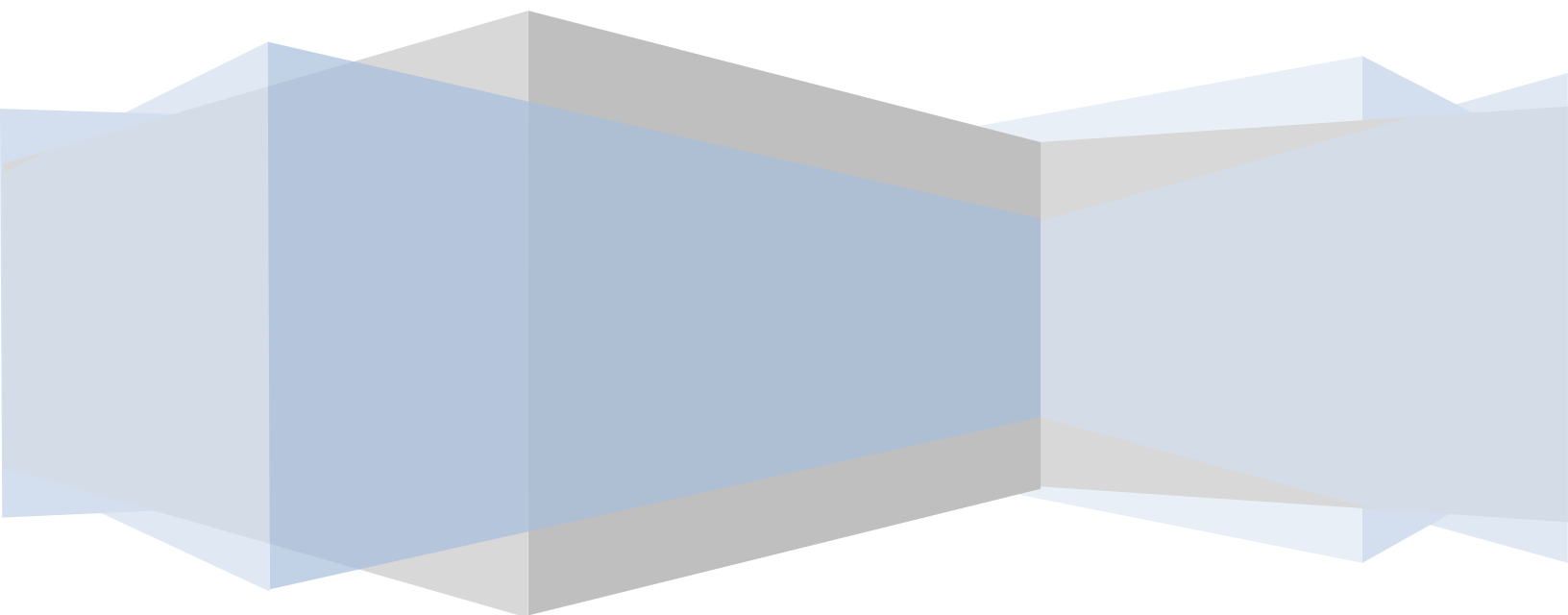


Windows PowerShell Essentials



Windows PowerShell Essentials – Edition 1.0

© 2009 Payload Media. This eBook is provided for personal use only. Unauthorized use, reproduction and/or distribution strictly prohibited. All rights reserved.

The content of this book is provided for informational purposes only. Neither the publisher nor the author offers any warranties or representation, express or implied, with regard to the accuracy of information contained in this book, nor do they accept any liability for any loss or damage arising from any errors or omissions.

