PROVIDER NAME
    FileSystem

DRIVES
    C, D

SYNOPSIS
    Provides access to files and directories.

DESCRIPTION
    The Windows PowerShell FileSystem provider lets you get, add, change, clear,
    and delete files and directories in Windows PowerShell.

    The FileSystem provider exposes Windows PowerShell drives that correspond to
    the logical drives on your computer, including drives that are mapped to
    network shares. This lets you reference these drives from within Windows
    PowerShell.

    The FileSystem provider lets you refer to files and folders in Windows
    PowerShell in the same way that you refer to them in Windows.

    To refer to a drive, specify the drive name followed by a colon. Like most of
    Windows PowerShell, the FileSystem provider is not case-sensitive. For
    example, to get the files and folders on the C drive, you refer to the "C:"
    drive or the "c:" drive.

    A fully qualified name includes the drive name, followed by a colon, any
    directory and subdirectory names, and the file name (when applicable). Each
    element of the fully qualified name must be separated either by a backslash
    (\) or a forward slash (/).

    The following example shows the fully qualified name for the Shell.dll file in
    the System32 subdirectory of the Windows directory on the C drive:

        C:\Windows\System32\shell.dll

    If any element in the fully qualified name includes spaces, you must enclose
    the name in quotation marks.

    For example,

        "C:\Program Files\Internet Explorer\iexplore.exe"

    The current location in the file system is represented by a dot or period
    character (.)

    For example, the current location is the C:\Windows\System32 directory, then
    you can refer to the Shell.dll file in that directory as the following:

        .\Shell.dll

    To use the FileSystem provider to view and manage files and folders, use the
    provider cmdlets, such as Get-ChildItem ("dir", "ls") and Set-Location ("cd").
    Windows PowerShell also includes a "mkdir" function (alias = "md") that uses
    the New-Item cmdlet to create a new directory.

    Beginning in Windows PowerShell 3.0, you can get customized help topics for
    provider cmdlets that explain how those cmdlets behave in a file system drive.

    To get the help topics that are customized for the file system drive, run a
    Get-Help command in a file system drive or use the Path parameter of Get-Help
    to specify a file system drive.

    For example,

        PS C:\> get-help get-childitem

        PS HKLM:\> get-help get-childitem -path c:

    Also, beginning in Windows PowerShell 3.0, the FileSystem provider supports
    credentials for the New-PSDrive cmdlet. For more information, see the help
    topic for the New-PSDrive cmdlet.

CAPABILITIES
    Filter, ShouldProcess

TASKS

TASK: Splitting a Large File

-------------------------- EXAMPLE 1 --------------------------

By default, the Get-Content cmdlet uses the end-of-line character as its delimiter, so it gets a file as a collection of strings, with each line as one string in the file.

You can use the Delimiter parameter to specify an alternate delimiter. If you set it to the characters that denote the end of a section or the beginning of the next section, you can split the file into logical parts.

The first command gets the Employees.txt file and splits it into sections, each of which ends with the words "End of Employee Record" and it saves it in the $e variable.

The second command uses array notation to get the first item in the collection in $e. It uses an index of 0, because Windows PowerShell arrays are zero-based.

For more information about arrays, see about_Arrays.

```
$e = get-content c:\test\employees.txt -delimited "End Of Employee Record"

$e[0]
```

TASK: Navigating the File System

-------------------------- EXAMPLE 1 --------------------------

This command gets the current location:

```
get-location
```

The Get-Location cmdlet includes the functionality of commands like the cd command in the Windows Command Prompt and the pwd command in UNIX. For more information, type:

```
    get-help get-location
```

-------------------------- EXAMPLE 2 --------------------------

This command sets the current location:

```
set-location C:
```

TASK: Getting File and Directory Information

-------------------------- EXAMPLE 1 --------------------------

This command gets all the files and directories in the current directory:

```
get-childitem
```

By default, the Get-ChildItem cmdlet does not recurse. If files and folders are present in the current directory when you run this command, a System.IO.FileInfo object and a System.IO.DirectoryInfo object are returned.

