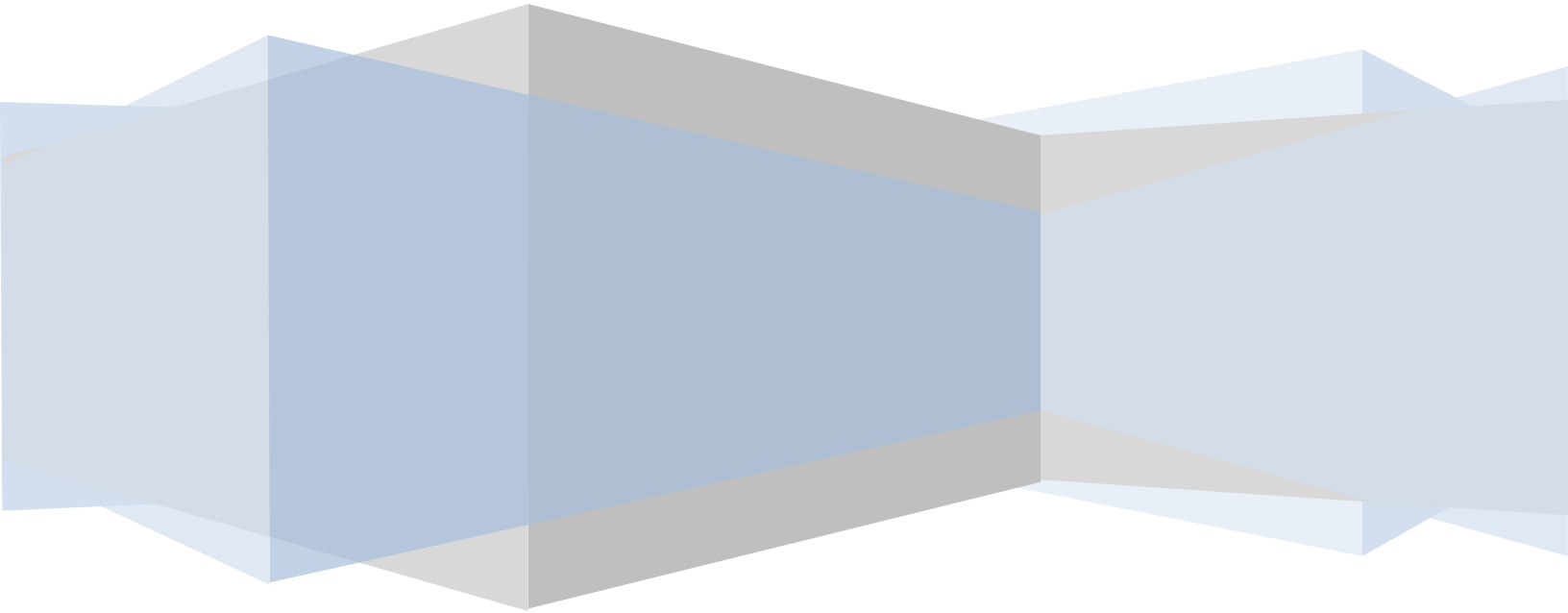
Windows PowerShell Essentials



Windows PowerShell Essentials – Edition 1.0

Table of Contents

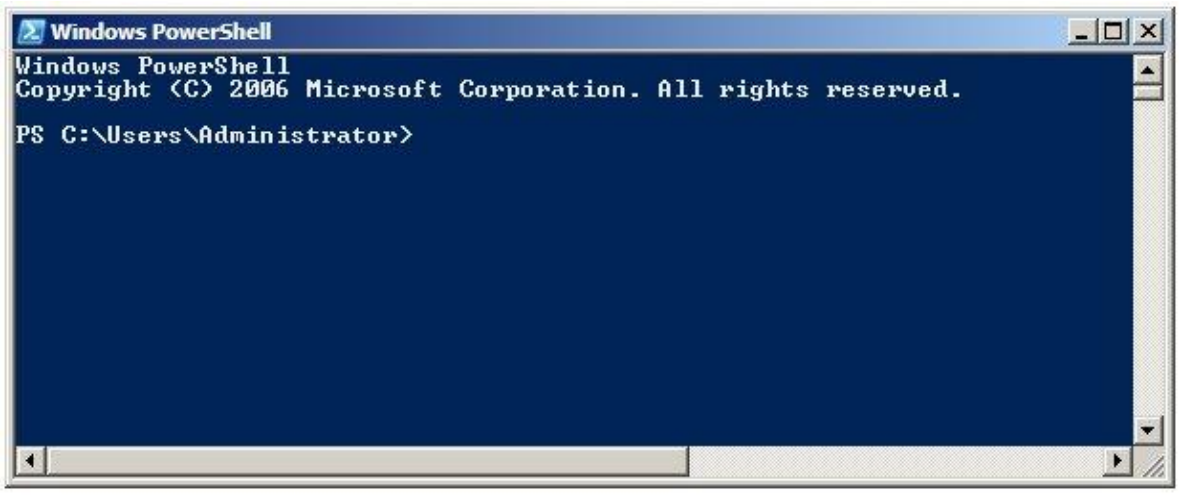
Chapter 1. Installing Windows PowerShell