Find more eBooks at www.ebookfrenzy.com

Table of Contents

| | |
|---|----|
| Chapter 1. Installing Windows PowerShell..... | 10 |
| 1.1 Installing Windows PowerShell on Windows Server 2008 | 10 |
| 1.2 Performing a Windows Server 2008 PowerShell Command Line Installation | 10 |
| 1.3 Installing PowerShell on Windows Server 2003, XP and Vista Systems | 11 |
| Chapter 2. The Basics of the Windows PowerShell Interactive Shell | 13 |
| 2.1 The Windows PowerShell Command Prompt | 13 |
| 2.2 PowerShell Command Line Editing Keys | 14 |
| 2.3 PowerShell Command Completion | 15 |
| 2.4 Customizing the PowerShell Window | 16 |
| Chapter 3. The Basics of Creating and Running Windows PowerShell Scripts..... | 19 |
| 3.1 What is a PowerShell Script?..... | 19 |
| 3.2 An Example Windows PowerShell Script | 19 |
| 3.3 PowerShell 1.0 Script Naming Convention | 19 |
| 3.4 Executing PowerShell Scripts | 19 |
| 3.5 Handling Script Arguments | 21 |
| 3.6 The PowerShell exit Keyword..... | 22 |
| Chapter 4. Windows PowerShell Commands and Aliases..... | 23 |
| 4.1 PowerShell 1.0 Command Summary..... | 23 |
| 4.2 PowerShell 1.0 Alias Summary..... | 30 |
| 4.3 Creating and Listing PowerShell Aliases..... | 33 |
| Chapter 5. Windows PowerShell String Quoting and Escape Sequences | 34 |
| 5.1 Using Double Quotes..... | 34 |
| 5.2 Using Single Quotes..... | 35 |
| 5.3 Using the PowerShell Escape Character | 36 |
| 5.4 PowerShell Special Escape Sequences..... | 38 |
| Chapter 6. Directing and Formatting Windows PowerShell Output | 39 |
| 6.1 The Default Output | 39 |

| | | |
|---|---|----|
| 6.2 | The Out-Host Cmdlet | 39 |
| 6.3 | The Out-Printer Cmdlet..... | 39 |
| 6.4 | The Out-File Cmdlet | 39 |
| 6.5 | The Out-Null Cmdlet | 40 |
| 6.6 | The Out-String Cmdlet..... | 40 |
| 6.7 | Formatting PowerShell Output | 40 |
| Chapter 7. Understanding and Creating Windows PowerShell Variables..... | | 44 |
| 7.1 | Naming a Variable in PowerShell | 44 |
| 7.2 | Creating and Assigning a Value to a PowerShell Variable | 45 |
| 7.3 | Accessing PowerShell Variable Values | 46 |
| 7.4 | Changing the Type of a PowerShell Variable | 46 |
| 7.5 | Pre-defined PowerShell Variables..... | 47 |
| Chapter 8. Basic Windows PowerShell Types..... | | 49 |
| 8.1 | PowerShell Numeric Types..... | 49 |
| 8.2 | Specifying Hexadecimal Numbers..... | 51 |
| 8.3 | PowerShell Boolean Type..... | 52 |
| 8.4 | PowerShell String Type..... | 52 |
| 8.5 | Windows PowerShell Type Casting | 53 |
| Chapter 9. Working with Arrays in Windows PowerShell | | 55 |
| 9.1 | Creating a Windows PowerShell Array | 55 |
| 9.2 | Creating Windows PowerShell Multidimensional Arrays | 55 |
| 9.3 | Obtaining the Length of an Array..... | 56 |
| 9.4 | Accessing Elements in a Windows PowerShell Array | 56 |
| 9.5 | Accessing Elements in a Windows PowerShell Multidimensional Array | 57 |
| 9.6 | Adding new Elements to a Windows PowerShell Array | 58 |
| 9.7 | Combining Windows PowerShell Arrays..... | 58 |
| Chapter 10. Windows PowerShell Hashtables | | 60 |
| 10.1 | Creating PowerShell Hashtables | 60 |

| | | |
|-------------|---|----|
| 10.2 | Accessing Hashtable Elements | 60 |
| 10.3 | Modifying Windows PowerShell Hashtable Elements | 61 |
| 10.4 | Adding Elements to a Windows PowerShell Hashtable | 62 |
| 10.5 | Removing Elements from a Windows PowerShell Hashtable | 62 |
| 10.6 | Clearing All Elements from a Windows PowerShell Hashtable | 63 |
| 10.7 | Combining Hashtables | 63 |
| 10.8 | Listing Hashtable Count, Keys and Values | 63 |
| Chapter 11. | Basic Windows PowerShell Operators | 65 |
| 11.1 | What is an Expression? | 65 |
| 11.2 | Windows PowerShell Basic Assignment Operator | 65 |
| 11.3 | Windows PowerShell Compound Assignment Operators | 66 |
| 11.4 | Windows PowerShell Arithmetic Operators | 67 |
| 11.4.1 | The Addition Operator | 68 |
| 11.4.2 | Multiplication Operator | 69 |
| 11.4.3 | Division, Subtraction and Modulus Operators | 70 |
| 11.5 | PowerShell Increment and Decrement Operators | 71 |
| 11.6 | Windows PowerShell & Operator | 72 |
| 11.7 | Windows PowerShell Bitwise and Logical Operators | 73 |
| Chapter 12. | Windows PowerShell Comparison and Containment Operators | 74 |
| 12.1 | Windows PowerShell Comparison Operators | 74 |
| 12.2 | Windows PowerShell Containment Operators | 75 |
| 12.3 | Performing Windows PowerShell Comparisons | 76 |
| 12.4 | Using Comparison Operators with Arrays and Collections | 77 |
| 12.5 | Using PowerShell Containment Operators | 77 |
| Chapter 13. | Windows PowerShell Pipes and Redirection | 78 |
| 13.1 | PowerShell Pipes | 78 |
| 13.2 | Windows PowerShell Redirection Operators | 78 |
| 13.3 | Windows PowerShell Redirection | 79 |

