CHAPTER 3

Connection and Communication Parameters

About this chapter

This chapter provides a reference for the parameters that establish and describe connections from client applications to a database.

Contents

Connection parameters

This section describes each of the connection parameters that can be included in connection strings or data sources.

Notes

- Connection parameters are case-insensitive.
- The Usage for each connection parameter describes the circumstances under which the parameter is to be used. Common usage entries include the following:
  - **Embedded databases** When Adaptive Server Anywhere is used as an embedded database, a personal server is started by the connection, and the database is loaded by the connection. When the application disconnects from the database, the database is unloaded and the server stops.
  - **Running local databases** This refers to the case where an Adaptive Server Anywhere personal server is already running, and the database is already loaded on the server.
  - **Network servers** When Adaptive Server Anywhere is used as a network server, the client application must locate the server on the network and connect to a database already loaded on to it.

Agent connection parameter

Function To specify a local or network connection.

Usage Anywhere

Values String. Must be **any** or **server**.

Default No value.

Description If you wish to ensure that you connect to a server on a network, and do not start a personal database server, specify **Agent=server** to ensure a connection to the appropriate agent.

Example

- The following entry in a file data source instructs the connection not to start a personal server:

```
...
agent=server
...
```

AutoStart connection parameter [Astart]

Function To prevent a local database server from being started if no connection is found.

Usage Anywhere.

Default Yes

Description By default, if no server is found during a connection attempt, and a database file is specified, then a database server is started on the same machine. You can turn this behavior off by setting the AutoStart parameter to OFF in the connection string.

Example

♦ The following data source fragment prevents a database server from being started if no network server is located:

```
[My Sample Database]
DatabaseFile=c:\asa6\asademo.db
Autostart=No
UserID=dba
ENG=network_server
```

AutoStop connection parameter [Astop]

Function To prevent a database from being unloaded as soon as there are no more open connections.

Usage Embedded databases.

Default Yes

Description By default, any server that is started from a connection string is stopped when there are no more connections to it. Also, any database that is loaded from a connection string is unloaded as soon as there are no more connections to it. This behavior is equivalent to **Autostop**=Yes.

If you supply **Autostop**=No, any database that you start in that connection is not unloaded when there are no more connections to it. As a consequence, the database server will not be shut down either.

The **AutoStop** parameter is used only if you are connecting to a database that is not currently running. It is ignored if the database is already loaded.

Example

♦ The following data source fragment prevents the database from being unloaded when the connection is dropped:

```
[Sample Embedded Database]
DatabaseFile=c:\asa6\asademo.db
Autostop=No
UserID=dba
```

CommAutoStop connection parameter [CAstop]

Function	To unload network communications ports as soon as there are no more open connections from the client machine.
Usage	Network server.
Default	No

Description

When the client library makes a connection over a network, it loads one or more **network ports** into memory.

By default, a network port that is started from a connection string is not unloaded when there are no more connections to it. This behavior is equivalent to **CommAutostop**=No.

If you supply **CommAutostop**=Yes, any network ports you start from that connection are unloaded when there are no more connections using them.

Example

♦ The following data source fragment instructs the client library to unload the network ports after the connection is dropped:

```
[Sample Connection]
ServerName=network_server
CommAutostop=Yes
UserID=dba
PWD=sql
CommLinks=tcpip
```

CommBufferSize connection parameter [CBSize]

Function	To set the maximum size of communication packets, in bytes.
Usage	Network server only.
Values	Integer
Default	1000

Description

The **CommBufferSize** parameter specifies the size of communications packets, in bytes. The minimum value of **CommBufferSize** is 280, and the maximum is 16000. If the specified packet size is larger than that of the database server, the server's packet size is used.

The maximum size of a packet on a network is set by the protocol stack. If you set **CommBufferSize** to be larger than that permitted by your network, the largest buffers are broken up by the network software. You should set the buffer size to be somewhat smaller than that allowed by your network, because the network software may add information to each buffer before sending it over the network.

A larger packet size improves performance for multi-row fetches and fetches of larger rows. As each connection has its own pool of buffers, a larget buffer size increases the memory usage. The application side uses four default-sized buffers per connection, and on the server side, there are two.

