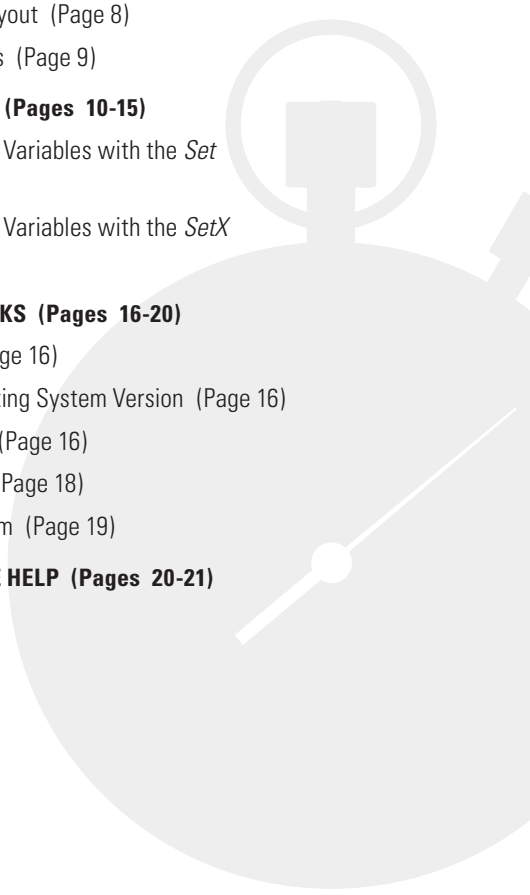
PART I
Command Line Basics

IN THIS PART

1

Configuring the Local Machine

IN THIS CHAPTER, YOU WILL LEARN TO:



You can access the command line anytime you want. Issuing commands will work just fine without doing anything special. However, if you want to have the best possible experience at the command line, then you need to perform a few configuration tasks before you proceed. This chapter describes basic configuration procedures you can use to enhance your command line experience and make it better.

Configure the Command Window

Many users start the command window, see the typical command prompt, and just assume that they'll never see anything else. Fortunately, you can easily configure the command window to appear as you want, at least within limits.

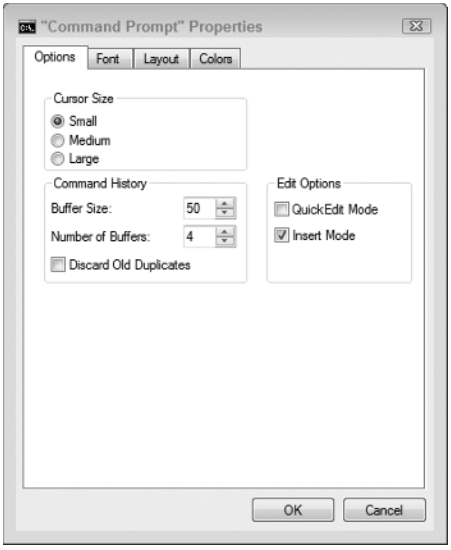
You can access these features using these steps:

1. Click the box in the upper left corner of the command window and choose Properties from the context menu. You'll see a properties dialog box with four tabs.
2. Set the properties on each tab to meet specific needs, such as displaying the text in another color. Each of these tabs is described in the sections that follow.

Set the Window Options

The Options tab shown in Figure 1.1 defines how the command window reacts when you open it. The Cursor Size option controls the size of the cursor, with small being the default. The Large option provides a block cursor that's very easy to see. The Display Options determine whether you see the command window full screen or as a window. Using the full screen mode when you have a number of tasks to perform is easier on the eyes.

Figure 1.1: The Options tab helps you control the appearance and behavior of the command window.



NOTE Older versions of Windows let you change the display mode through a property setting. However, Server Core (the version of Windows Server 2008 that comes without the usual GUI and relies exclusively on the command line for configuration) doesn't let you run the command window in full screen mode by changing the Display Options setting. This particular option is missing when you view the dialog box shown in Figure 1.1. In most cases, you don't want to run the command window in full screen mode when working with Server Core because the few graphical elements it provides can become inaccessible and it's already possible to maximize the screen real estate by maximizing the window. Without a Start menu, taskbar, or other graphical elements to consume space, using Windows shouldn't cause any problems. (If you really must work in full screen mode, you must modify the registry to do it.)