13.4 Redirecting Error Output..... 80

Chapter 14. Windows PowerShell Flow Control with *if*, *else* and *elseif*..... 83

14.1 Using the *if* Statement 83

14.2 Using *if ... else ..* Statements 84

14.3 Using *if ... elseif ..* Statements 84

14.4 Summary 85

Chapter 15. Windows PowerShell Looping with the *for* and *foreach* Statements..... 86

15.1.1 The PowerShell *for* Loop..... 86

15.2 Creating an Infinite *for* Loop 88

15.3 Breaking Out of a *for* Loop 88

15.4 Nested *for* Loops 89

15.5 Breaking From Nested Loops with Labels 89

15.6 Continuing *for* Loops 91

15.7 Windows PowerShell *foreach* Loops..... 91

Chapter 16. Windows PowerShell Looping with *do* and *while* Statements 93

16.1 The Windows PowerShell *while* Loop 93

16.2 Windows PowerShell *do ... while* loops 94

16.3 Breaking from *while* Loops..... 94

16.4 The *continue* Statement..... 95

Chapter 17. The Windows PowerShell *switch* Statement 97

17.1 Why Use a *switch* Statement? 97

17.2 Windows PowerShell *switch* Statement Syntax..... 98

17.3 A *switch* Statement Example 99

17.4 Explaining the Example 100

17.5 Using *break* in a Windows PowerShell *switch* Statement 100

17.6 The *switch \$_* Variable 102

17.7 Using Expressions in a Windows PowerShell *switch* Statement..... 102

17.8 Wildcards, Regular Expressions and Case Sensitivity in *switch* Statements..... 103

| | | |
|-------------|--|-----|
| 17.9 | Using switch Statements to Iterate through Ranges and Collections | 105 |
| 17.10 | Using the continue Statement..... | 106 |
| Chapter 18. | Windows PowerShell Functions..... | 108 |
| 18.1 | Creating and Calling a Simple Windows PowerShell Function | 108 |
| 18.2 | Passing Parameters to a Function | 108 |
| 18.3 | Specifying Named Parameters | 109 |
| 18.4 | Function Parameter Initialization..... | 110 |
| 18.5 | Specifying Function Parameter Types..... | 110 |
| 18.6 | Using Functions in a Pipeline | 112 |
| 18.7 | Returning from Functions | 113 |
| 18.8 | Windows PowerShell Function Administration | 114 |
| 18.9 | Loading Function Definitions from a Windows PowerShell Script File | 115 |
| Chapter 19. | Working with File Systems in Windows PowerShell..... | 117 |
| 19.1 | Windows PowerShell File System Cmdlets | 117 |
| 19.2 | Getting Disk Drive Information | 118 |
| 19.3 | Creating New Windows PowerShell Drives..... | 125 |
| 19.4 | Getting Information about Network Drives in PowerShell | 126 |
| 19.5 | Windows PowerShell File System Directory Listings | 127 |
| 19.6 | Copying, Remaining and Deleting Files and Directories | 128 |
| Chapter 20. | Windows PowerShell File Handling | 130 |
| 20.1 | Getting File Properties | 130 |
| 20.2 | Changing File Properties with Set-ItemProperty | 130 |
| 20.3 | Reading Text Files in Windows PowerShell..... | 131 |
| 20.4 | Reading Binary Files in Windows PowerShell | 132 |
| 20.5 | Writing to Files in Windows PowerShell | 132 |
| 20.6 | Searching for Strings in Files with Windows PowerShell | 132 |
| Chapter 21. | An Overview of Windows PowerShell and .NET | 135 |
| 21.1 | Exploring .NET Assemblies in Windows PowerShell | 135 |