This corresponds to the SQL Anywhere Version 5 *dbclient* –p command-line switch.

Examples

♦ The following data source fragment sets the buffer size to 400 bytes:

```
...
CommBuffSize=400
...
```

CommBufferSpace connection parameter [CBSpace]

Function To specify the amount of space to allocate on startup for network buffers, in kilobytes.

Usage Network servers only

Values Integer

Default 100.

Description Specify amount of space to allocate on startup for network buffers, in kilobytes. The default is 100.

The value is a global setting, for all connections.

Examples

♦ The following data source fragment instructs the network library to allocate 200 Kb for network buffers on startup.

```
...
CBSpace=200
...
```

CommLinks connection parameter [Links]

Function To specify network communications links.

Usage Connections to the network server only.

Default Use all communications links (network protocols) supported on the current operating system.

See also "Network communications parameters" on page 54
"Client/Server Communications" on page 685 of the book *Adaptive Server Anywhere User's Guide*.

Description

If no CommLinks parameter is specified in a connection string, the network library is not started and no search is made for a server other than on the current machine. This behavior is equivalent to CommLinks = None.

If a CommLinks parameter is supplied, the named communication links are started and used when searching for a database server. The CommLinks parameter is required for connections to a network server.

Available values of the CommLinks parameter are as follows:

- **NONE** Start no communications links
- **TCP/IP** Start the TCP/IP communications link. TCP/IP is supported on all operating systems.
- **IPX** Start the TCP/IP communications link. The IPX protocol is supported for Windows and NetWare clients.
- **NetBIOS** Start the NetBIOS communications link. NetBIOS is supported on Windows operating systems.
- **ALL** Start all available communications links.

You may wish to use a specific protocol, as opposed to **ALL**, for the following reasons:

- The network library starts slightly faster if unnecessary network links are not started.
- If you wish to tune the broadcast behavior of a particular protocol by providing additional network communications parameters, you must specify the link explicitly.

Additional network communications parameters may be provided for each link, to tune the broadcast behavior of the link.

The **CommLinks** parameter corresponds to the database server `-x` command-line switch. The default behavior of the network server is equivalent to -x ALL

Examples

- The following data source fragment starts the TCP/IP protocol only:

  ```
  CommLinks=tcpip
  ```

- The following data source fragment (which is held on one line in a data source) starts the TCP/IP and IPX protocols, searching for the host **kangaroo** in addition to servers on the immediate TCP/IP network:

  ```
  CommLinks=ipx, tcpip(HOST=kangaroo)
  ```

ConnectionName connection parameter [CON]

Function Names a connection, to make switching to it easier in multi-connection applications.

Usage Not available for ODBC.

Default No connection name.

Description An optional parameter, providing a name for the particular connection you are making. You may leave this unspecified unless you are going to establish more than one connection, and switch between them.

The Connection Name is not the same as the data source name.

Examples

♦ Connect, naming the connection FirstCon:

```
CON=First Con
```

DatabaseFile connection parameter [DBF]

Function The database file to which you want to connect.

Usage Embedded databases

Default There is no default setting.

Description To load and connect to a specific database file.

♦ If a database is loaded with a name that is the same as the DatabaseFile parameter, but without the *.db* extension, the connection is made to that database instead.

♦ If the filename does not include an extension, a file of name *.db* is looked for.

♦ The path of the file is relative to the working directory of the database server. If you start the server from the command prompt, the working directory is the directory that you are in when entering the command. If you start the server from an icon or shortcut, it is the working directory that the icon or shortcut specifies.

Example

♦ To load and connect to the sample database, installed in directory *c:\asa6*, use the following DBF parameter:

```
DBF=c:\asa6\asademo.db
```

DatabaseName connection parameter [DBN]

Function Identifies a loaded database to which a connection needs to be made.

Usage Running local databases or network servers.

Default There is no default setting.

Description Whenever a database is started on a server, it is assigned a database name. The default database name is the name of the database file with the extension and path removed.

