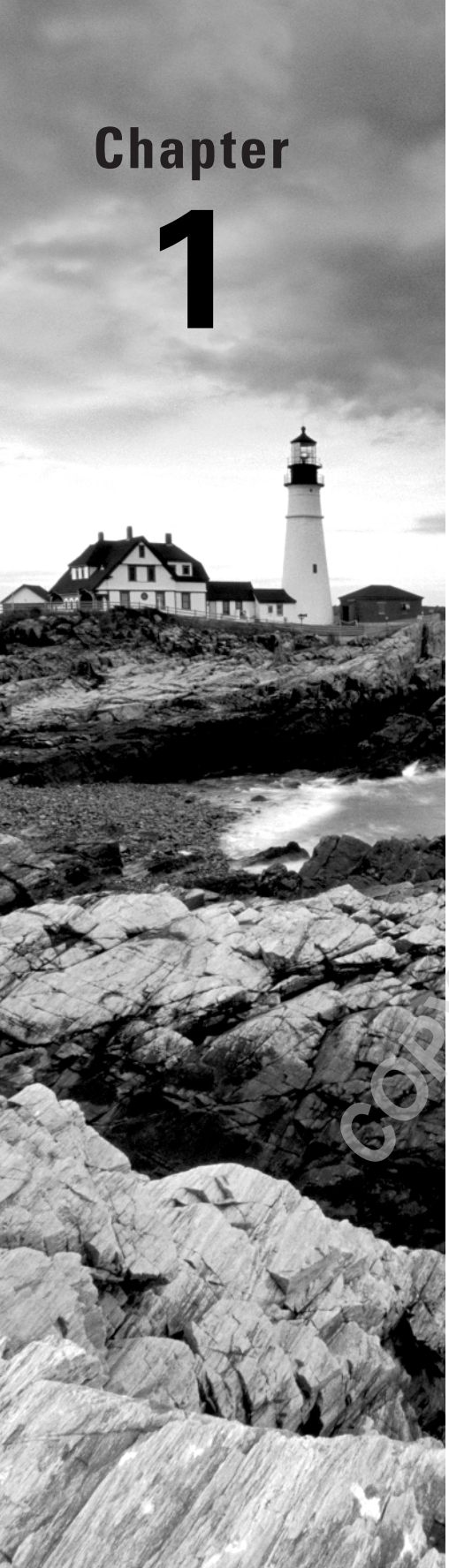


Chapter 1

Introducing Windows 8.1

70-687 MICROSOFT EXAM OBJECTIVES COVERED IN THIS CHAPTER:

✓ **Configure access to resources**

- Configure user rights
- Manage credentials
- Manage certificates
- Configure biometrics
- Configure picture password
- Configure PIN
- Set up and configure Microsoft account
- Configure virtual smart cards
- Configure authentication in workgroups or domains
- Configure User Account Control (UAC) behavior

Windows 8.1 has a significantly different interface than previous Windows versions. New users might take to it quite easily, but for many experienced Windows users it presents some challenges—and frustrations. It includes a Start screen with tiles and charms, but the Desktop is just a keystroke or a mouse click away. In this chapter, you'll learn about the new interface, including some ways to navigate it and reach tools you'll need to do common tasks.

Users prove who they are by authenticating, so it's important to understand the various account types available in Windows 8.1. This chapter includes exercises for creating different account types, including the new Microsoft Live account type. You'll also learn about new picture passwords and the new login PIN feature.

The command prompt and Windows PowerShell are both available in Windows 8.1, and they are required to do many of the tasks covered by the Windows 8.1 exams. PowerShell is much richer than the command prompt, but if you understand how PowerShell commands use the verb-noun format and how to get help, you'll be able to use it for common tasks. In this chapter, you'll learn how to start these tools and execute basic commands.

Navigating Windows 8.1

The Windows 8.1 interface can be a little confusing if you're not used to it. However, once you grasp a few simple concepts it becomes much easier to navigate.

Figure 1.1 shows the Windows 8.1 Start screen with some items highlighted. The username is displayed at the top right, the default tiles and apps are in the center, and some additional tiles and apps are on the right. If you scroll to the right, you'll reveal more apps. At the bottom right, you can see the negative symbol (–), which is the *semantic zoom* icon. This feature is useful when you have tiles that span multiple pages and allows you to quickly select, and move to, another area. You can also access this feature by pressing Ctrl+– (minus key), or by holding the Ctrl key and spinning the mouse wheel.

In addition to what you can see, several features aren't displayed. If you move the mouse to each of the four corners, the additional features appear:

Top Left When you move your mouse to this corner, Windows displays an icon of a recently used app and also displays one or more tabs on the left edge of the screen. If you move your mouse to one of the tabs, Windows displays a list of icons for other recently opened apps. If other apps are currently open, it displays icons you can select to switch apps.

Bottom Left This shows the Start button. Clicking it toggles between the traditional desktop and the Start screen. If you right-click the Start button you get the Start button menu, which gives you access to a number of applications and features, usually a list of programs commonly used by technicians. You can also use the Windows logo key on the keyboard to toggle between the Start screen and the Desktop, or between the Start screen and a running app. If you right-click over this display, it shows a preview menu.

FIGURE 1.1 Windows Start screen

The Windows logo key is between the Ctrl (or FN) and Alt keys on the left of the spacebar on most keyboards. This key is combined with other keys to invoke Windows 8.1 shortcuts.

Top Right Moving your mouse to this corner displays the Charms bar, allowing you to pick from one of the five charms. You can also display the Charms bar by pressing Windows logo key+C, which also brings up the Time display.

Bottom Right If you hover over this area, it will also display the Charms bar, similar to hovering over the top-right area. This area also includes the semantic zoom icon.

Using Gestures

Windows 8.1 is designed so that it can work on a tablet, with a touch screen, or with a mouse. If you have a smartphone or tablet device, you are probably familiar with many gestures used on a touch screen. However, if you're not familiar with these gestures, read on. The following list shows common gestures used with Windows 8.1 and how they can be simulated with a mouse and keyboard:

Two-fingered gestures are commonly done with a finger and a thumb. However, you can also use two fingers if desired.

Tap You can tap an item by touching it once with your finger. This is similar to a single mouse click.

Press and Hold This is similar to a tap, but you don't remove your finger. After a moment, a context menu or information box will appear, depending on the item. This is similar to right-clicking with a mouse.

Pinch You can touch the screen with a finger and thumb and bring them closer together as if you're pinching something between them. This causes the display to zoom in. This works similar to holding down the Ctrl key while spinning the mouse wheel in one direction. If the Start screen is displayed, this action invokes semantic zoom.

Stretch This is the opposite of pinch. You can zoom out by touching the display with your finger and thumb and spreading them apart. You can also mimic this with the Ctrl key and the mouse wheel by spinning the wheel in the opposite direction.

Rotate If an item can be rotated, you can place your finger and thumb on the screen and twist your hand either clockwise or counterclockwise. This only works with certain items in certain apps, such as when rotating an image or graphic in a photo-editing app.

Slide to Scroll If you touch and drag your finger across the screen, it allows you to scroll through the screen. This technique is the same as scrolling with a mouse and can be used to scroll horizontally and/or vertically, depending on what is currently displayed.

Slide to Move If you touch a movable item, you can keep your finger on the item and move it around the screen. This is similar to dragging an item with the mouse by selecting and holding it with the mouse button. For example, you can rearrange items on the Start screen by using the slide-to-move gesture (or by dragging an item with the mouse button held down).

Swipe to Select Some items allow you to quickly slide them a short distance and they will bring up different commands. This isn't repeatable with the mouse.

Swipe from Edge If you swipe your finger from the right edge, it brings up the Charms bar. If you swipe your finger from the left edge, it either shows open apps or allows you to manipulate how the apps are displayed. If you swipe from the top or bottom, it shows

additional commands for the current app. You can also close an app by swiping it down to the bottom of the screen.

Using Charms

A significant difference in Windows 8.1 is the use of charms. Instead of the traditional Start menu, Windows 8.1 starts with the Start screen (also called the Start charm). Apps, tools, and utilities are accessed through the Start screen or other charms. The easiest way to bring up the Charms bar is by pressing Windows logo key+C. There are five primary charms:

Start This is the primary display showing the default apps. You can also press the Windows logo key to toggle back and forth between the Start screen and the Desktop or between the Start screen and a running app. As mentioned earlier, clicking the Start button on the bottom left of the screen lets you toggle between Start and Desktop.

Search Selecting this brings up a Search text box. As you type, Windows looks for and displays apps, settings, or files that match the text. If the Start screen is displayed, you can just start typing and the Search charm automatically appears. You can press Windows logo key+F to search for files, Windows logo key+Q to search for apps, or Windows logo key+W for specific settings.

Share The Share charm lets you send links and photos to friends from within certain apps. You can also access the Share charm with Windows logo key+H.

Devices You can use this charm to send files and stream movies to TV, printers, and other devices connected on a network. This is not related to devices managed with the Device Manager. You can also access Devices with Windows logo key+K.

Settings This charm is used to do many basic tasks such as configure the network interface card (NIC), adjust the sound volume or brightness, configure notifications, and turn off or restart the computer. You can select Change PC Settings from here to modify many basic computer settings. You can also open Settings with Windows logo key+I.

It's worth your time to learn how the Search charm works. Using this charm makes it easy to find what you need, and you can start by simply typing from the Start screen. Additionally, the Settings charm is important for Windows 8.1 administrators. It includes a link to the PC Settings tool, which is used in Exercises 1.2 and 1.4, and elsewhere in this book.

Using Tiles

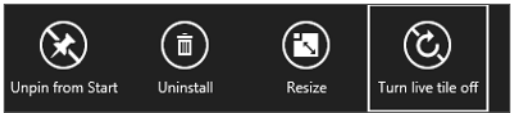
The Start screen includes several icons called *tiles* in Windows 8.1. Tiles can be dynamic and display live information based on the purpose of the underlying app. For example, the Weather tile shows weather for a specific location.

You can drag tiles around the screen to rearrange them. You can also create new groups and organize them in a new group. Exercise 1.1 (later in this chapter) shows how to manipulate some of these tiles.

Using Alt Menus

In traditional Windows applications, you can right-click an item and an alt menu appears. If you do this with items in the Start screen, you'll see menu selections appear on the bottom of the screen. For example, Figure 1.2 shows the alt menu that appears for the Weather tile.

FIGURE 1.2 Alt menu



For this selection, you have the following choices:

Unpin From Start Select this option to remove the app from the Start screen. You can pin items that are not already pinned to the Start screen by right-clicking over an empty area of the Start screen, selecting All Apps, right-clicking over an app, and selecting Pin To Start.

Uninstall For some apps, this option will uninstall the app after prompting you to confirm it. For other apps, it launches the Programs And Features Control Panel applet and you can uninstall it from there.

Resize This option reduces or enlarges the size of the tile on the Start screen. When you click this option, you get a submenu with Large, Wide, Medium, and Small as the tile size choices.

Turn Live Tile Off This option stops the animation for the tile. The app is still accessible but the icon returns to the default icon.

Using the Start Menu

The Start menu is available by right-clicking the Start icon at the bottom left of the screen. You can see the preview icon by dragging your mouse to the bottom left. When you right-click it, you'll see a display similar to Figure 1.3. These tools are valuable to administrators, and this menu is likely to be popular among many administrators.

You can also access the Start menu by pressing Windows logo key+X.

FIGURE 1.3 Start menu

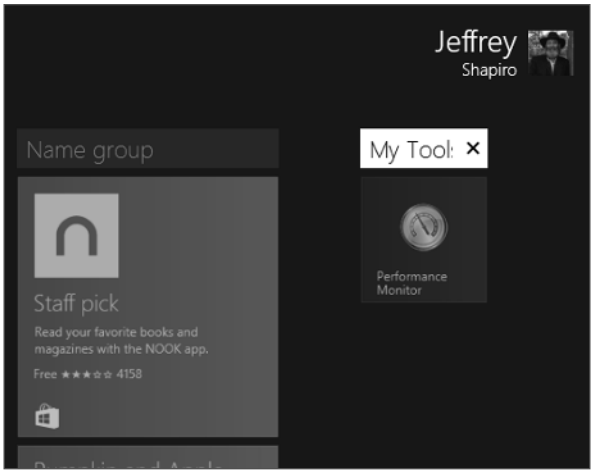
Showing Administrative Tools

Many administrators will likely want to add the Administrative tools menu to the Start screen. Exercise 1.1 shows how to do this. You'll also learn how to create a tile group and how to organize tiles in the group. This exercise assumes you're logged in and the Start screen is showing.