The objective of this chapter of *Windows PowerShell Essentials* is to cover the steps necessary to install Windows Powershell on Windows XP, Windows Vista and Windows Server 2008 systems.

1.1 Installing Windows PowerShell on Windows Server 2008

Windows PowerShell is included as a standard feature with Windows Server 2008, although the feature is not installed by default. The PowerShell feature may be added using the *Server Manager* tool which is invoked by selecting the *Start -> Server Manager* menu option. Once Server Manager is running, select *Features* from the left hand pane and in the *Features Summary* section of the main panel click on *Add Features* to invoke the *Add Features Wizard*. The first screen displayed will list all available features in alphabetical order. Scroll down the list and select the checkbox next to *Windows PowerShell* and click *Next* to proceed to the installation screen. Clicking *Install* will initiate the installation process. After a few minutes the installation will complete and the wizard may be closed using the *Close* button.

To launch Windows PowerShell, select *Start -> All Programs -> Windows PowerShell 1.0 -> Windows PowerShell*. Once loaded, a new window will appear displaying the Windows PowerShell *PS>* prompt at which commands may be entered:



1.2 Performing a Windows Server 2008 PowerShell Command Line Installation

The installation of PowerShell on a Windows Server 2008 may be performed from the command prompt using the **servermanagercmd** command. To achieve this, invoke the

command prompt with elevated privileges (right click on the Command Prompt entry in the Start menu and select *Run as administrator*) and execute **servermanagercmd -install powershell** as follows:

```
C:\Users\Administrator>servermanagercmd -install powershell
....

Start Installation...
[Installation] Succeeded: [Windows PowerShell].
<100/100>

Success: Installation succeeded.
```

Once installed, PowerShell may be run from the command prompt simply by entering *powershell*:

1.3 Installing PowerShell on Windows Server 2003, XP and Vista Systems

Unlike Windows Server 2008, other Windows versions do not ship with PowerShell bundled as a feature. As such, it is necessary to download and install PowerShell from the Microsoft Download Center. Pre-requisites for installing PowerShell are as follows:

- Windows Server 2003 Service Pack 1 (SP1) or later
- Windows XP Service Pack 2 (SP2) or later

- .Net Framework 2.0 or later (available for download from the Microsoft MSDN web site

Assuming the above pre-requisites have been met, Windows PowerShell may be downloaded from the Microsoft PowerShell download page. Once the installation package has been downloaded, double click on it to begin the installation and follow the instructions. Once installation is complete, the following dialog will be displayed by the install wizard:

Once installation is complete, PowerShell may be launched from the *Start* menu by selecting *All Programs -> Windows PowerShell 1.0 -> Windows PowerShell*. Alternatively, launch PowerShell from a command prompt window by typing:

```
powershell
```

Chapter 2. The Basics of the Windows PowerShell Interactive Shell

Windows PowerShell is both a scripting language and an interactive shell. In interactive shell mode, PowerShell provides a command line prompt at which commands may be entered and executed interactively. The purpose of this chapter is to cover the basics of using the PowerShell interactive shell.

2.1 The Windows PowerShell Command Prompt

The PowerShell interactive prompt may be accessed in two ways. One option is to launch PowerShell from the *Start* menu by selecting *Start -> All Programs -> Windows PowerShell 1.0 -> Windows PowerShell*. Alternatively, PowerShell may be launched from the Windows Command Prompt (cmd.exe) window simply by entering the command *powershell*. Once loaded, PowerShell will display copyright information, followed by the shell prompt:

```
Windows PowerShell
Copyright (C) 2006 Microsoft Corporation. All rights reserved.


PS C:\Users\Administrator>
```