The Command History is especially important. The Buffer Size option determines the number of commands the buffer will store. Every command requires memory, so increasing this number increases the amount of memory the command prompt requires. Increase this number when you plan to perform a number of complex commands. A smaller number will save memory for larger command line applications. The Number of Buffers Option controls the number of individual histories. You need one history for each command process (application environment) you create. Generally, the four buffers that Figure 1.1 show work fine.

To better understand how buffers work, try this experiment:

1. Open a command window.
2. Type a command such as **`CLS`** or **`Dir`** and press Enter.
3. Press the Up arrow.

   You should see the command you just typed—there's the buffer. The command you typed appears in the first buffer. Remember that you have four buffers that you can use at the command processor when using the default settings.
4. Press Esc. The command processor clears the command from the prompt so you have a blank prompt to use.
5. Type **`Cmd`** and press Enter. You've just created a new command processor. This command processor uses the second buffer.
6. Press the Up arrow. You don't see anything because this command processor is using its own buffer.
7. Now, type a different command (such as **`CLS`** or **`Dir`**) and press Enter.
8. Press the Up arrow. You'll see the command you just typed, but not any of the commands from the previous command processor.
9. Press Esc to clear the command.
10. Type **`Exit`** and press Enter to close the current command processor. You're back to the previous command processor.
11. Press the Up arrow twice and you'll see whatever command you typed earlier because this command processor is using the first buffer.

12. Press Esc to clear the command.
13. Type **`Cmd`** and press Enter. This action creates a new command processor that will use the second buffer.
14. Press the Up arrow.

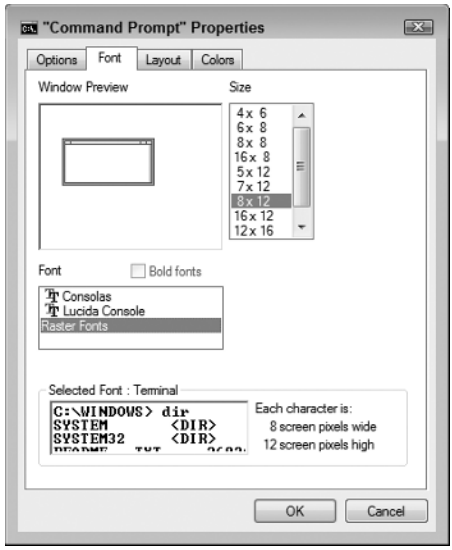
    Wait, what are you seeing here? You see the `Exit` command. Press the Up arrow again and you'll see the command you typed in step 7 for the second buffer. Each buffer retains its content, even if you close the command processor.
15. Type **`Exit`** and press Enter. You return to the first command processor.
16. Type **`Exit`** again and press Enter; the command processor window closes.

The Edit Options determine how you interact with the command window. Check the QuickEdit Mode when you want to use the mouse to work with the entries directly. The only problem with using this feature is that it can interfere with some commands such as `Edit` that have a mouse interface of their own. The Insert Mode option lets you paste text into the command window without replacing the text currently there. For example, you might copy some information from a Windows application and paste it as an argument for a command.

Change the Font

The Font tab shown in Figure 1.2 controls the font used to display text. The font size automatically changes when you resize the window, but you can also control the font size directly using this tab. The raster fonts give the typical command line font appearance that works well for most quick tasks. The Lucida Console font works better in a windowed environment. It's easier on the eyes because it's smoother, but you might find that some applications won't work well with it if they create "text graphics" using some of the extended ASCII characters. The extended ASCII characters include corners and lines that a developer can use to draw boxes and add visual detail.

Figure 1.2: Use the Font tab to control the size of the text in the command window.