Examples

♦ Connect to a database named Kitchener

```
DBN=Kitchener
```

DatabaseSwitches connection parameter [DBS]

Function To provide database-specific switches when starting a database.

Usage Connecting to a running server when the database is not loaded.

Default No switches

See also "The database server" on page 12
"StartLine connection parameter" on page 52

Description You should supply **DatabaseSwitches** only if you are connecting to a database that is not currently running. When the server starts the database specified by **DatabaseFile**, the server uses the supplied **DatabaseSwitches** as command line options to determine startup options for the database.

Only *database* switches can be supplied using this parameter. Server switches must be supplied using the START connection parameter.

Examples

♦ The following command, entered all on one line, connects to the default database server, loads the database file *\sa60\asademo.db* (DBF parameter), names it as **my_db** (DBS parameter) and connects to the database of that name (DBN parameter).

```
dbcollat -c
"uid=dba;pwd=sql;dbf=\sa60\asademo.db;dbn=my_db;
dbs=-n my_db" e:\temp\temp.col
```

DataSourceName connection parameter [DSN]

Function Tells the ODBC driver manager or Embedded SQL library where to look in the *odbc.ini* file or registry to find ODBC data source information.

Usage Anywhere

Default There is no default data source name.

See also "FileDataSourceName connection parameter" on page 49

Description It is common practice for ODBC applications to send only a data source name to ODBC. The ODBC driver manager and ODBC driver locate the data source, which contains the remainder of the connection parameters.

In Adaptive Server Anywhere, Embedded SQL applications can also use ODBC data sources to store connection parameters.

Examples

♦ The following parameter uses a data source name:

```
DSN=Dynamo Demo
```

Debug connection parameter [DBG]

Function To provide diagnostic information on communications links on startup.

Usage Network server only.

Default No diagnostic information.

See also "Logfile connection parameter" on page 51

Description If you are having trouble establishing a connection to a network server, set the Debug connection parameter to **Yes** and the Logfile parameter to a log file name. Diagnostic information is then placed in the log file.

Examples

♦ The following data source fragment says to use the Debug switch, with output to a file named *error.log*.

```
...
DBG=Yes
Log=ERROR.LOG
...
```

DisableMultiRowFetch connection parameter [DMRF]

Function To turn off multi-row fetches across the network.

Usage Network server only.

Default Yes

Description

By default, when the database server gets a simple fetch request, it fills 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.

Examples

♦ The following data source fragment requires no blocking of fetches:

```
...
DMRF=Yes
...
```

EngineName connection parameter [ENG]

Function

Synonym for **ServerName**. The name of a running database server to which you want to connect.

Usage

Network servers or running personal servers.

Default

The default local database server.

Description

EngineName is not needed if you wish to connect to a local database server and only one server is running.

You need to supply an **EngineName** only if more than one database server is running, or you wish to connect to a network server.

In the Sybase Central and Interactive SQL Connect dialog box, and in the ODBC Administrator, this is the **Server Name** field

Examples

♦ Connect to a server named **Guelph**:

```
ENG=Guelph
```

EncryptedPassword connection parameter [ENP]

Function

To provide a password, stored in an encrypted fashion in a data source.

Usage

Anywhere

Default

None

Description

Data sources are stored on disk as a file or in the registry. Storing passwords on disk may present a security problem. For this reason, when you enter a password into a data source, it is stored in an encrypted form.

If both **Password** and **EncryptedPassword** are specified, Password takes precedence.

Encryption connection parameter [ENC]

Function To encrypt packets transmitted from the client machine over the network.

Usage Network server only.

Values Boolean

Default No encryption.

Description You can use this parameter if you are concerned about the security of network packets. Encryption does affect performance marginally.

This parameter corresponds to the SQL Anywhere Version 5 *dbclient* `-e` command-line switch.

Using the `-e` switch on the *dbsrv6* command line encrypts packets for all clients regardless of whether the Encryption parameter is used at the client.

Examples

♦ The following connection parameter instructs the client to encrypt packets:

```
ENC=Yes
```

FileDataSourceName connection parameter [FileDSN]

Function The **FileDataSourceName** parameter tells the client library that there is an ODBC file data source holding information about the database to which you want to connect.

