. All rights reserved. Printed in the United States of America.

This publication pertains to Release 5.5.02 of the Sybase SQL Anywhere database management software and to any subsequent release until otherwise indicated in new editions or technical notes.

Information in this document may change without notice and does not represent a commitment on the part of Sybase, Inc. and its subsidiaries.

The software described herein is furnished under a license agreement, and it may be used only in accordance with the terms of that agreement.

No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Sybase, Inc.

Sybase, Inc. and its subsidiaries claim copyright in this program and documentation as an unpublished work, revisions of which were first licensed on the date indicated in the foregoing notice. Claim of copyright does not imply waiver of other rights of Sybase, Inc. and its subsidiaries.

PowerBuilder, Powersoft, S-Designer, SQL Smart and SYBASE are registered trademarks of Sybase, Inc. and its subsidiaries. InfoMaker, the Column Design, ComponentPack, Data Pipeline, DataWindow, NetImpact, NetImpact Dynamo, NetImpact Studio, ObjectCycle, PowerBuilder Foundation Class Library, PowerScript, PowerTips, Powersoft Optima++, Powersoft Portfolio, Powersoft Professional, SDP, StarDesigner, Sybase SQL Anywhere, Watcom, Watcom SQL, and Watcom SQL Server are trademarks of Sybase, Inc. and its subsidiaries. Certified PowerBuilder Developer and CPD are service marks of Sybase, Inc. and its subsidiaries. DataWindow is a proprietary technology of Sybase, Inc. (U.S. patent pending).

AccuFonts is a trademark of AccuWare Business Solutions Ltd.

All other company and product names used herein may be the trademarks or registered trademarks of their respective companies.

Network Guide

Copyright

-
- About This Manual
 - CHAPTER 1. Overview of SQL Anywhere
 - CHAPTER 2. Running the Database Server and Client
 - CHAPTER 3. The Database Server Display
 - CHAPTER 4. Network Communications
 - CHAPTER 5. SQL Anywhere Components

About This Manual

Subject

This manual describes how to use client applications with the Sybase SQL Anywhere database server over a network. It contains material on the components you need for network operation of SQL Anywhere, material on managing network communications, including troubleshooting and network performance tuning.

Audience

This manual is for administrators of SQL Anywhere database servers. It is a companion volume to the *SQL Anywhere User's Guide*.


CHAPTER 1. Overview of SQL Anywhere


About this chapter


This chapter presents an overview of SQL Anywhere architecture on a single computer and on a network.


An application developed in a standalone environment can be deployed in a multiuser network environment with no alteration to the code whatsoever.

This chapter also introduces terminology for relational database management systems, and describe the database management tools included in your SQL Anywhere package.

 The SQL Anywhere engine and the SQL Anywhere server

 Running SQL Anywhere on a single computer

 Running SQL Anywhere on a network

 Running mixed operating systems on a single computer



The SQL Anywhere engine and the SQL Anywhere server

SQL Anywhere includes two executables for managing databases:

- ◆ The network server, for managing databases using a client/server arrangement on networks
- ◆ The database engine, for managing databases on a single computer in a standalone mode

PowerBuilder and InfoMaker users—This chapter contains some information about connecting to a SQL Anywhere network server. The SQL Anywhere network server is not included with PowerBuilder and InfoMaker.

The SQL Anywhere server and SQL Anywhere engine manage databases in exactly the same way and are completely compatible. However, the SQL Anywhere engine has no support for network communications.

Server and engine—The terms **engine** and **server** are used in this manual to refer to the database engine and database server. In contexts where either the engine or the server could be used, the terms are used interchangeably. When the engine and server need to be distinguished, the terms **standalone engine** and **network server** are sometimes used to emphasize which is meant.

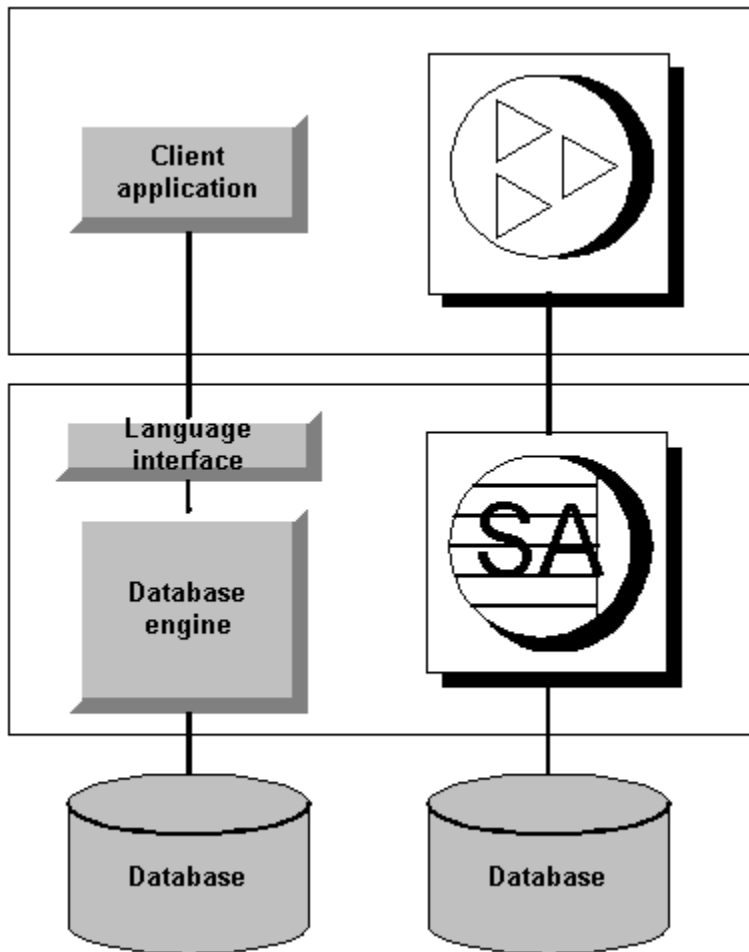
The following sections describes the architecture for running SQL Anywhere on a single computer using the SQL Anywhere engine and for running SQL Anywhere across a network using the SQL Anywhere network server.



For information on running more than one operating system on a single computer, see the section "[Running mixed operating systems on a single computer](#)".

Running SQL Anywhere on a single computer

The figure shows the architecture of a standalone SQL Anywhere installation, running a single database engine and working with a single database. All more complicated arrangements are elaborations of this basic setup, so you should understand how the basic setup works, even if you are operating a multiuser client/server installation.



Standalone SQL Anywhere components




The components of the basic standalone SQL Anywhere setup are:

- ◆ The client application
- ◆ The SQL Anywhere interface layer
- ◆ The SQL Anywhere database engine
- ◆ The database

Database users do not directly manipulate database files. Instead, their client application communicates with the **database engine**, using a programming interface supported by SQL Anywhere, and the database engine handles all manipulation of the actual database.

For information on a client application and a database server *for different operating systems* running on the same computer, see the section "[Running mixed operating systems on a single computer](#)".

-
- The client application
 - The SQL Anywhere database engine
 - The database

	Network Guide
	CHAPTER 1. Overview of SQL Anywhere
	Running SQL Anywhere on a single computer

The client application

A client application communicating with the SQL Anywhere database engine must do so using a **programming interface** supported by SQL Anywhere. The client application calls functions from one of the SQL Anywhere programming interfaces.

The client application together with the programming interface layer form the **client side** of the setup.

The SQL Anywhere database engine

The database engine and the database together form the **server side** of the setup. A client application manipulates a database by sending requests to the database engine.

SQL Statements

Communications between a client application and a database engine take the form of Structured Query Language (**SQL**) statements. For example, a SELECT statement, or query, is used to extract information from a database. An UPDATE statement may be used to modify the contents of one of the database tables.

The client application sends the SQL statements and the database engine processes them and sends the results back to the client application.

Running an application against a SQL Anywhere network server

Running an application against a SQL Anywhere network server generally requires an extra component to handle network communications from the client computer (see "[Running SQL Anywhere on a network](#)"). However, for applications on the same computer the database server can be run in exactly the same manner as the standalone database engine described here using a direct connection to the network server.

If a SQL Anywhere Client is running and the network server is running on the same machine, the Client will not be used by client applications connecting on the same machine; the connection will be direct.

■ For a discussion of running a client application and a database server for different operating systems on a single computer, see "[Running mixed operating systems on a single computer](#)".

The database

SQL Anywhere is a relational database system. The database itself is stored on one or more disk drives, and consists of the following objects:

<u>Database objects</u>	<u>Description</u>
Tables	Hold the information in the database
Keys	Relate the information in one table to that in another
Indexes	Allow quick access to information in the database
Views	Are computed tables
Stored procedures	Hold queries and commands that may be executed by any client application (stored procedures are not available in the SQL Anywhere Desktop Runtime system)
Triggers	Assist in maintaining the integrity of the information in the database (triggers are not available in the SQL Anywhere Desktop Runtime system)
System tables	Hold the information about the structure of the database

Multiple databases on a single database engine

A single SQL Anywhere database engine can manage access to several databases simultaneously. You can start and stop databases from database administration tools or client applications, and you may connect to any of the currently running databases on a database engine.

As far as the database user is concerned, interaction with a database engine is always through a **connection**. Each time users connect to a database, supplying valid user ID and password, they are connected to a specific database on a specific database engine. Once a connection is established, it provides a channel through which all communications go. The connection insulates the user from the other components of a running database system such as network sessions and interprocess communication mechanisms.

Multifile databases

SQL Anywhere supports multifile databases. When a SQL Anywhere database is first initialized, it is composed of one file—the **root file**. As tables and other database objects are added to the database, however, they may be stored in different files, which may be on different disk drives from the root file.

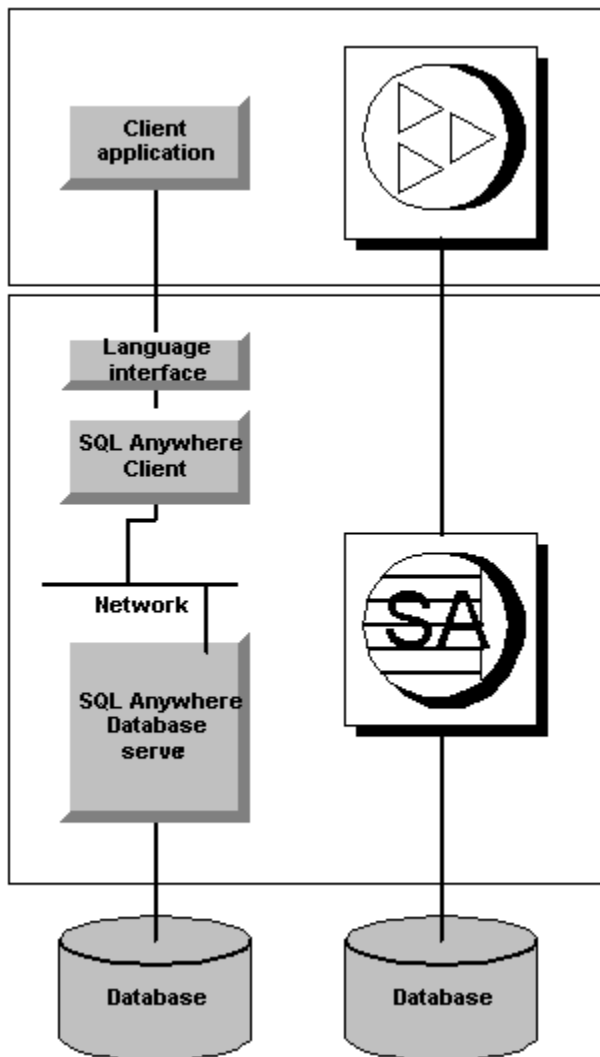
Users of the database (other than the database administrator) need not be aware of the physical location of database files. The database engine handles all access to the files and shields this complexity from the user.

Running SQL Anywhere on a network

SQL Anywhere Server products support connections from many users at a time, over a network. In this case, the database engine runs on one computer (the **database server computer**), while client applications run on other computers (**client computers**).

The SQL Anywhere server supports multiuser network access to SQL Anywhere. The SQL Anywhere standalone engine does not support multiuser access or network communications.

PowerBuilder and InfoMaker users—This chapter contains some information about connecting to a SQL Anywhere network server. The SQL Anywhere network server is not included with PowerBuilder and InfoMaker.



The client side of the SQL Anywhere setup sends SQL queries and commands over the network to the server side, which carries out commands and sends the results of queries back to the client.

The SQL Anywhere Client

The SQL Anywhere Client is a program that handles network communications to the SQL Anywhere

database server. The SQL Anywhere Client is a program named DBCLIENT (the Windows 3.x version is called DBCLIENW). The SQL Anywhere Client communicates with the SQL Anywhere network server. The standalone engine cannot handle communications from the SQL Anywhere Client.

For QNX, the database client is different from other operating systems. Instead of being a separate executable, the SQL Anywhere Client is a library named **dbclient**, which is loaded dynamically by client applications.




Standalone setup versus network setup

From the client application's point of view, there is no difference between the standalone setup and the network setup. In the single-user setup, a client application sends requests and commands to the database engine. In the multiuser setup, these requests are sent to the SQL Anywhere Client instead. In each case, the client application has a single point of contact with the DBMS. The additional complexity of handling requests in a multiuser, networked environment is hidden from the client application.

Standalone applications work with the SQL Anywhere network server—Once a client application is developed and working on a standalone SQL Anywhere setup on a single computer, no changes are needed to the application in order for it to work as a client application in a network environment against a SQL Anywhere server.

SQL Anywhere multiplatform support

Some database terms

	Network Guide
	CHAPTER 1. Overview of SQL Anywhere
	Running SQL Anywhere on a network

SQL Anywhere multiplatform support

The SQL Anywhere standalone database engine is available for the Windows 3.x, Windows 95 and Windows NT, OS/2, and DOS operating systems. The SQL Anywhere database server is available for Novell NetWare, Windows 95 and Windows NT, OS/2, Windows 3.x, DOS, and QNX operating systems.

One SQL Anywhere database server can support multiple clients operating on different operating systems, communicating via different network protocols.

Some database terms

With a single database running on a single database engine, the default settings make the engine name, the database name, and the database file the same, apart from the path and extension associated with the file. In this situation there is little ambiguity when talking about *the database*.

In environments with multiple databases, multiple database files, and several database engines operating simultaneously, it is important to distinguish among the different components that make up a running SQL Anywhere DBMS.

Terms used in this documentation

Term	Description
Database file	Even though the tables of a database may be held in several files on several disk drives, each database is identified by a single root file . Throughout this book, when reference is made to a database file , it is referring to the root file
Database name or alias	A SQL Anywhere database engine or server can run several databases simultaneously, managing access to each of them. When a database is started on a database engine, it is assigned a database name , also called the database alias . If no database name is explicitly assigned, the database receives the name of the root file with the path and extension removed
Server or engine name	When a database server or engine is started, it is assigned a server name or engine name . The server or engine name is entirely distinct from the name of the database engine program itself. By default, the server name is the first database name. For example, if a database engine is started with database C:\PB\EX\PSDEMODB.DB and no name is explicitly given, then the name of the engine is PSDEMODB
Clients and servers	Both the client side and the server side of a SQL Anywhere client/server setup consist of several components. The terms client and server themselves are commonly used to describe not only the computers on which each side of the setup sits, but also the programs that are communicating, and also the collection of software components on each of the computers. Throughout this guide, the terms <i>client</i> and <i>server</i> are qualified whenever possible to specify which of these meanings is used

Running mixed operating systems on a single computer

SQL Anywhere supports situations in which client applications and the database engine run under different operating systems on the same computer. This support requires a SQL Anywhere Client (DBCLIENT) for the client application operating system. You can think of the mixed operating system setup as a client/server network arrangement, with both client and server sides residing on the same computer.