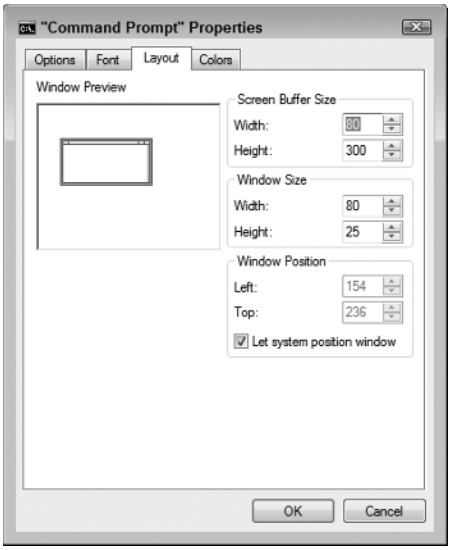
Choose a Window Layout

The Layout tab shown in Figure 1.3 has the potential to affect your use of the command window greatly when working in windowed mode. The Screen Buffer Size controls the width and height of the screen buffer, the total area used to display information. When the Window Size setting is smaller than the Screen Buffer Size, Windows provides scroll bars so you can move the window around within the buffer area and view all it contains. Some commands require a great deal of space for display purposes. Adjusting the Screen Buffer Size and Window Size can help you view all of the information these commands provide.

The Window Position determines where Windows places the command window when you first open it. Some people prefer a specific position on the screen so they always know where a new command window will appear. However, it's generally safe to check Let System

Position Window to allow Windows to place the command window on screen. Each command window will appear at a different, randomly chosen, position on screen.

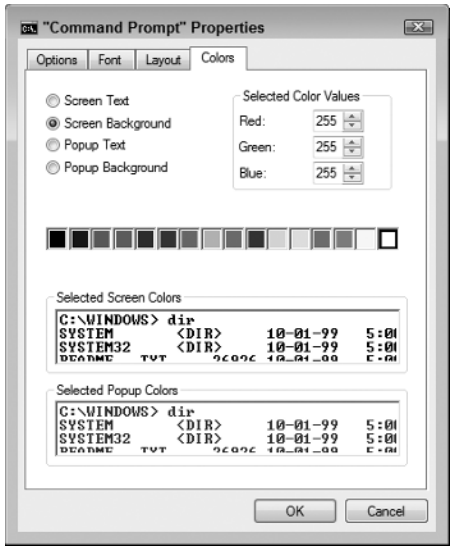
Figure 1.3: Change the size and positioning of the command window using the Layout tab.



Define the Text Colors

Microsoft assumes that you want a black background with light gray letters for the command window. Although DOS used this setting all those years ago, today, many people want a choice. The Color tab lets you choose different foreground, background, and pop-up colors for the command window (even though Figure 1.4 doesn't show the colors, it does present the dialog box layout). You can modify the window to use any of the 16 standard color combinations for any of the text options. Use the Select Color Values options to create custom colors.

Figure 1.4: Modify the text colors for an optimal display using the Colors tab.



Set the Environment

The command line environment is important because it controls how the command processor works and also changes the way the commands and utilities work in many cases. Configuring the command line lets you perform work faster and with greater ease. For example, you might need to create an environment variable to ensure that a command or utility can locate files or data that it needs. The following sections describe how to control the command line environment so that everything works as you anticipate.

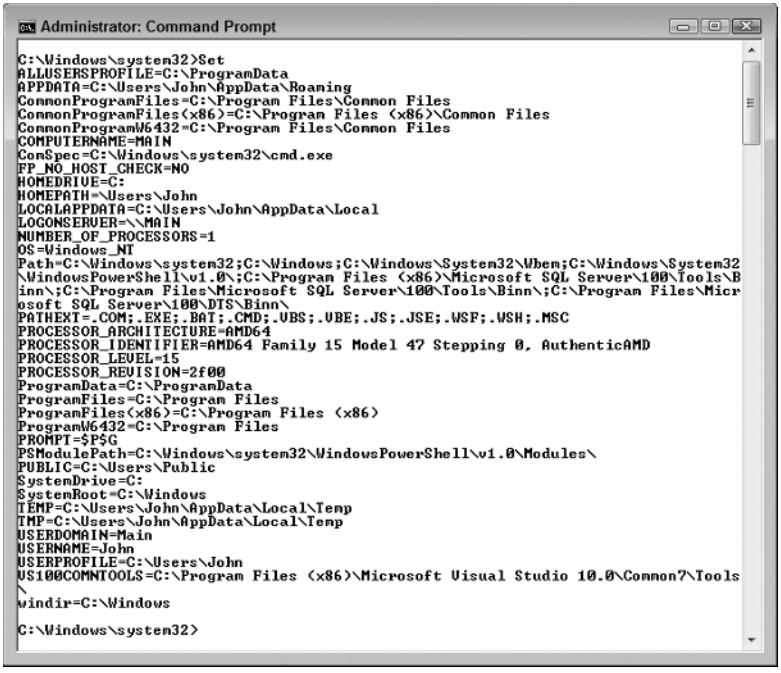
Manage Environment Variables with the *Set* Command