Both ODBC and Embedded SQL applications can use File data sources

Usage Anywhere

Default There is no default name.

See also "DataSourceName connection parameter" on page 46

Description File data sources hold the same information as ODBC data sources stored in the registry. File data sources can be easily distributed to end users, so that connection information does not have to be reconstructed on each machine.

Examples

♦ The following is a data source description held in a file data source:

```
[Sample File Data Source]
ENG = asademo
DBA = dba
PWD = sql
```

Integrated connection parameter [INT]

Function To use the integrated login facility.

Usage Anywhere

Default No

See also "LOGIN_MODE option" on page 161

Description The **Integrated** parameter has the following settings:

- **Yes** An integrated login is attempted. If the connection attempt fails and the LOGIN_MODE option is set to Mixed, a standard login is attempted.
- **No** This is the default setting. No integrated login is attempted.

For a client application to use an integrated login, the server must be running with the LOGIN_MODE database option set to Mixed or Integrated.

Examples

- The following data source fragment uses an integrated login:

```
INT=yes
```

LivenessTimeout connection parameter [LTO]

Function To control the termination of connections when they are no longer intact.

Usage Network server only, and only on TCP/IP and IPX communications protocols.

Values Integer

Default If no **LivenessTimeout** value is set, the liveness timeout is controlled by the setting on the server, which defaults to 120 seconds.

Description A **liveness packet** is sent periodically across a client/server TCP/IP or IPX communications protocol to confirm that a connection is intact. If the client runs for the liveness timeout period without detecting a liveness packet, the communication is severed.

Liveness packets are sent at an interval of one quarter of the **LivenessTimeout** value.

When the communication is severed, the client machine forgets the address of the server. It looks the address up next time there is a connection to the server from that machine, dropping all current connections to that server.

Examples

- The following sets a Liveness timeout value of 60 seconds

```
LTO=60
```

Logfile connection parameter [LOG]

Function To send client error messages and debugging messages to a file.

Usage Network server only.

Default No log file.

See also "Debug connection parameter" on page 47

Description If you want to save client error messages and debugging messages in a file, use the **Logfile** parameter.

If the file name includes a path, it is relative to the current working directory of the client application.

Examples

♦ The following data source fragment says to use the Debug switch, with output to a file named *error.log*.

```
...
DBG=Yes
Logfile=ERROR.LOG
...
```

Password connection parameter [PWD]

Function To provide a password for the connection.

Usage Anywhere

Default No password provided.

See also "EncryptedPassword connection parameter" on page 48

Description Every user of a database has a password. The password must be supplied for the user to be allowed to connect to the database.

The password parameter is not encrypted. If you are storing passwords in a data source, you should use the **EncryptedPassword** parameter. Sybase Central and the Adaptive Server Anywhere ODBC configuration tool both use encrypted parameters.

If both **Password** and **EncryptedPassword** are specified, Password takes precedence.

Examples

♦ The following connection string fragment supplies the user ID **DBA** and password **SQL**.

```
uid=dba;pwd=SQL
```

ServerName connection parameter [ENG]

Synonym for "EngineName connection parameter" on page 48.

StartLine connection parameter [START]

Function To start a database server running from an application.

Usage Embedded databases.

Default No StartLine parameter

Description You should supply a StartLine parameter only if you are connecting to a database server that is not currently running. The StartLine parameter is a command line to start a personal database server.

☞ For a detailed description of available command line switches, see "The database server" on page 12.

Examples

♦ The following data source fragment starts a personal database server with a cache of 8 Mb.

```
StartLine=dbeng6 -c 8M asademo.db
```

Unconditional connection parameter [UNC]

Function To stop a server using *dbstop* even when there are connections to the server.

Usage Anywhere

Default No

See also "The DBSTOP command-line utility" on page 103

Description The *dbstop* command-line utility shuts down a database server. If you specify **Unconditional=Yes** in the connection string, the server is shut down even if there are active connections. If Unconditional is not set to **Yes**, then the server is shut down only if there are no active connections.