The situations where this can occur are:

- ◆ DOS or Windows client applications with an OS/2 engine
- ◆ DOS or Windows 3.x client applications with a Windows 95 or Windows NT engine

Mixed operating systems can occur when running DOS or Windows 3.x client applications with a Windows 95 or Windows NT engine

DOS or Windows client applications on OS/2

DOS or Windows 3.x clients on Windows 95 or NT

DOS or Windows client applications on OS/2

The OS/2 standalone engine or network server can be accessed from a DOS or WIN-OS/2 client application on the same machine.

The client application communicates with the SQL Anywhere Client for the client application operating system. (The DOS and Windows 3.x SQL Anywhere Clients are installed with *SQL Anywhere for OS/2* or as a separate install with the DESKTOP RUNTIME SYSTEM FOR OS/2.) The SQL Anywhere Client handles the communication with the OS/2 database engine or server using named pipes or DDE.

Command lines for DOS or Windows 3.x clients

The command line to run the DOS client is:

dbclient *engine-name*

The command line to run the Windows 3.x client is:

dbclienw *engine-name*

Other servers

SQL Anywhere also supports access to network servers elsewhere on the network simultaneously via a separate communication link in the SQL Anywhere Client.

DOS or Windows 3.x clients on Windows 95 or NT

The Windows 95 or Windows NT database engine or server can be accessed from a DOS or Windows 3.x client application on the same machine.

The client application communicates with the SQL Anywhere Client for the client application operating system. (The DOS and Windows SQL Anywhere Clients are installed with SQL Anywhere.) The SQL Anywhere Client handles the communication with the Windows 95 or Windows NT database engine or server, using named pipes (Windows NT) or DDE (Windows 95).

Command lines for DOS or Windows 3.x SQL Anywhere clients

The command line to run the DOS SQL Anywhere Client is:


dbclient *engine-name*

The command line to run the Windows 3.x SQL Anywhere Client is:

dbclienw *engine-name*

Other servers








SQL Anywhere also supports access to network servers elsewhere on the network simultaneously via a separate communication link in the SQL Anywhere Client.

 For more information, see the SQL Anywhere Network Guide.

CHAPTER 2. Running the Database Server and Client

About this chapter

This chapter describes how to start and stop the database server and Client, and describes the options open to you under different operating systems.

-
-  Choices for running the server
 -  Running the database server as an NT service
 -  Running the NetWare database server
 -  Ensuring network communication software is running
 -  Running the database server executable
 -  Running the SQL Anywhere Client
 -  Running client and server on the same computer

Choices for running the server

All of the topics discussed in the SQL Anywhere User's Guide apply to both SQL Anywhere standalone engine and the SQL Anywhere network server. A database application cannot tell the difference between the standalone engine running on the same machine, and the network server running on a different node on your network.

How you run the database server depends on the operating system you are using:

- ◆ The NetWare database server runs as a NetWare Loadable Module.
 - For more information, see "[Running the NetWare database server](#)".
- ◆ The Windows NT database server can be run as an NT Service.
 - For information about running the database server in this manner, see the SQL Anywhere User's Guide. The advantages of running the database server as an NT service are described in "[Running the database server as an NT service](#)".
- ◆ The OS/2, Windows 3.x, Windows 95, Windows NT, DOS, and QNX database servers run as an executable program, although you may prefer to run the Windows NT database server as an NT service.
 - For information about running the database server executable, see "[Running the database server executable](#)".

Running the database server as an NT service

Many users will find that running the database server as a service is the most convenient and appropriate way of running the Windows NT database server.

For detailed information about running the server as an NT service, see the *SQL Anywhere User's Guide*. The relevant sections are also accessible as online Help from the SQL Anywhere Service Manager.

Understanding NT services

Although the database server can be run like any other Windows NT program rather than as a service, there are limitations to running it as a standard program.

For a full description of NT services and how to use them, see your Windows NT documentation.

Limitations of running as a standard executable

When you start a program, it runs under your NT login session: if you log off the computer, the program terminates. As only one person can be logged on to Windows NT (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 in to the computer running the database server in order for the database server to keep running. This also presents a security risk as the NT computer must be left in a logged on state.

Advantages of services

Installing an application as an **NT service** enables it to run even when you log off. The database server can run as an NT service.

When you start a service, it logs on using a special system account called LocalSystem (or using another account you specify). The service is not tied to the user ID of the person starting it, and therefore is not stopped when that person logs off. A service can also be configured to start automatically when the NT computer is started, before a user logs on.

Managing services


Although you can control most aspects of services from the Services option in the Windows NT Control Panel, SQL Anywhere includes a SQL Anywhere Service Manager which provides a more convenient and comprehensive way of managing SQL Anywhere services.




Full documentation of the SQL Anywhere Service Manager is provided in the *SQL Anywhere User's Guide*, accessible as online Help from the SQL Anywhere Service Manager interface.

Running the NetWare database server

The database server for NetWare is a NetWare Loadable Module (DBSRV50.NLM). An NLM is a program that you can run on your NetWare server.

 Starting the database server for NetWare

 Stopping the database server for NetWare

 Running under NetWare 3.11 and 3.12

Starting the database server for NetWare

To start the database server from the NetWare console, type:

```
load dbsrv50 database
```

The *database* specifies the directory and filename of the database file. The database file must be on a NetWare volume (a typical filename is of the form DB:\DATABASE\SALES.DB).

The database server has a command line switch to specify the amount of memory to use for caching. For example,

```
load dbsrv50 -c 20M database
```

will use 20 megabytes for caching. By default, DBSRV50 uses 2 megabytes for caching.

For a complete description of the database server command line, see "[SQL Anywhere Components](#)".

You can load DBSRV50 from a client machine by using the Novell remote console utility. See your Novell documentation for details.

You can put the LOAD DBSRV50 line into your Novell AUTOEXEC.NCF file so that DBSRV50 is automatically loaded each time you start the NetWare server.

Stopping the database server for NetWare


You can stop the database server by choosing **Exit** from the **File** menu on the server window. You can also stop the database server for NetWare by typing an UNLOAD command from the NetWare console:


```
unload dbsrv50
```

If there are still active connections from client machines, you will be prompted whether you really want to shut down the database server. When the database server is stopped, the NLM is unloaded and the memory that it used is released.

The database server is unloaded automatically if you DOWN the NetWare server while the database server is running.

 Network Guide

 CHAPTER 2. Running the Database Server and Client

 Running the NetWare database server

Running under NetWare 3.11 and 3.12

SQL Anywhere uses the Direct File System to maintain database files. The Direct File System is built into NetWare 4.0 and 4.1, but not in NetWare 3.11 or 3.12. Novell has provided a loadable module containing the DIRECTFS functions for version 3.11 and 3.12. DIRECTFS.NLM comes with SQL Anywhere and is installed during the installation if it is not found on your NetWare server. The DIRECTFS module is automatically loaded when you load DBSRV50.

Novell has also provided an updated CLIB.NLM for Netware 3.11. This update contains bug fixes that are necessary for SQL Anywhere to work properly. It is installed during the installation of SQL Anywhere if it is not already on your NetWare server.

Ensuring network communication software is running

Appropriate network communication software must be installed and running before you run the database server or client. 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 the chapter "[Network Communications](#)".

If you are running under DOS, your network communications software must be installed and running as a TSR. You should be careful about what other TSR's you run on your system. Some may affect the performance or reliability of the database server. It is better to remove TSR's if they are not required to run your network drivers.


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


For example, if you are using NetBIOS under Windows 95, Windows for Workgroups, or Windows NT you may want to confirm that the CHAT or WINPOPUP application is working properly between machines running client and database server software.


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.

Running the database server executable

There are two steps to starting the database server executable. First, you should ensure that the appropriate network software is running properly, and then you can start the executable.

 For information on networking software, see

 Starting the database server executable

 Stopping the database server executable

Starting the database server executable

The way in which you start the database server depends on the operating system you are using. The command lines in this section are for the simple case of running a single database with default settings.

For a full description of the command line switches available, see "[The database server](#)".

Starting the server for Windows 3.x

You can use a Program Manager icon to hold a command line, or enter the following command line in the Run dialog box:

```
dbsrv50w database
```

You can run only one Windows 3.x server on a given computer at one time.

Starting the server for OS/2

You can use an icon on the desktop to hold a command line, or enter the following command at an OS/2 command line prompt:

```
dbsrv50 database
```

The database server is not a Presentation Manager application. It runs in a full screen or windowed OS/2 session. To start the server in a separate session, use the OS/2 START command:

```
start dbsrv50 database
```

You can put the DBSRV50 line into your STARTUP.CMD or into your startup folder so that DBSRV50 is automatically loaded each time you start OS/2.

Starting the server for Windows NT

Many users will find that running database server as an NT service is more convenient, but you can also run the server as a program from your login user ID. You can use a Program Manager icon to hold a command line, or enter the following command at the command prompt:

```
dbsrv50 database
```

To start the server in a separate session, use the START command:

```
start dbsrv50 database
```

Starting the server for Windows 95

You can use a Program Manager icon to hold a command line, or enter the following command at the command prompt:

```
dbsrv50 database
```

Starting the server for DOS

Enter the following command line at the DOS prompt:

```
dbsrv50 database
```

The *database* specifies the database file. The root of the database file (that is, the name with the path and extension removed) becomes the server name.

Starting the server for QNX

The database server is a single program that can be started with the following command:

```
dbsrv50 database
```

The *database* specifies the database filename.

Setting the cache size

The size of the database cache is one of the most important factors in determining server performance. In general, the larger memory cache available to the server, the better its performance. Under all operating systems, the database server has a command-line switch to specify the amount of memory that it uses for caching. For example, under Windows NT,

```
dbsrv50 -c 20M database
```

uses 20 megabytes for caching. By default, the database server uses 2 megabytes for caching.

■ For more information about the database server command line see "[The database server](#)".

Stopping the database server executable

You can stop the database server by choosing Exit from the File menu on the database server display, or by using the DBSTOP command-line utility.

For information on DBSTOP command-line switches, see the chapter "SQL Anywhere Components" in the *SQL Anywhere User's Guide*.

On QNX, dbstop allows you to stop a database server running on any node of your network. You should not use the *slay* or *kill* commands to stop the QNX database server.

Stopping the OS/2 server—Under OS/2, if you close the OS/2 session where the database server is running without stopping the database server, the database will not shut down cleanly (see the chapter "Backup and Recovery" in the *SQL Anywhere User's Guide*). Recovery will happen automatically the next time you start the database server. It is better to stop the database server before closing the OS/2 session.


Running the SQL Anywhere Client

For client platforms other than QNX, the SQL Anywhere Client is a separate executable (DBCLIENT.EXE or DBCLIENW.EXE), which communicates with the database server using a supported protocol stack.

The QNX SQL Anywhere Client is a separate file (dbclient) that is loaded automatically by client applications. There is no separate executable to run for QNX clients: this section provides instructions for running the SQL Anywhere Client on client platforms other than QNX.


A database server must be running before you start the SQL Anywhere Client on the client machine. Any of the client programs can communicate with a running SQL Anywhere database server on any operating system, as long as appropriate network communications software is running and the protocol is supported.


When you start a SQL Anywhere Client, it connects to a running database server. Once a SQL Anywhere Client is running, several applications can connect to the same or a different database server using a single SQL Anywhere Client.


 For information on connecting to more than one database server, see "[Connecting to more than one database server](#)".

You should ensure that networking software is properly installed and running before you start the SQL Anywhere Client.

 For more information, see "[Ensuring network communication software is running](#)".

 [Starting the SQL Anywhere Client](#)

 [Stopping the SQL Anywhere Client](#)

 [Connecting to more than one database server](#)

Starting the SQL Anywhere Client

This section gives a brief description of how to start the SQL Anywhere Client for each supported operating system. Several applications can use the same Client at the same time communicating with the same or a different database server: you only need to run one Client once.

For a full description of DBCLIENT command-line options, see the chapter "[SQL Anywhere Components](#)".

Starting the SQL Anywhere Client in DOS

The SQL Anywhere Client can be started with the following command:

```
dbclient name
```

The *name* is the name of a database server that is currently running. A status will be reported within a few seconds indicating whether the Client was able to find the named database server on the network. Once the Client is started, any SQL Anywhere application can be run on the client machine and can communicate with a database server.

Starting the SQL Anywhere Client in Windows 3.x

The SQL Anywhere Client can be started with the following command:

```
dbclienw name
```

You can use a Program Manager icon to do this or type the command line in the Run dialog. The *name* is the name of a database server that is currently running. A status will be reported within a few seconds indicating whether the Client was able to find the named database server on the network. Once the Client is started, any SQL Anywhere application can be run on the client machine and can communicate with a database server.

Starting the SQL Anywhere Client in OS/2

You can use an icon on the desktop to hold a command line, or type the following command at an OS/2 command prompt:

```
dbclient name
```

The SQL Anywhere Client is not a Presentation Manager application. It runs in a full-screen or windowed OS/2 session. To start the SQL Anywhere Client in a separate OS/2 session, type:

```
start /c dbclient name
```

The *name* is the name of a database server that is currently running. A status will be reported within a few seconds indicating whether the Client was able to find the named database server on the network. Once the Client is started, any SQL Anywhere application can be run on the client machine and can communicate with a database server.

Starting the SQL Anywhere Client in Windows NT

The SQL Anywhere Client can be started with the following command:

```
dbclient name
```

You can use a Program Manager icon to do this or type the command at a Windows NT command prompt.

To start the SQL Anywhere Client in a separate Windows NT window, type:

```
start dbclient name
```

The *name* is the name of a database server that is currently running. A status will be reported within a few seconds indicating whether the Client was able to find the named database server on the network. Once the Client is started, any SQL Anywhere application can be run on the client machine and can

communicate with a database server.

Several applications can use the same Client at the same time communicating with the same or a different database server. You only need to start the Client once.

Starting the SQL Anywhere Client in Windows 95

The SQL Anywhere Client can be started with the following command:

```
dbclient name
```

You can use a Program Manager icon to do this or type the command at a command prompt. The *name* is the name of a database server that is currently running. A status will be reported within a few seconds indicating whether the Client was able to find the named database server on the network. Once the Client is started, any SQL Anywhere application can be run on the client machine and can communicate with a database server.

Starting the SQL Anywhere Client in QNX

The QNX SQL Anywhere Client (dbclient) is automatically loaded by the database interface library that is linked with all applications. You do not need to start the client separately.

The client is not a separate task. For better performance, it is loaded as part of the application. The communication software is maintained as an external file so that developers do not need to relink their applications with each new release of SQL Anywhere. Each new release of SQL Anywhere includes new dbclient files. An end-user can install the new database software and run existing applications without relinking.

Stopping the SQL Anywhere Client

The QNX SQL Anywhere Client stops automatically when the application program terminates.

In windowing operating environments you can stop the Client by closing its window.

You can also stop the SQL Anywhere Client using the DBSTOP utility. This section gives a brief description of how to stop SQL Anywhere Clients using DBSTOP.

For a full description of DBSTOP and its command-line switches, see the chapter "SQL Anywhere Components" in the *SQL Anywhere User's Guide*.

To stop a SQL Anywhere Client using DBSTOP, use the following command:

```
dbstop name
```

In Windows 3.x, the command is DBSTOPW *name*.

The *name* is the default database server as specified in the command line of a running SQL Anywhere Client.

Using DBSTOP on an operating system other than QNX stops the client software only, and does not stop the database server.

Connecting to more than one database server

You may have more than one database server running on your network.

You may also have one or more single user SQL Anywhere database engines running on the client computer.

A single SQL Anywhere Client can communicate with all standalone database engines on the same machine and all database servers on the network: there is no need to start more than one SQL Anywhere Client.

The default database server

When starting the SQL Anywhere Client (DBCLIENT) is started, a database server name can be provided on the command line. The named server is the default database server for all connections through that client which do not explicitly identify a database server name.

To connect to a database server other than the default, you must specify the database server to which you wish to connect. This is specified in the ODBC data source definition, if you are connecting through ODBC, or in the EngineName connection parameter.

■ For more information see the chapter "Connecting to a Database" in the *SQL Anywhere User's Guide*.

Running client and server on the same computer

You can run applications on the same computer as the database server. If the client application is native to the same operating system as the server, no SQL Anywhere Client is required to connect. Instead, the client application connects directly to the database server.

