

Chapter 3: Server and Client Configuration and Troubleshooting

With previous versions of SQL Server, as soon as it was installed, the DBA sat down to configure it. SQL Server 7.0's capability to configure itself removes much of that need. However, you may find some things useful in solving problems specific to your installation. This chapter covers the startup of SQL Server. Then, it looks at some of the configuration options that you may want to explore. Note that the most useful configuration options relate to how SQL Server performs. This chapter also covers SQL Server's interaction with Windows NT. The server-side discussion ends with a section on server troubleshooting. The discussion then turns to the client side, and covers client setup and configuration. Finally, because one of the most annoying tasks you will face as a SQL Server DBA involves dealing with connectivity problems, this chapter shows you a few tricks for dealing with this troublesome part of SQL Server.

Starting, Pausing, and Stopping SQL Server

When you installed SQL Server, you most likely checked the option to have SQL Server and the SQL Agent start when Windows NT starts. On some occasions, however, you may want to stop and start the server without taking down NT. You can do this with Enterprise Manager, or with the SQL Service Manager. I almost always use the Service Manager because it comes up quickly, and its traffic light icon is easily accessible from my Icon Tray. But there are other ways of stopping and starting the server, including the following:

- The Services applet in Control Panel
- From the command prompt

Starting SQL Server

You can start SQL Server from the command prompt in two different ways:

- `NET START mssqlserver`
- `sqlservr`

`NET START` starts SQL Server as a service just as Enterprise Manager, SQL Service Manager, or Control Panel do. When SQL Server is running as a service, you can log off NT and the server continues to run. If you start it by just typing `sqlservr` at the command prompt, however, it is not run as a service. This means that you must shut it down (by pressing Ctrl+C) before you log off of NT. Note that you shouldn't minimize the command window. When you start SQL Server this way, it runs as a foreground application. If you minimize the window, NT pages all the SQL Server memory out. You can specify some options on the command line; some of the most useful ones are discussed later in this chapter.

You can also start SQL Server from DMO with the `Start` method of a server object. The `Start` method has four parameters, as shown here.

Parameter	Description
StartMode	When set to true, an attempt is made to connect on successful start. When set to false, no attempt is made to connect after a successful start.
Server	This parameter is optional and is a string that specifies the SQL Server name. You can also provide the server name in the Name property of the SQL Server object.
Logon	This parameter is optional, and is a string specifying the SQL Server logon that will be used to connect after the server is started. (StartMode is true.)
Password	This parameter is optional and is a string specifying the SQL Server password that will be used to connect after the server is started. (StartMode is true.)

Pausing SQL Server

You can pause SQL Server with Enterprise Manager, SQL Service Manager, the Services applet in Control Panel, the command line, or DMO. You can pause the server only when it is running as a service. When you pause SQL Server, no new connections are allowed, but existing connections are allowed to complete their work. You should pause the SQL Server and broadcast a "Server going down in x minutes" message when you plan to take the server down. In Enterprise Manager, SQL Service Manager, and Control Panel, it is just a matter of clicking the Pause button. From the command line, just type the following:

```
NET PAUSE MSSQLSERVER
```

In DMO, use the Pause method of a SQL Server object. This method has no arguments.

You can restart a paused server with the Start/Continue button in SQL Service Manager, the Continue button in the Services applet, or the Continue choice in Enterprise Manager. To restart from the command line, use the following:

```
NET CONTINUE MSSQLSERVER
```

In DMO, use the Continue method of a SQL Server object. The method has no arguments.

General Tip - There doesn't seem to be any way to pause and resume SQL Server with Transact-SQL.

Stopping SQL Server

When you stop the SQL Server, SQL Server disallows new connections and allows existing connections to finish their work. Then, it takes a checkpoint in every database, and shuts down. It's possible to stop SQL Server with Enterprise Manager, Service Manager, the Services applet, the command line, DMO, and Transact-SQL. In the graphical tools, just click Stop. To stop SQL Server from the command line, use the following:

```
NET STOP MSSQLSERVER
```

General Tip - Note that if you don't start SQL Server as a service, you must use Ctrl+C to stop it.

In DMO, just use the Stop method of the SQL Server object. This method has no arguments.

In Transact-SQL you can issue the following shutdown command:

```
SHUTDOWN [WITH NOWAIT]
```

If you don't specify `WITH NOWAIT`, SQL Server stops in the same fashion described previously. If you specify `WITH NOWAIT`, existing connections are stopped, their transactions are rolled back, and no checkpoints are taken. You should use the `NOWAIT` option only in an emergency and when you are certain that the server needs to stop immediately.

Startup Switches

Some startup switches are used when SQL Server starts. The default values are stored in the Registry. Any switches you provide on the command line override the Registry values. The following three switches are required.

Switch	Description
<code>-dmaster_file_path</code>	The fully qualified name of the master database file (for example, C:\MSSQL7\DATA\MASTER.MDF)
<code>-lmaster_log_path</code>	The fully qualified name of the transaction log for the master database (for example, C:\MSSQL7\DATA\MASTER.LDF)
<code>-eerror_log_path</code>	The fully qualified name of the SQL Server Error log file (for example, C:\MSSQL7\LOG\ERRORLOG)