Examples

♦ The following command line shuts down the server unconditionally:

```
dbstop -c "uid=dba;pwd=sql;eng=server-name;unc=yes"
```

Userid connection parameter [UID]

Function	The user ID with which you log on to the database.
Usage	Anywhere
Default	None
Description	You must always supply a user ID when connecting to a database

Examples

♦ The following connection string fragment supplies the user ID **DBA** and password **SQL**:

```
uid=dba;pwd=SQL
```

Network communications parameters

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

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

```
dbsrv6 -x tcpip(PARM1=value1;PARM2=value2;. . .),IPX
```

From the client side, the communications parameters are entered as the CommLinks communication parameter:

```
CommLinks=tcpip(PARM1=value1;PARM2=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:

```
dbsrv6 -x "tcpip(PARM1=value 1;PARM2=value 2;...),IPX"
```

```
CommLinks="tcpip(PARM1=value 1;PARM2=value 2;...),IPX"
```

The quotation marks are required under UNIX if more than one parameter is given, because UNIX interprets the semicolon as a command separator.

Boolean parameters are turned on with YES, ON, or 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 [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

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. In this case, you must specify the server host with the HOST option.

For IPX only, you can also supply an address for the DOBROADCAST argument. The address serves as a mask for the HOST parameter, and allows you to specify a non-zero network number. This is intended for use when broadcasting over a router. The network number is assigned by the network administrator.

In IPX, a node address consists of six numbers (up to 255 each) and the network address consists of four digits, separated by colons.

Default Yes

Example

♦ The following command starts a client without broadcasting to search for a database server. Instead, the server is looked for only on the computer named **silver**.

```
dbclient -x tcpip(DOBROADCAST=NO;HOST=silver)
asademo
```

♦ On UNIX, the options must be enclosed in quotation marks:

```
dbclient -x "tcpip(DOBROADCAST=NO;HOST=silver)"
asademo
```

♦ On an IPX network, the following parameter specifies a broadcast over network 2:

```
-x ipx(dobroadcast=255:255:255:255:255:255/0:0:0:2)
```

DLL parameter

Usage

TCP/IP (Windows 95, Windows NT)

Description

To support untested TCP/IP protocol stacks where the required networking interface functions are in DLLs that differ from the default protocol stack. The client or server looks for its required functionality in the named DLLs.

Default

♦ On Windows and Windows NT, the default is *winsock.dll*.

Example

♦ The following command starts a server with the protocol interface functions in *abc.dll* and *xyz.dll:*

```
dbsrv6 -x tcpip(dll=abc.dll;dll=xyz.dll) asademo
```

EXTENDEDNAME parameter

Usage

IPX (platforms other than Windows 95 or NT)

Description

According to the Novell standard for legal SAP names, the following characters are not allowed:

```
\ / : ; , * ? + -
```

If you start a server named "asademo-1", the default behavior is to strip out the - and try to start a server asademo1. By turning on ExtendedName, the name is left untouched.

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

Default

No.

Example

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

```
load DBSRV6.nlm -x ipx(ExtendedName=Yes) asademo-1
```

HOST parameter [IP]

Usage

TCP/IP (all platforms)

Description

HOST specifies additional machines outside the immediate network to be searched by the client library. On the server, the search is carried out to avoid starting a server with a duplicate name.

For TCP/IP, the *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. Also, you can append a port number to an IP address, using a colon as separator.

IP is a synonym for HOST.

Default No additional machines.

Example

- The following data source fragment instructs the client to look on the machines "kangaroo" and 197.75.209.222 (port 2369) to find a database server called **asademo**:

```
...
ENG=asademo
CommLinks=tcpip(HOST=kangaroo;HOST=197.75.209.222:23
69)
...
```

- For UNIX, quotation marks are required around the tcpip options:

```
dbclient -x
"tcpip(HOST=kangaroo;HOST=197.75.209.222)" asademo
```

MAXLANA parameter

Usage NetBIOS

Description Each path through a NetBIOS 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

Example

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

```
dbsrv6 -x netbios(MAXLANA=10) asademo
```

MYIP parameter

Usage TCP/IP

Description

The MyIP parameter is provided for machines with more than one network adapter.

Each adapter has an IP address. By default, the database server uses the first network card it finds. If you wish your database server to use more than one network card, specify the address of each card in the MyIP parameter.

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 Windows NT, this option can be used multiple times for machines with multiple IP addresses.

You can optionally append a port number to the IP address, separated by a colon.

Example

- ♦ The following command line (entered all on one line) instructs the server to use two network cards, one with a specified port number.