If the client application is not native to the same operating system as the server, there are cases where you can still connect to the server using the SQL Anywhere Client. These cases include the following:

- ◆ DOS application against a Windows 3.x.
- ◆ DOS or Windows 3.x application against a Windows 95 or Windows NT server.
- ◆ DOS or Windows 3.x application against an OS/2 server.


Applications can connect to either a network server or standalone engine on the same computer using the SQL Anywhere Client in these situations.


■ Running DOS client applications on Windows 3.x

■ Running DOS and Windows 3.x client applications on OS/2

■ Running DOS and Windows 3.x applications on Windows 95 or NT

 Network Guide

 CHAPTER 2. Running the Database Server and Client

 Running client and server on the same computer

Running DOS client applications on Windows 3.x

If you want to run DOS applications in a Microsoft Windows 3.x DOS session, you must start the DOS SQL Anywhere Client (DBCLIENT) in that DOS session. DOS and Windows 3.x applications on the same machine can communicate with the database server at the same time.

Running DOS and Windows 3.x client applications on OS/2

If you want to run Windows 3.x applications in a WIN-OS/2 session, you must start the Windows 3.x SQL Anywhere Client (DBCLIENW).

If you want to run DOS applications in an OS/2 DOS session, you must start the DOS SQL Anywhere Client (DBCLIENT) in that DOS session.

DOS, Windows 3.x, and OS/2 applications on the same machine can all communicate with the database server at the same time. The DOS and Windows 3.x clients can communicate with both network server and standalone versions of the SQL Anywhere database engine on OS/2.

Running DOS and Windows 3.x applications on Windows 95 or NT







If you want to run Windows 3.x applications on Windows 95 or NT, you must start the Windows 3.x SQL Anywhere Client (DBCLIENW). If you want to run DOS applications on Windows 95 or NT, you must start the DOS SQL Anywhere Client (DBCLIENT) in that command line session.

You can run DOS and Windows 3.x clients in addition to the Windows 95 or NT SQL Anywhere Client (DBCLIENT), so that 16 and 32 bit applications can use the database server at the same time. The DOS and Windows 3.x clients can communicate with both network server and standalone versions of the SQL Anywhere database engine on Windows 95 and NT.

CHAPTER 3. The Database Server Display

About this chapter

The database server has a display that allows you to monitor its activity. This chapter describes the display and the network statistics you can monitor.

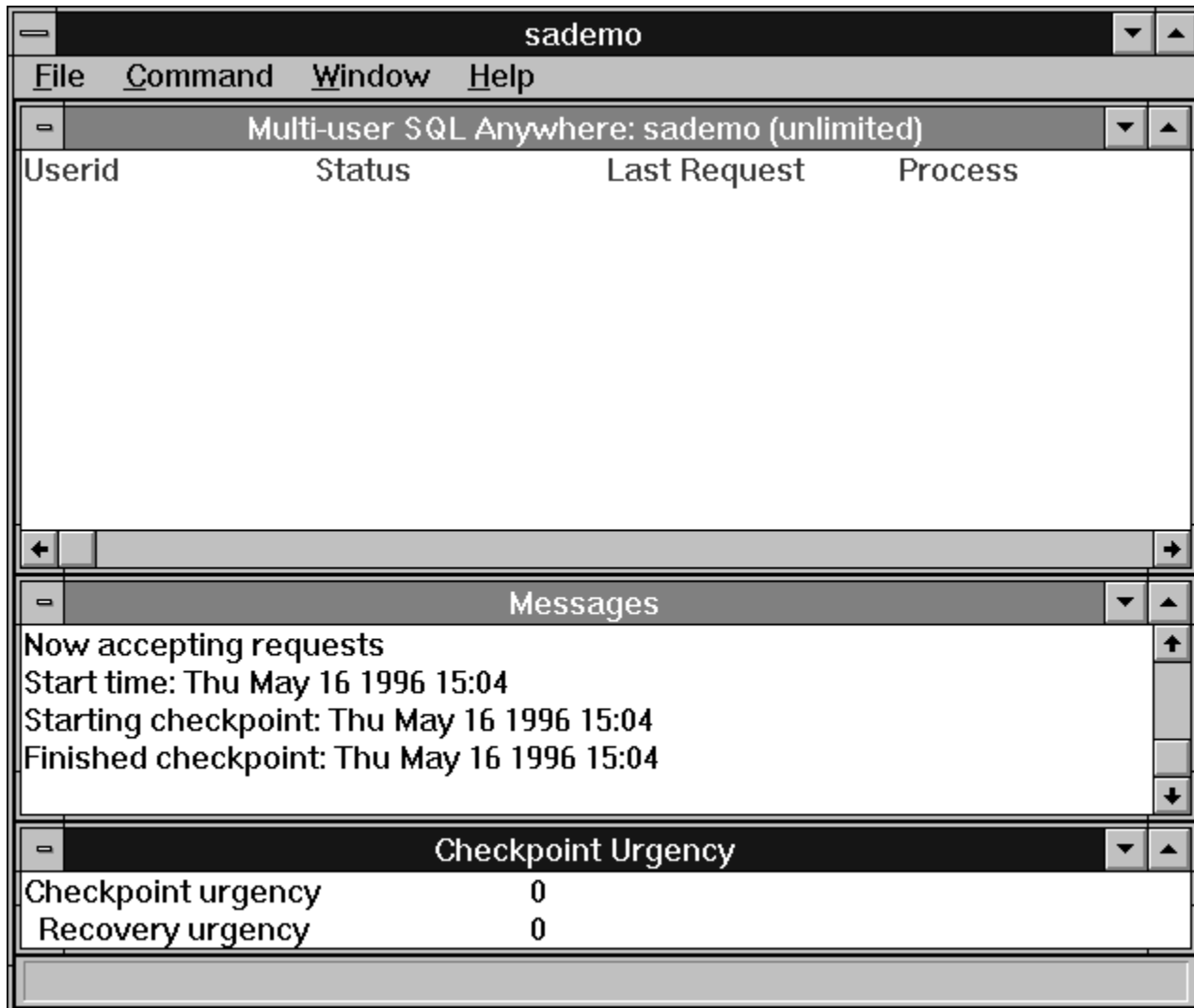
-
-  The server windows
 -  The Connection window
 -  The Messages window
 -  The Checkpoint Urgency window
 -  Network statistics
 -  Locking the keyboard

The server windows

The server screen has three windows:

- ◆ The Connection window
- ◆ The Messages window
- ◆ The Checkpoint Urgency window

Each of these is discussed in the following sections.





The Connection window

The connection window shows a list of all current client connections to the database server. You can use the mouse or the keyboard to scroll through the list or position on one connection. Double-clicking on a connection displays a detail dialog box, and right-clicking displays a short menu.

You cannot use the mouse on the database server for NetWare display.

Disconnecting a connection

To disconnect a database server connection from the database server display or from the remote monitoring facility:

- ◆ Position on a connection and press `ENTER` or double-click the mouse (not available on the database server for NetWare display) to display a detail dialog showing more detail about a connection.
- ◆ From within the detail dialog, you can terminate the connection.

You can also disconnect a user by positioning on a connection and right-clicking, then selecting Disconnect from the popup menu (not available on the database server for NetWare display).

You can also disconnect a user by positioning on a connection and right-clicking, then selecting Disconnect from the popup menu.

The Messages window

The database server uses the Messages window to display information messages. It allows you to scroll back through messages no longer on the screen. The contents of this window can be logged to a file using the -o switch on the database server.

The Checkpoint Urgency window

A SQL Anywhere database file is composed of pages. Before a page is updated (made **dirty**), a copy of the original is always made. The copied pages are the checkpoint log.

Dirty pages are not written immediately to the disk. For improved performance, they are cached in memory and written to disk when the cache is full or the server has no pending requests. A **checkpoint** is a point at which all dirty pages are written to disk. Once all dirty pages are written to disk, the checkpoint log is deleted.

Priority

The priority of writing dirty pages to the disk increases as the time and the amount of work since the last checkpoint grows. This is important when the database engine does not have enough idle time to write dirty pages. The database option CHECKPOINT_TIME controls the maximum desired time between checkpoints. The database option RECOVERY_TIME controls the maximum desired time for recovery in the event of system failure. Both times are specified in minutes.

When the database server is running with multiple databases, the CHECKPOINT_TIME and RECOVERY_TIME specified by the first database started is used, unless overridden by command-line switches.

■ For a description of the command-line switches, see "[The database server](#)".

The Checkpoint Urgency window shows the checkpoint urgency, which is the percentage of CHECKPOINT_TIME that has passed since the last checkpoint.

The Checkpoint Urgency window also shows the recovery urgency, which is the estimated time for recovery from a system failure as a percentage of the RECOVERY_TIME option. The recovery urgency will increase until a checkpoint is done.


If the percentages exceed 90% on a regular basis, your database server is near capacity. A faster hard disk, more memory for caching, or a faster computer could improve the performance of your database server.


■ For a full description of the recovery mechanism in SQL Anywhere, see the chapter "Backup and Recovery" in the *SQL Anywhere User's Guide*.

Network statistics

You can open the database server statistics window by pressing F2 while running the database server or the remote monitoring facility (DBWATCH). You can open the client statistics window by pressing F3 while running the remote monitoring facility. These statistics are useful after the SQL Anywhere Client and the database server have been running for a while under normal program load.

SQL Anywhere works with communication protocols that do not have guaranteed transmission: NetBIOS datagrams, IPX, and TCP/IP. It has a guaranteed transmission mechanism built in to the application protocol used by the database server and the client. Several of the statistics collected at the server and client do not apply to the guaranteed transmission protocols (NetBIOS sessions, Named Pipes, and QNX messages).

 Server statistics

 Client statistics

Server statistics

The following table lists the server statistics that can be monitored.

Statistic	Description
Bytes received	Total size of all packets received from clients.
Bytes sent	Total size of all packets sent to clients.
Free buffers	Number of buffers available for incoming packets.
Liveness packets	Number of duplicate request packets that have been received from clients.
Multi packets sent	Number of extra packets sent because a response could not fit within one packet.
Multi packets received	Number of extra packets received because a request could not fit within one packet.
Packets chopped	Number of packets received that were shorter than the length contained in the packet. These packets are discarded and the server will wait for a duplicate request packet.
Packets corrupted	Number of packets that did not checksum correctly. These packets are discarded and the server will wait for a duplicate request packet. The number of corrupted packets should be a small percentage of the number of packets sent (less than 1%). If this number is high, your wiring may be suspect.
Packets dropped	This happens when the server does not have an available buffer to store an incoming packet. These packets are discarded and the server will wait for a duplicate request packet. The server will allocate more memory for packet buffers.
Packets received	Number of packets received from clients.
Packets resent	Number of duplicate response packets that were sent to clients.
Packets sent	Number of packets sent to clients.
Remoteput_wait	This count is incremented when the communication link does not have buffers available to send information. It is collected for NetBIOS (both sessions and datagrams) and IPX protocols.
Send_failed	The number of packets incorrectly sent by the server.
Total buffers	Total number of buffers (used and

available).

■	Network Guide
■	CHAPTER 3. The Database Server Display
■	Network statistics

Client statistics

The following table lists the client statistics that can be monitored.

Statistic	Description
Bytes received	Total size of all packets received from the server(s).
Bytes sent	Total size of all packets sent to the server(s).
Free buffers	Number of buffers available for incoming packets.
Liveness packets	Number of duplicate response packets that have been received from the server(s).
Multi-packets sent	Number of extra packets sent because a request could not fit within one packet.
Multi-packets received	Number of extra packets received because a response could not fit within one packet.
Packets chopped	Number of packets received that were shorter than the length contained in the packet. These packets are discarded and the client will wait for a duplicate response packet.
Packets dropped	This happens when the client does not have an available buffer to store an incoming packet. These packets are discarded and the client will wait for a duplicate response packet.
Packets received	Number of packets received from the server(s).
Packets resent	Number of duplicate request packets that were sent to the server(s).
Packets sent	Number of packets sent to the server(s).
Packets corrupted	Number of packets that did not checksum correctly. These packets are discarded and the client will wait for a duplicate response packet.
Remoteput_wait	This count is incremented when the communication link does not have buffers available to send information. It is collected only for NetBIOS (sessions and datagrams) and IPX protocols.
Send_failed	The number of packets incorrectly sent by the client.
Total buffers	Total number of buffers (used and available).

Locking the keyboard

To improve the security of your database server, the database server allows you to disable keyboard input and most menu items by providing a password. This can be done with a command-line switch or through the menus. To enable the keyboard and the menus, a user must type the password. Only the server has input disabled. Other programs are not affected.












For QNX, dbstop requires the use of this password, using the -p command-line switch.

CHAPTER 4. Network Communications

About this chapter

SQL Anywhere has been tested in many network environments. The number of combinations of network adapter boards, wiring types, and network software drivers makes testing under all configurations impossible. Each network environment has its own peculiarities: this chapter describes those aspects of network communication relevant to the proper functioning of SQL Anywhere, and provides some tips for diagnosing network communication problems.









If you are running a single network, and your network communications software is running reliably, you should not need to read this chapter. If you are running several network protocols, or if you are having problems with communications between your the SQL Anywhere Client and the database server, you may need to ensure that your network communication software is configured properly. This chapter provides a description of how networks operate, and hints on running SQL Anywhere under each of the different protocols it supports.

-
-  [Network communication layers](#)
 -  [Real world protocol stacks](#)
 -  [Protocols supported by SQL Anywhere](#)
 -  [Using SQL Anywhere with IPX](#)
 -  [Using SQL Anywhere with NetBIOS](#)
 -  [Using SQL Anywhere with TCP/IP](#)
 -  [Using SQL Anywhere with Named Pipes](#)
 -  [Using SQL Anywhere with QNX messages](#)
 -  [Troubleshooting](#)
 -  [Configuring your network adapter board](#)
 -  [Evaluating network performance](#)

Network communication layers

Applications in a local area network communicate using a set of rules and conventions called an application protocol. Each application is isolated from the lower-level details of how information gets transported across the network by lower-level protocols, which form a **protocol stack**.

This section provides a brief description of how protocol stacks work. Hints regarding specific network issues later in this chapter assume a basic understanding of how protocol stacks operate.

-
-  [The protocol stack](#)
 -  [How information is passed across a network](#)
 -  [The physical layer](#)
 -  [The data link layer](#)
 -  [The network layer](#)
 -  [The transport layer](#)
 -  [The application layer](#)
 -  [Compatible protocol stacks](#)

■	<u>Network Guide</u>
■	<u>CHAPTER 4. Network Communications</u>
■	<u>Network communication layers</u>

The protocol stack

■ The figure shows the layers of a protocol stack, based on a simplification of the OSI Reference Model of network communications.

The OSI (Open Systems Interconnection) Reference Model, developed by the International Standards Organization (ISO), is a step towards international standardization of network protocols. Most current networking software and hardware conforms to some extent, but not exactly, to this model. Even though conformance is incomplete, the OSI model is a valuable aid in understanding how network communications work.

How information is passed across a network

When one network application (such as the SQL Anywhere Client) sends information to another network application (such as the database server) the information passes down one protocol stack, across the network, and up the other protocol stack to the other application.

Protocol stacks have layers

The protocol stack isolates the different functions needed for reliable data transfer. Each layer of the protocol stack is connected to the layer above it and that below it by an **interface**.

Each layer of a protocol stack treats information passed to it by the layer above it merely as data, labeling that data in such a way as to be identified and deciphered by the equivalent layer on the other computer. Only the physical layer is responsible for actually placing the data passed to it by the data link layer onto the wire—all other layers provide some well-defined level of functionality, such as error detection, correction, encryption and so on.




Each layer has a protocol

Although actual data transmission is vertical (down one stack, up the other), each layer is programmed as if it were in direct communication with the corresponding layer on the other stack (peer-to-peer communication). The rules and conventions that govern each level of peer-to-peer communication are called a **protocol** for that level. There exist transport protocols, network protocols, data link protocols, and application level protocols, among others.