EXERCISE 1.1

Adding Administrative Tools to the Start Screen

1. Access the Settings charm by pressing Windows logo key+I.
2. Select Tiles. Note that the Tiles selection is not available if you access the Settings charm from the Desktop.
3. The Show Administrative Tools selection appears. Click the icon next to No to change it to Yes.
4. Click a blank area on the Start screen. There is a slight delay before the tools appear. Keep in mind that Windows includes 22 administrative tools and will take up a substantial part of the Start screen when all tools are added.
5. There might be some administrative tools that you don't use. You can follow these steps to remove and add items to the Start screen:
   a. Right-click over ODBC Data Sources. You'll see a check mark indicating this item is selected and the alt menu appears along the bottom of the screen. Select Unpin From Start. Doing so removes the item from the Start screen. You can use this method to remove any items from the Start screen.
   b. Right-click in a blank area of the Start screen. Select All Apps from the lower-right corner.
   c. Right-click over ODBC Data Sources. You'll see a check mark indicating this item is selected and the alt menu appears along the bottom of the screen. Select Pin To Start from the alt menu.
6. The following steps will create a group for Administrative Tools items:
   a. Pin the Performance Monitor, or any other Administrative app, to Start. Click and hold the app's tile, and drag it up but don't release it. Once it's away from the other tiles, drag it to the right. When a white horizontal bar appears, release it. This puts the Performance Monitor in a separate group.
   b. Right-click the tile. It will immediately be checked as selected and allow you to customize the tile and its group.
   c. Select Name Group, which appears above the tile groups, and type **My Tools** in the text box. Your display will look similar to this:



d. Click a blank area of the Start screen to expand it to regular size. You can then drag and drop the tools that have been pinned to Start into the new group by dragging and dropping the tile onto the name of the group. You can right-click and select Unpin From Start for any tools that you don't want to keep in this group.

Snapping Apps

Windows 8.1 allows you to share your screen between different apps by snapping an app to the screen edge. With an app open, hold down the Windows logo key, toggle the period (.), and press an arrow button to position the application. You can also drag the app and snap it to the right or left edge. To return to full screen, simply pull it away from the edge and expand it as usual.

Snapping apps works best if the resolution is set to 1366×768 or higher. A default installation of Windows 8.1 starts with a minimum resolution of 1024×768, so this feature won't make sense unless the resolution is much higher. You can access the Screen Resolution page by right-clicking the Desktop and selecting Screen Resolution.

Using Windows 8.1 Shortcuts

Just as with most other Windows operating systems, you'll find that there are many shortcuts you can use to get to a desired app or tool. One of the most important tools is the Search charm, which starts as soon as you begin typing from the Start screen. In Figure 1.4 the Search charm is displaying the Control Panel tile after a user types **control**. You can use the Search charm to find many familiar tools if you know the name.

FIGURE 1.4 Using the Search charm to search apps

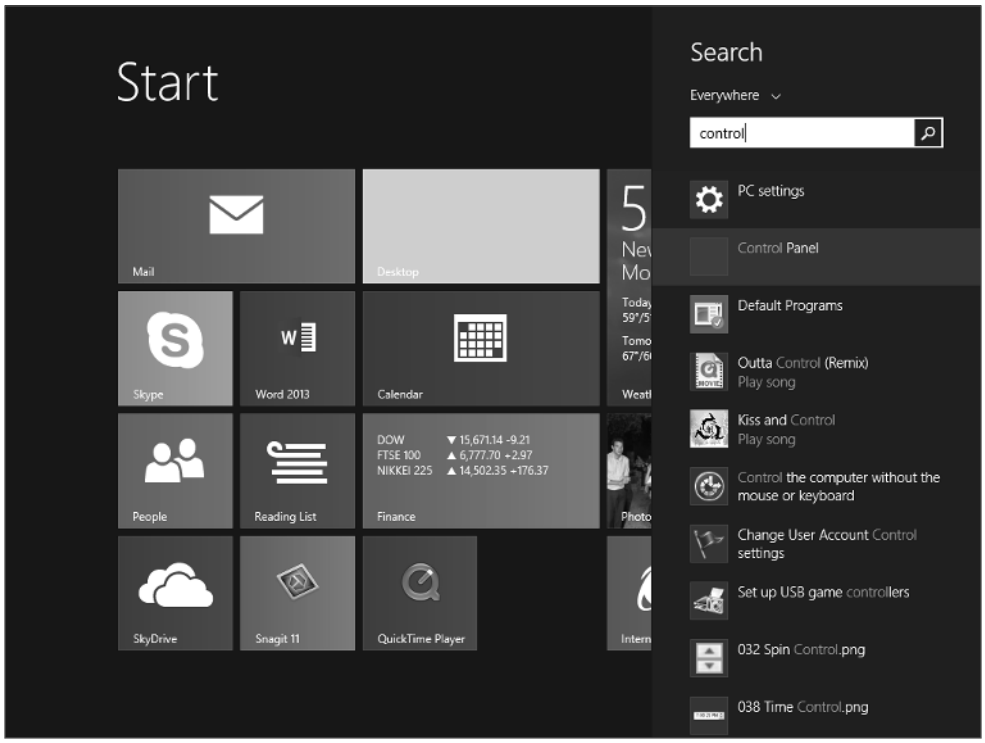


Table 1.1 shows shortcuts for accessing many common tools, and Table 1.2 shows the shortcuts you can use to access Windows 8.1 charms.

TABLE 1.1 Windows 8.1 shortcuts

Feature	Key combination
Toggle between Start and the Desktop.	Windows logo key
Show the Desktop when an app is running.	Windows logo key+D

Feature	Key combination
Cycle through apps (except Desktop apps).	Windows logo key+Tab (shows a taskbar on the left side of the screen)
Cycle through apps in reverse order (except Desktop apps).	Windows logo key+Shift+Tab (shows a taskbar on the left side of the screen)
Cycle through Desktop apps.	Alt+Tab
Cycle through Desktop apps in reverse order.	Alt+Shift+Tab
Open the App Switcher from the Start screen.	Alt+Tab
Show the preview menu (Programs And Features, Power Options, Event Viewer, and more).	Windows logo key+X
Lock display.	Windows logo key+L
Peek at the Desktop.	Windows logo key+, (comma)
Start File Explorer.	Windows logo key+E
Open the App Switcher.	Alt+Tab (toggle Alt to select different apps)
Zoom in.	Windows logo key ++ (plus key)
Zoom out.	Windows logo key + – (minus key)
Move Start screen and open apps to monitor on the left.	Windows logo key+Page Up
Move Start screen and open apps to monitor on the right.	Windows logo key+Page Down
Show Shut Down Windows menu (from the Desktop).	Alt+F4
Shut Down Windows (from Start screen).	Windows logo key+I, select Power

TABLE 1.2 Windows 8.1 charms

Charm	Key combination
Open the Charms bar.	Windows logo key+C
Open the Settings charm.	Windows logo key+I
Open the Search charm to search files.	Windows logo key+F
Open the Search charm to search everywhere.	Windows logo key+Q
Open the Search charm to search settings.	Windows logo key+W
Open the Devices charm.	Windows logo key+K
Open the Share charm.	Windows logo key+H

CERT OBJECTIVE Understanding Authentication and Authorization

As with all recent Windows operating systems, you need an account to use Windows 8.1. This is a core requirement for basic *authentication* and *authorization*. Here's the difference between these two terms:

Authentication A user claims an identity with a user account and proves this is the user's identity by authenticating with a password or other authentication method. Authentication methods supported in Windows 8.1 are text passwords, picture passwords, personal identification numbers (PINs), smart cards, and biometric methods.

Authorization After users are authenticated, they are granted access to resources based on their proven identity.

Account Types

There are multiple types of accounts you can use in Windows 8.1. Many of these are the same types of accounts you could use in previous versions, but the Microsoft Live account is new.

Local User Account When a computer is in a workgroup or HomeGroup, users have a local user account stored on the computer. They are able to log on to the local computer only with this account. If users need to log on to other computers, they need an account on each of these computers.

Domain Account When the computer is in a domain, users have an account within the domain. Their account information is stored on a domain controller and managed by Active Directory Domain Services (AD DS). By default, users can use this account to log on to any computer in the domain except for domain controllers.

Microsoft Live Account This is an email address that users can use to access multiple Microsoft services, including Windows Phone and Xbox LIVE. Microsoft hosts hotmail.com, outlook.com, and live.com email services, and you can use an account from one of these services. You can also use any email address after registering it with Microsoft.

Microsoft domain accounts can be used as a Microsoft Live account. Microsoft domain accounts support the use of user principal names (UPNs), which takes the format of *username@domain.extension*. For example, a user named Darril in the GetCertifiedGetAhead.com domain can have a UPN of Darril@GetCertifiedGetAhead.com and use it to log on and for email. This same account name can be used as a Microsoft Live account after registering it with Microsoft.

There are several benefits to using a Microsoft Live account. The primary benefit is that it provides single sign-on (SSO) capabilities for many Microsoft services. You need to sign in only once to Windows and you can then access other Microsoft services without signing in again. However, users might be prompted to provide a password again when starting certain transactions, such as purchasing an app from the Windows Store.

There are four types of local user accounts:

Administrator Account An administrator account is a member of the Administrators group and has full control over the computer. The first user account that is created when Windows 8.1 is installed is an administrator account by default. This is in addition to the built-in Administrator account, which is disabled by default.

Standard Account This is the default account designed for basic use and created as needed. Users have the ability to run applications but not make significant changes to the system. Standard accounts but can be changed to administrator accounts by adding them to the Administrators group.

Guest Account This account is used as a temporary account. It is disabled by default but can be enabled temporarily for a user instead of creating a new account. When the guest account is no longer needed, it should be disabled again.

HomeGroupUser Account This is a built-in account used to connect from one device in a HomeGroup to another device in the same HomeGroup.

Exercise 1.2 shows how to create a local user account. This account is created as a standard account by default. This exercise assumes you're logged in with an administrator account and the Start screen is showing.

EXERCISE 1.2

Creating a Local User Account

1. Access the Settings charm by pressing Windows logo key+I.
2. Select Change PC Settings from the bottom of the display.
3. Click Accounts.
4. Click Other Accounts.
5. Click Add An Account.
6. Click Sign In Without A Microsoft Account (at the bottom of the screen). Ignore the "Not Recommended." warning.
7. Select Local Account at the bottom of the page.
8. Enter the user's name, the user's password (twice), and a password hint. The hint is displayed if the user forgets the password. Click Next.
9. Click Finish.

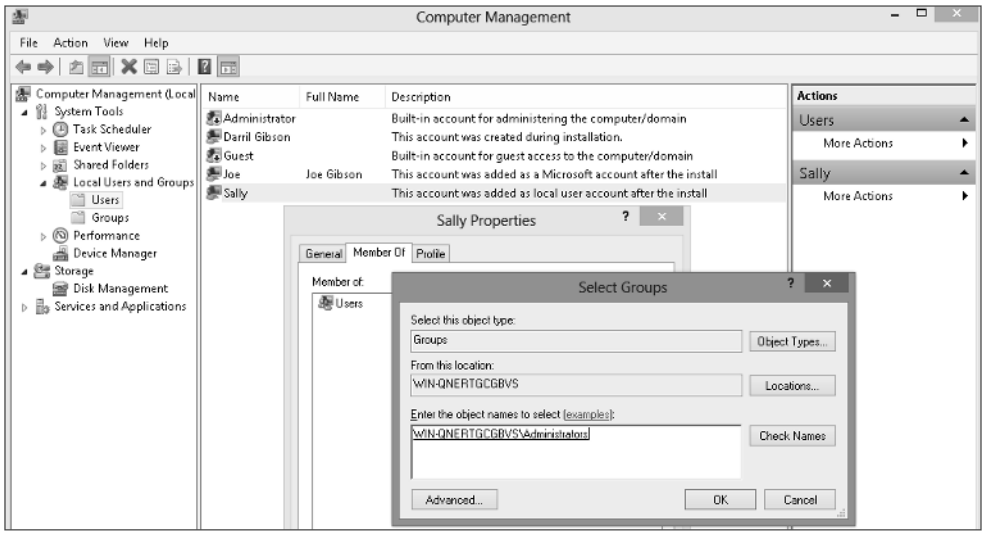
Exercise 1.3 shows how you can change a standard user account to an administrator account. This exercise assumes you're logged in with an administrator account and the Start screen is showing.