| | | |
|-------------|---|-----|
| 21.2 | Loading .NET Assemblies into Windows PowerShell | 137 |
| 21.3 | .NET Static Members and Properties | 137 |
| 21.4 | Instance Members and Properties | 138 |
| 21.5 | Listing the Methods and Properties of a .NET Class | 139 |
| Chapter 22. | Creating GUIs in Windows PowerShell with WinForms | 142 |
| 22.1 | An Overview of WinForms | 142 |
| 22.2 | Loading the Winforms .NET Assembly | 142 |
| 22.3 | A Basic Windows PowerShell GUI | 142 |
| 22.4 | Windows PowerShell and WinForms Events | 143 |
| 22.5 | Setting WinForms Properties | 145 |
| 22.6 | Bringing it all Together | 145 |
| Chapter 23. | Drawing Graphics using PowerShell and GDI+ | 147 |
| 23.1 | An Overview of GDI+ | 147 |
| 23.2 | Loading the GDI+ and WinForms .NET Assemblies | 147 |
| 23.3 | Creating Drawing Objects..... | 147 |
| 23.4 | Setting Properties of a GDI+ Drawing Object..... | 148 |
| 23.5 | Creating the WinForms Form and the Graphics Object..... | 149 |
| 23.6 | The Paint Event Handler..... | 150 |
| 23.7 | Drawing Graphics with Windows PowerShell and GDI+ | 150 |
| 23.8 | Bringing it All Together..... | 152 |
| Chapter 24. | Using COM with Windows PowerShell | 155 |
| 24.1 | Listing Available COM Objects | 155 |
| 24.2 | Creating COM Object Instances in Windows PowerShell | 156 |
| 24.3 | Listing the Properties and Methods of a COM Object | 156 |
| 24.4 | Interacting With COM Objects | 158 |
| 24.5 | Interacting with the Windows Shell | 158 |
| 24.6 | Using the WScript.Shell Class | 161 |
| 24.7 | Summary | 163 |

Chapter 25. Windows PowerShell Security 164

- 25.1 Windows PowerShell Script Execution Policy 164
- 25.2 Identifying and Changing the Current Execution Policy..... 165
- 25.3 Signing Windows PowerShell Scripts 165
- 25.4 Setting up a Local Certificate Authority 166
- 25.5 Creating a Certificate..... 166
- 25.6 Signing a Windows PowerShell Script 167
- 25.7 Protecting Certificates with Private Key Encryption 169

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*:



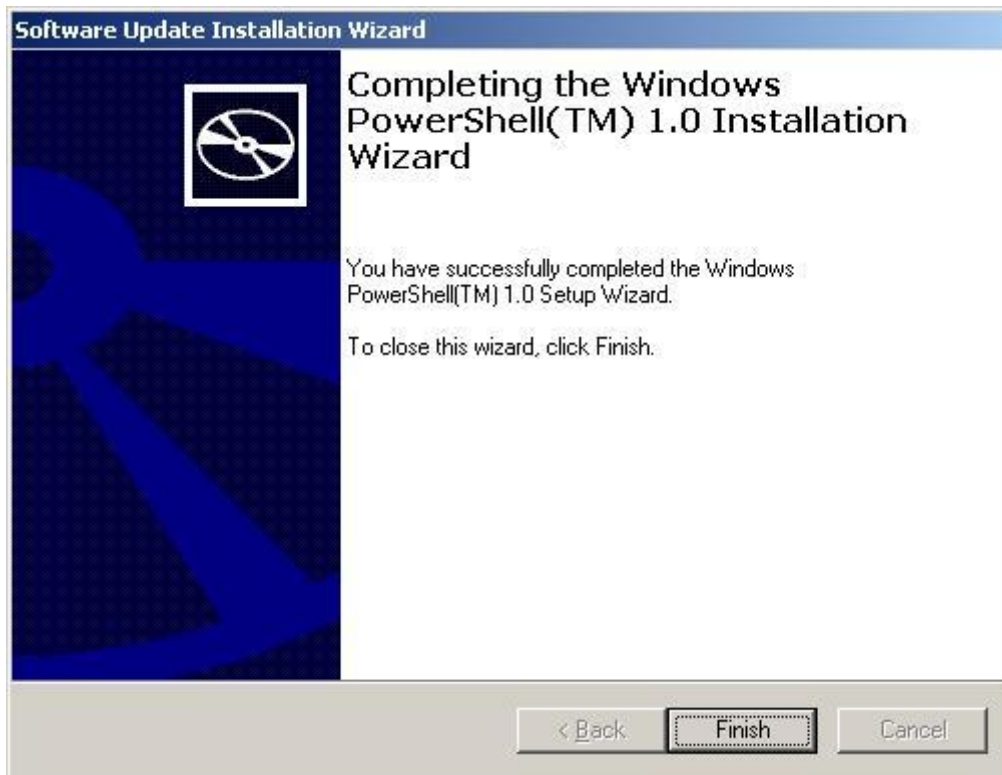
1.3 Installing PowerShell on Windows Server 2003, XP and Vista Systems