Protocol stack compatibility

The protocol stacks on each side of the communication must be compatible at each level for network communications to work properly. If they are not compatible, the layer on the receiving stack does not understand the information being passed to it by its corresponding layer on the sending stack.

Software that manages lower levels in a protocol stack is often called a **driver**. The different layers of a protocol stack have the following functions:

	<u>Network Guide</u>
	<u>CHAPTER 4. Network Communications</u>
	<u>Network communication layers</u>

The physical layer

Your **network adapter**, or **network card**, and the network wiring form the physical layer for network communication. The higher layers in the protocol stacks ensure that software does not have to be aware of the details of network wiring or network adapter design.

The data link layer

The job of a data link layer is to handle the safe passing of information across the network at the level of individual sets of bits. At higher levels of the protocol stack, data is not described in terms of individual bits: it is at the data link layer that information is broken down to this elementary level.

Specification The data link layer's interface with the network layer above it in the protocol stack commonly conforms to one of two specifications: the Network Driver Interface Specification (**NDIS**) developed jointly by IBM and Microsoft, and the Open Device Interface (**ODI**) developed by Novell.

ODI and NDIS data link layers can be made compatible with each other using a **translation driver**.

■ For more information, see "[Working with multiple protocol stacks](#)".

The network layer

The network layer takes a **packet** of information from the transport layer above it and gets it to the corresponding network layer on the receiving protocol stack. Issues of routing are handled by the network layer.

Information at a higher level is broken down into packets of a specified size (in numbers of bytes) for transmission across a network.

Examples of network layer protocols

Internet Protocol (IP) and Novell's IPX are widely-used network layer protocols. SQL Anywhere has interfaces to both network layer protocols and transport layer protocols. SQL Anywhere has an interface directly to the IPX network protocol.

The transport layer

The principal task of the transport layer is to guarantee the transmission of information between applications. It accepts information directly from a network application, such as a database server, splits it up if necessary, passes the packets to the network layer and ensures that the pieces all arrive correctly at the other end, where it assembles the packets before passing them up the stack to the application layer.

Examples of transport protocols

Novell's SPX, Microsoft and IBM's NetBEUI, and Named Pipes are widely-used transport protocols. The TCP/IP suite of protocols includes more than one transport layer. NetBIOS is an interface specification to the transport layer from IBM and Microsoft that is commonly (but not necessarily) paired with the NetBEUI protocol.

SQL Anywhere supports both the NetBIOS interface to the transport layer and a direct interface to the UDP transport protocol for TCP/IP stacks. In addition, SQL Anywhere has an interface to Named Pipes for same computer communications only.

The TCP/IP protocol suite has more than one transport layer protocol. SQL Anywhere employs the User Datagram Protocol (UDP) which, while it is referred to here and elsewhere as a transport protocol, does not provide guaranteed transmission.

SQL Anywhere applies its own checks to the data passed between client application and database engine, to further ensure the integrity of data transfer.

The application layer

Database servers and client applications are typical application layers in a protocol stack, from a networking point of view. They communicate using an application-defined protocol. This protocol is internal to SQL Anywhere programs.

Typical data for transmission includes a SQL query or connect statement (from client application to database server) or the results of a SQL query (from database server to client application).

Passing information down the protocol stack

The SQL Anywhere Client and database server have an interface at the transport level (in the case of NetBIOS, UDP (TCP/IP), or Named Pipes), or at the network level (in the case of IPX), passing the information to the network communications software. The lower levels of the protocol stack are then responsible, independent of SQL Anywhere, for transmitting the data to the equivalent layer on the other computer. The receiving layer hands the information to SQL Anywhere on the receiving machine.

The database server and the SQL Anywhere Client perform a set of checks and functions to ensure that data passed across the network arrives correctly, in a proper form.


Compatible protocol stacks


For two protocol stacks to be compatible, they must be operating the same transport layer (say UDP) on each stack, and the same network protocol on each stack (say IP). If one stack employs a NetBIOS interface to the transport layer, so must the other. A SQL Anywhere Client running on a protocol stack employing NetBIOS cannot communicate with a database server using a TCP/IP protocol stack.

At the data link layer, ODI-based protocol stacks can be made compatible with NDIS-based protocol stacks using translation drivers, as discussed in "[Working with multiple protocol stacks](#)".

Real world protocol stacks

The OSI model is close enough to current networking software and hardware to be a useful model for thinking about networks. There are inevitable complications because of the different ways in which software companies have designed their own systems. Also, there are complications when an individual computer is running more than one protocol stack, as is increasingly common.

 Common protocol stacks

 Working with multiple protocol stacks

Common protocol stacks

Widely-used network software packages implement their own protocol stacks. Here are some networking software vendors together with their most common protocol stacks.

- ◆ **Novell** Novell NetWare typically operates using an SPX transport level atop an IPX network protocol (often written as SPX/IPX). Novell's IPX requires an ODI data link layer. The NetWare Client installation installs this protocol stack on Novell NetWare client computers. SQL Anywhere has an interface directly to the IPX protocol, and does not rely on the services of the SPX layer.

NetWare, IPX, and ODI are often used interchangeably when discussing network protocols.

- ◆ **Microsoft** Microsoft NT 3.5 comes with networking software for NetBIOS, IPX, and TCP/IP. Windows 95 installs IPX as default networking software, and also comes with NetBEUI software. Windows for Workgroups typically uses NetBIOS on top of NetBEUI as the transport level protocol. The NDIS data link layer protocol was jointly developed by Microsoft and IBM, and is employed by all Microsoft's higher-level protocols.
- ◆ **IBM** IBM LAN Server and other IBM networking software use a NetBIOS interface to a NetBEUI transport layer on top of an NDIS data link layer. The NDIS data link layer protocol was jointly developed by Microsoft and IBM, and is employed by all IBM's higher-level protocols.

In each case the installed data link layer may be changed from ODI to NDIS (or vice versa) by other networking software if more than one network protocol is installed. In these cases a translation driver ensures that both ODI and NDIS data link layer interfaces are available to the higher levels of the protocol stack. Working with more than one protocol stack is discussed in more detail in the next section.

Working with multiple protocol stacks

For two network applications (such as a SQL Anywhere client and a database server) to communicate, the client computer and the server computer must have compatible protocol stacks at each level. When each computer has a single network protocol installed, it is fairly straightforward to ensure that each is running the same stack. However, when each computer is running multiple protocols, the situation becomes more complex.

At the transport and network layers, there is no problem with more than one protocol running at once; as long as there is a compatible path through the protocol stacks, the two applications can communicate.

ODI and NDIS interfaces

There are two widely-used interfaces between the data link layer and the network layer above it: ODI and NDIS. As the data link layer communicates directly with the network adapter, only one data link driver can be installed at a time. However, as most data link layers do not modify the information placed on the network wire, an ODI data link driver on one computer is compatible with an NDIS driver on another computer.

Each network layer is written to work with one, and only one, of ODI or NDIS. However, translation drivers are available that enable network level protocols that require an NDIS (or ODI) data link interface to work with ODI (or NDIS) data link layers, and consequently a protocol stack built on an ODI data link layer can be compatible with a protocol stack based on an NDIS data link layer as long as the upper layers are compatible.

ODI and NDIS translation drivers

For example, Novell provides a driver named ODINSUP, which enables support for NDIS network protocols on an ODI data link layer, as well as a driver named ODI2NDI, which translates in the other direction. Microsoft provides an ODI to NDIS mapper called ODIHLP.EXE with Windows for Workgroups and Windows 95.

The translation drivers enable NDIS-based protocol stacks to be compatible with ODI-based protocol stacks on other computers, and to coexist on the same computer. You can think of the network adapter driver, the NDIS or ODI interface, and (if present) the translation driver as together forming the data link layer. For instance, you can configure a computer running OS/2 with an NDIS driver to communicate (via ODI2NDI) with a Novell NetWare server using IPX on ODI drivers while, at the same time, communicate (via the NDIS driver) with a Windows NT computer using TCP/IP on an NDIS driver.

Get the latest versions—The translation driver technology is relatively new, and not all translation drivers achieve complete compatibility. Be sure to get the latest available version of the driver you need. Although we provide some tips concerning network troubleshooting in this booklet, the primary source of assistance in troubleshooting a particular protocol stack is the documentation for the network communications software you install.



Protocols supported by SQL Anywhere

Properly configured SQL Anywhere packages run under the following networks and protocols:


- ◆ The NetWare database server runs on all Novell networks using the IPX or TCP/IP protocols.
- ◆ The Windows NT and Windows 95 database servers run on networks using the NetBIOS, TCP/IP, or IPX protocols.
- ◆ The OS/2 database server runs on networks using the NetBIOS, TCP/IP, or IPX protocols.
- ◆ The Windows 3.x database server runs on networks using the NetBIOS, TCP/IP, or IPX protocols.
- ◆ The DOS database server runs on networks using the NetBIOS or IPX protocols.
- ◆ The QNX database server runs on networks using either QNX messages or the TCP/IP protocol.
- ◆ The OS/2 and Windows NT database servers use the Named Pipes transport level protocol for communication with DOS and Windows 3.x client applications running on the same machine as the database engine. Named Pipes are not used for network communications.
- ◆ The Windows 3.x database server uses a transport based on Windows messaging to communicate with a client running on the same computer.

The SQL Anywhere Client for each platform supports the same protocols as the corresponding server.

In order for SQL Anywhere to run properly, the protocol stack on the client and server computers must be compatible at each layer.

Using SQL Anywhere with IPX

IPX is a network level protocol from Novell. SQL Anywhere for NetWare, Windows NT, Windows 95, Windows 3.x, OS/2, and DOS can all employ the IPX protocol. This section provides some tips for using IPX under different operating systems.

 [Using IPX with Windows 3.1](#)

 [Using IPX with Windows for Workgroups](#)

 [Using IPX with Windows 95](#)

 [Using IPX with OS/2](#)

 [Using IPX with Windows NT](#)

 [Using IPX with DOS](#)

 [Tuning SQL Anywhere performance under Novell IPX](#)

Using IPX with Windows 3.1

IPX support for Windows 3.1 is generally associated with Novell NetWare. You must have the Novell support for Windows 3.1 installed and have Windows or WIN-OS/2 setup for Novell NetWare. This setup is done by the Novell NetWare requestor installation, and places three drivers into your SYSTEM.INI: VIPX.386, VNETWARE.386, and NETWARE.DRV. Many network adapters require an

`EMMExclude=XXXX-YYYY`

line in the SYSTEM.INI file that excludes memory used by the network adapter.

The SQL Anywhere installation installs two Novell dynamic link libraries (DLLs): NWCALLS.DLL and NWIPXSPX.DLL. You should check to see that there is not an older version of any DLLs used by SQL Anywhere that are in the Windows system directory or in the PATH before the SQL Anywhere directory.

If you are running Windows in standard mode, you must run the task-switched buffer management program (TBMI2) before running Windows. You should not run this program if you are using Windows in enhanced mode.

■ For information about using IPX in a WIN-OS/2 session, see "[Using IPX with OS/2](#)".

Using IPX with Windows for Workgroups

Windows for Workgroups comes with NetBEUI as its default transport protocol. Microsoft NetBEUI employs a NetBIOS interface.

You can use both NetBEUI-based Windows for Workgroups network software and IPX-based Novell network software on a single network adapter board by installing NetWare support as an additional network.




NetWare support

Installing NetWare support allows a database server for Windows or a SQL Anywhere Client (DBCLIENW) to use an NDIS-based protocol stack while also allowing access to a NetWare file server, even though the NetWare file server will generally be using an ODI driver. Installation is carried out using the Network Setup utility in the Network Program Manager group. See NETWORKS.WRI in your Windows for Workgroups directory for details on installing NetWare support. You can view or print the file with the Write accessory in the Accessories Program Manager group.

IPX/SPX Compatible Transport

Windows for Workgroups can also support IPX communication even when the machine is not configured as a NetWare client, through the IPX/SPX Compatible Transport that comes with Windows for Workgroups. IPX/SPX Compatible Transport is installed by default on machines with enough memory. A Windows for Workgroups client can then communicate with SQL Anywhere using the NetBIOS protocol or IPX protocol.

IPX/SPX Compatible Transport is only necessary when Windows for Workgroups is not configured for NetWare and you would like a SQL Anywhere for Windows to communicate with a SQL Anywhere Client on a NetWare client machine or you would like a SQL Anywhere Client (DBCLIENW) to communicate with a SQL Anywhere using IPX.

	<u>Network Guide</u>
	<u>CHAPTER 4. Network Communications</u>
	<u>Using SQL Anywhere with IPX</u>

Using IPX with Windows 95

The IPX/SPX network protocol is the default network installed by Windows 95. If you installed Windows 95 without network support, you can install it at a later date from the Control Panel, Network Settings.

Using IPX with OS/2

If you are using Novell's Workstation for OS/2 and you wish to use the IPX protocol from a DOS or WIN-OS/2 session, you must have Novell network access from Virtual DOS and Windows installed (VIPX.SYS and VSHELL.SYS). You must run the correct version of NETX or use GLOBAL NetWare resources in order to use IPX from a DOS or WIN-OS/2 session. For details, see your Novell Workstation for OS/2 documentation.

You must use WIN-OS/2 in standard mode, not enhanced mode, for the Novell IPX driver to work properly. You must run the task-switched buffer management program (TBM12) before running WIN-OS/2. Adding a line to the AUTOEXEC.BAT of your WIN-OS/2 session is a convenient way of doing this.

Using IPX with Windows NT

Windows NT 3.5 ships with IPX network software that use NDIS network drivers. This software can be installed from the Windows NT Control Panel, Network Settings. This software allows a SQL Anywhere for Windows NT or a SQL Anywhere client to use Windows NT IPX running on NDIS, while also allowing access to a NetWare file server, even though the NetWare file server will generally be using an ODI driver.

You must install NWLink IPX/SPX Compatible Transport and Client Service for NetWare to use this feature. In some environments problems have been found when the default setting (Auto-detect) has been chosen for the frame type. In this case, the frame type settings for client and server should be the same as each other.

■	<u>Network Guide</u>
■	<u>CHAPTER 4. Network Communications</u>
■	<u>Using SQL Anywhere with IPX</u>

Using IPX with DOS

IPX support for DOS is most commonly associated with NetWare. If you have the NetWare client software installed and running on your computer, you should be able to use SQL Anywhere for DOS or a DOS SQL Anywhere client.

■ For information about using IPX in a DOS OS/2 session, see "[Using IPX with OS/2](#)".

Tuning SQL Anywhere performance under Novell IPX

IPX parameters are found in the "Protocol IPXODI" section of your NET.CFG file. The Novell IPX driver has only one parameter that affects SQL Anywhere performance:

IPX PACKET SIZE LIMIT The IPX PACKET SIZE LIMIT is usually 576. The default maximum packet size used by the SQL Anywhere is 512 bytes plus 30 bytes for IPX addresses.

If you are fetching long records in your multiuser system, you may want to increase the IPX packet size and specify the new packet size less 30 bytes as the packet size parameter (-p) on both the SQL Anywhere Client and database server command lines.


Using SQL Anywhere with NetBIOS

NetBIOS is an interface for interprocess communications, not a protocol specification. Programs written to the NetBIOS interface should operate over a variety of protocol stacks. For instance, implementations of the NetBIOS interface exist for OS/2 using TCP/IP—that is, TCP/IP is used as the communication mechanism for the NetBIOS interface for the OS/2 operating system.


The protocol stacks implementing the NetBIOS interface must be compatible between the SQL Anywhere client and server machines. For instance, a NetBIOS interface using SPX/IPX as the communication mechanism is incompatible with another NetBIOS implementation using NetBEUI.


As Windows 95, Windows for Workgroups and IBM networking software both use NetBIOS with NetBEUI as a standard protocol, configuration of the NetBIOS protocol stack for these environments is carried out by the network software installation.