EXERCISE 1.3

Changing a Standard Account to an Administrator Account

1. Type **computer**. Select Computer Management. You can also access the Computer Management applet by right-clicking on the Start button.
2. Expand Computer Management ➢ System Tools ➢ Local Users And Groups, and select Users.

3. Right-click the user you want to modify and select Properties.
4. Select the Member Of tab and click Add.
5. Type **administrators** and click Check Names. This should change the computer name to a backslash (\) followed by the word Administrators, as shown in the following graphic. Click OK twice.



CERT OBJECTIVE PIN and Picture Passwords

A new feature available with Windows 8.1 is the ability to use a personal identification number (PIN) or a picture password instead of a standard password. A PIN is a four-digit code similar to a code you might use with an automated teller machine (ATM).

Picture passwords are useful on touch screens. Instead of a user typing a PIN or a password, it's possible for the user to use gestures on the screen as a password. The user picks an image and then enters three gestures as their password. Supported gestures are taps, straight lines, and circles. Picture passwords aren't restricted to only touch screens. The gestures can also be drawn with a mouse.

For example, we've created a picture password with a graphic from the Administrative Tools tiles. For the first gesture, we tapped Performance Monitor. For the second gesture, we drew a circle around Resource Monitor. Figure 1.5 shows the third gesture, where we're drawing a straight line from the Defragment and Optimize Drives tool to the Disk Cleanup tool.

FIGURE 1.5 Using a picture password

When the picture password is configured, users have the choice of logging on with the picture password or with a regular password.

Exercise 1.5 shows how to set up a picture password. This exercise assumes you're logged in with an administrator account and the Account screen is showing.

EXERCISE 1.5

Creating a Picture Password

1. Select Sign-in options from the left side of the screen.
2. Click the Add button under Picture Password.
3. Verify your account information and enter your password on the next screen.
4. On the Welcome Picture Password screen, click Choose Picture.

5. On the Files screen, select a picture. Click Open.
6. If necessary, move the picture until it is in the position you want for your picture password. Click Use This Picture.
7. Draw three gestures to indicate the gestures you want to use for your picture password. You can use any combination of straight lines, taps, or circles. Be sure to remember where you drew the shapes.
8. Redraw all three shapes to confirm your password.
9. Click Finish.

Using the Windows Command Prompt

The *command prompt* is used to type commands from a prompt rather than using the Windows graphical user interface (GUI). Although Windows and other operating systems depend heavily on the GUI for ease of use, the command prompt is extremely important for administrators.

The command prompt may look out of fashion to some, but it is indeed an integral tool needed to accomplish many basic and advanced administrative tasks. In short, the command prompt is not going away. It's stronger than ever. One of the biggest reasons is that anything that can be executed at the command prompt can be scripted and anything that can be scripted can be scheduled or programmed to respond to specific events.

In this section, you'll learn some basics related to the command prompt and perform some exercises to start and use it. The next section covers Windows PowerShell basics, and it's important to realize that many of the same concepts that apply to the command prompt also apply to PowerShell.

Launching the Command Prompt

This is a participative sport. You can't just read about it—you need to get your hands on the keyboard and execute these commands to see how they work. Start by launching the command prompt. In Windows 8.1, you can launch it using one of these two methods:

Typing method

1. Type **cmd**. As soon as you start typing, the Search Apps text box appears.
2. Click on Command Prompt to start it.

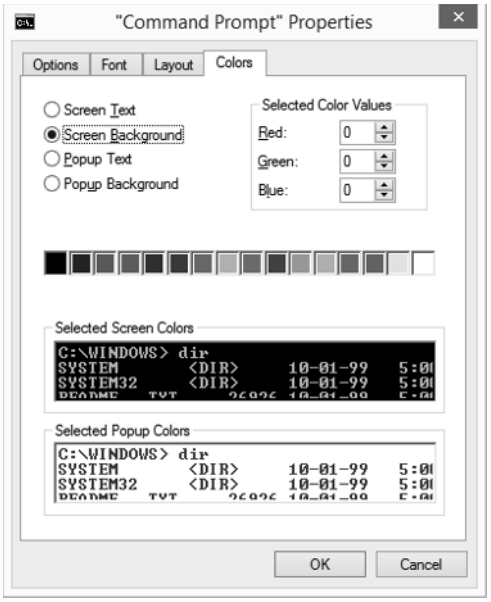
Keyboard shortcut method

1. Press Windows logo key+X to open the Start/preview menu.
2. Click on Command Prompt to start it.

The command prompt starts with white text on a black screen, though you can modify it if you desire. After launching the command prompt, right-click the title bar, select Properties, and then select the Colors tab, as shown in Figure 1.6.

You can select different colors for the background and the text. For example, when showing commands to an audience we often change the background to white and the text to black so that it is easier for participants to see it. Similarly, you can modify the fonts from the Font tab, and we often increase the font size when doing demonstrations so that everyone can easily see the commands—even from the cheap seats.

FIGURE 1.6 Modifying the display of the command prompt



Launching with Administrator Privileges

The command prompt will launch without administrative permissions by default. However, some commands require administrative permissions to execute, so you need to start the command prompt with administrative permissions. Exercise 1.6 shows how. This exercise assumes you're logged in and the Start screen is showing.

EXERCISE 1.6

Launching the Command Prompt with Administrative Privileges

1. Locate the Command Prompt app on the Start page.
2. Right-click over it.
3. The context menu will appear next to the app.
4. Click Run As Administrator.
5. If prompted by User Account Control, click Yes to continue. If you are logged in with a nonadministrative account, you'll be prompted to provide the credentials for an administrative account.

When you start the command prompt normally, it will start in the `\users\username` folder by default, where *username* is the name of the currently logged-on user. When you start it with administrative privileges, the prompt starts in the `\windows\system32` folder by default. Also, the title bar changes from Command Prompt to Administrator: Command Prompt.

Command Prompt Basics

There are many basics related to the command prompt that are important to understand. These apply if you're using basic commands, advanced commands, or commands in scripts. Additionally, many of these same basics apply to PowerShell commands.

Commands Are Modified with Switches

A switch is identified with either a `/` symbol or a – symbol. The `/` symbol was used often in older DOS commands, and the – symbol was common in Unix and Unix derivatives. However, at this point most commands will accept either switch.

As a simple example, you can use the `dir` command to view the contents of the current directory (by typing **`dir`** and pressing Enter). If you want to find a specific file named `file.txt` that might be in any subdirectory, you can use the `/s` switch like this: **`dir file.txt /s`**.

You can view a full listing of switches for any command by entering the command and the `/?` switch.

Help Is Always Available

You can find help for just about any command by typing the command with the help switch (`/?`). In other words, to get help on the `manage-bde` command, you can enter **`manage-bde /?`** at the command prompt.

Also, for many commands you can access help by typing **`help`** and then the command. For example, if you know you can restart the computer into the Windows Recovery Environment (RE) with the `shutdown` command but you don't remember how, you can enter **`help shutdown`**. (In case you're interested, the actual command to reboot and start in the Windows RE is `shutdown /r /o`.)

You can also access help available from the TechNet Command-Line Reference page:

`http://technet.microsoft.com/library/cc754340.aspx`

The `manage-bde` command is a BitLocker Drive Encryption tool. For example, you can type **`manage-bde -status`** to list drives that have BitLocker enabled. A new switch (`-WipeFreeSpace` or `-w`) will erase all data fragments in a volume's free space. BitLocker is explored in more depth in Chapter 9, "Managing Files and Disks on Windows 8.1."

Spelling Counts

At some point in the future, computers will do what we want them to do, not what we ask them to do. For now, they interpret our commands literally. For example, if you want to extend the display to a second monitor you can do so with this command:

```
displayswitch /extend
```

However, if you accidentally enter the following command instead (`displaywitch` instead of `displayswitch`), the command prompt gives a syntax error:

```
displaywitch /extend

'displaywitch' is not recognized as an internal or external command,
operable program or batch file.
```

The error indicates the command is not recognized because it is trying to execute a command called `displaywitch` (which doesn't exist). When things don't work, it's worth your time to read the error message. It will often point you in the right direction.

Commands Are Not Case Sensitive

Most command prompt commands are not case sensitive. In other words, a command is interpreted the same if it's entered as all upper case, all lower case, or a mixture of both

upper case and lower case. For example, the `msiexec` (Windows Installer) command allows you to install, modify, and perform other Windows Installer actions from the command line. You would see the same result if you entered it as `MSIEXEC`, `MsiExec`, or even `mSiExEc`.

It's rare, but occasionally you'll run across a parameter used within a command prompt command that needs to be a specific case. When this true, it will be stressed.

For consistency, commands will be listed in lower case within this book. If the command must be entered using a specific case, it will be accompanied by a comment saying so.

Commands Can Use Wildcards

A *wildcard* is a character that can take the place of other characters. They are often used for search and copy operations. The command prompt includes two wildcard characters: the asterisk (`*`) and the question mark (`?`).

The `*` symbol will look for any instance of zero or more characters in the place of the `*` symbol. For example, if you want to know if you have any text files (with the `.txt` extension) in the current directory, you can use the following command:

```
dir *.txt
```

Similarly, if you want a listing of all files that start with `app` and end with `.exe`, you can use this command:

```
dir app*.exe
```

Note that it will include a file named `app.exe`, if it exists, in addition to any files that that have letters after `app`.

The `?` wildcard will take the place of a single character. It isn't used as often, but you can use it if you're looking for something more specific. For example, if you were looking for any files that had an extension of `.ex` and any third character, you could use this command:

```
dir *.ex?
```

It would not include any files that ended with `.ex` without a third character in the extension. In other words, the `?` wildcard specifies that a single character must exist for a match to occur. This is different from the `*` symbol, which will match for zero or more characters.

Strings with Spaces Need Quotes

Many commands will accept parameters, and when the parameters include spaces, the parameter usually needs to be enclosed in quotes.

For example, imagine you were using the `msiexec` command to install an application named `success for me.msi`. The filename includes spaces, so you need to include the entire filename with quotes like this:

```
msiexec /i "success for me.msi" /qb
```

In this example, the `/i` switch indicates that the MSI file should be installed. The `/qb` switch specifies a basic user interface level that will display a progress bar and progress messages if the MSI file includes them. Notice the name of the MSI file is included in quotes because it has spaces.

However, the following command (without the quotes) would be interpreted quite differently:

```
msiexec /i success for me.msi /qb
```

In this second example, `success` will be interpreted as the name of the file to install. Even if a file exists named `success` without an extension of `.msi`, the `msiexec` command won't know how to interpret the word `for`, which comes right after the space. This causes an error.

You can occasionally get away without using the quotes. For example, if you're at the root of C (C:\) and want to change to the `Program Files` folder (which has a space), the following command will work:

```
cd program files
```

This is only because the `CD` command has been programmed to accept it, but that wasn't always the case. You could also enter this command like this:

```
cd "program files"
```

Doskey Saves Typing

Doskey is a utility that is constantly running in the background of the command prompt and can be very valuable—if you know how to use it. Every time you enter a command in a command prompt session, it is recorded by Doskey and can be recalled.

For example, suppose you're testing connectivity with a server using the following `ping` command:

```
ping dc1.training.getcertifiedgetahead.com
```

You could execute the command and then realize the NIC wasn't configured correctly, or the host cache needed to be cleared with `ipconfig /flushdns`, or something else needed to be done. After resolving the issue, you want to execute the `ping` command again. Instead of typing it in from scratch, you can simply use the up arrow to recall it and press Enter, and you've executed it again (without retyping it).

The up arrow can be used to retrieve any previous command that you've entered in the current command prompt session (up to the limit of the buffer, which is rather large). This can be valuable when you're entering very long commands or even short commands if you use the hunt-and-peck method of typing.

Typos are also common at the command line. However, you don't have to retype the entire command. If you get an error, you can use the up arrow to recall the command, and then use the left and right arrows to position the cursor where you want to modify the text. Make your corrections and press Enter, and the corrected command executes.

You can also use the F7 key to display a pop-up window that shows a history listing. It includes all of the commands you've entered in the current session. You can then use the up or down arrows to select the desired command or press the Esc key to dismiss the window.

Doskey is a command prompt utility itself and includes some commands you can use. For example, if you want to view all the commands that have been entered in the current session, enter **`doskey /history`**.

By default, the system includes a buffer size of 50 commands. If you need more, you can modify the buffer size. For example, the following command changes the buffer to 99:

```
doskey /listsize=99
```

Using *runas*

You can also start a separate instance of the command prompt with the `runas` command. This allows you to start it in the context of a specific user account. The basic syntax is as follows:

```
runas /user:domain\username [/profile or /noprofile] application
```

Table 1.3 shows some common switches used with the `runas` command.