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>
```

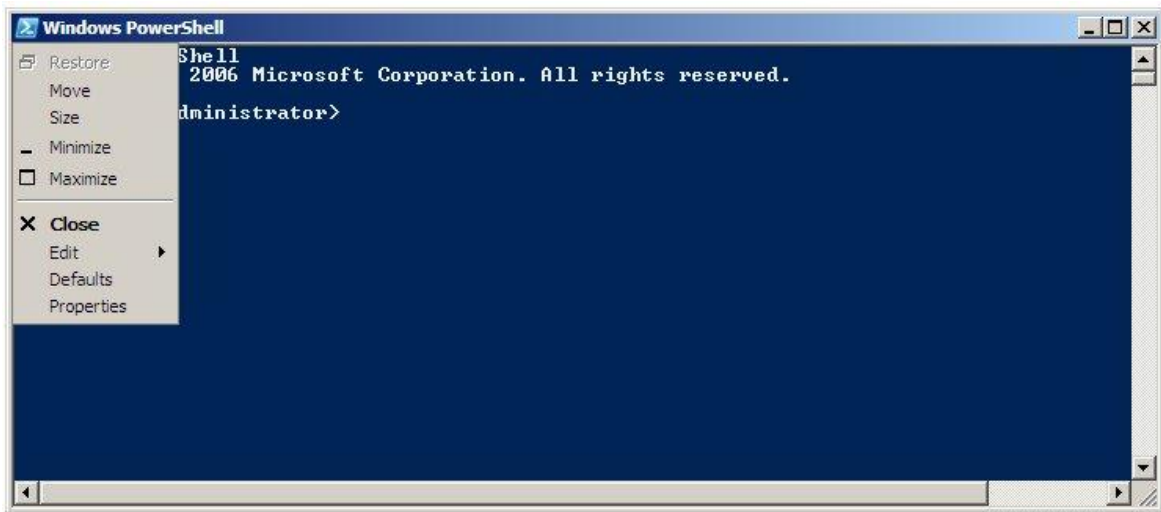
results in PowerShell completing the file name with the first closest match:

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

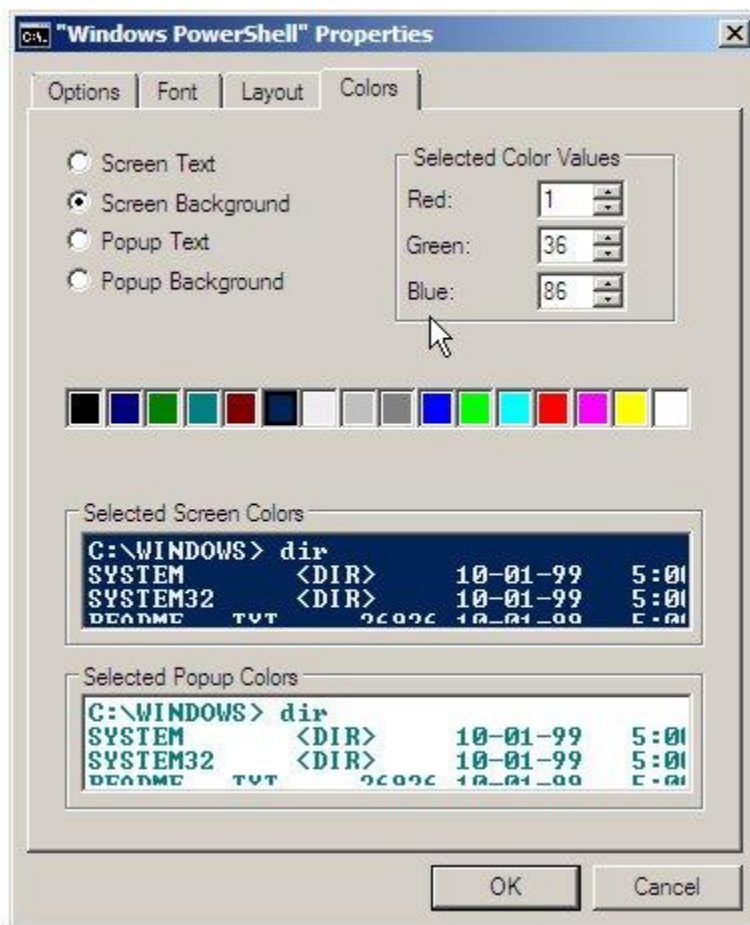
2.4 Customizing the PowerShell Window

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 as 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.