 [Using NetBIOS with OS/2](#)

 [Using NetBIOS with Windows NT](#)

 [Using NetBIOS with DOS](#)

 [NetBIOS session support in SQL Anywhere](#)

 [Tuning SQL Anywhere performance under the NetBIOS datagram service](#)

█	<u>Network Guide</u>
█	<u>CHAPTER 4. Network Communications</u>
█	<u>Using SQL Anywhere with NetBIOS</u>

Using NetBIOS with OS/2

If you are using IBM's Network Transport Services/2 or LAN Server requestors, you must have the OS/2 LAN Virtual Device Driver installed in order to use NetBIOS from a DOS or WIN-OS/2 session. The LAN VDD is documented in the NTS/2 documentation. Add the following lines to your CONFIG.SYS:

```
DEVICE=C:\IBMCOM\PROTOCOL\LANVDD\OS2
```

```
DEVICE=C:\IBMCOM\PROTOCOL\LANPDD\OS2
```


Using NetBIOS with Windows NT

Windows NT 3.5 ships with NetBIOS interfaces to a number of different protocol stacks (NetBEUI, NW IPX/SPX transport, TCP/IP) that use NDIS network drivers. Install **NetBIOS Interface** from the Windows NT Control Panel, Network Settings. SQL Anywhere for NT and the SQL Anywhere Client for Windows NT work with any of these protocol stacks, but you must be sure to use compatible protocol stacks on the client and server sides of the communication.

You can have more than one NetBIOS interface active at one time. Each interface appears to the SQL Anywhere software as a different LAN adapter number. SQL Anywhere can simultaneously use different LAN adapter numbers, and so can simultaneously communicate on multiple NetBIOS interface protocol stacks.

Using NetBIOS with DOS

If you have Windows for Workgroups, you can run NetBIOS software from DOS by typing the following command:

```
net start netbeui
```

You should not use this command if you intend to start Windows. First, you should unload the NetBEUI support with the following command:

```
net stop
```

NetBEUI is not available from DOS when NetWare support is installed.

NetBIOS session support in SQL Anywhere

Within NetBIOS, communications can take place using a NetBIOS **session** or using the **datagram** service. NetBIOS sessions, or virtual circuits, are the default NetBIOS link for SQL Anywhere. In a NetBIOS session, all communication between the client and server applications takes place in the context a reliable two-way connection between the two applications. Broadcast messages are not required in a session, so that network traffic outside the two communicating computers is affected less than when the datagram service is used.

Tuning SQL Anywhere performance under the NetBIOS datagram service

The NetBIOS datagram service is supported only for reasons of compatibility with releases of Watcom SQL earlier than 4.0a. You should use the NetBIOS session service instead of the NetBIOS datagram service to run SQL Anywhere.

The information in this section is only for those employing the NetBIOS datagram service. To employ this service, the SQL Anywhere Client and the database server must be started with the -x **NetDG** option. If you are employing this service, the areas of concern are:

Maximum datagram packet size The default packet size for the SQL Anywhere Client and the database server is 512 bytes. Some NetBIOS implementations have a smaller maximum datagram size. In this case, the client may not be able to communicate at all with the server, or it may have problems only when a packet is attempted that is bigger than the supported size. The maximum packet size should be at least 600 to allow for protocol information in the packet.

You may need to change the packet size used by the SQL Anywhere multiuser support to 256 bytes (-p parameter on both the database server and the SQL Anywhere Client—see the chapter "[SQL Anywhere Components](#)").

Directory name service for datagrams In standard NetBIOS implementations, all datagrams are broadcast to all nodes on the network. The directory name service caches the most recently used NetBIOS names and remembers the addresses. Subsequent datagrams to that name will be sent directly to the appropriate node and not broadcast. This name service will dramatically improve network performance. The client should have a name cache of at least the number of servers plus four. The server should have a name cache of at least the number of clients plus four.

Timeout and retry settings The timeout and retry settings will affect performance, but mostly the time it takes the server and the client to start up. Reduced timeout and retry settings will make the server and client start up faster.

IBM Network Transport Services/2 for OS/2

Use LAPS (LAN Adapter and Protocol Support) to configure NetBIOS parameters. These parameters are stored in the PROTOCOL.INI file. The following parameters have worked well with NTS/2:

```
USEMAXDATAGRAM=Y  
NAMECACHE=10  
NETBIOSRETRIES=4  
DATAGRAMPACKETS=10
```

The following parameters have worked well with SQL Anywhere for OS/2 using NTS/2 to support 36 clients:

```
USEMAXDATAGRAM=Y  
NAMECACHE=40  
NETBIOSRETRIES=4  
DATAGRAMPACKETS=10
```

IBM LAN Support Program for DOS

IBM's Local Area Network Support Program has two different configurations for NetBIOS. If you require both NetBIOS and IEEE 802.2 interface support, you will use DXMT0MOD.SYS. If you do not require IEEE 802.2 and you are using an NDIS adapter, you should use DXMJ0MOD.SYS. The installation aid program (DXMAID) on the install diskette will help you install the appropriate software. See the LAN Support Program documentation for a full description of installation and configuration.

NetBIOS Protocol Driver (DXMJ0MOD)

The following parameters have worked well with the NetBIOS Protocol Driver for NDIS Adapters (DXMJ0MOD.SYS) for SQL Anywhere for Windows and SQL Anywhere for DOS to support 36 clients:

```
USEMAXDATAGRAM=Y
NAMECACHE=40
NETBIOSRETRIES=4
DATAGRAMPACKETS=20
```

The following parameters have worked well with the NetBIOS Protocol Driver for NDIS Adapters (DXMJ0MOD.SYS) with the SQL Anywhere Client:

```
USEMAXDATAGRAM=Y
NAMECACHE=10
NETBIOSRETRIES=4
DATAGRAMPACKETS=10
```

These parameters are found in the PROTOCOL.INI file in the [DXMJ0MOD_MOD] section.

NetBIOS Device Driver (DXMT0MOD)

The following parameters have worked well with the NetBIOS Device Driver (DXMT0MOD.SYS) with SQL Anywhere for Windows and SQL Anywhere for DOS to support 36 clients:

```
DATAGRAM.MAX=Y
DHB.SIZE=600
REMOTE.DATAGRAM.CONTROL=Y
REMOTENAME.DIRECTORY=40
TRANSMIT.TIMEOUT=1
TRANSMIT.COUNT=1
```

The recommended CONFIG.SYS setting using the short forms for the above mentioned parameters is:

```
device=DXMT0MOD.SYS DG=Y DS=600 RND=40 RDC=Y TC=1 TT=1
```

The following parameters have worked well with the NetBIOS Device Driver (DXMT0MOD.SYS) with the SQL Anywhere Client:

```
DATAGRAM.MAX=Y
DHB.SIZE=600
REMOTE.DATAGRAM.CONTROL=Y
REMOTENAME.DIRECTORY=10
TRANSMIT.TIMEOUT=1
TRANSMIT.COUNT=1
```

The recommended CONFIG.SYS setting using the short forms for the above mentioned parameters is:

```
device=DXMT0MOD.SYS DG=Y DS=600 RND=10 RDC=Y TC=1 TT=1
```

Using SQL Anywhere with TCP/IP

TCP/IP is a suite of protocols originally implemented by the University of California at Berkeley for BSD UNIX. TCP/IP is gaining widespread use with the expansion of the Internet and the World-Wide Web. Different TCP/IP implementations rely on particular data link drivers in order to work correctly. For example, TCP/IP for OS/2 relies on NDIS drivers as the data link protocol, whereas TCP/IP for NetWare relies on ODI drivers for correct operation.

Unlike NetBEUI and IPX, the TCP/IP protocol is not associated with any specific software vendor. There are many implementations of TCP/IP available from different vendors, and many different programming interfaces to TCP. Consequently, SQL Anywhere supports only certain TCP/IP implementations on each platform. For details, see the subsections below for each platform. As all TCP/IP implementations do implement the same protocol suite, they are all compatible.

User Datagram Protocol

There are several entries into the TCP/IP protocol stack. SQL Anywhere employs the User Datagram Protocol (UDP). While it is called a transport protocol here and elsewhere, UDP provides little more than a user interface to the network layer IP. In particular, UDP is not a guaranteed transmission protocol.

-
- [Verified TCP/IP protocol stacks](#)
 - [Using TCP/IP with OS/2](#)
 - [Using TCP/IP with Windows 3.x](#)
 - [Using TCP/IP with Windows 95 and Windows NT](#)
 - [Using TCP/IP with QNX](#)
 - [Tuning SQL Anywhere performance under TCP/IP](#)

Verified TCP/IP protocol stacks

Many vendors supply TCP/IP protocol stacks and associated software. SQL Anywhere communications have been explicitly verified with the following TCP/IP implementations:

- ◆ **OS/2** IBM TCP/IP for OS/2 version 2.0
- ◆ **Windows** Chameleon TCP/IP For Windows, Novell LAN Workplace for Windows, Microsoft Winsock version 1.1, Trumpet TCP/IP for Windows, FTP Software TCP/IP 1.2, SunSoft PC-NFS Pro 1.x.
- ◆ **NetWare** TCP/IP For NetWare.
- ◆ **Windows NT** Microsoft Winsock version 1.1.

■	Network Guide
■	CHAPTER 4. Network Communications
■	Using SQL Anywhere with TCP/IP

Using TCP/IP with OS/2

The TCP/IP implementation of SQL Anywhere for OS/2, and of the SQL Anywhere OS/2 client, is written to the IBM TCP/IP for OS/2 2.0 specification. In order to use TCP/IP under OS/2, you need to obtain compatible TCP/IP software from a TCP/IP vendor or obtain the **TCP/IP for OS/2 Base Kit** from IBM.




Generally, TCP/IP software for OS/2 uses NDIS drivers. If you are running Novell NetWare in addition to TCP/IP, you will need a translation driver to ensure that both NDIS-based and ODI-based network protocols can run, as discussed in "[Working with multiple protocol stacks](#)".

Using TCP/IP with Windows 3.x

The TCP/IP implementation of SQL Anywhere for Windows 3.x, 95, and NT, and of the corresponding SQL Anywhere Clients, is written to the Winsock 1.1 standard. Your TCP/IP software must support this standard in order for SQL Anywhere to operate.

In order to use TCP/IP under Windows 3.x, or under Windows for Workgroups, you need to obtain TCP/IP software that conforms to the Winsock 1.1 standard from a TCP/IP vendor, or obtain Microsoft TCP/IP from Microsoft.

Generally, TCP/IP software for Windows uses NDIS drivers. If you are running Novell NetWare IPX in addition to TCP/IP, you will need a translation driver to ensure that both NDIS and ODI network layers can run. Windows for Workgroups comes with the ODIHLP translation driver.

	<u>Network Guide</u>
	<u>CHAPTER 4. Network Communications</u>
	<u>Using SQL Anywhere with TCP/IP</u>

Using TCP/IP with Windows 95 and Windows NT

Windows NT 3.5 ships with TCP/IP software that uses NDIS network drivers. Install TCP/IP Protocol from the Windows NT Control Panel, Network Settings.

This software allows a SQL Anywhere for Windows NT or a SQL Anywhere client to use Windows NT TCP/IP.

Using TCP/IP with QNX

In addition to the native QNX networking protocol, the QNX database server supports TCP/IP. This enables non-QNX clients to communicate with a QNX database server.

For performance reasons, you should consider running the Socket process on each QNX node using the TCP/IP link.

Tuning SQL Anywhere performance under TCP/IP

Although the default packet size for TCP/IP is 512 bytes, a larger 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 parameter on both the the SQL Anywhere Client and the database server command lines.

Using SQL Anywhere with Named Pipes

Named pipes are a facility for interprocess communication. Named pipes can be used for communication between processes on the same computer or on different computers.

SQL Anywhere for Windows NT and for OS/2 use local named pipes. This allows DOS and Windows applications running on Windows NT or on OS/2 to communicate with a SQL Anywhere client or standalone engine on the same machine.

SQL Anywhere does not use remote named pipes for communicating with client machines. You do not need to install remote named pipe support for local named pipes to work.

Using SQL Anywhere with QNX messages

QNX messages is the native QNX network protocol. You can use QNX messages to communicate between a QNX SQL Anywhere Client and a QNX database server.

By default, both server and client start all available protocols when they start up. If you are using QNX messages only, you may wish to disable the TCP/IP protocol on your client and server command lines.

Preventing the client from starting TCP/IP

If you experience lengthy delays when an application is first connecting to the database, it may be due to the client trying to start the TCP/IP link to the database. You can prevent the client from starting the TCP/IP link in the following ways:

- ◆ Set the SQLCLIENT environment variable. For example:

```
export SQLCLIENT="dbclient -x qnx"
```

- ◆ If you have the source code for the application, you can achieve the same effect by calling the `db_client_start_line` function in the database library. For example:

```
db_client_start_line( "dbclient -x QNX" );
```










Preventing the server from starting TCP/IP

You can prevent the server from starting the TCP/IP link by using the `-x` switch on the server command line. For example:

```
dbsrv50 -x qnx -n dbname sademo.db
```

Troubleshooting

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 you are using compatible protocols](#)
 -  [Ensure you have current drivers](#)
 -  [Switch off your computer between attempts](#)
 -  [Diagnose your protocol stack layer by layer](#)
 -  [Verify that the data link layer is working](#)
 -  [Testing a NetBIOS protocol stack](#)
 -  [Testing a TCP/IP protocol stack](#)
 -  [Diagnosing wiring problems](#)
 -  [A checklist of common problems](#)

Ensure you are using compatible protocols

If you have more than one protocol stack installed on the client or server computer, you should ensure that the SQL Anywhere Client and the database server are using the same protocol. The -x command line switch for the SQL Anywhere Client and for the database engine selects a protocol for the application to use. You can use this option to ensure that each application is using the same protocol. For more information about the -x switch, see the chapter "[SQL Anywhere Components](#)".

By default, both the database server and the SQL Anywhere Client attempt to use all available protocol stacks. The server supports client requests on any active protocol, and the client searches for a server on all active protocols.

Ensure you have current drivers




Old network adapter drivers are a common source of communication problems. You should ensure that you have the latest version of the NDIS or ODI driver for your network adapter, as appropriate. You should be able to obtain current network adapter drivers from the manufacturer or supplier of the card.

Network adapter manufacturers and suppliers make the latest versions of drivers for their cards available. Most card manufacturers have a BBS (bulletin board system) that you can phone and download the latest versions of NDIS and ODI drivers.

You may also be able to obtain a current network adapter driver from the provider of your networking software.




Novell makes available the latest versions of Novell client software through its distributors and resellers or from CompuServe. The NOVFILES CompuServe forum contains all of the most commonly downloaded files which includes client software for the supported operating systems. The main Novell forum is called NOVELL. If you do not have a CompuServe account, the NetWare manuals describe how to get one.

When you download Novell client software, it has ODI drivers for some network adapters in addition to the Novell software that is used for all network adapters.

	<u>Network Guide</u>
	<u>CHAPTER 4. Network Communications</u>
	<u>Troubleshooting</u>

Switch off your computer between attempts

Some network adapter boards do not reset cleanly when you reboot the computer. When you are troubleshooting, it is better to turn the computer off, wait a few seconds, and then turn it back on between attempts.

	<u>Network Guide</u>
	<u>CHAPTER 4. Network Communications</u>
	<u>Troubleshooting</u>

Diagnose your protocol stack layer by layer

If you are having problems getting your client application to communicate with a SQL Anywhere engine through the SQL Anywhere Client, you need to ensure that the SQL Anywhere Client and the database server are using compatible protocol stacks.




Many other network communication problems are independent of SQL Anywhere. A helpful method of isolating network communication problems is to work up the protocol stack, testing whether each level of communication is working properly. Here we give some suggestions for how you may do this.

Verify that the data link layer is working

If you can connect to the server computer in any way, then the data link layer is working, regardless of whether the connection is made using the same protocols you will be using for SQL Anywhere at higher layers of the protocol stacks.

For example, you may want to try to connect to a disk drive on the computer running the database server from the computer running the SQL Anywhere Client.

Having verified that the data link layer is working, the next step is to verify that other applications using the same network and transport layers as SQL Anywhere are working properly.