TABLE 1.3 `runas` switches

Feature	Key combination
`/profile`	Specifies that the user's profile should be loaded. This is the default. It cannot be used with `/netonly`.
`/noprofile`	Specifies that the user's profile should not be loaded. This causes the application to load more quickly, but can cause some applications to malfunction.
`/env`	Specifies that the current environment will be used instead of the specified user's environment.
`/netonly`	Use if the credentials specified are for remote access only. This cannot be used with `/profile`.
`/savecred`	Use credentials previously saved by the user.
`/smartcard`	Use if the credentials are to be supplied from a smart card.

Say you are logged on as a regular user but have an account named JoeAdmin with administrator privileges within a domain named GetCertifiedGetAhead.com. You can use the `runas` command to run the command with the administrator account with this command:

```
runas /user:getcertifiedgetahead\joeadmin /profile yourapp.exe
```

System Variables

Many *system variables* are available within Windows 8.1, and you'll see these often when using the command prompt and Windows PowerShell. System variables are settings that are global to all users, not just the currently logged-in user (the latter are commonly referred to as *environment variables*). System variables are useful in identifying specific values about the environment. As a simple example, every computer has a computer name, but the computer names are different. The system variable `%computername%` holds the value of the local computer's computer name.

System variables are easy to identify in text. They always start and end with a percent symbol (%). An easy way to view the value of any variable is by using the `echo` command in the following format:

```
echo %variablename%
```

Table 1.4 shows many of the commonly used system variables and their value. You use the `echo` command with each to see its value.

TABLE 1.4 Commonly used system variables

Variable	Value
`%windir%` `%systemroot%`	Both `%windir%` and `%systemroot%` identify the folder where Windows was installed, typically `C:\Windows`.
`%systemdrive%`	The folder where the system boot files are located, typically `C:\`.
`%computername%`	The name of the local computer.
`%username%`	The name of the user logged on to this session.
`%date%`	Holds the value of the current date in the format ddd `mm/dd/yyyy`. The first three letters are an abbreviation of the day of the week, such as Mon, Tue, Wed, and so on. The remaining format is all numbers with `mm` for the month, `dd` for the day, and `yyyy` for the year.

%time%	Holds the value of the current time in a 24-hour format as hh.mm.ss.ms.
%errorlevel%	Indicates whether the previous command resulted in an error. If it didn't result in an error, the value is 0.
%ProgramFiles%	Points to the location of the Program Files folder, which is normally C:\Program Files.
%Public%	The location of the Public folder, typically C:\Users\Public.

Commands and Paths

When you execute commands from the command prompt, the system needs to know where to find the command. It first tries to execute it in the current path and then looks for it in predefined paths. A path identifies a location on the hard drive.

For example, when you first launch the command prompt, it will start in the c:\users\%username% path by default, where %username% will be replaced with the username you're logged on with. If you launch it with administrator privileges, it will start in the c:\%windir%\system32 folder.

If you execute a command (such as ipconfig), it will look for the command in the current folder first. If it isn't located in the current folder, the system will search the folders identified in the predefined paths. If the command isn't located in any of the known paths, you'll see an error. For example, if you type **xyz** and press Enter, you'll see this error:

```
'xyz' is not recognized as an internal or external command,
operable program or batch file.
```

Documentation commonly uses the terms *folders* and *directories* interchangeably. In the early DOS days, they were almost always called directories. When the Windows GUI came out, they were referred to as folders because the icon looks like a folder. It matches a metaphor users could easily understand; that is, files are placed in folders in the real world and they are placed in folders in Windows. However, there is no difference between a folder and a directory; both terms mean the same thing.

If the system didn't have predefined paths, it would search only the current folder, and commands would be a lot harder to enter and execute. However, the system starts with several predefined paths. On a new installation, this path includes all of these directories:

- C:\Windows\system32
- C:\Windows

- C:\Windows\System32\Wbem
- C:\Windows\System32\WindowsPowerShell\v1.0\

You can execute the path or the set path command to view the predefined path for your system. Some applications will modify the path, and you can also modify the path yourself.

Identifying Executables

When you execute the set path statement, you also see something else valuable: a list of file types that are known to be executables. Some files can be executed or run, whereas others are simply data files used by executable programs.

So what is an executable? It is any file that can be run. For example, you can run the ipconfig.exe file because it's an executable. If a file named ipconfig.txt existed, it could not be executed. The extension .exe identifies the first file as an executable, whereas the extension .txt identifies the second file as a text file.

Known executable files are defined by the system variable pathext (path extension). The following are the path extensions, or file extensions, known to be executables. They are listed as they appear on this author's system:

- .com (executable file)
- .exe (command file or executable file)
- .bat (batch file)
- .cmd (command file)
- .vbs (Visual Basic Script file)
- .vbe (Visual Basic executable file)
- .js (JavaScript file)
- .jse (JavaScript executable file)
- .wsf (Windows Script file), .wsh (Windows Script Host file)
- .msc (Microsoft Management Console file)

So when you execute the ipconfig command, it searches for a file that starts with ipconfig and ends with one of the identified known executable extensions. Since the ipconfig program ends with .exe, the ipconfig.exe command is located and executed.

Modifying the Path to Executables

If you need to modify the known paths of the system, you can do so with either the set path statement or via the GUI. For example, you may have an executable in the c:\app

path, and you may want this path included in the `path` variable. You can use one of these methods.

Set Path Command

Before modifying the path, take a look at what it currently is with the command `set path`.

You can use the `set path` statement to modify the path to include the `c:\app` folder with the following command:

```
set path = c:\app
```

After you modify the path, view the current path by executing `set path` again. You'll notice that there are two paths currently set—the original default path and another `path = c:\app` path that you just created, as shown in Listing 1.1 and Listing 1.2. Listing 1.1 is what appears before you execute the `set path = c:\app` statement. Listing 1.2 shows what appears after you execute the statement, with the next path highlighted.

Listing 1.1: Output of the set path statement

```
Path=C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;
C:\Windows\System32\WindowsPowerShell\v1.0\
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
```

Listing 1.2: Output of the set path statement after appending the path

```
Path=C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;
C:\Windows\System32\WindowsPowerShell\v1.0\
path = c:\app
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
```

This behavior of the `set path` statement is different in Windows 7 and Windows 8.1 than it was in previous versions of Windows. In previous versions, if you used the `set path = c:\app` statement, it would overwrite the previous path and only `c:\app` would be included in the path. However, when you execute the `set path` statement in Windows 7 and Windows 8.1, it appends, not replaces, the current path.

This modified path will be modified for only the current session. In other words, if you exit the command prompt window, launch it again, and enter **`set path`**, you'll see only the original system default path.

Modifying the Path with the GUI

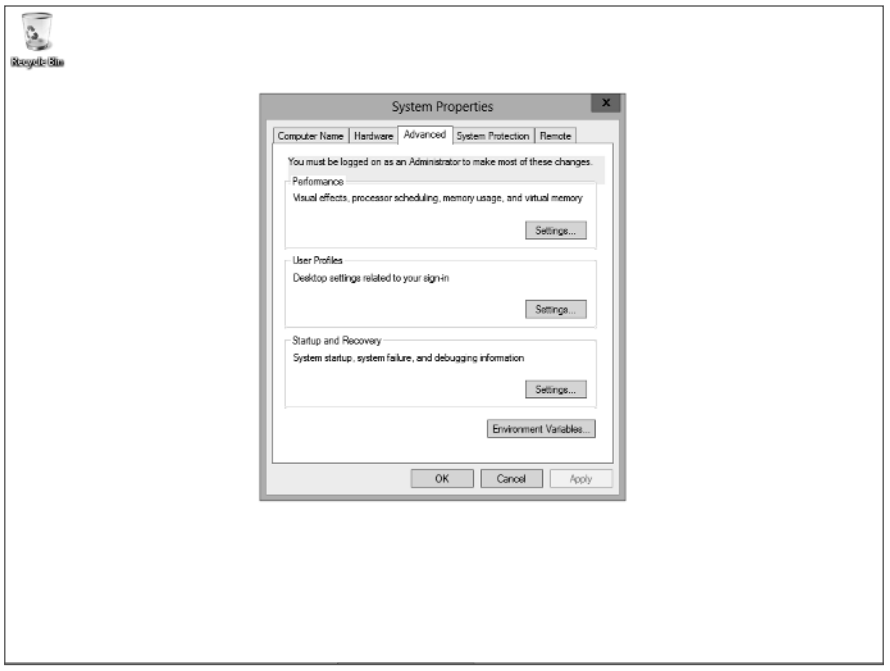
The path can also be modified by modifying the system variables in the GUI, as shown in Exercise 1.7.

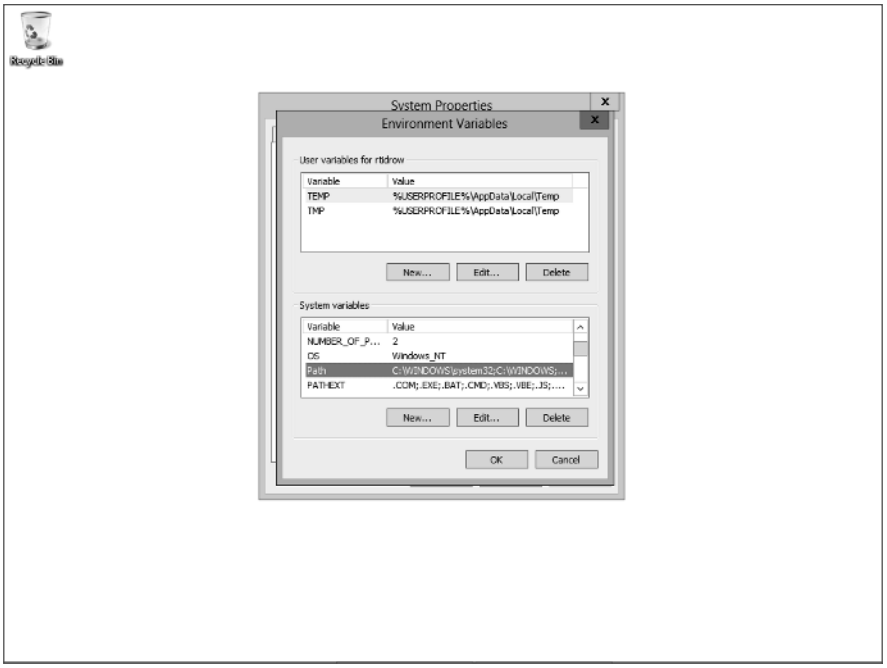
EXERCISE 1.7

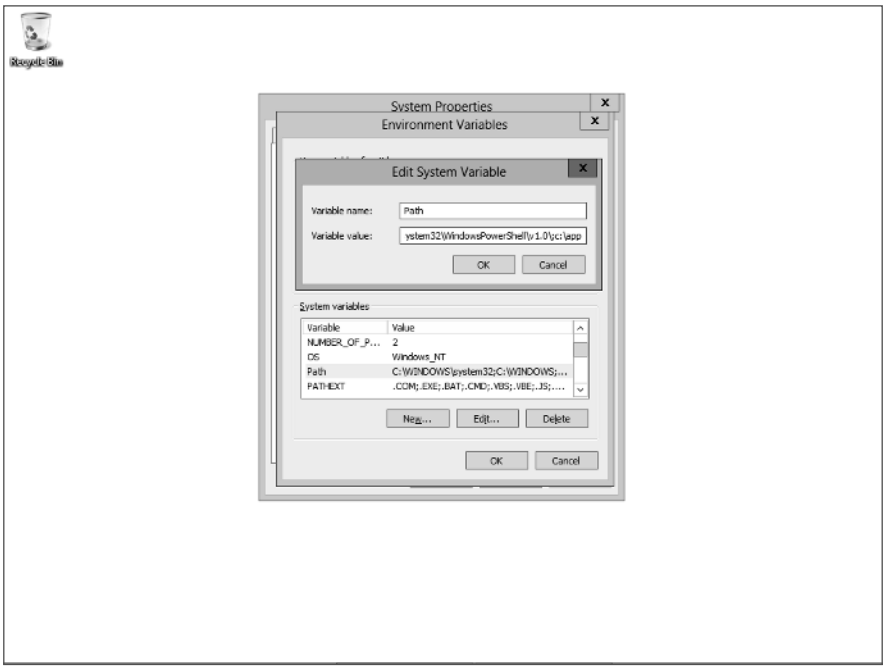
Modifying the Path System Variable

1. Drag your mouse to the lower-left corner of the screen and right-click on the Start button.
2. When the Start menu appears, select System.
3. Select Advanced System Settings. Ensure the Advanced tab is selected.
4. Click the Environment Variables button.
5. Scroll to Path in the System Variables pane. Select Path and click Edit.
6. Scroll to the end of the text (or press Ctrl+End) in the Variable Value text box.