Environment variables are important because they let you define the value of something. An environment variable acts as a storage container that the command processor or you can later access to work more efficiently. For example, the PATH environment variable contains a list of locations to search for executable files. The command processor uses the PATH environment variable to locate the commands you want to execute. The following sections tell how to work with environment variables.

Display Environment Variables

The operating system automatically creates some environment variables when you open a command prompt. To see a list of these environment variables, type **Set** and press Enter. Figure 1.5 shows typical output from this command.

Figure 1.5: Display every environment variable using the `Set` command.



All of the environment variables shown in Figure 1.5 are defined by default—you don't create any of them. To display a single environment variable, you can use either of the following commands.

```
Set VariableName
```

or

```
Echo %VariableName%
```

In both cases, you type the command, followed by the name of the variable you want to see. For example, if you want to see the value of the `PATH`

environment variable, you type either **`Set PATH`** or **`Echo %PATH%`** and press Enter. The second form relies on variable expansion. You tell the command processor to expand an environment variable by surrounding the environment variable name with percent signs (%). Environment variable expansion has a lot of uses, but you normally use it when systems have the same setting with a different setting value. For example, the location of the Windows directory can differ between machines, but every machine will have a Windows directory. Using environment variable expansion makes it possible to find the Windows directory location on each machine.

NOTE The PATH environment variable provides a third display method that isn't available to other environment variables. You can simply type **PATH** and press Enter to display the path.

Create or Change an Environment Variable

In some cases, you must create or change an environment variable. To create or change an environment variable temporarily, use the following command line syntax.

```
Set VariableName=Value
```

The *VariableName* defines the name of the variable you want to create or change, while Value defines the content of the variable. If you type the name of an existing variable, the command processor changes its value. When you need to make more permanent changes, you must use the SetX utility described in the "Manage Environment Variables with the SetX Utility" section of the chapter instead.

Expand an Environment Variable

Several applications can share an environment variable. The most common example of a shared environment variable is PATH, but there are other examples. You may find that you need to expand the environment variable content, rather than change it. For example, you might need to add another path to the PATH environment variable. In this case, you expand the current environment variable content and add it to the new content using this approach:

```
Set VariableName = %VariableName%;Value
```

or

```
Set VariableName = Value;%VariableName%
```

For example, let's say you want to add C:\ to the beginning of the existing PATH environment variable. In this case, you type **Set PATH=C:\;%PATH%** and press Enter (remember that paths are separated in the PATH environment variable using semicolons). Using %PATH% expands the content of the PATH environment variable, just as if you had typed **Echo %PATH%**. Likewise, if you want to add C:\ to the end of the PATH environment variable, you type **Set PATH= %PATH%;C:** and press Enter.

Use Equations in Environment Variables

Normally, the command process places the precise value you type in the environment variable. For example, if you type **Set MyVar=2*3** and press Enter, MyVar will contain the value 2*3. However, if you type **Set /A MyVar=2*3** and press Enter, MyVar will contain the value 6 instead. The /A command line switch tells Set to interpret the value as an equation.

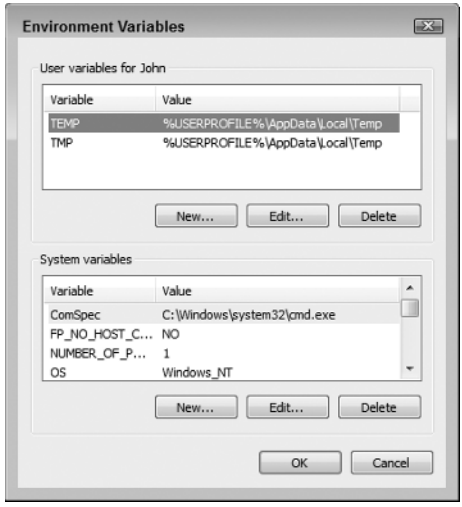
Get User Input

You might not know what value to place in an environment variable when you create a batch file. In this case, you can prompt the user to obtain the value. For example, if you want to provide a value for MyVar, you might type **Set /P MyVar="Type a value for MyVar "** and press Enter. In this case, the user sees the prompt "Type a value for MyVar" at the command line. To add a value to MyVar, the user types it at the command line and presses Enter.