The interactive shell prompt consists of the letters *PS* to indicate that the user is working within the PowerShell, followed by the current working directory. The prompt is terminated with a '>' character after which the cursor appears awaiting user input. The current working directory indicates that any file system based operations will take place relative to that directory on the file system. Typing the *dir* or *ls* command, therefore, will provide a listing of files and sub-directories in the current working directory (C:\Users\Administrator) in this case:

```
PS C:\Users\Administrator> dir



    Directory: Microsoft.PowerShell.Core\FileSystem::C:\Users\Administrator



Mode                LastWriteTime     Length Name
----                -------------     ------ ----
```

```
d-r--         11/7/2008   4:26 PM            Contacts
d-r--         11/7/2008   4:26 PM            Desktop
d-r--         11/7/2008   4:26 PM            Documents
d-r--         11/7/2008   4:26 PM            Downloads
d-r--         11/7/2008   4:26 PM            Favorites
d-r--         11/7/2008   4:26 PM            Links
d-r--         11/7/2008   4:26 PM            Music
d-r--         11/7/2008   4:26 PM            Pictures
d-r--         11/7/2008   4:26 PM            Saved Games
d-r--         11/7/2008   4:26 PM            Searches
d-r--         11/7/2008   4:26 PM            Videos
```

By default, this will be the home directory of the user who invoked PowerShell. The directory may be changed using *cd* command, for example:

```
PS C:\Users\Administrator> cd \windows
PS C:\Windows>
```

2.2 PowerShell Command Line Editing Keys

PowerShell provides a number of special key sequences that speed the task of creating and editing commands in the interactive shell, each of which is outlined in the following table:

Key Sequence	Description
Left Arrow	Moves the command line cursor one character to the left.
Right Arrow	Moves the command line cursor one character to the right.
Ctrl+Left Arrow	Moves the command line cursor one word to the left.
Ctrl+Right Arrow	Moves the command line cursor one word to the right.
Delete	Deletes the character in the command line at the current cursor position.

Backspace	Deletes the character immediately to the right of the current cursor position.
Insert	Toggles between character insert and overwrite modes.
Home	Relocates the cursor to the beginning of the command line.
End	Relocates the cursor to the end of the command line.
Tab	Performs command completion operation whereby the shell attempts to guess at the remainder of the command being entered.
F7	Displays a new window containing the command history from which previous commands may be selected.

2.3 PowerShell Command Completion

After a few characters of a command, parameter, file name, variable or variable property have been entered at the PowerShell command line prompt, the Tab key may be pressed to instruct PowerShell to complete the remainder of the entry. For example, if the first few characters of a command have been entered, PowerShell will try to find the best match for a command and fill in the remainder of the command name. If the command selected by PowerShell is not the correct one, pressing the Tab key repeatedly will cycle through all the possible matches until the correct one is displayed. Similarly, if a partial file name is entered followed by the Tab key, PowerShell will populate the command line with file name from the current directory. If, once again, the first match displayed by PowerShell is not the desired file, the Tab key can be pressed repeatedly until the correct match is displayed.

Tab completion is particularly useful for finding the properties and methods of a variable. As an example, suppose a variable named *$mynumber* is assigned a numerical value and we want to find out what the type of that value is. We assume there must be a method we can call on the variable to get the type but do not know what that method is called. To cycle through the available methods we simply enter the name of the variable, followed by a '*.*' character and then the first few characters of the method (we are assuming it will begin with *Get*), We then press the *Tab* key until we find the method that does what we need, in this case, *GetType()*:

```
PS C:\Users\Administrator> $mynum.GetType(
```

We then enter the closing parentheses '*)*' since we do not need to pass any arguments through to the method, and then press enter to execute the command:

```
PS C:\Users\Administrator> $mynum.GetType()


IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     Double                                   System.ValueType
```

Tab completion may also be used in conjunction with wildcards, for example where the * character is used to represent one or more characters in a file name. This allows, for example, both the beginning and end characters of a file name to be specified before using the Tab key to complete the entire file name. For example, entering:

```
PS C:\Users\Administrator> type my*file.txt <tab>
```

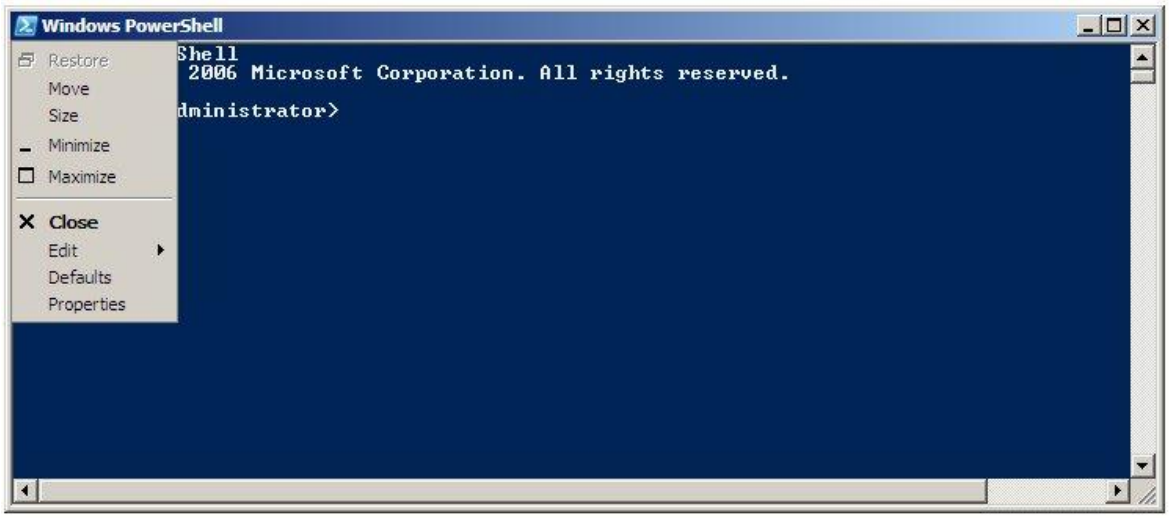
results in PowerShell completing the file name with the first closest match:

```
PS C:\Users\Administrator> type mydatafile.txt
```