   Warning: Make sure you separate each path with a semicolon. If the semicolon is omitted, the path will be interpreted as a part of the previous path and the previous path will no longer be accessible.
7. Enter a semicolon (**;**) and then the path you want added, as shown here:







8. Click OK three times to dismiss all of the windows. If you launch the command prompt, you'll see that the new path has been appended to the system path.

Changing the Current Path with *cd*

When you open the command prompt, the default path is `c:\users\%username%`, with `%username%` replaced with your username. For example, if you logged on with a username of Darril, the default path would be `c:\users\darril`.

You can change the path with the `cd` command (short for change directory):

- `cd \` will take you to the root of the current drive.
- cd .. will take you up one folder.
- `cd foldername` will take you into the folder specified as long as the folder is in the current folder.
- `cd \1st folder\2nd folder\3rd folder` will take you to the third folder, as long as the full path is valid.

Use Exercise 1.8 to see this in action.

EXERCISE 1.8

Using the *cd* Command

1. Launch a command prompt and note the current path. Unless you launched it with administrative privileges, it will point to a directory with your username in the `c:\users` directory.
2. Type **`cd ..`** and press Enter. This will take you up one folder to the `c:\users` folder.
3. Type **`cd \`** and press Enter. This will take to you the root of `c:\`.
4. Type **`dir`** and press Enter to view the contents of the root.
5. Type the following command to change the directory to the `windows\system32` folder:

   ```
   cd \windows\system32
   ```

   Note that the backslash (\) before `windows` causes the path to start from the root of the C drive.
6. Type **`dir`** and press Enter to view the contents of this folder.
7. Press the up arrow twice to recall the `cd \` command, and press Enter. You'll be returned to the root.
8. Enter the following command to change to the `system32` folder using the `windows` variable:

   ```
   cd %windir%\system32
   ```

Changing the Current Path with File Explorer

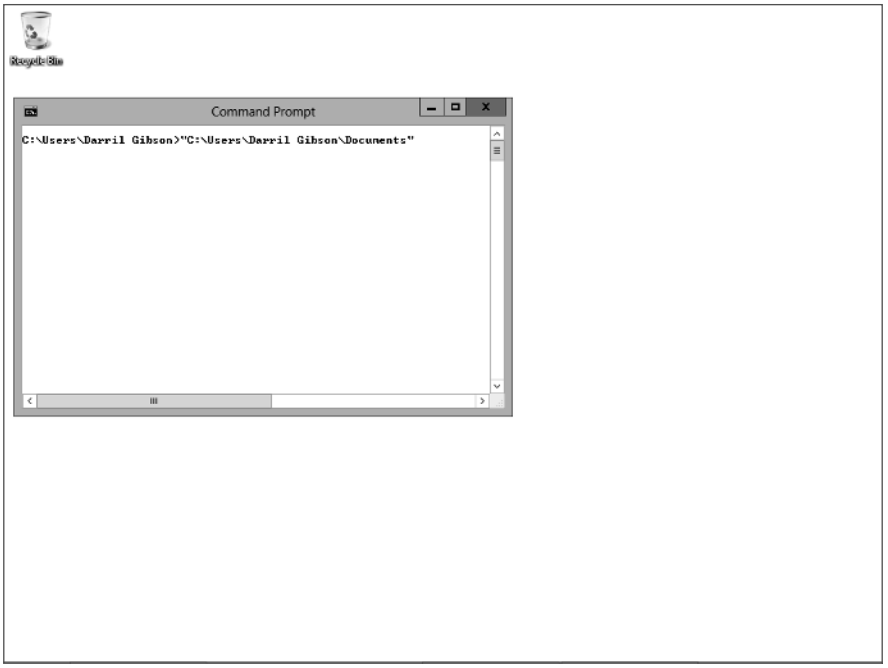
A neat feature that's available with the command prompt is the ability to use drag and drop from File Explorer to copy the path. This technique doesn't change the directory, but you can use it to make things a lot easier.

This feature is not available when you run the command prompt with administrator privileges. Exercise 1.9 shows how this technique can be useful.

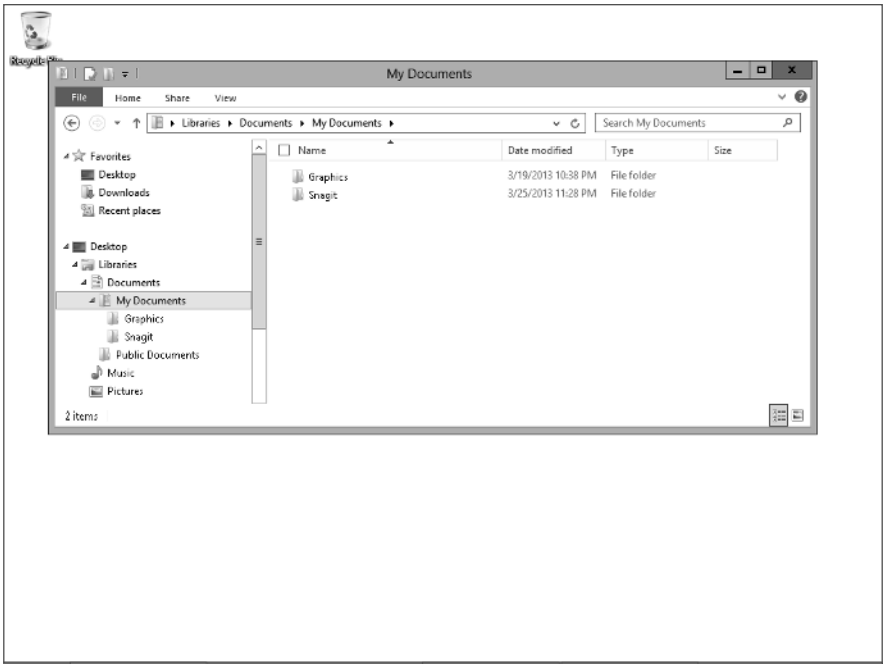
EXERCISE 1.9

Using Drag and Drop with the Command Prompt

1. If a command prompt is not already open, launch a command prompt.
2. Launch File Explorer and browse to the `Desktop\Libraries\Documents\My Documents` folder.
3. Position File Explorer and the command prompt window side by side.
4. Click `My Documents` in File Explorer, drag it to the command prompt window, and release it. You'll notice that the path is now displayed in the window. Your display will be similar to this:



EXERCISE 1.9 *(continued)*



5. In the graphic, the username is Darril Gibson, so the default path of Darril's Libraries starts as `C:\Users\Darril Gibson` and the actual path to the `Libraries\Documents\My Documents` folder is `C:\Users\Darril Gibson\Documents`. When the `My Documents` folder is dragged and dropped into the command prompt window, the path is typed out. However, you're not finished yet.

6. Use the left arrow (or the Home key) to position your cursor to the left of all the text. Type **cd** and a space to modify the command, and press Enter. Your path will be changed to the equivalent of the `My Documents` folder.

The previous exercise showed how you can easily change the path using File Explorer, but you can also launch the command prompt to any folder's location from File Explorer. With File Explorer open, press the Shift key, right-click the folder, and select Open Command Window Here. The command prompt will be launched with the directory set at the same folder as File Explorer.

Using Copy and Paste with the Command Prompt

At first glance it looks like you can't copy and paste to or from the command prompt. It doesn't work exactly as it does in Windows, causing a lot of people to assume you can't, but doing so is possible.

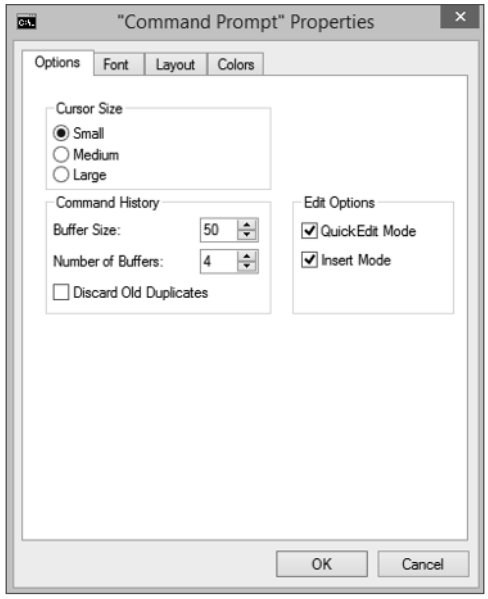
You can actually copy and paste by default, but enabling the QuickEdit mode makes it a little easier. This can be valuable when you are testing commands that you want to paste into a script file. Exercise 1.10 shows how copy and paste works from the command prompt, how to enable QuickEdit mode, and its benefits.

EXERCISE 1.10

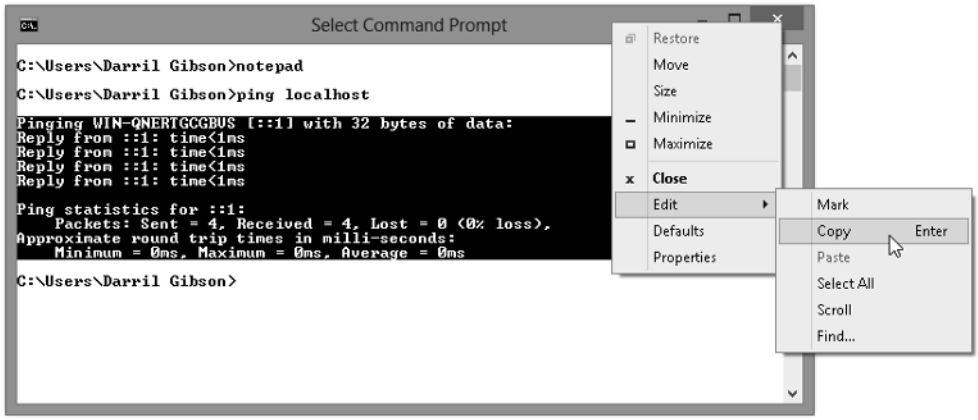
Using Copy and Paste from the Command Prompt

1. Open a command prompt window, type **notepad**, and press Enter. This will launch an instance of Notepad that you'll use to copy and paste text to and from the command prompt window.
2. Type **ping localhost** in the Notepad window. Press Ctrl+A to select the text and Ctrl+C to copy it.
3. Click in the command prompt window. Right-click and select Paste. You'll see the command pasted into the command prompt window. Press Enter and the command will execute.
4. Right-click the command prompt title bar and select Edit ➢ Mark.
5. Use your mouse to highlight the output from the `ping` command. When the text has been highlighted, press Enter to copy it to the Clipboard.
6. Click in Notepad and press Ctrl+V to paste the output into the text file.
7. Enable QuickEdit mode with these steps:
   a. Right-click the title bar and select Properties. If necessary, select the Options tab.
   b. Select the check box next to QuickEdit Mode, as shown in the following graphic. Click OK.

EXERCISE 1.10 *(continued)*



8. Now you can select text within the command prompt window without selecting Mark. Simply use the mouse to select the text and press Enter. The following graphic shows how text can be selected by highlighting it and using the menu to select Edit ➢ Copy, you can simply press the Enter key.



Capturing the Output

It's common to need to capture the output of commands executed at the command prompt. This can easily be done by redirecting the output to a text file using the > symbol.

For example, you may want to document the current status of all of the services in a text file so that you can review them later. The following command will list all of the services on the system, including the current state, such as running or stopped:

```
sc query state= all
```

The `state= all` option must be entered without a space after `state` and with a space after the equals (=) symbol. Other combinations result in an error.

If you execute this command at the command prompt, you'll see that output quickly scrolls off the screen. However, you can redirect the output to a text file by modifying the command like this:

```
sc query state= all > servicestatus.txt
```

You can then view the file in Notepad with the following command:

```
notepad servicestatus.txt
```

Of course, you can also redirect the file to any location by including the path in the filename. The following command creates and stores the file in the `c:\data` folder:

```
sc query state= all > c:\data\ servicestatus.txt
```

Creating a Batch File

A *batch file* is a listing of one or more command prompt commands within a text file. When the batch file is called or executed, the commands are executed. The best way to understand this is to do it. Although there are sophisticated text editors you can use, Notepad will work.