	<u>Network Guide</u>
	<u>CHAPTER 4. Network Communications</u>
	<u>Troubleshooting</u>

Testing a NetBIOS protocol stack

If you are using Windows for Workgroups, Windows 95, or Windows NT, and if you are using the native protocol for SQL Anywhere, try using the **chat** or **WinPopup** application to test whether the computers on which the SQL Anywhere Client and the database server are running can communicate with each other.

If you are running NetBIOS network communications software under OS/2, you should confirm that you can establish a connection between the computers running the SQL Anywhere Client and the database server using one of the applications that come with your network software.

You should ensure that these applications that come with your networking software are running properly before testing SQL Anywhere.

Testing a TCP/IP protocol stack

If you are running under TCP/IP, there are several applications that you can use to test the compatibility of the client computer and server computer TCP/IP protocol stack. The ping utility provided with many TCP/IP packages is useful for testing the IP network layer.

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 named IP-protocol stack.

First determine if your own protocol stack is operating correctly by "pinging" yourself first. If your IP-address is 191.72.109.12, you would enter:

```
ping 191.72.109.12
```

at the command line prompt and wait to see if the packets are being routed at all. If they are, the output will appear similar to:

```
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, and 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 computer running the SQL Anywhere Client before proceeding.

Using telnet or ftp to test the TCP/IP stack

To further test the TCP/IP stack you can start a server application on one computer, a client program on the other computer, and test whether they can communicate properly.

There are several applications commonly provided with TCP/IP implementations that can be used for this purpose: here we show how to use **telnet** to test the TCP/IP stack.

- ◆ Start a telnet server process (or **daemon**) on one machine. Check with your TCP/IP software for how to do this. For a typical command line telnet program you would type the following instruction at the command prompt: telnetd
- ◆ Start the telnet client process on the other machine and see if one gets a connection. Again, check with your TCP/IP software for how to do this. For command line programs, you would typically type the following instruction:

```
telnet server_name
```

where *server_name* is the name or IP address of the computer running the telnet server process.

If a telnet connection is established between these two machines, the protocol stack is stable and the SQL Anywhere client and server should be able to communicate using the TCP/IP link between the two computers. If a telnet connection cannot be established, there is a problem. You should ensure that your TCP/IP protocol stack is working correctly 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 will only occur on one machine, it may be a wiring problem or a hardware problem.

For information on detecting wiring problems under NetWare, see your Novell NetWare manuals. The Novell LANalyzer program is useful for tracking down wiring problems with Ethernet or TokenRing networks. Your NetWare authorized reseller can also supply you with the name of a Certified NetWare Engineer who can help diagnose and solve wiring problems.

A checklist of common problems

The section "[Network communications parameters](#)" lists many command-line switches for tuning network communications. You should check that section to see if any are relevant to your installation.

The following list presents some common problems and their solutions.

If you receive the message "Unable to start — server not found" when trying to start the client, the client cannot find the database server on the network. Check for the following problems:

- ◆ You have not specified the proper database server name when starting the SQL Anywhere Client. You must specify the name of the database server.
- ◆ The network configuration parameters of your network driver on the client machine are different from those on the server machine. For example, two Ethernet adapter cards should be using a common frame type. For Novell NetWare, the frame type is set in the NET.CFG file under DOS, Windows, and OS/2. In Windows NT the settings can be accessed through the Control Panel Network Settings. In Windows for Workgroups, it can be accessed through Network Settings in the Windows Setup.
- ◆ Under the TCP/IP protocol, clients search for database servers by broadcasting a request. Such broadcasts will typically not pass through gateways, so that if the database server is on a machine in another (sub)network, it will not be found. If this is the case, supply the host name of the machine on which the server is running using the -x command-line option.
- ◆ Your network drivers are not installed properly or the network wiring is not installed properly.
- ◆ The network configuration parameters of your network driver are not compatible with SQL Anywhere multiuser support. See "[Configuring your network adapter board](#)" for a description of configuration parameters that affect performance and may affect operation of the client and server.
- ◆ If your network communications are being carried out using TCP/IP, and you are operating under Windows for Workgroups or Windows NT, check that your TCP/IP software conforms to the Winsock 1.1 standard.
- ◆ If you are using NetBIOS, check that your SQL Anywhere Client and database server are Release 4.0A or later. Earlier releases of SQL Anywhere employ datagrams for communication, while Release 4.0A introduced support for NetBIOS sessions, which is the recommended communication method.

If you receive the message "Unable to initialize any communication links", no link can be established. The probable cause is that your network drivers have not been installed. The server and the client try to start communication links using all available protocols unless you have specified otherwise using the -x option. Check your network documentation to find out how to install the driver you wish to use.

Under DOS and Windows, SQL Anywhere may be affected by Terminate and Stay Resident programs (TSR) that you are running on the machine. This may show up as errors reported when starting the server, frequent crashes of the server machine, poor performance of the server or database recovery problems after a power failure. Remove any TSR not required by your network drivers and try the server again.

Configuring your network adapter board

Network adapter boards have configuration settings that allow them to use different interrupt request (IRQ) levels. The network adapter board may work when it is using the same IRQ level as another adapter board in your computer (for example, your parallel card). Performance may suffer when the network adapter board shares an IRQ level.

The drivers for some adapter boards have parameters. The most important type of parameter to look out for is one that controls buffer size or number of buffers. For example, some versions of the NDIS driver for an SMC PLUS Ethernet adapter have these configuration parameters:

```
ReceiveBuffers = 12
```

```
ReceiveBufSize = 768
```

The default packet size for the SQL Anywhere Client and the database server is 512 bytes. The maximum buffer size used by the adapter board should be at least 600 to allow for protocol information in the packet. The computer running the database server might need more than the default number of buffers used by a driver.

Evaluating network performance

The SQL Anywhere multiuser tools provide some statistics for evaluating network performance. These are gathered at both the server and the client ends of the communication. The server statistics can be displayed by pressing F2 when the server is running. A server communication statistics window will be displayed. Server statistics can also be accessed from SQL Central and the Windows NT Performance Monitor.







The client statistics can be displayed by running the remote monitoring facility (DBWATCH) and pressing F3. For more information about the server and DBWATCH display, see the chapter "[The Database Server Display](#)".

DBWATCH has a **test mode**. In test mode, it sends packets to the server as rapidly as it can receive response packets from the server. Activate test mode by choosing **Test Mode** from the **Command** menu.

CHAPTER 5. SQL Anywhere Components







About this chapter




This chapter presents reference information on the SQL Anywhere network server, the SQL Anywhere Client, and the server monitoring facility (DBWATCH). It includes descriptions of command-line switches, which can be entered on the command line or held in a configuration file.

-
-  Environment variables
 -  Software component return codes
 -  The database server
 -  The SQL Anywhere Client
 -  The DBWATCH server monitoring facility
 -  Network communications parameters

Environment variables

The following is a list of the environment variables that are used by SQL Anywhere and a description of where they are used.

-
-  SQLANY environment variable
 -  SQLCONNECT environment variable
 -  SQLPATH environment variable
 -  SQLREMOTE environment variable
 -  SQLSTART environment variable
 -  TMP environment variable

	Network Guide
	CHAPTER 5. SQL Anywhere Components
	Environment variables

SQLANY environment variable

SQLANY = *path*

Description

The SQLANY environment variable is used to contain the directory where SQL Anywhere is installed. The default installation directory is c:\sqlany50. The install procedure automatically adds the SQLANY environment variable to your startup environment. The SQLANY variable is used by the batch files or command files that build the Embedded SQL examples.

In QNX, SQL Anywhere is installed in a fixed directory and the SQLANY variable is not required.

SQLCONNECT environment variable

SQLCONNECT = *keyword=value* ; ...

SQLCONNECT = *keyword#value* ; ...




Description

The SQLCONNECT environment variable specifies connection parameters that are used by several of the database tools to connect to a database engine or network server. This string is a list of parameter settings of the form **KEYWORD=value**, delimited by semicolons. The number sign "#" is an alternative to the equals sign, and should be used when setting the connection parameters string in the SQLCONNECT environment variable, as using "=" inside an environment variable setting is a syntax error.

The keywords are from the following table.

Verbose keyword	Short form
Userid	UID
Password	PWD
ConnectionName	CON
EngineName	ENG
DatabaseName	DBN
DatabaseFile	DBF
DatabaseSwitches	DBS
AutoStop	AutoStop
Start	Start
Unconditional	UNC
DataSourceName	DSN

For a full description of the connection parameters, see "Connecting to a Database", in the *SQL Anywhere User's Guide*.

	Network Guide
	CHAPTER 5. SQL Anywhere Components
	Environment variables




SQLPATH environment variable

SQLPATH = *path*;

PATH = *path*;

Description

ISQL searches along SQLPATH for ISQL command files and help files before searching the system path.




	Network Guide
	CHAPTER 5. SQL Anywhere Components
	Environment variables

SQLREMOTE environment variable

SQLREMOTE = *path*

Description

Addresses for the FILE message link in SQL Remote replication are subdirectories of the SQLREMOTE environment variable. This variable should point to a shared directory.

	Network Guide
	CHAPTER 5. SQL Anywhere Components
	Environment variables

SQLSTART environment variable

SQLSTART = *start-line*

SQLSTARTW = *start-line*

Description

Historical. The SQLSTART environment variable information has been added to the SQLCONNECT environment variable as the **Start** parameter.

TMP environment variable

TMP = *directory*

TMPDIR = *directory*

TEMP = *directory*

Description

The database engine creates temporary files for various operations such as sorting and performing unions. These temporary files will be placed in the directory specified by the TMP, TMPDIR, or TEMP environment variables. (The database engine takes the first one of the three that it finds.)

If none of the environment variables is defined, temporary files are placed in the current directory.

Software component return codes

All database components use the following executable return codes. The header file SQLDEF.H has constants defined for these codes.

Code	Explanation
0	Success
1	General failure
2	Invalid file format, and so on
3	File not found, unable to open, and so on
4	Out of memory
5	Terminated by user
6	Failed communications
7	Missing required database name
8	Client/server protocol mismatch
9	Unable to connect to database engine
10	Database engine not running
11	Database server not found
254	Reached stop time
255	Invalid parameters on command line

The database server

{button See also,PI('`DBENGINES_sa')}

Syntax

```
dbsrv50 [ server-switches ] [ database-file [database-switches ]]*
```

Windows 3.x syntax

```
dbsrv50w [ server-switches ] [ database-file [database-switches]]*
```

QNX Syntax

```
dbsrv50 [ server-switches ] [ database-file [ database-switches ]]*
```

Switch	Description
@filename	Read in switches from configuration file
@envvar	Read in switches from environment variable
-b	Run in bulk operations mode
-c cache-size	Set maximum cache size
-C console	Run server on specified console (QNX only)
-d	Disable asynchronous I/O (OS/2, Windows NT, NetWare only)
-df	Force direct I/O (Windows 3.1, DOS only)
-di	Use direct I/O if possible (Windows 3.1, DOS only)

-e	Enable packet encryption
-ga	Automatically unload database after last connection closed
-gb level	Set database process priority class to <i>level</i> .
-gc num	Set checkpoint timeout period
-gd level	Set database starting permission
-ge size	Sets the stack size for threads that run external functions
-gf	Disable firing of triggers
-gk level	Set permission for stopping the server using DBSTOP
-gl level	Set integrated login level (Windows NT, Windows 95 only)
-gn num	Set number of threads
-gp size	Set maximum page size
-gr num	Set maximum recovery time
-gs size	Set thread stack size
-gw num	Set the interval (in milliseconds) for background processing
-gx	Disable dual threading
-l password	(lower case L) Lock the keyboard with specified password
-m	Truncate transaction log after checkpoint, for all databases
-n name	Name the database server
-o filename	Output messages to file
-p packet-size	Set maximum network packet size
-q	Quiet mode—suppress output
-r	Disable multiple-row fetching
-ta sec	Scan time for terminated applications:—default 30 seconds
-ti min	Client idle time before shutdown:—default 240 minutes
-tl sec	Default liveness timeout for clients in seconds—default is 120 seconds
-tq time	Set quitting time
-tr sec	Active request termination if no retries—default 60 seconds
-u	Use buffered disk I/O (Windows 95 and Windows NT only)
-v	Log old values of all columns on UPDATE or DELETE for all databases
-x list	Comma separated list of communication links to provide
-y	Run as a Windows 95 service
-Z	Provide diagnostic information on communication links

Recovery

Switch	Description
--------	-------------

-a log-file Apply named transaction log file
-f Force database to start without transaction log

Database

Switch	Description
-m	Truncate transaction log after checkpoint
-n name	Name the database
-v	Log old values of all columns on UPDATE or DELETE

Description

Starts the database server.

The *database* specifies the database filename. If *database* is specified without a file extension, SQL Anywhere first looks for *database* with extension WRT (a write file.) followed by *database* with extension .DB.

NetWare database server—In the case of database server for NetWare, the database file and the transaction log file must be on a NetWare volume, and the paths must be fully specified. The files cannot be on a DOS partition. NetWare allows you to have volumes that span 2 or more hard disks. When the database file is on one of these volumes, NetWare can sometimes perform multiple I/O operations at the same time, improving the performance of the SQL Anywhere.

Switches

@filename Read in command-line switches from the supplied file.

The file may contain line breaks, and may contain any set of command line switches. For example, the following command file holds a set of command line switches for an engine that starts with a cache size of 4Mb, a name of myserver, and loads the sample database:

```
-c 4096
-n myserver
c:\sqlany50\sademo.db
```

If this configuration file is saved as C:\CONFIG.TXT, it can be used in an command line as follows:

```
DBENG50 @c:\config.txt
```

@environment-variable Read in command-line switches from the supplied environment variable. The environment variable may contain any set of command line switches. For example, the first of the following pair of statements sets an environment variable holding a set of command line switches for a database server that starts with a cache size of 4Mb and loads the sample database. The second statement starts the database server:

```
set envvar=-c 4096 c:\sqlany50\sademo.db
DBENG50 @envvar
```

Environment variable given priority—If you have both a file and an environment variable with the value of your @ command-line switch, the environment variable is used.

-b Use bulk operation mode.

This is useful when loading large quantities of data into a database.

The database server allows only one connection by one application. It does not keep a rollback log or a transaction log, and the multiuser locking mechanism is turned off. You should use a new log file when starting the database server after loading data with the -b switch.

Bulk operation mode does not disable the firing of triggers.

-c cache-size Set the size of the cache. The database server uses extra memory for caching database pages if it is set aside in the cache. Any cache size less than 10000 is assumed to be K-bytes (1K = 1024 bytes). Any cache size 10000 or greater is assumed to be in bytes. The cache size may also be specified as nK or nM (1M = 1024K). By default, the database server uses 2 megabytes of memory for caching. The more cache that can be given the engine, the better will be its performance.

NetWare database server—There is a trade off between memory for 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 not already loaded) and select "Resource Utilization". Buying extra memory for your NetWare server computer could improve database performance and/or file server performance dramatically.

-C console This option is for QNX only. It specifies on which console number the database server display should run. If this option is omitted, then the database server uses the current console if running in the foreground, or the first unused console if running in the background.

The server display appears on a console, as follows:

- ◆ By default, the server display uses the current console.
- ◆ If you start the database server as a background task (using &) the server display uses a different console.
- ◆ With a `-C 0` (zero) command line, the database server uses the first unused console.
- ◆ With a `-C n` command line, the database server uses console *n*. For example,