-------------------------- EXAMPLE 2 --------------------------

This command gets all the files in the current directory:

```
get-childitem | where-object {!$_.psiscontainer}
```

The command uses the Get-ChildItem cmdlet to get all files and directories. It pipes the results to the Where-Object cmdlet, which selects only the objects that are not (!) containers.

-------------------------- EXAMPLE 3 --------------------------

The command uses the Get-ChildItem cmdlet to get all files and directories. It pipes the results to Where-Object, which select only the objects that are containers.

```
get-childitem | where-object {$_.psiscontainer}
```

-------------------------- EXAMPLE 4 --------------------------

This command displays the properties of a directory:

```
get-item -path c:\ps-test | format-list -property *
```

The command uses the Path parameter of the Get-Item cmdlet to get the C:\ps-test directory. It pipes the directory object to the Format-List cmdlet, which displays all (*) the properties and values of the directory in a list.

-------------------------- EXAMPLE 5 --------------------------

This command displays the properties of a file:

```
get-item -path test.txt | format-list -property *
```

The command uses the Path parameter of the Get-Item cmdlet to get the test.txt file. It pipes the file object to the Format-List cmdlet, which displays all (*) the properties and values of the file in a list.

TASK: Copying Files and Directories

-------------------------- EXAMPLE 1 --------------------------

This command copies the A.txt file from the C:\A directory to the C:\A\Bb directory:

```
copy-item -path C:\a\a.txt -destination C:\a\bb\a.txt
```

It overwrites files in the destination directory without prompting for confirmation.

-------------------------- EXAMPLE 2 --------------------------

This command copies all the files in the C:\A\Bb directory that have the .txt file name extension to the C:\A\Cc\Ccc\ directory:

```
copy-item -path C:\a\bb\*.txt -destination C:\a\cc\ccc\
```

It uses the original names of the files. The command overwrites the existing files in the destination directory without prompting for confirmation.

-------------------------- EXAMPLE 3 --------------------------

Copies all the directories and files in the C:\a directory to the C:\c directory. If any of the directories to copy already exist in the destination directory, the command will fail unless you specify the Force parameter.

```
copy-item -path C:\a\* -destination C:\c -recurse
```

TASK: Moving Files and Directories

    -------------------------- EXAMPLE 1 --------------------------

    This command moves the C.txt file in the C:\A directory to the C:\A\Aa
    directory:

    move-item -path C:\a\c.txt -destination C:\a\aa

    The command will not automatically overwrite an existing file that has the
    same name. To force the cmdlet to overwrite an existing file, specify the
    Force parameter.
    -------------------------- EXAMPLE 2 --------------------------

    This command moves the C:\A directory and all its contents to the C:\B
    directory:

    move-item -path C:\a -destination C:\b

    You cannot move a directory when that directory is the current location.

TASK: Managing File Content

    -------------------------- EXAMPLE 1 --------------------------

    This command appends the "test content" string to the Test.txt file:

    add-content -path test.txt -value "test content"

    The existing content in the Test.txt file is not deleted.
    -------------------------- EXAMPLE 2 --------------------------

    This command gets the contents of the Test.txt file and displays them in
    the console:

    get-content -path test.txt

    You can pipe the contents of the file to another cmdlet. For example, the
    following command reads the contents of the Test.txt file and then
    supplies them as input to the ConvertTo-HTML cmdlet:

    get-content -path test.txt | convertto-html
    -------------------------- EXAMPLE 3 --------------------------

    This command replaces the contents of the Test.txt file with the "test
    content" string:

    set-content -path test.txt -value "test content"

    It overwrites the contents of Test.txt. You can use the Value parameter of
    the New-Item cmdlet to add content to a file when you create it.

TASK: Managing Security Descriptors

    -------------------------- EXAMPLE 1 --------------------------

This command returns a System.Security.AccessControl.FileSecurity object:

get-acl -path test.txt | format-list -property *

For more information about this object, pipe the command to the Get-Member cmdlet. Or, see "FileSecurity Class" in the MSDN (Microsoft Developer Network) library at http://go.microsoft.com/fwlink/?LinkId=145718.