Exercise 1.11 shows the steps used to create a simple batch file, and then it builds on the simple batch file to add extra capabilities. Ultimately, you'll end up with a batch file you can use to create a list of all services on a computer and their current state. The batch file can then copy the file to a share on another computer.

EXERCISE 1.11

Creating a Batch File

1. Launch the command prompt.
2. Type **`notepad servicestatus.bat`** and press Enter. Notepad will launch, and since a file named `servicestatus.bat` doesn't exist, you'll be prompted to create it. Click Yes.

   Note that the file will be created in the same directory in which the command prompt window was launched.
3. Type in the following text in Notepad:

   ```
   @echo off
   echo Hello %username%. Today is %date%.
   ```

   Press Ctrl+S to save the file, but don't close it.
4. Return to the command prompt, type **`servicestatus`**, and press Enter. Notice that because the batch file is considered one of the executable types, it is automatically located and executed. You'll see a greeting with today's date. This is okay but not very useful.
5. Open Notepad and type the following text after your first two lines:

   ```
   sc query state= all > %computername%servicestatus.txt
   ```

   This command will create a list of updates currently installed on this system and store the updates in the file named *`computername`*`servicestatus.txt` (the computer name will be different for each computer where it is executed). Press Ctrl+S to save the file.
6. Return to the command prompt, press the up arrow, and press Enter to execute the batch file again. Notice that it almost seems as though it's the same as before. A greeting appears, it pauses for a second or two, and then the command prompt returns.
7. Provide some user feedback by adding the following line to the batch file:

   ```
   echo A list of services is stored in the
   %computername%servicestatus.txt file.
   ```

   Press Ctrl+S to save the file.

EXERCISE 1.11 *(continued)*

8. Access the command prompt, press the up arrow to retrieve the last command, and press Enter to view the difference. Notice that instead of `%computername%`, your actual computer name is used.

9. You could also open the file for the user by adding this command to the batch file:

   ```
   %SystemDrive%\Users\%Username%\Documents\%computername%servicestatus.
   txt
   ```

   If you add this to the batch file to test it, make sure you remove it before moving on.

10. If you want to copy the file to a network share (such as a central computer that will hold files from multiple computers), you can use the `net use` command. For this set of commands, we're assuming you have a share named services on a server named srv1 that you can access in the network and that you have permissions to copy the file. You're accessing it using the `\\srv1\services` UNC path. You can use any server (or another Windows 8.1 computer) and any share that has appropriate permissions.

    ```
    net use z:  /delete
    net use z: \\srv1\services
    copy %computername%servicestatus.txt z:
    net use Z: /delete
    ```

    The first command ensures that the Z drive isn't already mapped to something else. The next command maps the Z drive to the UNC path using the `\\serverName\shareName` format. The third line copies the file to the Z drive using the `copy` command, and the fourth line returns the environment to normal.

The UNC path always uses the format of `\\`*`servername`*`\`*`sharename`*. The *`servername`* is the name of any computer that can share folders and can be another Windows 8.1 system.

Now that the file is created, it can be configured to execute automatically based on a schedule. Windows 8.1 includes the built-in *Task Scheduler* that can be used to schedule tasks.

Real World Scenario

Preparing a Classroom

As a trainer, this author (Darril) teaches many technical courses and the courses require different student materials. Since I'm never sure what previous students might have done, I often refresh the files on computer systems before a class.

Walking around the room with my USB and touching as many as 18 student computers could easily take an hour or so. However, I've created scripts to load these materials onto the systems for the different courses. I simply turn on the student computers and launch my script from the instructor computer. A few minutes later, I verify that the script ran successfully and I'm done.

The script ensures that the process is always exactly the same for each student. Additionally, it saves me a lot of time and effort.

Using Windows PowerShell and the PowerShell ISE

Windows PowerShell is an extensible version of the command prompt. It is integrated with the Microsoft .NET Framework, which gives it extensive capabilities well beyond the command prompt. Windows 8.1 comes with both Windows PowerShell and the *Windows PowerShell Integrated Scripting Environment (ISE)*.

The PowerShell commands can be used to perform and automate many administrative tasks, such as managing services, managing event logs, modifying the Registry, and interacting with Windows Management Instrumentation (WMI). PowerShell was designed with scripting in mind so you'll find that you can create elegant scripts that can automate many of your administrative tasks.

One of the challenges with PowerShell is that it is so rich in capabilities and features that it can be intimidating. However, you can start using it without understanding everything about it. You can learn as you go.

You'll get the most out of this section if you're able to launch PowerShell, execute the commands, and see them in action. You can launch PowerShell normally or with administrative permissions. However, we strongly recommend that you launch it normally unless you specifically need to use administrative permissions (such as when you need to change the execution policy as shown later in this chapter).

Just as the command prompt has its own environment, PowerShell too has its own environment. It also has a distinctive look and feel. Launch it and see for yourself by following these steps from the Start screen:

1. Type **powershell**. As soon as you start typing, the Search Everywhere text box appears.
2. Click on Windows PowerShell to start it.

In addition to seeing Windows PowerShell in the search results, you'll see the *Windows PowerShell ISE*, which is the integrated scripting environment (ISE). If you're running a 64-bit system, you'll see the following four choices:

- Windows PowerShell (x86) (32-bit version)
- Windows PowerShell ISE (x86) (32-bit version)
- Windows PowerShell (64-bit version)
- Windows PowerShell ISE (64-bit version)

We haven't run across any issues using the 64-bit Windows PowerShell and ISE for any scripts, but if you need to step down to the 32-bit version, you can.

You can also launch PowerShell from the command prompt with the following command: **`powershell_ise.exe`**. If you add the `-noprofile` switch (**`powershell_ise.exe -noprofile`**), the command will not run the default profile script and PowerShell will start quicker.

If you need to run PowerShell with administrative permissions, right-click the app when it appears and select Run As Administrator from the alt context menu.

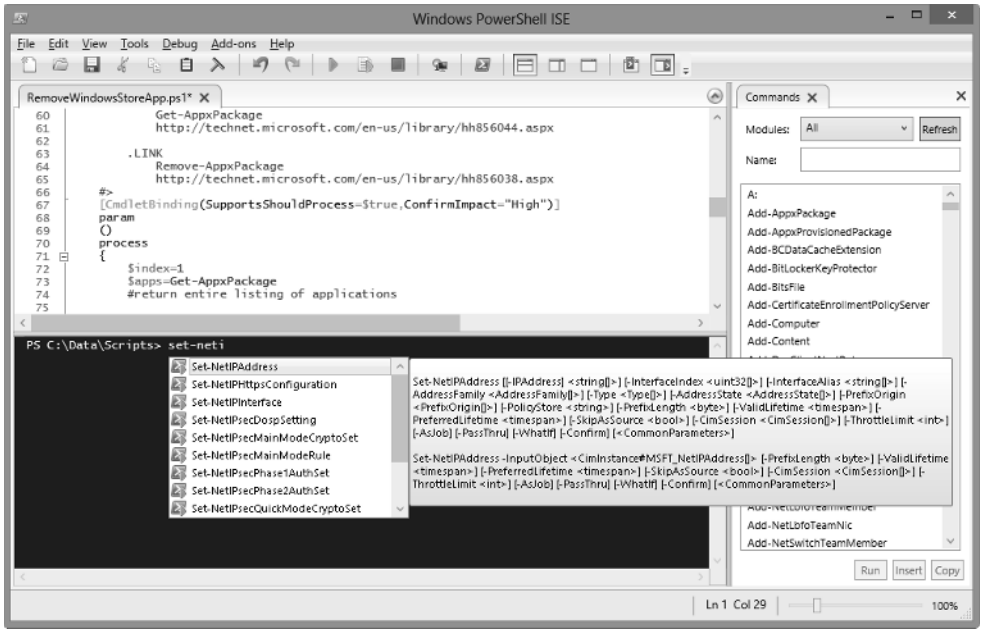
Although the look and feel of PowerShell is different from the command prompt, you can right-click over the title bar and you'll see you have access to the same menu as the command prompt. You can also copy and paste to and from the PowerShell window and enable QuickEdit mode, so this is a little easier. Copy and paste works exactly the same as shown with the command prompt earlier in this chapter.

Windows PowerShell ISE

The Windows PowerShell Integrating Scripting Environment can be used just as easily as the Windows PowerShell prompt, but it gives you a lot more capabilities. Figure 1.7 shows the PowerShell ISE.

You can see that it has a familiar menu across the top just like any other Windows application. Below the menu are several icons. As with other applications, you can hover over the icon to see the name. If you're working with scripts, the most common icons you'll use are Save, Run Script (green arrow), and Run Selection. The Run Script icon is the right arrow icon, and if you highlight any text, the icon next to it will be enabled so that you execute the selected text.

FIGURE 1.7 Windows PowerShell ISE



In Figure 1.7, we've opened a script and opened the script pane by clicking the down arrow labeled Script. The icon changes to an up arrow and clicking it will hide the script.

The bottom pane is the PowerShell prompt. Any commands you can enter at the PowerShell prompt can also be entered at this prompt in the ISE. You can even save the script you're writing and execute it using the full path and name from this prompt.

An extremely useful feature is IntelliSense. In Figure 1.7, we've typed **`set-neti`** and IntelliSense is displaying a list of cmdlets that start with `set-neti`. Additionally, IntelliSense is showing the specific syntax required for the `Set-NetIPAddress` cmdlet.

At the right is a listing of PowerShell commands. It defaults to show all PowerShell modules (all commands), but you can filter it to show only specific commands based on the module name. For example, you can select NetAdapter and it will show only the commands related to network adapters. You can also type the name of a command in the Name text box and get details on a specific command.

At the bottom right, you'll notice some status information that can be useful if you're working on scripts. Right now, we have the cursor at the beginning of line 5, and the status shows it at "Ln 1 Col 29." Sometimes, an error message will indicate there's an error at a specific line and column and you can use this information to determine the exact location of the error.

Last, you can easily zoom in to make the text bigger or smaller. Currently, it's set at 100 percent, but the pointer can be moved to the right to increase the size or to the left to decrease the size.

Getting Help on PowerShell

You'll find a rich set of help available to you within PowerShell. Many people are uneasy about using the help provided from the command prompt or PowerShell. However, this help has become much richer in recent years. If you overlook it, you'll be missing a lot. Here are some of the commands you can use to get help:

`Help` or **`Get-Help`** Either will display generic help information on how to execute help commands and their results. Both `Help` and `Get-Help` will work.

`Get-Help` ***`commandName`*** You can request help on any PowerShell command by simply typing **`Help`** followed by the command name. The command can be a cmdlet, a function, or an alias. When you request help for an alias, it provides help on the associated cmdlets. For example, if you type **`Get-Help dir`** it will return help on the `Get-Childrens` cmdlet since `dir` is an alias for `Get-Childrens`.

`Get-Help` ***`commandName`*** **`-Examples`** Examples are only a few keystrokes away just by adding the `-examples` switch to your `Get-Help` request. For example, you can use the following command to see examples with descriptions of the `Get-Service` command. There are several pages, so the `More` command will allow you to view a page at a time. Press the spacebar to scroll to the next page.

```
Get-Help Get-Service -Examples | More
```

`Get-Help` ***`commandName`*** **`-detailed`** The `-detailed` switch can be used to provide more detailed help than the basic `Help` command. It will include examples.

`Get-Help` ***`commandName`*** **`-full`** The `-full` switch provides all of the available help on the topic. This will often provide more information on parameters used within the command.

Updating Help

Some of the newer advanced PowerShell cmdlets have only basic help by default. For example, if you use the following command you will only get basic help by default:

```
Get-Help Set-NetIPAddress
```

The output will include a Remarks section that indicates the `Get-Help` cmdlet cannot find advanced help. However, you can update the basic help with the `Update-Help` cmdlet. You can use Exercise 1.12 to update the help topics. This exercise assumes you're logged in, the Start screen is showing, and you have access to the Internet.

EXERCISE 1.12

Updating Help

1. Type **powershell**. Right-click Windows PowerShell and select Run As Administrator. If prompted by User Account Control, click Yes.
2. Enter the following command to start the process:

   ```
   Update-Help
   ```

3. This process will take some time but you'll see progress indicators showing that the help files are being updated. When it completes, close PowerShell.

Many help files are available within PowerShell that provide more information about specific topics (after updating help). They are referred to as "about" topics. For example, if you want information on the `pipelines` command, you can enter **`Help about_pipelines`**. To see a full listing of all of these "about" topics, enter **`help about`**.