```
dbsrv6 -x
tcpip(MyIP=192.75.209.12:2367,192.75.209.32)
c:\asa6\asademo.db
```

- ♦ The following data source fragment instructs the client to make no attempt to determine addressing information.

```
...
CommLinks= tcpip(MyIP=NONE)
...
```

REGISTERBINDERY parameter [REGBIN]

Usage

IPX (Windows 95 and NT only)

Description

The database server attempts 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. In this case, the client library must be able to locate the database server over IPX by broadcasting packets.

Default

TRUE

SEARCHBINDERY parameter [BINSEARCH]

Usage

IPX (Windows 95 and NT only)

Description

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

Default Yes

SERVERPORT parameter [PORT]

Usage TCP/IP (all platforms)

Description The Internet Assigned Numbers Authority has assigned the Adaptive Server Anywhere database server port number 2638 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 another application.

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

In a data source, 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 that is specified on the **ServerPort** parameter to find the server.

Default 2638

Example

1. Start a network database server:

   ```
   dbsrv6 -x tcpip -n server1
   ```

   Port number 2638 is now taken.

2. Attempt to start another database server:

   ```
   dbsrv6 -x tcpip -n server2
   ```

   This fails with an error Unable to Initialize Communication Links, because the port is currently allocated.

3. Start another database server, assigning a different port number to it:

   ```
   dbsrv6 -x tcpip(ServerPort=2639) -n server2
   ```

   This should succeed as long as 2639 is not a reserved port, and no other application has allocated it.

SESSIONS parameter

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.

Example

♦ The following statement starts a server with a database named **asademo**, allowing 200 NetBIOS connections.

```
dbsrv6 -x netbios(sessions=200) asademo.db
```

TDS parameter

Usage TCP/IP, NamedPipes

Description To disallow TDS connections to a database server, set TDS to NO. If you want to ensure that only encrypted connections are made to your server, these port options are the only way to disallow TDS connections.

Default YES

Example

♦ The following command starts a database server using the TCP/IP protocol, but disallowing connections from Open Client or jConnect applications.

```
dbsrv6 -x tcpip(TDS=NO) ...
```

THREADS parameter

Usage IPX (Windows 95 and Windows NT)

Description THREADS specifies the number of threads that are used 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 that you can tune.

Default 2

Example

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

```
dbsrv6 -x ipx(threads=3) c:\asa6\asademo.db
```

TIMEOUT parameter [TO]

Usage TCP/IP, IPX (all platforms)

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

Default

5 seconds.

Example

- The following data source fragment starts a TCP/IP communications link only, with a timeout period of twenty seconds.

  ```
  ...
  CommLinks=tcpip(TO=20)
  ...
  ```

THREADSTATS parameter [STATS]

Usage

IPX (Windows 95 and Windows NT only)

Description

This option creates 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

Example

- The following statement places the statistics in the file *ipxstat.txt* in the current directory.

  ```
  dbsrv6 -x ipx(threadstats=ipxstat.txt)
  c:\asa6\asademo.db
  ```

WSAVERSION parameter [WSAVER]

Usage

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

Description

The database server for 95, and NT requires 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

- The following command (to be entered all on one line) starts a database server even though the Winsock DLL is version 1.0.

  ```
  dbsrv6 -x tcpip(wsaversion=0x100) c:\asa6\asademo.db
  ```

- The following data source fragment instructs a client to start even though the Winsock DLL is version 1.0.

  ```
  ...
  CommLinks=tcpip(wsaversion=0x100)
  ENG=asademo
  ...
  ```