```
dbsrv50 -C 4
```

uses the console `/dev/con04`.
- ◆ With a `-C device` command line, the database server uses the specified device. This allows the database server display to use a different node from the current node. For example, `dbserver -C //1/dev/con03` uses console 3 on node 1 for the server display.
- ◆ If you start the database server using the `-q` switch, there is no server display.

-d disable asynchronous I/O Use synchronous I/O rather than asynchronous I/O. Asynchronous I/O is generally the preferred option.

Since DOS and Windows 3.x systems use synchronous I/O by default, this option applies only to Windows NT, OS/2 and NetWare systems, which use asynchronous I/O by default.

-df force direct I/O For DOS and Windows 3.x database engines and servers only. The default I/O method for DOS and Windows 3.x database servers is to use normal DOS input and output instead of direct input and output.

This option forces the use of direct, or asynchronous, I/O, rather than normal DOS calls. Since asynchronous I/O is the default setting for Windows NT, OS/2 and NetWare systems, this option has no effect in those environments.

Asynchronous I/O is not supported in Windows 95 environments.

-di use direct I/O if possible For DOS and Windows 3.x database engines and servers only. The default I/O method for DOS and Windows 3.x database engines and servers is to use normal DOS input and output instead of direct input and output.

When the `-di` option is supplied, the database server tests to see if it is possible for direct I/O to be used before it is implemented.

With this switch, the server will not use direct I/O under Windows for Workgroups 3.11 or for some highly fragmented database files.

Asynchronous I/O is not supported in Windows 95 environments.

-e Encrypt all packets transmitted to and from all clients over the network. By default, packets are not encrypted, thus opening a potential security risk. If you are concerned about the security of network packets, use the **-e** switch. Encryption does marginally affect performance.

-ga Applications can cause databases to be started and stopped by the engine. Specifying this switch causes the engine to shutdown when the last database is stopped.

Automatically unload each database after the last connection to it is dropped. The database server itself does not shut down.

-gb level OS/2 and Windows NT only. Set the database process priority class to *level*. Level must be one of **idle**, **normal** (the default), **high**, or **maximum**. Idle is provided for completeness, and maximum may interfere with the running of your computer. Normal and high are the commonly used settings.

-gc num Set the maximum desired length of time (in minutes) that the database server will run without doing a checkpoint. The default value is 60 minutes.

For more information, see the description of the CHECKPOINT_TIME option in the *SQL Anywhere User's Guide*.

When a database server is running with multiple databases, the checkpoint time specified by the first database started will be used unless overridden by this switch. If a value of 0 is entered, the default value of 60 minutes is used.

-gd level Set the database starting permission to *level*. This is the permission level required by a user to cause a new database file to be loaded by the server. The level can be one of the following:

- ◆ **dba** Only users with DBA authority can start new databases.
- ◆ **all** All users can start new databases.
- ◆ **none** Starting new databases is not allowed.

The default setting is **all**.

-ge size Sets the stack size for threads running external functions, in bytes. The default is 16384 (16K). This switch is used only for OS/2, Windows 95 and NT, and NetWare.

-gf Disable firing of triggers by the server.

-gk level Set the permission required to stop the database server using DBSTOP to *level*. The level can be one of the following:

- ◆ **dba** Only users with DBA authority can use DBSTOP to stop the engine (the default).
- ◆ **all** All users can use DBSTOP to stop the server.
- ◆ **none** The server cannot be stopped using DBSTOP.

-gl level Windows NT, Windows 95 only. Sets the level of permitted integrated logins to *level*. This option can permit the use of a single user ID and password for both database connections and operating system or network logins. The *level* can be one of the following:

- ◆ **Standard (s)** This is the default setting, which disallows integrated logins.
- ◆ **Integrated (i)** All logins to the database must be made using integrated logins.
- ◆ **Mixed (m)** Both integrated and standard logins are allowed.

Network clients choice using Integrated login level—Only Windows NT and Windows 95 network

clients can work in conjunction with the integrated login option. As a result, network servers started with the `-gli` option will only permit connections from network clients using Windows NT or Windows 95.

■ For more information about integrated logins please see using_an_integrated_login.]

-gn num Sets the number of execution threads that will be used in the database server while running with multiple users.

■ For more information, see the description of the `THREAD_COUNT` option in the *SQL Anywhere User's Guide*.

When a database server is running with multiple databases, the thread count specified by the first database started will be used unless overridden by this switch.

-gp size Sets the maximum page size allowed, in bytes. The size specified must be one of: 512, 1024, 2048, or 4096. When a database server is running with multiple databases, the page size specified by the first database will be used unless overridden by this switch. Without using this option, an attempt to load a database file with a page size larger than the page size of the database first loaded will fail.

-gr num Set the maximum desired length of time (in minutes) that the database server will take to recover from system failure.

■ For more information, see the description of the `RECOVERY_TIME` option in the *SQL Anywhere User's Guide*.

When a database server is running with multiple databases, the recovery time specified by the first database started will be used unless overridden by this switch.

-gs size Sets the stack size of every internal execution thread in the server. The value entered is multiplied by four to produce the stack size in bytes.

The default number of execution threads and the stack size of each thread is operating system dependent. The number of execution threads is controlled by the `THREAD_COUNT` database option, and has a default value of twenty for all engines and servers except for the 16-bit Windows 3.x engine (DBENG50S.EXE). The default value of *size* is operating system dependent. You may want to use the `-gs` option to lower the memory usage of the database server in environments with limited memory.

-gw num Sets the interval for background processing. At each interval, the server carries out one I/O operation. The default setting is 500 (half a second).

-gx Disable dual operating-system threading. This option is available for the Windows 95 and Windows NT versions only.

On machines with more than one processor, the network request management and the request processing are run by default as two separate operating system threads. Use the `-gx` switch if you want the database server to never use more than one processor of a multi-processor machine.

-l password Lock keyboard with specified *password*.

-m 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 (also settable 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 provided against media failure on the device containing 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.

Replicated databases—Do not use the `-m` option with databases that are being replicated as replication

inherently relies on transaction log information.

-m 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 (also settable 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 provided against media failure on the device containing 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.

Replicated databases—Do not use the -m option with databases that are being replicated as replication inherently relies on transaction log information.

-n name Set the name of the database server. 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:\SQLANY50\SADEMO.DB and no -n switch is specified, then the name of the server will be sademo.

The server name specifies the name that is used when the SQL Anywhere Client (DBCLIENT) is started. It is also used on the connect statement issued by client applications.

There are two n switches—The -n switch is positional. If it appears after a database filename, it has a different meaning. See "Database switches" later in this section.

The server name can be used on the connect statement to specify to which server you wish to connect. In all environments, there is always a default database server that will be used if no server name is specified provided at least one database server or SQL Anywhere Client (DBCLIENT) is running on the computer.

-o filename Print all server message window output to a file (in addition to displaying it on the screen).

-p packet-size Set the maximum size of communication packets. The default is 512 bytes. The minimum value is 200 bytes.

-q Do not display the server screen (no console for the server).

-r Fetch 1 row per network request. By default, when the database server gets a simple fetch request, it will fill one network packet with several rows so that subsequent sequential fetches do not require network traffic. This is often referred to as **blocking** of fetches. This switch disables multiple-row fetching for all clients.

-ta seconds For Windows 3.x, Windows NT, and Windows 95 only. The database server periodically scans the connection list and disconnects any connections associated with terminated applications connected *directly* to the server. To disconnect client/server connections from terminated applications use the -ta switch on the SQL Anywhere Client. The scan period can be controlled using the -ta switch, and has a default value of 30 seconds. Setting the value to zero prevents scanning.

-ti minutes Disconnect connections that have not submitted a request for *minutes* minutes. The default is 240 (4 hours). A client machine in the middle of a database transaction will hold locks until the transaction is ended or the connection is terminated. The -i option is provided to disconnect inactive connections, freeing their locks.

-tl seconds A liveness packet is sent periodically across a client/server TCP/IP or IPX communications protocol to confirm that a connection is intact. If the server runs for a liveness timeout period (default 2

mins) without detecting a liveness packet, the communication is severed. The server drops any connections associated with that client. DOS and QNX clients do not do liveness.

The `-tl` switch on the server sets the liveness timeout for all clients that do not specify a `-tl` switch.

Liveness packets are sent at an interval of the (liveness timeout)/4.

-tq time Shut down the server at a specified time. This is useful for setting up automatic off-line backup procedures (see the chapter "Backup and Recovery" in the *SQL Anywhere User's Guide*). 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".

-tr seconds An active request and connection is terminated if the server does not receive a retransmit within the specified number of seconds.

The default setting is 60 seconds. The client retry time is controlled by the `-ts` client command-line switch.

-u Files are opened using the operating system disk cache in addition to the database cache. This option applies to the Windows 95 and Windows NT database servers only.

While the operating system disk cache may improve performance in some cases, in general better performance is obtained without this switch, 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).

-v Cause the network server to record in the transaction log the previous values of each of the columns whenever a row of any loaded database is updated or deleted. By default, the server will only record enough information to uniquely identify the row (primary key values or values from a not null unique index). This switch is useful for working on a copy of a database file.

-x list Use only the listed communication links.

For example,

```
-x NamedPipes,IPX
```

allows only Named Pipes and IPX communications.

The default is to try all settings supported by database server on your operating system.

The *list* is a comma-separated list of settings taken from the following list: IPX, TCPIP, NetBIOS, NetDG, NamedPipes, Windows, or QNX.

The NetDG setting is provided for backward compatibility only. It uses a NetBIOS driver employing datagram-based communication rather than session-based communication. Session-based communication is more efficient. The NamedPipes and Windows settings are for local (same-computer) communications only.

The supported settings on each operating system are as follows:

- ◆ The NetWare database server supports IPX and TCP/IP links.
- ◆ The Windows NT and Windows 95 database servers support all listed links, except for Windows.
- ◆ The OS/2 database server supports all listed links, except for Windows.
- ◆ The Windows 3.x database server supports all listed links, except for NamedPipes.
- ◆ The DOS database server supports IPX, NetBIOS, and NetDG links.

- ◆ The QNX database server supports TCP/IP and QNX links.

For some protocols, additional parameters may be provided, in the format

```
-x tcpip{PARAM1=value1;PARAM2=value2;...}
```

For QNX, quotation marks are required if more than one parameter is supplied:

```
-x "tcpip{PARAM1=value1;PARAM2=value2;...}"
```

■ For a description of available parameters, see "[Network communications parameters](#)".

-y Runs server as a Windows 95 service. By registering the server as a Windows 95 service it continues to operate whether as users log on or off and shutdown commands are ignored.

-Z Provides diagnostic information on communications links on startup. This should only be used when tracking problems. Option only accepts upper case letter.

Recovery switches

-a log-file Apply the named transaction log. This is used to recover from media failure on the database file (see the chapter "Backup and Recovery" in the *SQL Anywhere User's Guide*). When this option is specified, the database server will apply the log and then terminate—it will not continue to run.

-f This option is used for recovery: to force the database server to start after the transaction log has been lost (see the chapter "Backup and Recovery" in the *SQL Anywhere User's Guide*).

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.

Database switches

-m Truncate (delete) the transaction log when a checkpoint is done, either at shutdown or as a result of a checkpoint scheduled by the engine. 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 (also definable 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 provided against media failure on the device containing 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 switch is the same as the **-m** engine switch, but applies only to the current database or the database identified by the *database-file* command-line variable.

Replicated databases—Do not use the **-m** option with databases that are being replicated as replication inherently relies on transaction log information.

-n name Set the name of the database. 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 file with the path and extension removed. For example, if the server is started on C:\SQLANY50\SADEMO.DB and no **-n** switch is specified, then the name of the database is sademo.

-v Causes the server to record in the transaction log the previous values of each of the columns whenever a row of the specified database is updated or deleted. By default, the server will only record enough information to uniquely identify the row (primary key values or values from a not null unique index). This switch is useful for working on a copy of a database file.

The SQL Anywhere Client

{button See also,PI('`,`DBCLIENT_sa')}

Syntax

dbclient [*switches*] *server-name*

Windows 3.x syntax

dbclienw [*switches*] *server-name*

Switch	Description
-b <i>max-packets</i>	Set the maximum number of packets for multi-row fetches
-c	Purge previous connections
-e	Encrypt all network packets
-ga	Automatically shut down after last connection closed
-o <i>filename</i>	Output messages to a file
-p <i>packet_size</i>	Set maximum network packet size
-q	Quiet mode—suppress output
-r	Disable multiple-row fetching
-s <i>buffer_space</i>	Memory for buffers in K
-ta <i>sec</i>	Scan time for terminated applications:—default 30 seconds
-tl <i>seconds</i>	Client liveness timeout in seconds—default is server setting which defaults to 120 seconds
-ts <i>t1[,t2]</i>	Client min [and max] retry time in 100ths of a second
-x <i>list</i>	Comma separated list of communication links to run, with any parameters
-y	Start with no server connection
-Z	Provide diagnostic information when starting communication links

Description

In a client/server arrangement using the SQL Anywhere network server, the SQL Anywhere Client runs on the client application computer. Your application connects to the database via the SQL Anywhere Client, which manages the network .

A server with the same *server name* must have already been started on a machine on the network. The SQL Anywhere Client is a separate executable and it manages many programs doing database requests simultaneously.

On OS/2 and Windows NT, the DOS or Windows 3.x SQL Anywhere Client can be used to access a database server from a DOS, WIN-OS/2, or Windows 3.x session. See "[DOS or Windows client applications on OS/2](#)" and "[DOS or Windows 3.x clients on Windows 95 or NT](#)".

Description (QNX)

The SQL Anywhere Client for QNX is not a separate task. For performance reasons it is loaded as part of the application. It is maintained as an external file so that developers do not need to relink their applications with each new release of SQL Anywhere. There is a link from /bin/dbclient to dbclient.

Each new release of SQL Anywhere includes new dbclient files. End-users can install the new database software and run existing applications without relinking.

Overriding default QNX startup parameters

If you wish to override the default client startup parameters for the QNX client, you may create the SQLCLIENT environment variable. For example,

```
export SQLCLIENT="dbclient -p 4096"
```

causes the client to use 4096-byte buffers instead of the default 512-byte buffers.

SQLCLIENT may also be used to specify where to find the client file. For example:

```
export SQLCLIENT="//11/home/steve/dbclient.a -p 4096"
```

If no path is specified, the PATH environment variable is used to find the client file.

Switches

-b num-packets Set the maximum number of packets received at one time for multi-row fetches. The value must be in the range from 1 to 32, inclusive. The default setting is half of the available buffers.

-c Purge previous connections from this client computer. If a client machine is turned off or rebooted while connections exist to the database server, one or more connections may be left open. These connections may be closed manually on the server (see the chapter "[The Database Server Display](#)"), or they can be closed by starting the SQL Anywhere Client with this switch on the client machine.

-e Encrypt all packets transmitted by this client over the network. By default, packets are not encrypted. If you are concerned about the security of network packets, use the -e switch. Encryption does affect performance marginally.

Using the -e switch on the DBSRV50 command line will encrypt packets for all clients regardless of whether the -e switch is used on the client command line.

-ga Automatically shut down the client after the last connection to a server is closed.

-o filename Output all SQL Anywhere Client error messages and debugging messages (see -Z) to a file instead of the console. Normally, there are no messages.

-p packet-size Set the maximum size of communication packets. The default is 512 bytes. The minimum value is 200 bytes. If the specified packet size is larger than that of the database server, the server's packet size is used.

-q Operate quietly. Do not print messages.

-r Fetch 1 row per network request. By default, when the database server gets a simple fetch request, it will fill one network packet with several rows so that subsequent sequential fetches do not require network traffic. This is often referred to as **blocking** of fetches. This switch disables multiple-row fetching.

-s buffer_space Specify amount of space to use for network buffers in kilobytes. The default for the DOS DBCLIENT is 64 (you may want to reduce this number to reduce the footprint); for all other clients it is 100.

-ta seconds For Windows 3.x, Windows NT, and Windows 95 only. The client periodically scans the connection list and disconnects any connections associated with terminated applications. The scan period can be controlled using the -ta switch, and has a default value of 30 seconds. Setting the value to zero prevents scanning.

-tl seconds A liveness packet is sent periodically across a client/server TCP/IP or IPX communications protocol to confirm that a connection is intact. If the client runs for a liveness timeout period (default 2 mins) without detecting a liveness packet, the communication is severed. The client forgets about the address of the server and looks it up next time there is a connection for it dropping all current connections

to that server. DOS and QNX clients do not do liveness.

If no -tl switch is set, the liveness timeout is controlled by the setting on the server, which defaults to 120 seconds.

Liveness packets are sent at an interval of the (liveness timeout)/4.

-ts t1 [, t2] Specify the minimum and maximum times (in hundredths of a second) before the client reissues a request if there has been no response. The default value for t1 of 25/100ths of a second is usually adequate. The default value for t2 is 500, and the maximum is 3000.

-x list Use only the listed communications links (NamedPipes, IPX, TCPIP, NetBIOS, NetDG, Windows, QNX). The default is to try all communication links supported by the version of dbclient.

Named Pipes and Windows links are for local communications only.

By default, the SQL Anywhere Client tries to use a local Named Pipe or Windows link first (if applicable), followed by IPX, TCP/IP, NetBIOS, and then NetDG when locating a server. The SQL Anywhere Client starts slightly faster if unnecessary network drivers are not started.

Additional parameters may be provided, in the format