PowerShell Command Overview

PowerShell includes three types of commands that can be executed or scripted:

Cmdlets PowerShell *cmdlets* are built-in commands that come with PowerShell. They are different from command prompt commands in that they are tightly integrated with Microsoft's .NET Framework and thus provide a much richer set of capabilities. You can think of them as mini-programs. Many cmdlets can accept parameters and return values. Values can be displayed, assigned to variables, or passed to other cmdlets or functions.

For example, the `Get-Command` cmdlet will retrieve a list of all PowerShell commands. You can modify the command with parameters to modify the results. The following three commands can be used to retrieve a list of only the cmdlets, only the aliases, and only the functions. In these examples, the `Get-Command` cmdlet is being used with the `-CommandType` switch and `cmdlet`, `alias`, and `function` are the parameters passed to the `Get-Command` cmdlet.

```
Get-Command -CommandType cmdlet
Get-Command -CommandType alias
Get-Command -CommandType function
```

PowerShell commands can be entered with any case. However, they are commonly displayed with camel case (capping the first letter of a word) to make them easier to read. You'll see them displayed this way in this chapter and elsewhere in the book when they are used.

Aliases An *alias* is another name for a cmdlet. Many command prompt commands have been rewritten as PowerShell commands. Although the actual PowerShell command is different from the command prompt command, many aliases have been created so that you enter the command prompt command and it will launch the PowerShell command. For example, the `cd` command prompt command is used to change the current directory. The PowerShell cmdlet is `Set-Location`, but `cd` is recognized as an alias for `Set-Location`.

Try it. These two commands will achieve the same result of changing the current path to the root of the C drive.

```
CD \
Set-Location \
```

All command prompt commands have not been rewritten. If you enter a command such as `path` that works at the command prompt, you'll see it doesn't work here. You can enter the `Get-Alias` command to get a list of all the aliases supported within PowerShell. Additionally, some longer cmdlets have been rewritten as aliases. For example, the `Get-WMIObject` cmdlet has an alias of `gwmi`.

Functions A *function* is a type of command that you can run within PowerShell or PowerShell scripts. They are very similar to cmdlets. They can accept parameters and return values that are displayed, assigned to variables, or passed to other functions or cmdlets.

A commonly used function is `Help`, which provides help on PowerShell topics and concepts. When executed without a parameter, it provides one set of information. When a valid PowerShell command is added as a parameter (such as `Help Get-Command`), it provides specific help on the command. Another function that is commonly used is the drive letter (such as `C:`, `D:`, and so on). It will change the current PowerShell prompt location to the named drive. You can also create your own functions.

Any of these commands can be executed from the PowerShell prompt or embedded into PowerShell scripts. Additionally, many server applications (such as Internet Information Services [IIS], used to serve web pages, or Exchange Server 2013, used for email) are heavily intertwined with PowerShell commands. In other words, what you learn here for Windows 8.1 will be useful when you're managing servers as well.

Verbs and Nouns

PowerShell cmdlets are composed of verbs and nouns in the format of `verb-noun`, with the dash (-) separating the two. You may remember from your English classes that verbs denote action and nouns are things. Common PowerShell verbs are `Get`, `Set`, and `Test`. You can combine them with nouns to get information on objects or to set properties on objects.

Earlier, you saw the `Set-Location` cmdlet, which is similar to the command prompt change directory command (`cd`). `Set` is the verb and `Location` is the noun. Similarly, `Get-Service` uses the verb `Get` to retrieve information on services, and `Out-File` uses the verb `Out` to send information to a file.

If you can remember `Get`, `Set`, and `Test`, you're halfway there because PowerShell will give you a lot of clues. Try this. Type **`Get-`** and then press the Tab key. PowerShell will display each of the legal commands that can be executed starting with `Get-` (from `Get-ACL` to `Get-WSManInstance`).

You can do the same thing with the `Set` and `Test` verbs. Type **`Set-`** and press Tab to see all of the objects (nouns) that can have properties set: `Set-ACL` to `Set-WSManQuickConfig`. If you type **`Test-`** you'll be able to tab through the choices, from `Test-AppLockerPolicy` to `Test-WSMan`.

Functions don't necessarily follow the verb-noun format but instead are just commands. For example, the `E:` function will set the current drive to `E:` by calling the `Set-Location` cmdlet using the parameter `E:`. However, the `Clear-Host` function does use a verb-noun format to indicate the host screen (the noun) is being cleared (the verb).

Sending Output to a Text File

Many times you'll want the output of the PowerShell command to be written to a file instead of the screen. This can be especially useful when you're creating documentation. While learning, you may want to send the output of some of these commands so you can read them later. There are two ways this can be done.

The first is the same method used with the command prompt using the redirection symbol (>). For example, if you want to send a listing of all aliases to a text file named `PSAlias.txt` and then open the text file, you could use these commands:

```
Get-Alias > PSAlias.txt
Notepad PSAlias.txt
```

PowerShell also uses the `Out-File` cmdlet to send the output to a file. If you want to send a listing of services and their current status to a file, you can use the `Get-Service` cmdlet and the `Out-File` cmdlet in the same line.

Just as pipelining can be used at the command line, it can be used in PowerShell. When you want the output of one PowerShell command to be used as the input to another command on the same line, you separate the commands with the pipe symbol (|), which is Shift+Backslash (\) on most keyboards. The following shows how this is done:

```
Get-Service | Out-File Service.txt
Notepad Service.txt
```

Of course, if you wanted to save the file to a different location, you could include the full path. For example, you can save the file in the `Data` folder on the C drive with this command:

```
Get-Service | Out-File C:\Data\Service.txt
```

PowerShell Syntax

Many of the rules that apply to the command prompt also apply to PowerShell. As a reminder, here are some of these rules:

- Spelling counts.
- Commands are not case sensitive.
- Commands are modified with switches.
- Spaces usually need to be enclosed in quotes.
- Help is always available.
- Doskey saves typing.

There are a few other items you should know about PowerShell. These include variables, comparison operators, and command separators (such as parentheses, brackets, and braces), which are all covered in the following sections.

Variables Created with a $ Symbol

Many times, you'll need to create a variable that will be used to store information and later retrieve it. This can be as simple as loading a number (such as 5) into a variable like this:

```
$num = 5
```

and then retrieving the value with the variable like this:

```
$num
```

You can also use variables to hold collections of information. Collections can hold several similar items. For example, the following command will load a list of all the event logs into the variable named `$Log` and store them as a collection:

```
$Log = Get-EventLog -list
```

Once the collection is created, data about any of the items stored in the collection can be retrieved.

Comparison Operators

There are many comparison operators you can use to specify or identify a condition. Comparison operators will compare two values to determine whether a condition exists. Most comparison operators will return True if the condition exists and False if it doesn't exist.

Let's look at a simple example. Suppose a variable called `$num` has a value of 5. The comparison `$num -eq 5` evaluates as True, whereas `$num -eq 100` evaluates as False.

Comparisons can also compare text. However, when comparing text data, the text string needs to be enclosed in quotes. For example, if you wanted to know if a variable named `$str` has a value of "Success," you'd use this comparison:

```
$str -eq "Success"
```

String comparisons are case insensitive by default. In other words, both of these evaluate to True:

```
$str -eq "success"
$str -eq "SUCCESS"
```

However, if you want the comparison to be case sensitive, you can add the letter `c` to the operator switch, like this:

```
$str -ceq "Success"
```

Table 1.5 shows a listing of some of the commonly used comparison operators. You can enter these at the command line to see the result. For example, to see how the `-eq` command works, you could populate the variable `$num` with 5 and then use the variable in a comparison like this:

```
$num = 5
$num -eq 100
```

PowerShell will return False.

TABLE 1.5 Comparison operators

Operator	Description
`-eq`	Equals, as in `$x -eq 100`, or `$x -eq "y"`
`-ne`	Not equal, as in `$x -ne 100` or `$x -ne "y"`
`-gt`	Greater than, as in `$x -gt 100`
`-ge`	Greater than or equal to, as in `$x -ge 100`
`-lt`	Less than, as in `$x -lt 100`
`-le`	Less than or equal to, as in `$x -le 100`

`-like`	Compares strings using the wildcard character * and returns True if a match is found. The wildcard character can be used at the beginning, middle, or end to look for specific matches. If a variable named `$str` holds the string "MCSA Windows 8.," then all of the following comparisons will return True:
	`$str -like "MCSA*"`
	`$str -like "*Win*"`
	`$str -like "*8"`

`-notlike`	Compares strings using the wildcard character * and returns True if the match is not found.

You'll see comparison operators used in many different ways. For example, they are used in cmdlet switches when a comparison is needed and in `IF` statements when you're scripting.

Parentheses, Brackets, and Braces

PowerShell commands can include parentheses (), brackets [ ], and braces { }. Braces are also referred to as "curly brackets" and even "funky brackets" by some. Each is interpreted differently within PowerShell.

Parentheses ( ) Parentheses are commonly used to provide arguments. For example, when a script needs to accept a parameter, you can use `Param($input)`. Here `$input` is identified as an argument for the `Param` command.

The terms *parameters* and *arguments* are often used interchangeably. On one level, they are the same thing, but there is a subtle difference between the two. A parameter is provided as input to a piece of code, and an argument is what is provided. The value is the same, but the perspective is different. Parameters are passed in, and the value of this parameter is used as an argument within the code.

Brackets [ ] Brackets are used for various purposes. You'll see them used when accessing arrays when using `-like` comparisons and with some parameters. For example, the following command uses brackets to indicate the output should only include names that start with the letters a through g:

```
Get-wmiObject Win32_ComputerSystem | Format-List [a-g]*
```

Braces { } Braces are used to enclose a portion of code within a statement that is interpreted as a block of code. You'll see them in condition statements (like `ForEach`). The following example shows how braces are used to separate the block in the `Where` clause in a single line of code:

```
Get-service | Select * | Where {$_.name -like "Win*"}
```

The point here is not that you need to memorize when these characters need to be used. Instead, the goal is to let you know that all three could be used and, when you're executing code and writing scripts, to recognize each. If you replace curly braces with parentheses when you type your own code, you'll find things simply don't work.

Running PowerShell Commands

Exercise 1.13 gives you some practice running PowerShell commands. This exercise assumes you're logged on and the Start screen is showing.

EXERCISE 1.13

Running PowerShell Commands

1. Type **powershell**. Right-click Windows PowerShell and select Run As Administrator. If prompted by User Account Control, click Yes. Note that you don't always have to run PowerShell with administrative permissions, but some of the commands in this exercise require administrative permissions.
2. Enter the following command to retrieve help information on the `Get-NetIPAddress` cmdlet and store it in a text file:

   ```
   Get-Help Get-NetIPAddress -Full > netIP.txt
   ```

3. Use the following command to open the text file you just created. You can browse this file while entering commands.

   ```
   Notepad netIP.txt
   ```

4. Enter the following command to retrieve information on network adapters in the system:

   ```
   Get-NetIPAddress
   ```

   This command displays information on all of the network adapters. The wired Ethernet adapter (assigned an InterfaceAlias of Ethernet) is typically the first one in the list and assigned an InterfaceIndex of 12. Note the InterfaceIndex for your Ethernet adapter.

5. Enter the following command to create a table of information on the adapters showing only the index, alias, and IP address:

   ```
   Get-NetIPAddress | format-table interfaceindex, interfacealias,
   ipaddress
   ```

You'll see a display similar to the following output:

```
InterfaceIndex InterfaceAlias                 IPAddress
-------------- --------------                 ---------
12             Ethernet                       fe80::4973:2314:eca0:8dfc%12
1              Loopback Pseudo-Interface 1    ::1
12             Ethernet                       192.168.59.100
1              Loopback Pseudo-Interface 1    127.0.0.1
```

6. Enter the following command to get information on the Ethernet adapter:

```
Get-NetIPAddress -InterfaceIndex 12
```

You will normally see both IPv6 and IPv4 data listed.

7. Enter the following command to set a new IPv4 address of 192.168.59.121 with a subnet mask of 255.255.255.0 for the adapter with an InterfaceIndex of 12:

```
New-NetIPAddress -IPAddress 192.168.59.121 -PrefixLength 24
-InterfaceIndex 12
```

8. If the adapter doesn't currently have an IP address assigned, you can assign the address with the `Set-NetIPAddress` command:

```
Set-NetIPAddress -IPAddress 192.168.59.121 -PrefixLength 24
-InterfaceIndex 12
```

Running PowerShell Scripts

Creating and running PowerShell scripts has a couple of hurdles that can stop you in your tracks if you don't know what they are. These hurdles aren't hard to overcome, though. They are:

- PowerShell Execution Policy
- Path usage in PowerShell

Both of these issues are explained next.