You can change the default values for startup switches in several ways. In Enterprise Manager, highlight the server, right-click, and choose Properties. On the General tab, click the Startup Parameters button. Or you can edit the Registry, although I don't recommend doing that. The startup parameters are stored in the following:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSSQLServer\MSSQLServer\Parameters
```

The parameters are named `SQLArg0`, `SQLArg1`, `SQLArg2`, and so on. The order doesn't matter because the switch is part of the parameter.

You can also use the following useful switches at startup.

Switch	Description
-c	Starts SQL Server so that it does not run as a service. This shortens startup time, but there is no other advantage. When you start SQL Server this way, you cannot log off of NT without stopping SQL Server first.
-f	Starts SQL Server with a Minimal configuration. This is useful for recovering from a failed configuration.
-m	Starts SQL Server in single-user mode, and enables the Allow Updates to System Tables option (discussed later). This option is most often used when restoring master (see Chapter 6, "Backup and Recovery," for more information).
-pprecision_level	Specifies the maximum level of precision used for decimal and numeric datatypes. The actual precision allowed is 38; but by default, SQL Server only allows 28. If you need the additional 10 digits, use this startup switch.
-sregistry_key	Starts SQL Server with a set of parameters stored under the key named in registry_key. You can only use this option from the command line, but it enables you to have many previously defined startup options.
/Ttrace_flag#	Specifies that the server should start with a trace flag.
-x	Turns off the recording of CPU and cache-hit ratio statistics. This can improve performance, but you will not be able to see these values in Performance Monitor. It may also interfere with Performance Condition alerts (see Chapter 8, "Jobs and Alerts," for more information on alerts).

When I'm going to use a switch once—for example, putting the server in single-user mode so that I

can restore the master database—I set the switch with Control Panel. To do this, use the Services applet. Highlight the MSSQLServer service. You will see the display shown in [Figure 3.1](#).

Figure 3.1

Specifying startup switches in Control Panel.

You can just type the `-m` switch (or another switch name) in the Startup Parameters box at the bottom of the screen. It applies for a single startup of SQL Server. I think this is much easier than going through Enterprise Manager to change the parameters, and then having to remember to change them back.

General Tip - There's no way to specify startup switches when you start the server with DMO. It uses the default settings from the Registry.

Setting the Polling Interval

Both Enterprise Manager and SQL Service Manager routinely poll the SQL Server to determine whether it (as well as SQL Server Agent, MSDTC, and Full-Text Search) is still running. If you look closely at the Service Manager icon at the bottom right of your screen, you will periodically see a little red "blip" when this polling happens. You can control the polling interval. In Enterprise Manager, highlight a server and choose Tools, Options. Choose the target service and specify the polling interval in seconds. If you uncheck the Poll Server box, Enterprise Manager will not poll. The only downside of this is that you won't see little red, green, and yellow indicators of server state in the console tree. The upside of not polling is that there's less demand on the server.

To specify SQL Service Manager's polling interval, open Service Manager and select the service. Then right-click the icon in the System Tray and choose Options. Enter the polling interval in seconds.

General Tip - There's no relationship between Service Manager's polling interval and Enterprise Manager's. They're both client programs operating independently.

Configuring SQL Server

SQL Server is largely self-configuring, and in many cases it works correctly "right out of the box." You should understand some options, however, in case your circumstances require changes from the default behavior. You can change all configuration options with Transact-SQL, many of them with Enterprise Manager, and all of them with SQL DMO.

This section discusses generic options, covering options that relate to specific tasks as the task is discussed. Configuration options are classified on two different dimensions:

- **Standard versus advanced.** Standard options are the most commonly changed, whereas

advanced options are infrequently changed.

- **Static versus dynamic.** Static options do not take effect until the server is restarted, whereas the dynamic options take effect immediately.

In SQL Server 7.0, the advanced options are visible by default. If you turn off the Show Advanced Options, you can't modify any of the advanced options using Transact-SQL. In the following discussions on options, these categories are in parentheses at the end of the description of the option:

- **(S, S).** Standard option, requires restart
- **(S, D).** Standard option, takes effect immediately
- **(A, S).** Advanced option, requires restart
- **(A, D).** Advanced option, takes effect immediately

Use these category descriptions to understand when options take effect and when you are required to restart SQL for them to take effect.

Miscellaneous Options

Some options apply to the server as a whole and don't fit into a neat category such as "memory." I've grouped them here.

- **Allow Updates.** When this option is set to true, direct updates to the system tables are permitted. In most cases, you do not want this option to be turned on. In those rare cases in which you must modify system tables, I recommended that you do it with the server in single-user mode. Note that if this setting is true when you create a stored procedure, the stored procedure can modify system tables even after the option is turned off. (S, D)

Danger - Updating system tables directly can cause your server to fail to start or to behave erratically.

- **Default Language.** This option specifies the ID (taken from syslanguages) for the default language, and controls default formats for displaying dates and gives meaning to the date parts.

If you have a localized version of SQL Server (French, German, Spanish, Japanese), the default language specifies the language in which error messages display. If you have a Japanese SQL Server, and the default language is Japanese, for example, SQL Server error messages display in Japanese. If the default language on a localized SQL Server is not the local language, the messages displays in U.S. English. If you have a French SQL Server and specify Italian as the default language, for example, all error messages will be in U.S. English. (S, D)

- **Language in Cache.** This option specifies the number of languages that can be held in cache. The default is 3. (S, S)

- **Network Packet Size (B).** The network packet size specifies the network packet size in bytes. By default, SQL Server uses 4096 bytes as the packet size. If you routinely send large amounts of data across the network, you may benefit from a larger packet size. Conversely, if most of your transmissions are small, you might want to set this to 512 bytes, which is sufficient for many small data transmissions. If you have multiple protocols, set the network packet size to that which is appropriate for the most commonly used protocol. Note that a client application can specify a packet size that differs from the one specified here. (A, D)
- **Recovery Interval (Min).** This option specifies the number of minutes it will take SQL Server to recover all databases on the system at startup. It influences the frequency with which SQL Server takes checkpoints. By default, it is 0, which means that SQL Server configures it appropriately. In general, you should leave this option alone unless you find that checkpoints are being taken too frequently (see Chapter 6 for a discussion of checkpoints). In that case, you may want to experiment with increasing the value in small increments. (A, D)
- **Show Advanced Options.** This option determines whether you can see or change the advanced options when you run `sp_configure`. By default, it is set so that advanced options are visible.

Note that if you run the Upgrade Wizard (see Chapter 2, "Installation and Upgrade," for more information on this) and ask to migrate SQL Server 6.5 configuration options, you will find that this option is set to off. (S, D)

Memory-Related Options

The options in this section all relate to how SQL Server uses memory. In SQL Server 7.0, memory is largely self-configuring. If you have a machine running only SQL Server, you probably won't have to modify these settings. If you are running other applications, however, you may want to adjust some of these settings.

By default, SQL Server monitors its environment. When the operating system has less than 5MB (\pm 200KB) of free memory, SQL Server gives memory it would have normally kept for its available pool back to the operating system. When there is more than 5MB free, SQL Server takes the memory back.

The memory options available are as follows:

- **Extended Memory Size (MB).** This option is primarily for forward compatibility. It will be available in future versions of SQL Server running on a future version of Windows NT that has support for 64-bit addressing. Some hardware vendors may support this capability under Windows NT 4.0. (A, S)
- **Locks.** In previous versions of SQL Server, the amount of space available for locks was fixed, and it was necessary to tinker with this option. In SQL Server 7.0, SQL Server allocates 2% of the available memory for locks. When additional memory is needed, SQL Server allocates it unless doing so would reduce the amount of free memory for the OS to less than 5MB. If this happens, SQL Server issues a message that you are out of locks. You should change the number of locks only if you receive this message. The default for this option is 0, which means that it is self-configuring. If you explicitly specify this value, the value must be between 5,000 and 2,147,483,647. (A, S)

- **Min Server Memory (MB), Max Server Memory (MB).** These two options work together. If you set Min Server Memory and Max Server Memory to the same value, SQL Server uses a fixed portion of memory. If SQL Server is the only application running on the machine, leave these settings alone. If other applications are running, setting a Max Server Memory influences how quickly they start up, because there is normally a delay between when the competing application begins and when SQL Server frees memory for it. If you want to guarantee that SQL Server has a set minimum amount of memory no matter what, specify a Min Server Memory. The smallest value you can specify for Max Server Memory is 4MB; the highest value is limited by the resources available on your system. If you specify Min Server Memory, and the server is involved with replication, the value must be greater than or equal to 16.

If your server is periodically idle for long periods of time, you may want to set Min Server Memory so that the memory is immediately available when a query does come. If you have other critical applications on the machine that might also have idle applications, you might want to set Max Server Memory so that memory is available to those applications when they become active. (A, S)

General Tip - If you have installed Full-Text Indexing and are running the MS Search Service, you must specify a Max Server Memory so that there is enough room for the Search Service to run. In this case, you must configure NT's virtual memory so that there is virtual memory (virtual memory for SQL Server and virtual memory for any other concurrently running applications equal to 1.5 times the machine's physical memory for the search service):

NT virtual memory "e 1.5 x physical memory

You also need to specify a Max Server Memory if you are running SQL Server on the same machine as Exchange Server.

-
- **Set Working Set Size.** This option reserves physical memory for SQL Server. If Windows NT must page, it must swap out other processes. By default, this option is off, and idle SQL Server memory can be swapped out. It is best to leave it off. (A, S)
 - **Open Objects.** An open object is any table, stored procedure, view, and so on currently in use. Note that no matter how many users there are for an object, it is one open object. This option is self-configuring by default, and in most cases you should leave it that way. If you receive repeated instances of the following message in your Error log, you may want to consider setting it to some value:

Warning - OPEN OBJECTS parameter may be too low; attempt was made to free up descriptors in localdes(). Run `sp_configure` to increase parameter value.

I would start with 10,000 and keep increasing it until the messages go away. Note - this

reserves memory for open objects that cannot be reused for other purposes. (A, S)

Look, But Don't Touch Options

All the options listed in this section enable you to look at information. You should never modify the sort order ID or the Unicode information. The only successful way to change these is to reinstall SQL Server. Likewise, there is nothing to be gained from modifying the User Connections option.

Following are the "look, but don't touch" options:

- **Default SortOrder ID, Unicode Comparison Style, Unicode Locale ID.** These options show you the sort order ID, Unicode comparison style, and Unicode locale that you chose when you installed SQL Server. These options are numbers; to find out what they mean, issue the following command:

```
sp_helpsort
```

- **User Connections.** In previous versions of SQL Server, it was necessary to specify the maximum number of concurrent connections to SQL Server. In this release, SQL Server manages the number of connections dynamically and there is no reason to specify a value for this option. If you set it too low, users may be denied access to SQL Server. (A, S)

General Tip - Don't confuse connection with user. Each user may have many simultaneous connections to the server. You can't use the User Connections option to limit concurrent users.

Options That Affect Applications

Some of the SQL Server options can change the behavior of applications. The following list identifies these options:

- **Two Digit Year Cutoff.** Since time immemorial, SQL Server has observed the rule that, if dates are entered without a century, values less than 50 are assumed to be in 20xx and values greater than 50 are assumed to be 19xx. Therefore, 1/14/44 is interpreted as 1/14/2044, and 1/14/56 is interpreted as 1956. However, the OLE automation interface, including the capability to maintain data in Enterprise Manager (and ADO as well), has a cutoff of 30. With this rule, 1/14/44 becomes 1/14/1944. If at the time you are reading this you still have any applications that are not providing the century as part of the date, you probably have bigger things to worry about than this option. Your best option is to leave it alone and make sure the century is always supplied.

Danger - It is very easy to enter incorrect date data in Enterprise Manager's Open Table, Return All Rows dialog box because the dates display in mm/dd/yy format without century. Enterprise Manager, being a graphical tool going through OLE automation, uses 30 as the pivot point.

- **Nested Triggers.** When you are working with triggers, you can set up your applications so that a trigger on Table A performs some operation on Table B. With nested triggers, if there is a trigger on Table B, the trigger fires when Table A's trigger modifies Table B. This makes triggers much smaller and much more modular. It is unusual, although not impossible, for there to be situations in which this behavior is undesirable. In such a case, you turn nested triggers off. In most cases, you do not want to do so. Nested triggers has been SQL Server's default since 4.2; be very careful changing this if you have legacy applications that have been upgraded. This is a serverwide option. Don't confuse it with "recursive triggers," which are requested at a database level. (S, D)
- **User Options.** Many session-level (connection-level) settings control the behavior of applications. These options are specified with the `SET` statement. Some of these can be specified at the server level by using the appropriate bit masks for the User Options configuration option. The following options can be set at the server level.

SET Option<	Bit Mask
DISABLE_DEF_CNST_CHK	1
IMPLICIT_TRANSACTIONS	2
CURSOR_CLOSE_ON_COMMIT	4
ANSI_WARNINGS	8
ANSI_PADDING	16
ANSI_NULLS	32
ARITHABORT	64
ARITHIGNORE	128
QUOTED_IDENTIFIER	256
NOCOUNT	512
ANSI_NULL_DFLT_ON	1024
ANSI_NULL_DFLT_OFF	2048

See the discussion of the `SET` statement in your SQL Server documentation for the meaning of these settings. If you are not comfortable working with bit masks, you can set these on or off through the Enterprise Manager graphical user interface. (S, D)

Changing Configuration Options with Transact-SQL

To change a configuration option in Transact-SQL, use the following command:

```
sp_configure [option [, value]
```

If you just issue the command `sp_configure` in Query Analyzer, a report appears showing all the configuration options available together with their minimum, maximum, current, and next values. This report looks like the one shown in here.

Name	Minimum	Maximum	config_value	run_value
Affinity Mask	0	2147483647	0	0
Allow Updates	0	1	0	0
Cost Threshold for Parallelism	0	32767	5	5
Cursor Threshold	-1	2147483647	-1	-1
Default Language	0	9999	0	0
Default SortOrder ID	0	255	51	51
Extended Memory Size (MB)	0	2147483647	0	0

The Name column in this report is the name of the particular option, and what you use if you want to change that option. The column headed `config_value` is the value that will be used the next time SQL Server is started. The column headed `run_value` is the value SQL Server is currently using. A minimum and maximum of 0 and 1, respectively, indicates that it is a true-false option. If there is a minimum and maximum, but the `run_value` is 0, it's most likely one of the options that SQL Server manages dynamically.

If you issue the `sp_configure` command with just the name of an option, you will see the line of the preceding report for only that option. For example:

```
sp_configure 'allow updates'
```

This shows you the settings for just the Allow Updates option. You don't need to spell the option name out in full; any unique left substring works. Therefore, for Allow Updates, you can use 'al', 'allow', and so on.

To change the value of an option, add a value to the command such as this:

```
sp_configure 'allow', 1
```

After you have done this, you must use the `reconfigure` (or, in some cases, the `reconfigure with override`) command to make the option take effect. If the option is dynamic, the change is made immediately. If it's a static option, you must stop the server and restart it before the change takes effect.

Changing Configuration Options with Enterprise Manager

Not all configuration options can be changed in Enterprise Manager. To change options using

Enterprise Manager, right-click the server in the console pane and choose Properties. The SQL Server Properties dialog box displays, which includes several various tabs and options. Two screens in this dialog box can be confusing if you've never worked in them before. The first is the Memory Configuration screen shown in [Figure 3.2](#).

Figure 3.2

Memory Configuration screen.

The check box labeled Reserve Physical Memory for SQL Server is the same as the Set Working Set Size option discussed previously.

Another confusing screen is the Connection Options screen shown in [Figure 3.3](#).

Figure 3.3

Connection Options screen.

The check boxes for the Default Connection Options correspond to the bit masks for the User Options configuration option described previously.

In Enterprise Manager, the Apply and OK buttons cause the dynamic options to take effect. In addition, Enterprise Manager asks you whether you want to stop and restart the server when you modify a static option. If you are ready to stop the server, choose Yes. If you want to wait until a later time to stop the server (remembering that the option won't take effect until you do so), choose No.

Changing Configuration Options with DMO

The SQL Server object has a configuration object, which contains the ConfigValues collection. Each ConfigValue object has the following useful properties.

Property	Description
CurrentValue	Long
RunningValue	Long
MaximumValue	Long
MinimumValue	Long
Name	String
DynamicReconfigure	True if it's a dynamic option; false if it's a static option

To change a configuration option, you set the value for the appropriate option (using the names specified) as follows:

```
Set oConfigValue = oSQLServer.Configuration.ConfigValues("Show Advanced _Optio
```

When you are ready to apply the changes, you use the ReconfigureCurrentValue method of the configuration object. If the configuration option requires `with override` as does the Allow Updates option, use the ReconfigureWithOverride method instead. If any of the changes are to static options,

you must use the Stop and Start methods of the SQL Server object to make the changes take effect.

Behind the Scenes

Information about configuration option settings is contained in two different tables, `sysconfigures` and `syscurconfigs`:

- **sysconfigures.** This table contains the options SQL Server started with as well as any changes made to dynamic options since the server was started.
- **syscurconfigs.** This table contains values used the next time SQL Server is started. Note that `syscurconfigs` is a dynamic table built only when it is referenced.

Setting Up and Configuring Remote and Linked Servers

One SQL Server can interact with other SQL Servers as well as many other heterogeneous data sources. This can be done in two ways: *remote* servers and *linked* servers. When you set up a remote server, another SQL Server can execute stored procedures that reside in the remote server. Remote servers must be SQL Servers. This architecture is far more limited than the linked server architecture, which permits users to issue select, insert, update, and delete queries against the linked server. The linked server need not be a SQL Server; in fact, using linked servers, it is completely possible to join data from tables residing in SQL Server, Oracle, and Excel. In most cases, you will want to use linked servers rather than remote servers, because the linked servers are more powerful. I am going to cover remote servers as well as linked servers, however, because you may have legacy systems using the older architecture and because replication is implemented with remote servers.

Remote and Linked Server Configuration Options

Four configuration options deal with remote and linked servers. In most cases, the defaults are what you want, but you must make sure that the settings have not been changed from the default when you set up remote servers.

Options That Apply to Both Remote and Linked Servers

The following options apply to both remote and linked servers:

- **Remote Login Timeout (S).** This configuration option affects both remote and linked servers. It specifies how long to wait when connecting to a remote server. If you try to connect to a remote server, and that server is down, for example, you might wait forever. The default is 0, which means that you want to wait indefinitely. You may want to set this to an interval, such as 5 or 10 seconds. (S, D).

General Tip - The Remote Login Timeout option cannot be set in Enterprise Manager; you must use Transact-SQL or DMO to set it.

- **Remote Query Timeout (S).** This option specifies the number of seconds that SQL Server

should wait after issuing a request to a linked server or a remote server before timing out if no results have been returned. The default is 0, which means that you wait indefinitely. (S, D)

Options That Apply Only to Remote Servers

The following options apply only to remote servers:

- **Remote Access.** The default is 1, which allows other servers to access this server remotely. If you are planning to use remote servers, you should leave this option alone. If you set it to 0, other servers can't access the server remotely. (S, S)
- **Remote Proc Trans.** If this option is set to 1, all operations using remote servers will be distributed transactions managed by MSDTC. Because a lot of overhead is required to manage distributed transactions, you should leave this option set to 0 (the default) and explicitly request a distributed transaction only when you need one. (S, D)

General Tip - You can read more about distributed transactions in the Books OnLine.

Remote Servers

Remote servers are set up in pairs. First, each server must be aware of the other's existence. Then, you must set up logins for the remote servers. These are effectively mappings between the login ID used on one of the servers to the login ID used on the other server.

Setting Up Remote Servers with Transact-SQL

This procedure must be done on both servers. First, use the following command:

```
sp_addlinkedserver 'server', 'SQL Server'
```

Suppose, for example, that you have two servers, one named ServerA and one named ServerB. On ServerA, issue the following command:

```
sp_addlinkedserver 'ServerB', 'SQL Server'
```

On ServerB, issue the following command:

```
sp_addlinkedserver 'ServerA', 'SQL Server'
```

Now, you must map the logins. The way you do this depends on whether the logins are SQL Server Authentication logins or NT Authentication logins. (See Chapter 9, "Security," for details on the different methods of authentication.)

Mapping SQL Server Authentication Logins

If the user has the same login name and password on both SQL Servers, it is not necessary to perform this step. If Tom has a login on ServerA and a login on ServerB, for example, the mapping is

automatic. If the names differ, however, you must define the mappings. Assume for this example that ServerA has a login named Theresa, and you want that login to map to a login named RemoteUsers on ServerB. Use the command `sp_addremotelogin`, as follows:

```
sp_addremotelogin 'remoteserver', 'login', 'remote_name'
```

For example, on ServerB:

```
sp_addremotelogin 'ServerA', 'RemoteUsers', 'Theresa'
```

After this has been done, Theresa, running on ServerA, can execute any stored procedures that RemoteUsers have been given permission to on ServerB. If you want bidirectional logins (that is, from ServerB to ServerA), use a similar process on ServerA.

The final step is to specify whether a password is needed when Theresa wants to run a procedure on ServerB. In most cases, it is easiest if you do not request that the password be checked, because remote stored procedures are likely to be executed from local stored procedures and it is difficult to interact with a user for a password. You specify this with the command `sp_remotoption`, as follows:

```
sp_remotoption 'remoteserver', 'loginame',  
'remotename', trusted, {true | false}
```

To continue the preceding example, the command should look like this:

```
sp_remotoption 'ServerA', 'RemoteUsers', 'Theresa',  
trusted, true
```

Mapping NT Authentication Logins

Mapping NT Authentication logins is done differently. Use the command `sp_addlinkedssrvlogin`, as follows:

```
sp_addlinkedssrvlogin 'remoteserver', {true | false},  
'locallogin', 'remotename', 'rmtpassword'
```

The second argument must be false for an NT Authentication login. Assume that PINE\SHARON has an NT Authentication login on ServerA, and that you want to map this login to RemoteUsers on ServerB with no password. The command, issued on ServerB, should look like this:

```
sp_addlinkedssrvlogin 'ServerA', false, RemoteUsers,  
'PINE\SHARON', NULL
```

Setting Up Remote Servers with Enterprise Manager

It's much easier to set up remote servers with Enterprise Manager than with Transact-SQL, but it's not possible to map NT Authentication logins in Enterprise Manager. In the Security folder in the console pane, right-click on Remote Servers and choose New Remote Server. The screen shown in [Figure 3.4](#) displays.

Figure 3.4

Remote Server Setup screen.

It is important to check the box labeled RPC near the top of the screen. You can map all users to a single login on the remote machine or you can map them individually as shown in [Figure 3.3](#). To skip checking passwords on the remote server, don't check the box in the Check Password column.

Setting Up Remote Servers with DMO

You work with the RemoteServers collection, the RemoteServer object, the RemoteLogins collection, and the RemoteLogins object to set up remote servers with DMO.

To add a remote server, create a RemoteServer object. Specify its Name property and add it to the RemoteServers collection of a connected SQLServer object.

General Tip -You will see some other properties in the documentation and in the list that appears in Visual Basic for the object, but they all have to do with replication. You do not need to worry about setting these for a simple remote server.

To set up remote logins, create a RemoteLogin object. Set the RemoteName, LocalName, and Trusted properties. (Set these to true if password is not to be checked; otherwise set them to false.) Add the object to the RemoteLogins collection of a RemoteServer object.

General Tip -The RemoteLogin object seems to only handle SQL Server authentication logins.

Linked Servers

Linked servers are a new feature in SQL Server 7.0. They allow direct queries against many different data sources, including Oracle, Excel, Access, and ODBC data sources. After you have set up a linked server, you can query it by just including the linked server name as part of the table name, as follows:

```
SELECT * from MySpreadsheet...Sheet1$
SELECT * from ORACLEDB..Scott.Emp
```

Setting Up Linked Servers with Transact-SQL

Use the `sp_addlinkedserver` command to set up a linked server with Transact-SQL. This is the same command I described under remote servers, but there is a lot more information that you must provide when linking a heterogeneous server, as follows:

```
sp_addlinkedserver 'server', 'product_name',
provider_name', 'data_source', 'location',
'provider_string', 'catalog'
```

Server is a name you assign to the server, such as MyLinkedServer. The following table lists possibilities for the other parameters:

Remote Data	product name	provider name	Data Source	Location	provider string	Catalog
SQLServer	SQLServer (default)	SQLOLEDB (optional)	Network Name of SQLServer (optional)	N/A	N/A	Database Name (optional)
Oracle	Doesen't matter	MSDAORA	SQL*Net alias for Oracle database	N/A	N/A	N/A
Access/Jet	Doesen't matter	Microsoft. Jet.OLEDB. 4.0	Full path name of Jet database file	N/A	N/A	N/A
ODBC data source	Doesen't matter	MSDASQL	System DSN of ODBC data source*	N/A	ODBC connection string*	N/A
File system	Doesen't matter	MSIDXS	Indexing Service catalog name	N/A	N/A	N/A
Microsoft Excel spreadsheet	Doesn't matter	Microsoft. Jet.OLEDB. 4.0	Full name of Excel file	N/A	Excel 5.0	N/A
Text file***	Doesn't matter	Microsoft. Jet.OLEDB. 4.0	Full name of text file	N/A	Text	N/A

* You establish this alias with Oracle client tools. Oracle connectivity must be installed on the machine to do this.

** Provide either the ODBC DSN or the full ODBC connection string, not both.

*** You must have a SCHEMA.INI file in the same directory as the text file if you want to query text files. Preparation of this file is described in the Jet documentation that accompanies Microsoft Access.

If you want to set up an Excel spreadsheet as a linked server, the command should look like the following:

```
sp_addlinkedserver 'ExcelSpreadsheet', '',
'Microsoft.Jet.OLEDB.4.0',
'\EXCEL:\MySpreadsheets\LastMonth.xls', '', 'Excel 5.0'
```

Remember that the path names are from the SQL Server's point of view; you must use share names accessible to SQL Server. Do not use your mapped drive letters. If the files do not reside on the SQL Server computer, SQL Server must be logging in with a domain account and that account must have permission to access the files.

When you set up linked servers, it's also necessary to configure logins for those servers. Use the command `sp_addlinkedsevrlogin`, as follows:

```
sp_addlinkedsevrlogin 'servername',{true | false},
  'SQL Server loginname',
  'linked server login name',
  'linked server password']
```

When you specify true for the second argument, the SQL Server login name is used to connect to the remote server. You can only use this if the SQL Server login is a SQL Server Authentication login. You may specify null for the SQL Server login name. In this case, all SQL Server logins will log in to the linked server with the linked server login name and password. If you provide a value, it must be a SQL Server login or a Windows NT login that has been granted access to the SQL Server. Assume, for example, that the Excel spreadsheet allows an Admin login with no password, and you want all SQL Server users to be able to log in to this linked server. The command to set this up is as follows:

```
sp_addlinkedsevrlogin 'Excelspreadsheet', FALSE, NULL, 'Admin',NULL
```

Setting Up Linked Servers with Enterprise Manager

It's very simple to set up linked servers with Enterprise Manager. In the Security folder, right-click Linked Servers and choose New. The screen shown in [Figure 3.5](#) displays.

Figure 3.5

Setting up a linked server.

As annoying as it is, all linked server names must be in uppercase letters. Select the OLE DB Provider from the drop-down list, and provide the other elements using the parameters table in the preceding section. Leave Collation Compatible unchecked unless you are absolutely certain that the remote data source has the same character set as SQL Server. The RPC and RPC Out boxes only make sense for SQL Servers; they specify whether remote procedure calls from and to the server are allowed.

It's also possible to set up the logins with Enterprise Manager. You do so on the Security tab illustrated in

Figure 3.6

Figure 3.6 Specifying linked server logins in Enterprise Manager.

If you choose No Security Context Will Be Used, no logins at all are used. This makes sense for things such as Excel, if the spreadsheet isn't password protected, and text files, which have no security other than file-level permissions. If you choose They Will Be Impersonated, the SQL Server login is used to authenticate the user at the linked server. This only works for SQL Server Authentication logins. Checking this makes sense if the remote server is a SQL Server with the same logins as the server on which the linked server is defined. If you choose They Will Be Mapped To, all SQL Server users will connect to the linked server with the specified username and password. You should choose They Are Not Allowed to Access if you want to specify the logins individually in the grid at the bottom of the screen.

In the grid, provide the SQL Server login. In most cases, you won't choose Impersonate, because doing so requires that the logins be the same on both servers.

Refer back to [Figure 3.6](#). With this setup, Tigger logs in to the linked server as a user named Scott with a password of Tiger. Any other users will log in as a user named Admin with a password of secret.

Setting Up Linked Servers with DMO

Each SQL Server object has a LinkedServers collection that contains LinkedServer objects. Each LinkedServer object has a LinkedServerLogins collection that contains LinkedServerLogin objects.

To add a new LinkedServer, instantiate a LinkedServer object and provide its Name property. Use the same properties you used for setting up linked servers with Transact-SQL to set up properties for DMO. When the properties of the LinkedServer object are complete, add it to the LinkedServers collection of a connected SQLServer object.

To define linked server logins, instantiate a LinkedServerLogin object. This object has four properties.

Property	Description
LocalLogin	SQL Server login
Impersonate	True if SQL Server login should be passed to the linked server; otherwise false
RemoteLogin	Username for linked server
RemotePassword	Password for linked server

Then, add the LinkedServerLogin object to the appropriate LinkedServerLogins collection.

Windows NT and SQL Server

When you install SQL Server, it modifies a couple of Windows NT settings. As long as the machine is dedicated to SQL Server, you should leave these settings alone. If the machine will run other applications, however, you may find it useful to change them.

SQL Server modifies the following Windows NT settings:

- **Application Performance.** SQL Server sets this so that foreground and background tasks are equally responsive. (The NT default gives priority to foreground tasks.)
- **Maximize Throughput.** SQL Server sets this to maximize throughput for network operations. (The NT default maximizes throughput for file-sharing operations.)

To change the setting for Application Performance, right-click My Computer and choose Properties. The System Properties dialog box shown in [Figure 3.7](#) displays. The Application Performance option is on the Performance tab.

Figure 3.7

NT's Application Performance setting.

If you want to give more priority to foreground applications, move the slider to the right.

To modify the Maximize Throughput option, open the Network applet in Control Panel. On the Services tab, highlight Server and choose Properties. The screen shown in [Figure 3.8](#) displays.

Figure 3.8

Server Properties dialog box in NT.

You can change the settings as necessary to benefit other applications on the machine. It's also a good idea to uncheck Make Browser Broadcasts to LAN Manager 2.x Clients if you don't have any of those (and you probably don't!). This broadcast could announce the name of your server to hackers.

Troubleshooting Server Problems

When SQL Server fails, it's often difficult to know where to start in terms of fixing it. This section examines how you can interpret SQL Server's Error log. Then, this section discusses something called trace flags, which you may be able to use to get information about what's happening. This section also shows you how to get the diagnostics you need when you call Microsoft Technical Support and how you can turn on SQL Server's "flight recorder." Most server troubleshooting is done "by guess and by golly," and there aren't many tools to help you. In the main, Enterprise Manager is useless; in fact, because it consumes so many resources, it may not even start when your server is having problems. You need to be comfortable with Query Analyzer, as well as operating system tools, such as the NT Event Viewer, and the always, reliable Notepad as well as the DOS prompt!

Error Log

SQL Server maintains a log of error and information messages. By default, this log is kept in the \MSSQL7\LOG folder, although you can change this by changing the location on the SQL Server startup (see the "Startup Switches" section earlier in this chapter). When you start SQL Server, SQL Server creates a new log and archives the old one. When you look in the folder where the Error logs are kept, you see the following files (assuming, for this example, that you have started SQL Server at least seven times):

File	Description
ERRORLOG	Current error log
ERRORLOG.1	Most recent error log
ERRORLOG.2	Second most recent log

ERRORLOG.3	Third most recent log
ERRORLOG.4	Fourth most recent log
ERRORLOG.5	Fifth most recent log
ERRORLOG.6	Oldest log

If your server crashed, and you have restarted it, you may find the reason for the crash in ERRORLOG.1; if you haven't been able to restart it, look at ERRORLOG. You can view Error logs in Notepad. (It's also possible to view Error logs in Enterprise Manager; the logs are visible in the MANAGEMENT folder.) Many of the messages in the Error log are also visible in NT's Application Event log.

When you start SQL Server, it records the success and/or failure of its attempts to recover databases in the Error log. Normally, you see a series of messages and discover that all databases have been successfully recovered. Occasionally, you see a message that a database has been marked *suspect*. This usually means that one of the files that make up the database has been damaged or removed, and you will need to correct that problem. If you see errors that are Exception Access Violations, there is probably a bug in SQL Server and you will need to contact Microsoft Support. Other messages appear from time to time; for example, anytime that someone performs a non-logged operation, a message to that effect appears in the Error log. Following is one message that users often find troubling, but you shouldn't worry about it:

```
Failed to obtain TransactionDispenserInterface: XACT_E_TMNOT AVAILABLE.
```

This message means that the MSDTC service isn't running. If you're not performing distributed transactions, you don't need MSDTC.

If your server crashes, you should start it in single-user mode and check the Error log to make sure that all the databases were successfully recovered and that there are no problems before you make the server available for general use.

Errors issued by applications in your organization may appear in the Event log as well. (Chapter 8 discusses this in more detail.)

Trace Flags

Trace flags are used to temporarily change the characteristics of SQL Server, or to turn off some particular behavior. Some trace flags are available for backward compatibility with previous versions of SQL Server. They can also provide additional information about locks, deadlocks, and query plans. Microsoft sometimes provides trace flags as a way of working around a bug. Trace flags normally write information to the Error log; there's a trace flag that you can use to override this and send the information back to a client. A few of the more useful trace flags are discussed here; you can get details about others from Books OnLine.

Trace Flags That Provide Detailed Information About Locks and Query Plans

Chapter 10, "Performance Tuning," discusses query plans in more detail, but the trace flags listed here can give you additional information when you are studying performance or contention problems:

Flag	Meaning
325	Prints information about the cost of using a non-clustered index or a sort to process an <code>ORDER BY</code> clause
326	Prints information about the estimated and actual cost of sorts
330	Enables full output when using the <code>SET SHOWPLAN</code> option, which gives detailed information about joins
1204	Returns the type of locks participating in the deadlock and the current command affected
1205	Returns more detailed information about the command being executed at the time of a deadlock

Managing Tape Compression

It's often desirable to compress data that is written to tape so that it takes up less space. By default, if a tape drive supports hardware compression, backups will be written in a compressed format. You can use the following trace flag to disable this compression if you want to exchange tapes with other sites or use other drives that don't support compression.

Flag	Meaning
3205	Disable hardware compression

Managing Trace Flags

As mentioned earlier, many trace flags send output to the Error log. You may want the output to come directly to you. Two trace flags control where the output is sent.

Flag	Meaning
3604	Sends trace output to the client. Used only when setting trace flags with <code>DBCC TRACEON</code> and <code>DBCC TRACEOFF</code> .
3605	Sends trace output to the Error log. (If you start SQL Server from the command prompt, the output also appears onscreen.)

Suppressing Unneeded Messages

The following trace flag that suppresses unneeded messages is an important flag, and you may want to include it on the SQL Server startup. Any time a statement that does not return results is executed in a stored procedure, SQL Server sends a Done in Proc message back to the client. For example, the following statement sends such a message:

```
IF @@error <> 0
```

Flag	Meaning
3640	Eliminates the sending of DONE_IN_PROC messages to the client for each statement in a stored procedure. This is similar to the session setting of SET NOCOUNT; but when set as a trace flag, every client session is handled this way.

The client has absolutely no interest in the Done in Proc message. It is a short message and can create bottlenecks across WANs. It also impairs the performance of stored procedures when they are run through the Job Scheduler.

Using Trace Flags

You can use trace flags in two ways:

- As a SQL Server startup parameter
- With DBCC TRACEON/OFF

To specify a trace flag as a SQL Server startup parameter, use **-T** or **/T** followed by the trace flag, as follows:

```
/T3640
```

Be sure to use a capital T; lowercase is for Microsoft internal trace flags.

To set a trace flag with DBCC, use the following command:

```
DBCC TRACEON (trace# [,...n])
```

If you are studying a problem and want the output to come back to the client (Query Analyzer, for example) instead of being written to the Error log, you should issue the following commands:

```
DBCC TRACEON (3604)
```

```
DBCC TRACEON (1205)
```

To turn off a trace flag, use the following command:

```
DBCC TRACEOFF (trace# [,...n])
```

To determine whether a trace flag is on or off, use the following command:

```
DBCC TRACESTATUS (trace# [,...n])
```

The resulting display shows the trace flag and a 0 if it is off, or a 1 if it is on.

Diagnostics

When you encounter problems with SQL Server, you need information that will help you solve the problem. If you have to contact Microsoft, they will ask you about your server configuration and want details as well. You have two ways of getting this information:

- The `sqldiag` utility
- SQL Server's "flight recorder"

You can also use these tools to collect information whether or not you are having problems. The flight recorder can help you isolate problem queries, and the output from `sqldiag` can provide a record of configuration settings, and other items as detailed in the following sections.

`sqldiag`

The `sqldiag` utility comes with SQL Server and gives you information whether or not SQL Server is running. When SQL Server is running, the report includes the following:

- Text of all Error logs
- Registry information
- DLL version information
- Output from the following:

- `sp_configure`
- `sp_who`
- `sp_lock`
- `sp_helpdb`
- `xp_msver`
- `sp_helpextendedproc`
- `sysprocesses`

- Input buffer SPIDs/deadlock information
- Microsoft Diagnostics Report for the server, including the following:
 - n Contents of <SERVERNAME>.TXT file
 - n Operating system version report
 - n System report
 - n Processor list
 - n Video display report
 - n Hard drive report
 - n Memory report
 - n Services report
 - n Drivers report
 - n IRQ and port report
 - n DMA and memory report
 - n Environment report
 - n Network report
- The last 100 queries and exceptions (if you have turned on the recording of these)

If SQL Server is not running, information about SPIDs, configuration information, and the output of the various stored procedures is not included. To run the diagnostics program, issue the following command at a DOS prompt:

```
sqldiag [{-U login_ID} [-P password] | [-E]] [-O output_file]
```

You must run sqldiag on the server; you cannot run it from a client workstation. You can specify a login ID and password, or use the **-E** switch to specify a trusted connection. By default, the output files are named SQLDIAG.TXT and SQLDIAG.TRC (if the query recorder is running) and are placed in the MSSQL7\LOG folder. You may specify a different name and location.

The Flight Recorder

It's possible to have SQL Server save the last 100 queries. If you do this, they are included in the output from sqldiag. The sqldiag utility produces a file that can be read by SQL Profiler. You can also have this file produced at any time (without running sqldiag) after the flight recorder has been

enabled. The queries are stored in a wraparound list; when query 101 comes along, it overwrites query 1 in the list. To turn on the flight recorder, issue the following command:

```
xp_trace_setqueryhistory 1
```

After you have issued the command, `xp_trace_setqueryhistory` stored procedure is automatically executed each time SQL Server starts. To turn off the recorder, issue the following command:

```
xp_trace_setqueryhistory 0
```

You do not have to run `sqldiag` to find the most recent queries. Instead, you can issue the following command:

```
xp_trace_flushqueryhistory 'filename'
```

With this command, you can specify a full path as part of the filename. After the file has been created, you can view it using SQL Profiler. (See Chapter 10 for more information on SQL Profiler.)

General Tip -I could not get the flight recorder to work until I stopped the server and restarted it. The commands appeared to work, but no query history file was created. The documentation did not mention the need to stop the server and restart it after issuing the `xp_trace_setqueryhistory` command.

After you have enabled the query history, SQL Server saves a trace automatically whenever there is a SQL Server–issued error with severity level 17 or higher. Note that if an application raises an error of level 17 or higher, the trace is not written. The trace is written to a file named `BLACKBOX.TRC` and saved to the `\MSSQL7\LOG` folder.

Client Configuration

Thus far, this chapter has discussed the server configuration. Now it's time to take a look at the client side.

Client/Server Architecture

SQL Server is a client/server database management system. That means that client processes, which are often running on remote machines, must communicate with SQL Server over the network, as shown in [Figure 3.9](#).

Figure 3.9

Client/server architecture.

Client processes communicate with SQL Server in the following ways:

- **Open Database Connectivity (ODBC).** This is a native SQL Server driver used by many applications and developer tools including Visual Basic and Visual C++. The *Data Access*

Objects (DAO), *Remote Data Objects* (RDO), and *Microsoft foundation classes* (MFC) all use ODBC.

- **OLE DB.** This is also a native SQL Server provider based on the *Component Object Model* (COM) and the *Distributed Component Object Model* (DCOM). It is less widely supported today, but support for it is expected to grow. It can be used from Visual Basic and Visual C++ as well as Visual Interdev. The *ActiveX Data Objects* (ADO) use OLE DB.
- **DB-Library (dblib).** This is the original *application programmer's interface* (API) to SQL Server. It is supported for backward compatibility, but has not been enhanced with SQL Server 7.0 features.

Regardless of which of these is being used, the communication process is identical, as [Figure 3.10](#) shows.

Figure 3.10

Client/server communications.

What's important is the network library. Chapter 2 discussed choosing network libraries for your server. It is important that the library each client is using match one on which the SQL Server is listening. Otherwise, there will be no communication.

The client and server network libraries are DLLs. The relationship between the netlib and the actual DLL is shown in the following table.

Netlib	32-bit DLL	16-bit DLL
multiprotocol	DBMSRPCN.DLL	DBMSRPC3.DLL
named pipes	DBNMPNTW.DLL	DBNMP3.DLL
TCP/IP Sockets	DBMSSOCN.DLL	DBMSSOC3.DLL
IPX/SPX	DBMSSPXN.DLL	DBMSSPX3.DLL
AppleTalk	DBMSADSN.DLL	n/a
Banyan VINES	DBMSVINN.DLL	DBMSVIN3.DLL
Shared Memory	DBMSSHRN.DLL	n/a
DECnet	DBMSDECN.DLL	n/a

Installing Client Connectivity

You install client connectivity with the same Setup program that you used to install SQL Server. You must choose a Custom install. The client workstation must have the Client Connectivity libraries installed. You can install the management tools (Enterprise Manager, Query Analyzer, and so on) and Books OnLine, but you do not need to. You might want to put these tools on developer workstations. Only C and C++ programmers need the development tools. Developers might also benefit from the

code Samples. You specify what you want to install on the Components Selection screen shown in [Figure 3.11](#).

Figure 3.11

Client Components Selection screen.

When you install Client Connectivity on Windows NT or Windows 9x, by default the client is using named pipes. If that client wants to connect to a server running on Windows 9x, you must change the default network library to TCP/IP. Do this with the Client Network utility.

Warning -A common misunderstanding has to do with how multiprotocol works. Although it can work over the NT network protocols—NetBIOS, TCP/IP, and NWLink IPX/SPX—it doesn't necessarily mean that the client can send requests using any of those network libraries. If you are using multiprotocol on the server, you must also use it on the client.

Troubleshooting Connectivity Problems

When a user reports that he or she can't connect, the first thing to do is to find out what message the user is getting. If the message says, Login Failed for User, the user most likely forgot his or her password. You can reset the password with `sp_password`. The command looks like this:

```
sp_password NULL, newpassword, loginid
```

Only the system administrator can issue this command.

IMO -The `sp_password` command should be available to members of the security admin role (Chapter 9 discusses that role in detail), but it is not.

If the user is getting messages, such as Specified SQL Server Not Found (named pipes) or General Network Error (Sockets), you most likely have a connectivity problem. These messages can also be issued if the server is down, so you want to make sure the server is running before doing anything else.

Unfortunately, most client connectivity problems have to be solved at the client workstation; so unless you have a utility such as PC Anywhere or Remotely Possible, you will have to visit the workstation of the user with the problem. You can try several things when you are at the user's workstation.

First, use the Client Network utility to check out the network library the client is using. This was installed along with Client Connectivity.

When you start the utility, the General tab shows you what the default network library is. A client can be set up to connect to more than one server, using different network libraries. You will see these

aliases on the General tab as well. If the default network library or the one associated with a particular alias is not the one that SQL Server is listening on, just change the library. You can either select a different default, modify an existing alias entry, or add a new network library and alias. The Network Libraries tab shows you the network libraries available on the client workstation.

You might also need to check that the client and server are using the same underlying network protocol. If the server is using TCP/IP and the client is using NetBEUI, for example, there will be no communication even if the SQL Server netlibs are the same. You can find the installed network protocols by right-clicking on Network Neighborhood, and then choosing Properties. On a Windows 9x machine, highlight the network adapter and choose Properties. On an NT machine, look at the information shown on the Protocols tab.

If you can't resolve the problem with the Client Network utility, try the methods described in the following sections.

Troubleshooting Named Pipes Connectivity

At the operating system prompt, on the client workstation, type the following:

```
NET VIEW\\SERVERNAME
```

This tells you whether the workstation can see the server. If you get back information about the server, the workstation can see the server. If you get an Error 5 – Access Denied, there is a problem with the NT Authentication login. If the user can't see the server at all, you receive an Error 53. In this case, or if you get any other message, a network problem exists and you should contact the network administrator.

If this succeeds, but you still can't connect to the SQL Server, you should test the pipe with `makepipe` and `readpipe`. Unfortunately, you need to do part of this on the server and part of this on the client. `makepipe` runs on NT only; `readpipe` runs on NT, Windows 9x, and Windows 3.11. These tools install automatically when you install SQL Server. They are in the `\MSSQL7\BINN` directory. There is no icon for them, and they do not appear in the Start menu.

makepipe

You run `makepipe` on the server with the following command:

```
> makepipe
```

Upon running this command, you should see output that looks something like the following:

```
Making PIPE:\pipe\abc  
read to write delay (seconds):0  
Waiting for Client to Connect...
```

At this point, you can return to the client workstation and issue the `readpipe` command.

readpipe

You run `readpipe` on the client with the following command:

```
> readpipe /Sservername /Dstring
```

Assuming, for example, that you have run `makepipe` on a server named `MyServer`, run the following:

```
> readpipe /SmyServer /Dhello
```

If this command is successful, you should see output that looks like this:

```
SvrName:\\myserver
PIPE  :\\myserver\pipe\abc
DATA  :hello
Data Sent: 1 : hello
Data Read: 1 : hello
```

If you do not see the Data Read message, you have a network problem and should contact your network administrator. Press `CTRL+C` on the server to terminate the `makepipe` program.

Troubleshooting Sockets Connectivity

To test TCP/IP connections, use `ping` at the client workstation. It takes either a server name or an IP address, as shown here:

```
> ping MyServer
```

or

```
> ping 11.11.11.11
```

You should see results that look like this:

```
Pinging myserver [1.1.1.40] with 32 bytes of data:
Reply from 1.1.1.40: bytes=32 time<10ms TTL=128
Reply from 1.1.1.40: bytes=32 time<10ms TTL=128
Reply from 1.1.1.40: bytes=32 time<10ms TTL=128
Reply from 1.1.1.40: bytes=32 time<10ms TTL=128
Ping statistics for 1.1.1.40:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milliseconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

If you get a message that the request timed out, you have a network problem and should contact your network administrator.

Troubleshooting ODBC Connectivity

If you have network connectivity but still can't connect to SQL Server, you may have a problem with ODBC connectivity. You can use `ODBCPING` to diagnose these problems. `ODBCPING` is in the `\MSSQL7\BINN` directory. There is no icon for it in your Start menu. You can use it to test a direct ODBC connection to a server and to test an ODBC *data source name* (DSN).

To test a direct ODBC connection, use the following command:

```
ODBCPING -Sservername -Uusername -Ppassword
```

If you don't get a response, you probably need to install a newer version of the ODBC driver.

If this works, you need to identify which ODBC DSN the client is using (you can probably determine this by looking at the ODBC applet in Control Panel), and then issue the following command:

```
ODBCPING -Ddatasourcename -Uusername -Ppassword
```

If the direct ODBC connection worked, but this one did not, you need to check the DSN for correctness. For example, is does the DSN specify a database the user does not have access to or provide an incorrect server name? Do this with the ODBC applet in Control Panel.

General Tip -You can also test connectivity to an ODBC data source with the ODBC applet. Highlight the data source and choose Configure. Click Next, and then Finish. On the final screen, click the Test Data Source button.

If none of these tricks help, you can try using SQL Profiler (described in Chapter 10) to see what login message is actually getting to SQL Server. Tools such as the NT Network Monitor and the Network General Sniffer can also help you diagnose and correct connectivity problems. Ask your network administrator for help with these last two tools.

© Copyright Pearson Education. All rights reserved.