```
-x tcpip{PARAM1=value1;PARAM2=value2;...}  
-x "tcpip{PARAM1=value1;PARAM2=value2;...}"
```

The quotation marks are required for the QNX client and server only, and only if more than one parameter is supplied.

For a description of available parameters, see "[Network communications parameters](#)".

-Z Provides diagnostic information on communications links on startup. This is useful if -o is also specified.

The DOS and Windows 3.x SQL Anywhere Clients can be used to access an OS/2 or Windows 95 or NT database engine running on the same machine.

■ For more information see "[DOS or Windows client applications on OS/2](#)" and "[DOS or Windows 3.x clients on Windows 95 or NT](#)".

The DBWATCH server monitoring facility

{button See also,PI(`,`DBWATCH_sa')}

Syntax

`dbwatch [switches]`

Windows 3.x syntax

`dbwatchw [switches]`

QNX syntax

`dbwatch [switches]`

Switch	Description
<code>-c</code>	Supply database connection
<code>"keyword=value;parameters ..."</code>	

Description

Start the server monitoring facility.

The server monitoring facility displays the database server screen on a client machine. This is useful to check who is logged on to a database server that is running elsewhere on your network. You can also use it to display both the server and the client statistics on your local client screen. DBWATCH can also be used to disconnect users and to configure the database server.

For QNX, the SQL Anywhere Client must be running before you run `dbwatch`.

Switches

`-c "keyword=value; ..."` Specify connection parameters.

For a description of connection parameters, see the chapter "Connecting to a Database" in the *SQL Anywhere User's Guide*.

Network communications parameters

If you experience problems with client/server network communications, there are a number of command line parameters provided for both the client and the server to work around peculiarities of different TCP/IP and IPX implementations.

You supply the network communication parameters on the server or client command line as in the following example:

```
DBSRV50 -x tcpip{PARAM1=value1;PARAM2=value2;. . .},IPX
```

If there are spaces in a parameter, the network communication parameters must be enclosed in quotation marks to be parsed properly by the system command interpreter:

```
DBSRV50 -x "tcpip{PARAM1=value 1;PARAM2=value 2;...},IPX"  
dbsrv50 -x "tcpip{PARAM1=value1;PARAM2=value2;. . .}"
```

The quotation marks are required under QNX if more than one parameter is given, as QNX interprets the semicolon as a command separator. Quotation marks are also required if there are space characters in any of the parameters.

Binary parameters are turned on with any of YES, ON, and 1, and are turned off with any of NO, OFF, 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 or client command-line switch to invoke the configuration file.

The parameters currently available are as follows:

-
- BROADCAST parameter (synonym BCAST)
 - DOBROADCAST parameter
 - DLL parameter
 - EXTENDEDNAME parameter
 - HOST parameter (synonym IP)
 - MAXLANA parameter
 - MYIP parameter
 - RECEIVEBUFFERSIZE parameter (synonym RCVBUFFSZ)
 - REGISTERBINDERY parameter (synonym REGBIN)
 - SEARCHBINDERY parameter (synonym BINSEARCH)
 - SENDBUFFERSIZE parameter (synonym SNDBUFFSZ)
 - ServerPORT parameter
 - SESSIONS parameter
 - THREADS parameter
 - TIMEOUT parameter (synonym TO)
 - THREADSTATS parameter (synonym STATS)
 - WSAVERSION parameter

BROADCAST parameter (synonym BCAST)

Usage

TCP/IP

Description

BROADCAST specifies the special IP address used by your TCP/IP protocol implementation to identify a broadcast message. The most common broadcast IP address is 255.255.255.255, the default setting. Some TCP/IP implementations instead use a broadcast address consisting of the network IP address portion, with 255 as the remaining integers. For example, if the network portion of your IP address is 197, some TCP/IP implementations use 197.255.255.255 as the broadcast IP address. If your network portion is 192.023, the broadcast IP address would be 197.023.255.255.

Default

255.255.255.255

DOBROADCAST parameter

{button Example,JI(>Examples','TCPERM_2_')}

Usage

TCP/IP (all platforms), IPX (Windows 95 and NT only)

Description

With DOBROADCAST=YES, a broadcast is performed to search for a server if the server is not found in the bindery.

With DOBROADCAST=NO, 0, or OFF, no broadcast is performed to search for a database server. If you do set DOBROADCAST=NO, you must specify the server host with the HOST option.

Default

Yes

DLL parameter

{button Example,JI(>Examples',`TCP32DLL_3_')}

Usage

TCP/IP (Windows 95, Windows NT, OS/2)

Description

To support untested TCP/IP protocol stacks under OS/2 where the required networking interface functions are in DLLs that differ from IBM OS/2's TCP/IP protocol stack. If supplied, the client or server looks for its required functionality in the named DLLs.

Default

On OS/2, the defaults are TCP32DLL.DLL, SO32DLL.DLL.

On Windows and Windows NT, the default is WINSOCK.DLL.

EXTENDEDNAME parameter

{button Example,JI(>Examples',`TCPPRM_4_')}

Usage

IPX (platforms other than Windows 95 or NT)

Description

According to the Novell standard for legal SAP names, certain characters are not allowed. They are: \ / : ; , * ? + - . If you start a server named "sademo-1", the default behaviour is to strip out the - and try to start a server sademo1. By turning on ExtendedName, the name is left untouched.

Caution *Users should be wary of using this option as it is contrary to the SAP standard.*

Default

No.

HOST parameter (synonym IP)

{button Example,JI(>Examples',`TCPPRM_5_')}

Usage

TCP/IP (all platforms)

Description

HOST specifies additional machines outside the immediate network to be searched to avoid starting a server with a duplicate name or, in the case of the Client, to give the location of the server.

For TCP/IP, the *hostname* or a dot-separated IP address may be used. For IPX, 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 this addressing information during startup if the *-Z* switch is used.

You can use a semicolon-separated list of addresses to search for more than one machine.

Default

No additional machines.

MAXLANA parameter

{button Example `,JI(>Examples',`TCPPRM_6_')}`

Usage

NetBIOS and NetDG

Description

Each path through a NetBIOS or NetDG protocol stack is assigned a LAN Adapter number. By default, the server looks through all possible numbers up to 255. To speed up server startup, you can truncate the search for valid LAN adapters at a specified value using the MAXLANA parameter.

Default

255

MYIP parameter

{button Example,JI(>Examples',`TCPPRM_7_')}

Usage

TCP/IP

Description

MyIP informs the server of the IP-address(es) of the host machine on which the server is to listen for client packets.

If the keyword NONE is supplied as the IP number, no attempt is made to determine the addressing information. This option is intended primarily for clients on operating systems where this operation is expensive.

Under Windows 95 or NT this option can be used multiple times for machines with multiple IP addresses.

RECEIVEBUFFERSIZE parameter (synonym RCVBUFSZ)

{button Example,JI(>Examples',`TCPPRM_8_')}

Usage

TCP/IP (Windows 95 and NT, OS/2, and Netware)

Description

This option preallocates memory in the protocol stack for receiving TCP/IP packets destined for the SQL Anywhere client or server. Preallocation of memory inside the protocol stack may increase client/server performance for network-intensive applications.

Default

65536

REGISTERBINDERY parameter (synonym REGBIN)

Usage

IPX (Windows 95 and NT only)

Description

The database server will attempt to register its name with any active binderies on the network when loading the IPX link. To disable this name registration, set RegisterBindery to NO, FALSE or 0. If this registration is not performed, SQL Anywhere clients must be able to locate the database server over IPX by broadcasting packets.

Default

TRUE

SEARCHBINDERY parameter (synonym BINSEARCH)

Usage

IPX (Windows 95 and NT only)

Description

With SEARCHBINDERY=NO, 0, or OFF no search is made of a NetWare bindery for a database server.

Default

Yes

SENDBUFFERSIZE parameter (synonym SNDBUFSZ) **{button Examples,JI(`>Examples',`TCPPRM_11_')}**

Usage

TCP/IP (Windows 95 and NT, OS/2, and NetWare)

Description

This option preallocates memory in the protocol stack for sending TCP/IP packets. Under supported platforms, sending of packets is done asynchronously and preallocation of a memory block may allow the protocol stack and client/server to operate concurrently for a longer period of time, thus achieving better performance for network-intensive applications.

Default

65536

ServerPORT parameter

{button Example,JI(>Examples',`TCPPRM_12_')}

Usage

TCP/IP (all platforms)

Description

The SQL Anywhere database server is designated port number 1498 to use for TCP/IP communications. However, applications are not disallowed from using this reserved port, and this may result in an addressing collision between the database server and some other application.

In the case of the database server, the ServerPORT option designates the port number on which to communicate using TCP/IP.

In the case of the SQL Anywhere Client, the ServerPORT 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 named using the ServerPORT parameter to find the server.

The SQL Anywhere Client does not require a designated port number and merely uses one granted from the TCP/IP implementation.

Default

1498

SESSIONS parameter

{button Example,JI(>Examples',`TCPPRM_13_')}

Usage

NetBIOS

Description

Sets the maximum number of clients that can communicate with the server at one time through a single LAN adapter. The default setting is operating-system specific. The value is an integer, with maximum value 254.

Default

Operating system specific. On Windows NT, the default is 16.

THREADS parameter

{button Example,JI(>Examples',`TCPPRM_14_')}

Usage

IPX (OS/2, Windows 95 and NT), TCP/IP (OS/2, NetWare, Windows 95 and NT) NetBIOS (OS/2, Windows 95 and NT), NetDG (OS/2, Windows 95 and NT)

Description

THREADS specifies the number of threads for reading network communications. Integers from one to ten are allowed. It has been found that two threads produces good performance, but the option is provided as a performance parameter you can tune.

Default

2

TIMEOUT parameter (synonym TO)

{button Example,JI(>Examples',`TCPPRM_15_')}

Usage

TCP/IP (all platforms)

Description

TIMEOUT specifies the length of time, in seconds, to wait for a response when establishing TCP/IP communications. You may wish to try longer times if you are having trouble establishing TCP/IP communications.

Default

5 seconds.

THREADSTATS parameter (synonym STATS)

{button Example,JI(>Examples',`TCPPRM_16_')}

Usage

IPX (Windows 95 and NT only)

Description

This option allows a user to have the SQL Anywhere client or server create a file into which IPX thread statistics are written. Currently, the only statistic written to the file is the number of packets received by each executing IPX thread.

Default

NULL

WSAVERSION parameter

{button Example,JI(>Examples',`TCPPRM_17_')}

Usage

TCP/IP (Windows 3.x, 95, and NT), IPX (Windows 95 and NT only)

Description

The SQL Anywhere Client and server for Windows 3.x, 95, and NT require a version of the winsock dll of 1.1 or higher. This requirement can be relaxed to a lesser version of winsock if the same functionality as version 1.1 has been implemented by a vendor. The major version number appears in the high byte of the value, the minor version number appears in the low byte of the value.

Default

0x101 (version 1.1)

Example for DOBROADCAST parameter

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**.

```
dbclient -x tcpip{DOBROADCAST=NO;HOST=silver} sademo
```

On QNX, the options must be enclosed in quotation marks:

```
dbclient -x "tcpip{DOBROADCAST=NO;HOST=silver}" sademo
```

Example for DLL parameter

The following command starts a client with the protocol interface functions in ABC.DLL and XYZ.DLL:

```
dbclient -x tcpip{dll=abc.dll;dll=xyz.dll} sademo
```

Example for EXTENDEDNAME parameter

The following command starts a NetWare server with name sademo-1.

```
load DBSRV50.nlm -x ipx{ExtendedName=Yes} sademo-1
```

Example for HOST parameter (synonym IP)

The following command line instructs the client to look on the machines "kangaroo" and 197.75.209.222 to find a database server called sademo.:

```
dbclient -x tcpip{HOST=kangaroo;HOST=197.75.209.222} sademo
```

For QNX, quotation marks are required around the tcpip options:

```
dbclient -x "tcpip{HOST=kangaroo;HOST=197.75.209.222}" sademo
```

Example for MAXLANA parameter

The following command line looks only at LAN adapters with numbers less than 10 to identify active protocol stacks:

```
DBSRV50 -x netbios{MAXLANA=10} sademo
```


Example for MYIP parameter

```
DBSRV50 -x tcpip{MyIP=192.75.209.12} c:\sqlany50\sademo.db
```

The following command line instructs the client to make no attempt to determine addressing information.

```
dbclienw -x tcpip{MyIP=NONE} sademo
```

Example for RECEIVEBUFFERSIZE parameter (synonym RCVBUFSZ)

The following command line starts a server with a fixed preallocated receiving buffer:

```
dbsrv50 -x tcpip{receivebuffersize=16384} sademo.db
```

Examples for **SENDBUFFERSIZE** parameter (synonym **SNDBUFSZ**)

The following command line sets the buffer size for the server to 16384 bytes.

```
dbsrv50 -x tcpip{sendbuffersize=16384} c:\sqlany50\sademo.db
```

The following command line sets the buffer size for a client to 16384 bytes.

```
dbclient -x tcpip{sendbuffersize=16384} sademo
```

Example for ServerPORT parameter

- ◆ Start a SQL Anywhere network server:

```
dbsrv50 -x tcpip c:\sqlany50\sademo.db
```

Port number 1498 is now taken.

- ◆ Attempt to start another database server:

```
dbsrv50 -x tcpip c:\sqlany50\sademo2.db
```

This fails with an error "Unable to initialize communication links", as the port is currently allocated.

- ◆ Start another database server, assigning a different port number to it:

```
dbsrv50 -x tcpip{ServerPort=1499} c:\sqlany50\sademo2.db
```

This should succeed as long as 1499 is not a reserved port and no other application has allocated it.

- ◆ The following command line allows a client to communicate with servers on both ports:

```
dbclient -x tcpip{ServerPort=1499;ServerPort=1498} sademo
```

Example for SESSIONS parameter

The following statement starts a server with a database named sademo, allowing 200 NetBIOS connections.

```
DBSRV50 -x netbios{sessions200} sademo.db
```

Example for THREADS parameter

The following command starts a database server to use the IPX protocol only, using three threads.

```
DBSRV50 -x ipx{threads=3} c:\sqlany50\sademo.db
```

The following command starts a database client using the TCP/IP protocol only, using three threads.

```
dbclient -x tcpip{threads=3} sademo
```

Example for TIMEOUT parameter (synonym TO)

The following command starts a database client on a TCP/IP communications link only, with a timeout period of twenty seconds.

```
dbclient -x tcpip{TO=20} sademo
```

Example for THREADSTATS parameter (synonym STATS)

The following statements place the statistics in the file ipxstat.txt in the current directory.

```
DBSRV50 -x ipx{threadstats=ipxstat.txt} c:\sqlany50\sademo.db  
dbclient -x ipx{threadstats=ipxstat.txt} sademo
```


Example for WSAVERSION parameter

The following statement starts a database server even though the Winsock DLL is version 1.0.

```
DBSRV50 -x tcpip{wsaversion=0x100} c:\sqlany50\sademo.db
```

The following statement starts a SQL Anywhere client even though the Winsock DLL is version 1.0.

```
dbclient -x tcpip{wsaversion=0x100} sademo
```


The SQL Anywhere Client
The DBWATCH server monitoring facility

"The database server".

The SQL Anywhere Client
The database server