PowerShell Execution Policy

Microsoft has embraced a secure by default mindset, and this is reflected in the *PowerShell Execution Policy*. This policy has several settings that you can modify to allow or disallow the execution of various types of scripts.

If you don't modify the policy, you'll find that each attempt to execute a PowerShell script will result in an error that says "the execution of scripts is disabled on this system." This message is telling you that the execution policy is set to Restricted, the default setting. There are several possible settings for the execution policy:

Restricted The Restricted setting prevents any scripts from being executed and is the default setting. You can still execute individual PowerShell commands.

RemoteSigned It's common to change the execution policy to RemoteSigned. This will let you execute any scripts on your local system but will prevent scripts that don't have a digital signature from being executed remotely, such as over an Internet connection.

Signed scripts have a digital signature added to them that are associated with a code-signing certificate from a trusted publisher. The idea is that if a script is signed with a certificate, you can identify the writer. Since malicious script writers don't want to be known or identified, they won't sign their scripts.

AllSigned This setting is a little more secure than RemoteSigned. Whereas RemoteSigned will allow the execution of unsigned local scripts, AllSigned will not allow the execution of any unsigned scripts. All scripts must be signed.

Unrestricted Just as it sounds, Unrestricted allows the execution of any scripts. This setting will warn you before running scripts that are downloaded from the Internet.

Bypass This is similar to Unrestricted in that it allows the execution of any scripts, but it does not give any warnings. This setting would be used when an application is using scripts and wouldn't be able to respond to any warnings.

You can use the `Get-ExecutionPolicy` cmdlet to determine the current setting for the execution policy. You can change the policy using the `Set-ExecutionPolicy` cmdlet. You must be running Windows PowerShell with administrative permissions to change the execution policy.

Exercise 1.14 shows how to change the execution policy. The exercise assumes you're logged on and the Start screen is showing.

EXERCISE 1.14

Changing the Execution Policy

1. Type **powershell**. Right-click Windows PowerShell and select Run As Administrator. If prompted by User Account Control, click Yes.
2. Enter the following command to identify the current execution policy:

   ```
   Get-ExecutionPolicy
   ```

3. Enter the following command to set the policy:

   ```
   Set-ExecutionPolicy RemoteSigned
   ```

4. When prompted, type **Y** and press Enter.
5. Enter the following command to verify the policy is changed:

   ```
   Get-ExecutionPolicy
   ```

6. Close the Administrator window.

Path Usage in PowerShell

When you're using the command prompt, it will always recognize the current path and you don't have to specify it. However, if you're using PowerShell, the current path is not recognized and must be specified.

For example, consider a script used to remove multiple Windows Store apps from a user account in Windows 8.1. It is named `RemoveWindowsStoreApps.ps1` and is available for free here:

```
http://gallery.technet.microsoft.com/scriptcenter/Remove-Windows-
Store-Apps-a00ef4a4
```

You could place the script in the `c:\data\scripts` folder and change the path to the `c:\data\scripts` folder with the `cd` command. However, if you tried the `RemoveWindowsStoreApps.ps1` command, it would fail. Instead, you must specify that the script is in the local folder by using this command:

```
.\RemoveWindowsStoreApps.ps1
```

If you were in a different folder, you could use this command:

```
C:\Data\Scripts\RemoveWindowsStoreApps.ps1
```

Just a Glimpse

Remember, this chapter is not intended to make you an expert on the command prompt or PowerShell. There's no way it can in these few pages. Entire books are written on both the command prompt and PowerShell and if you want to dig deeper, we strongly encourage you to do so.

Just as the angel (Don Cheadle) tells Nicholas Cage in *Family Man*, this is "just a glimpse." It provides you with a glimpse of the possibilities. What you do with this glimpse is up to you.

Summary

The Windows 8.1 interface might be a little different than you're used to, but there are simple ways to get to familiar tools. The charms are simply mini-menus that you can access by pressing Windows logo key+C. The Search charm automatically appears if you just start typing from the Start screen. You can access additional commands by right-clicking any tile to see the alternate menu. You can also access a list of useful tools from the preview menu (by pressing Windows logo key+X).

A new type of account supported in Windows 8.1 is the Microsoft Live account. This is simply an email address that is registered with Microsoft, and domain accounts can be used as a Microsoft Live account. When users use a Microsoft Live account, it streamlines the use of many cloud-based services.

This chapter also included information on the command prompt and Windows PowerShell. There are similarities between the two, and both are heavily used by administrators. Even though these tools aren't specifically mentioned in the objectives, you can fully expect to see both types of commands on the job and on the exams.

Exam Essentials

Know the five charms. Know the five charms: Start, Search, Share, Devices, and Settings. You can access a menu for each by pressing Windows logo key+C. The Search charm appears when you start typing from the Start screen.

Understand the difference between authentication and authorization. Users prove who they are by authenticating with a username and password or other authentication method. Authenticated users are authorized to access resources based on the permissions they are granted.

Know the different types of accounts. Local accounts are used in workgroups and domain accounts are used in domains. A Microsoft Live account can be used in a workgroup or a domain. An administrator account is in the Administrators group and has full control on a computer. A standard user account can perform most actions on a computer but does not have administrative permissions.

Understand picture passwords. Picture passwords allow users to authenticate with gestures instead of typing a password. Picture passwords can be used with touch interfaces or with a mouse.

Know the command prompt. The command prompt is a separate interface from the typical Windows GUI where commands are typed at the command prompt. Many commands exist that can be used to perform a wide array of tasks.

Understand Windows PowerShell. Know how to launch and use Windows PowerShell. Windows PowerShell is an extension of the command prompt. It's integrated into the Microsoft .NET Framework and provides access to significantly more capabilities. PowerShell commands take the form of a verb-noun such as `Get-Alias` that gets (the verb) a listing of each alias command (the noun). Three common verbs are `Get`, `Set`, and `Test`. By entering the verb and a dash (such as `Get-`), you can tab through all available nouns that can be combined with the verb to form cmdlets.

Review Questions

You can find the answers in Appendix A.

1. What charm can be used to add Administrative Tools items to the Start screen?
   - **A.** Devices
   - **B.** Search
   - **C.** Share
   - **D.** Settings

2. A Windows 8.1 user is pressing Windows logo key+. (period) to snap an app to the screen location, but this is not working. What is the most likely reason?
   - **A.** Apps are snapped to the screen with Windows logo key+S.
   - **B.** Apps are snapped to the screen with Windows logo key+I.
   - **C.** The screen resolution isn't at least 1024×768.
   - **D.** Apps are snapped to the screen with the Windows logo +. and then using the arrow keys to snap the application to a screen location.

3. Of the following choices, what accurately describes the differences between authentication and authorization? (Choose all that apply.)
   - **A.** Users prove an identity with authentication.
   - **B.** Users prove an identity with authorization.
   - **C.** Authorization methods are used to grant users access to resources.
   - **D.** Authentication methods are used to grant users access to resources.

4. You need to ensure that users can access Microsoft resources like Windows Store and SkyDrive with their user account. What type of account should they use?
   - **A.** Local account
   - **B.** Domain account
   - **C.** Microsoft Live account
   - **D.** Administrator account

5. Which of the following methods can you use to create a Microsoft Live account?
   - **A.** Computer Management ➢ System Tools ➢ Local Users and Groups
   - **B.** Settings charm ➢ Change PC Settings ➢ Accounts
   - **C.** Administrative Tools ➢ Users
   - **D.** Control Panel ➢ Users

6. Users are using touch screens with Windows 8.1 devices. Of the following choices, what can they use for authentication rather than typing in a traditional password? (Choose all that apply.)
   - **A.** PIN
   - **B.** Picture password
   - **C.** Gestures
   - **D.** Windows Live ID

7. Which of the following methods can be used for authentication in Windows 8.1? (Choose all that apply.)
   - **A.** Picture passwords
   - **B.** Biometrics
   - **C.** BitLocker
   - **D.** Smartcards

8. Users in a domain currently use domain-based accounts to access domain resources. The company wants users to be able to access Microsoft-based resources with their Windows 8.1 systems. Of the following choices, what is the easiest solution?
   - **A.** Have users create a separate Microsoft Live account.
   - **B.** Have users register their domain account as a Microsoft Live account.
   - **C.** Create a separate domain account for users to use as a Microsoft Live account.
   - **D.** Create a separate local account for users to use as a Microsoft Live account.

9. You need to start the command prompt to run a command. Which of the follow methods will work? (Choose all that apply.)
   - **A.** Access the charms by pressing Windows logo key+C and selecting Command Prompt.
   - **B.** Type **command** at the Start screen and select Command Prompt.
   - **C.** Access the Settings charm by pressing Windows logo key+K and selecting Command Prompt.
   - **D.** Access the preview menu by pressing Windows logo key+X and selecting Command Prompt.

10. When you try to run a command from the command prompt on a Windows 8.1 computer, you see an error indicating it requires elevated permissions. What should you do?
    - **A.** Run the command using PowerShell.
    - **B.** Run the command with administrative permissions.
    - **C.** Run the command from the GUI.
    - **D.** Log on with an administrative account and run the command.

11. You are an administrator in a domain named Success.com and you have an administrator account named ITAdmin. You are helping a user resolve a problem and realize you need to run a program named `App.exe` from the command prompt with administrative permissions. The program needs access to the ITAdmin account environment. Which of the following commands should you use?

A. `runas /user:success\itadmin /env app.exe`

B. `runas /user:success\itadmin /profile app.exe`

C. `runas /user:success\itadmin /noprofile app.exe`

D. `runas /user:success\itadmin /netonly app.exe`

12. You are an administrator in a domain named Success.com and you have an administrator account named ITAdmin. You are helping a user resolve a problem and realize you need to run a program named `App.exe` from the command prompt with administrative permissions. You want to minimize the amount of time the program takes to run. Which of the following commands should you use?

A. `runas /user:success\itadmin /env app.exe`

B. `runas /user:success\itadmin /profile app.exe`

C. `runas /user:success\itadmin /noprofile app.exe`

D. `runas /user:success\itadmin /netonly app.exe`

13. You are running Windows 8.1 with two monitors. You want to display applications on both monitors. What should you do? (Choose all that apply.)

A. Select Extend from the Start screen.

B. Select Extend from the Settings charm.

C. Select Second Screen ➢ Extend from the Devices charm.

D. Run the `displayswitch /extend` command.

14. A computer is running Windows 8.1. What command can you use to remove leftover data fragments on a disk drive that is encrypted with BitLocker Drive Encryption?

A. `manage-bde -w`

B. `wipefreespace -bde`

C. `bitlocker -wipe`

D. `format -bde`

15. Which of the following commands can you use to install a program named `success.msi` on a Windows 8.1 computer from the command prompt?

A. `execmsi /i success.msi`

B. `install /i success.msi`

C. `installmsi /i success.msi`

D. `msiexec /i success.msi`

16. You've tried to execute a PowerShell command but get an error indicating that access is denied. What is the most likely solution?
    A. Change the PowerShell Execution Policy.
    B. Run the command from the PowerShell ISE.
    C. Start PowerShell using the `runas` command.
    D. Start PowerShell with administrative permissions.

17. You are trying to get help on a PowerShell command but only see limited help available for a specific command. What command should you use to get full help?
    A. `Update-Help`
    B. `Get-Help` *`command`* `-Full`
    C. `Help` *`command`* `-Full`
    D. `Get-Help` *`command`* `-Update-Help`

18. You need to set the IP address of a network adapter. Which of the following PowerShell commands can be used? (Choose all that apply.)
    A. `Set-NetIPAddress`
    B. `Get-NetIPAddress`
    C. `New-NetIPAddress`
    D. `ipconfig`

19. You are trying to execute a PowerShell script but keep getting an error indicating scripts cannot be executed. What should be done?
    A. Modify the PowerShell Execution Policy to Restricted.
    B. Modify the PowerShell Execution Policy to RemoteSigned.
    C. Run the script using `runas`.
    D. Launch PowerShell with administrative permissions.

20. You are trying to execute a PowerShell script in the current directory but keep getting an error indicating the script cannot be located. What is a likely solution?
    A. Use `.\` before the script name.
    B. Modify the PowerShell Execution Policy.
    C. Run the command from the PowerShell ISE.
    D. Enclose the script name in quotes.