Manage Environment Variables with the *SetX* Utility

Any environment variable that you create or change using the Set command is only valid for the current session. The moment that you close the command prompt, the environment variable changes back to its original value or it disappears entirely. You can create or change environment variables permanently using the Environment Variables dialog box shown in Figure 1.6 (accessed by clicking Environment Variables on the Advanced tab of the System Properties dialog box). Unfortunately, having to manually change permanent environment variables using this approach won't work when you need to automate tasks. The SetX utility makes it possible to make such changes from the command prompt.

Figure 1.6: The Environment Variables dialog box shows permanent environment variables.



As you can see from Figure 1.6, permanent environment variables can affect either a single user or the system as a whole. The `SetX` utility can create or change environment variables at either level. Use the `Set` command to display the environment variables you create using the `SetX` utility. You can also use the `SetX` utility to change environment variables on other machines. The following sections tell how to use the `SetX` utility.

Change the User-Level Environment

User-level environment variables affect only the current user. The system stores them in the `HKEY_CURRENT_USER` hive of the registry. To create a permanent user-level environment variable, use the following command line syntax.

```
SetX VariableName Value
```

As with the `Set` command, `VariableName` contains the name of the variable you want to create, while `Value` contains the information you want to place within the variable. Unlike the `Set` command, there's no equals sign between `VariableName` and `Value`. The environment variable you create won't affect the current session—to create an environment variable for the current session, you must also use the `Set` command.

Change the System-Level Environment

System-level environment variables affect every user of a particular machine. The system stores them in the HKEY_LOCAL_MACHINE hive of the registry. To create a permanent system-level environment variable, use the following command line syntax.

```
SetX /M VariableName Value
```

The /M command line switch tells SetX to create a system-level (also known as a machine-level) environment variable. As before, VariableName contains the name of the variable you want to create, while Value contains the information you want to place within the variable. The new environment variable only affects future sessions. You must use the Set command to create an environment variable for the current session.

Set Environment Variables on Other Machines

There's a whole class of commands and utilities that perform tasks on other machines from a local machine. Administrators can use the commands and utilities to interact with client systems without actually going to the client system. In general, you must supply these values:

- Machine name (/S command line switch)
- Username (/U command line switch)
- (Optional) Password (/P command line switch)

The machine name is the fully qualified name of the machine you want to access. The username must be an account that the person using the command can access. Adding the password to a batch file is a security risk. Consequently, you should always omit the password. However, the command won't execute sometimes if you omit the /P command line switch. To get around this problem, you can use /P *, which isn't documented, but always works.

As an example, let's say you want to add a system-level environment variable named NewVar to a machine named WinMachine using the Administrator credentials. In this case, you type **SetX /S WinMachine /U Administrator /P * /M NewVar Value** and press Enter. The SetX utility prompts you for the Administrator password. Type the password and press Enter to complete the task.

Perform Common Tasks

Several common tasks performed at the command line don't necessarily involve the work you're trying to accomplish. For example, when the screen becomes cluttered with too much old information, you might want to clear it so that all of your new commands are easier to see. The following sections describe a few of these common tasks.

Clear the Display

After you execute a number of commands, you might find that the display is getting cluttered. Too much information on the screen can slow you down, so cleaning up every once in a while is a good idea. To clear the display, type **`CLS`** and press Enter. There isn't any way to just clear part of the display—you must clear the entire display. However, clearing the display doesn't clear the command history. You can still press the Up and Down arrows to move between previously typed commands.

Determine the Operating System Version

Not every version of the command processor supports every command and utility. Consequently, you often need to know which version of the command processor is present on the user's machine. To perform this task, type **`Ver`** and press Enter. You'll see an operating system version number, such as Microsoft Windows [Version 6.1.7600], which indirectly tells you which version of the command processor is installed. (The command prompt also displays the version number automatically when you open the window.)