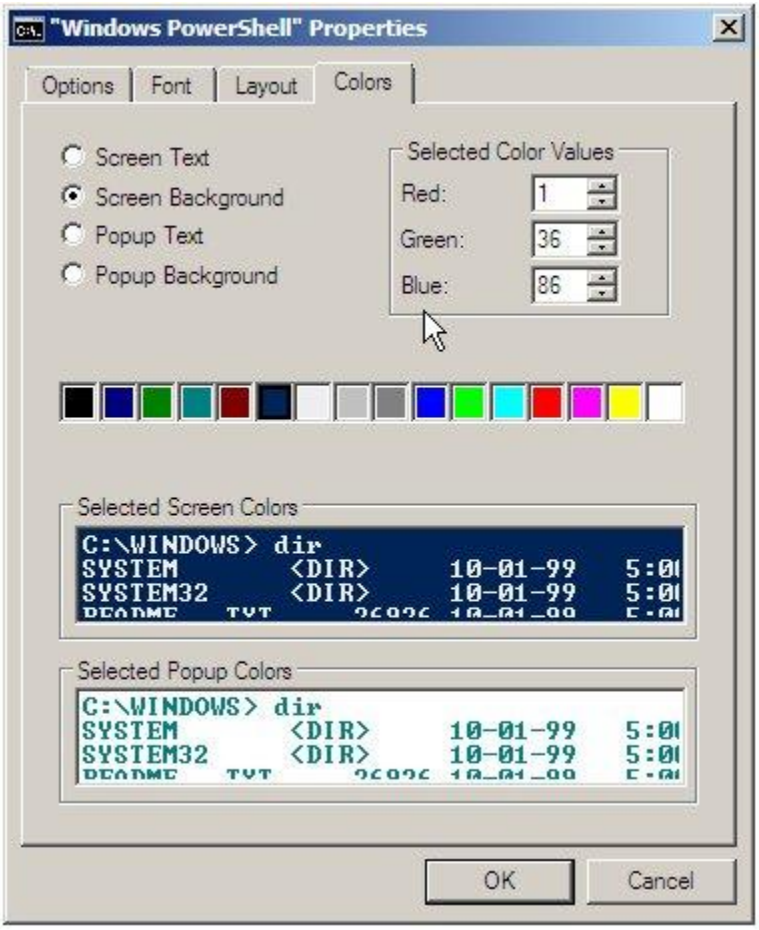
2.4 Customizing the PowerShell Window

The Windows PowerShell interactive shell is invoked either from the *Start* menu, or from a command prompt (cmd.exe) window. In either case, the shell will be operating in a window with some pre-set characteristics. For example, the command prompt displays white text on a black background and is restricted to 80 characters in width. The PowerShell window displays white text on a blue background and is typically limited to 157 characters in width.

Fortunately just about every aspect of these windows can be changed by clicking with the mouse pointer on the small icon in the top right hand corner of the window title block and selecting the *Properties* option from the resulting menu as illustrated in the following figure:



Once the *Properties* option has been selected, the properties dialog will appear. A number of tabs are available which, when selected, allow different categories of settings to be changed. The following figure, for example, shows the *Color* screen which is used to change settings such the background and foreground colors of the window:



The settings contained on each tab are as follows:

- **Options** - Includes cursor size settings, the size of the command history buffer, whether the shell is displayed as a window or full screen and editing options.
- **Font** - The style and size of font used to display text in the PowerShell window.
- **Layout** - Settings such as the initial height, width and position of the PowerShell window and size of the screen buffer (the amount of text retained after it has scrolled out of the viewable window area).
- **Colors** - The foreground and background colors of both the screen and any popups.