Start an Application

Sometimes you want to start a command line application in a separate window. Perhaps this application requires a special environment to run and you don't want to change the current environment to support it. In many other cases, the command line application requires enhanced rights to run or has some other requirement that makes it impossible or seriously irresponsible to run it in the current window. For example, you don't want to run commands that modify the registry in a window with user-level access—using a separate window is

more secure. It's also possible to run an application with a higher priority or assign it to a specific processor to help spread the load among processors on a multi-processor system. The `Start` utility provides the answer to all of these needs.

You can use the `Start` utility with either command line or Windows GUI applications. For example, if you want to start a separate window to execute the `Ver` command, you'd type **`Start Ver`** and press Enter. The command processor will open a separate window, execute the `Ver` command in it, and wait for you to close the window before continuing. Of course, you might not want the command to execute in a separate window, in which case you'd type **`Start /B Ver`** and press Enter. The `/B` command line switch tells the command processor to use the existing window. The difference is that using `/B` disables Ctrl+C usage; you must issue a break command using Ctrl+Break instead.

It's also possible to start Windows applications using the `Start` utility. In fact, you have a wide range of ways to use the `Start` utility to start a Windows application. The easiest method is to simply use the name of the application, such as Notepad. For example, if you want to start a copy of Notepad, you'd type **`Start Notepad`** and press Enter. Unlike a command line utility, `Start` automatically resumes once it starts the Windows application unless you use the `/Wait` command line switch. For example, if you type **`Start /Wait Notepad`** and press Enter, the `Start` utility will wait until you exit Notepad before it begins the next step of a batch file or returns control to the command prompt. You can also start Windows applications as minimized using the `/Min` command line switch or maximized using the `/Max` command line switch.

`Start` can also provide some file-related alternatives for starting applications. For example, if you type **`Start Test.TXT`** and press Enter, the command processor will look in the registry for the default application for opening files that have a TXT extension, start that application, and pass the name of the file to it. In this case, you'll likely see a copy of Notepad open with `Test.TXT` loaded in it. When you want to be sure that the command process uses a specific application, you provide the name of the application and the file. For example, let's say you want to open a specific URL in Firefox. In this case, you might type **`Start "C:\Program Files\Mozilla Firefox\Firefox" http://antwrp.gsfc.nasa.gov/apod/`** and press Enter. The command processor will open the URL at `http://antwrp.gsfc.nasa.gov/apod/` in Firefox.

Work with Services

The Service Control (SC) utility provides a number of methods for interacting with services on a system. In fact, there are enough ways that you're unlikely to use them all. Most administrators need to know how to start, stop, pause, continue, and view a service. The following sections describe how to perform common SC tasks.

View Service Status

SC provides an amazing 10 ways to query information about a service. The most common way is to use the Query or QueryEx options because they provide you with basic information about the service status and the features it supports. For example, if you want to see the status of the W32Time service, you type **SC Query W32Time** and press Enter. The output will tell you about the service type, whether it's running, the commands it supports (such as Pause), and some flag information that can prove useful at times when troubleshooting a particular service (you'd need documentation about these flags before the flag information becomes useful).

Sometimes you need a little more information. For example, you might want to know the service description. In this case, you type **SC QDescription W32Time** and press Enter. You see the service name and the full description provided for it in the Services console. A more practical bit of information are the privileges the service requires to work properly. In this case, you type **SC QPrivs W32Time** and press Enter. The output provides a list of privileges that you can decipher at http://svchost-exe.net/. Of course, administration normally means knowing how a service is configured. To see the configuration for a service, type **SC QC W32Time** and press Enter. The output contains the content of the TYPE, START_TYPE, ERROR_CONTROL, BINARY_PATH_NAME, LOAD_ORDER_GROUP, TAG, DISPLAY_NAME, DEPENDENCIES, and SERVICE_START_NAME fields.

Start, Pause, Continue, or Stop a Service

Administrators commonly need to perform four common tasks with services: start, pause, continue, and stop. Pausing a service differs from stopping a service in that the service retains all of its data. When you continue the service after a pause, the service begins right where it left off. Stopping a service requires that you issue a Start command to reload it, which means that the service begins from scratch. Almost every

service supports starting and stopping—pause support requires special programming. Here are typical examples of the four service status changing commands.

- `SC Start WinMgmt`
- `SC Pause WinMgmt`
- `SC Continue WinMgmt`
- `SC Stop WinMgmt`

In all four cases, the `SC` utility either displays an error message, such as "[SC] StartService FAILED 1056: An instance of the service is already running" or it displays a success message that basically repeats the output of the `Query` switch. The important field of a successful command is `STATE`, which tells you the current operational state of the service.

Shut Down the System

Administrators often need to perform a shutdown of the system using something other than the standard GUI. For example, the system may require a shutdown after running a script or batch file that performs an update. Fortunately, the `ShutDown` utility provides the means of performing both local and remote shutdowns as described in the following sections.

Log Off the System

Logging off the system simply means that you end the current session. The system remains operational and someone else can log into it. This is the most common use of `ShutDown` for administrators because it lets the administrator log off the system after performing maintenance and lets the user take over. To log off a system, type **`ShutDown \L`** and press Enter.

Perform a Shutdown

In some cases, you need to perform a system shutdown, such as after installing a new application. In this case, you have a number of choices. The easiest method is to simply type **`ShutDown /S`** and press Enter. The system will perform an orderly shutdown after 30 seconds. If you want to shut down a remote system, you must also provide the machine name such as, **`ShutDown /S /M \\RemoteComputer`**, and press Enter.

The larger your organization is, the greater the need to provide a reason for the shutdown. Of course, you have the three major categories: Planned, Unexpected, and Expected. The three major categories have a major and minor subcategory associated with them. For example, when you install a new application, the major category is 4 and the minor category is 2. You can see a list of these codes at `http://ss64.com/nt/shutdown.html`. These codes feed into the Shutdown Event Tracker, which creates events that you can see in the Event Viewer (the article at `http://www.topbits.com/the-shutdown-event-tracker.html` provides some excellent information about the Shutdown Event Tracker). To shut down a system with an event, use the `/D` command line switch. For example, to show that the system is being shut down as the result of an application installation, you'd type **`ShutDown /S /D P:4:2`** and press Enter.

A 30-second shutdown is standard. However, you might find that you need to perform an immediate shutdown in some cases. To perform an immediate shutdown, type **`ShutDown /S /T:0`** and press Enter. The `/T` command line switch provides a timeout interval for the shutdown—a value of 0 means that the shutdown is immediate.

Shut Down and Restart the System

A shutdown actually turns off the computer in most cases. If you want to use the computer after the shutdown occurs, then you need to perform a shutdown and restart. The `ShutDown` utility provides two forms of restart. The standard restart simply starts up Windows as a fresh session. To perform this kind of restart, type **`ShutDown /R`** and press Enter.

In some cases, you might want to restore the Windows applications that were open at the time of the shutdown. In this case, you type **`ShutDown /G`** and press Enter. Most applications will automatically reopen when the system restarts.

Obtain Command Line Help

There are a number of ways to obtain help at the command line. In most cases, you'll type the name of a command or utility, followed by the `/?` command line switch to learn more about it. For example, to discover more about the `Start` utility, type **`Start /?`** and press Enter. Some utilities provide layered help. For example, the `Net` utility provides multiple layers. If you type **`Net /?`** and press Enter, you see the top layer

that lists the subcommands you can type. To discover more about the `Accounts` subcommand, type **`Net Accounts /?`** and press Enter.

Unfortunately, Microsoft decided to make things difficult in some cases. For example, some utilities require that you use the `/Help` command line switch instead or you might have to use the `Help` utility to learn more about the command or utility in question. To see a list of commands and utilities that `Help` supports, type **`Help`** and press Enter. You could then learn more about a specific command, such as `Ver`. In this case, you'd type **`Help Ver`** and press Enter

The help supplied at the command line isn't always complete or accurate. For example, the `Start` utility seems to indicate that you can run 16-bit applications in a separate memory space using the `/Separate` command line switch. However, the `/Separate` command line switch is only useful when you're working with 32-bit Windows. The 64-bit version of Windows won't run 16-bit applications, even if you use the `/Separate` command line switch.