-------------------------- EXAMPLE 2 --------------------------

This command returns a System.Security.AccessControl.DirectorySecurity object:

get-acl -path test_directory | format-list -property *

For more information about this object, pipe the command to the Get-Member cmdlet. Or, see "DirectorySecurity Class" in the MSDN library at http://go.microsoft.com/fwlink/?LinkId=145736.

TASK: Creating Files and Directories

-------------------------- EXAMPLE 1 --------------------------

This command creates the Logfiles directory on the C drive:

new-item -path c:\ -name logfiles -type directory

-------------------------- EXAMPLE 2 --------------------------

This command creates the Log2.txt file in the C:\Logfiles directory and then adds the "test log" string to the file:

new-item -path c:\logfiles -name log.txt -type file

-------------------------- EXAMPLE 3 --------------------------

Creates a file called Log2.txt in the C:\logfiles directory and adds the string "test log" to the file.

new-item -path c:\logfiles -name log2.txt -type file -value "test log"

TASK: Renaming Files and Directories

-------------------------- EXAMPLE 1 --------------------------

This command renames the A.txt file in the C:\A directory to B.txt:

rename-item -path c:\a\a.txt -newname b.txt

-------------------------- EXAMPLE 2 --------------------------

This command renames the C:\A\Cc directory to C:\A\Dd:

rename-item -path c:\a\cc -newname dd

TASK: Deleting Files and Directories

-------------------------- EXAMPLE 1 --------------------------

This command deletes the Test.txt file in the current directory:

```
remove-item -path test.txt
```

-------------------------- EXAMPLE 2 --------------------------

This command deletes all the files in the current directory that have the .xml file name extension:

```
remove-item -path *.xml
```

TASK: Starting a Program by Invoking an Associated File

-------------------------- EXAMPLE 1 --------------------------

The first command uses the Get-Service cmdlet to get information about local services.

It pipes the information to the Export-Csv cmdlet and then stores that information in the Services.csv file.

The second command uses Invoke-Item to open the Services.csv file in the program associated with the .csv extension:

```
get-service | export-csv -path services.csv

invoke-item -path services.csv
```

TASK: Getting Files and Folders with Specified Attributes

-------------------------- EXAMPLE 1 --------------------------

This command gets system files in the current directory and its subdirectories.

It uses the File parameter to get only files (not directories) and the System parameter to get only items with the System attribute.

It uses the Recurse parameter to get the items in the current directory and all subdirectories.

```
get-childitem -file -system  -recurse
```

-------------------------- EXAMPLE 2 --------------------------

This command gets all files, including hidden files, in the current directory.

It uses the Attributes parameter with two values, !Directory+Hidden, which gets hidden files, and !Directory, which gets all other files.

```
Get-ChildItem -attributes !Directory,!Directory+Hidden
```

"dir -att !d,!d+h" is the equivalent of this command.
-------------------------- EXAMPLE 3 --------------------------

This command gets files in the current directory that are either
compressed or encrypted.

It uses the Attributes parameter with two values, Compressed and
Encrypted. The values are separated by a comma (,) which represents the OR
operator.

Get-ChildItem -attributes Compressed,Encrypted

DYNAMIC PARAMETERS
-Encoding <Microsoft.PowerShell.Commands.FileSystemCmdletProviderEncoding>
Specifies the file encoding. The default is ASCII.

Unknown
The encoding type is unknown or invalid. The data can be treated
as binary.

String
Uses the encoding type for a string.

Unicode
Encodes in UTF-16 format using the little-endian byte order.

Byte
Encodes a set of characters into a sequence of bytes.

BigEndianUnicode
Encodes in UTF-16 format using the big-endian byte order.

UTF8
Encodes in UTF-8 format.

UTF7
Encodes in UTF-7 format.

ASCII
Uses the encoding for the ASCII (7-bit) character set.

Cmdlets Supported: Add-Content, Get-Content, Set-Content

-Delimiter <System.String>
Specifies the delimiter that Get-Content uses to divide the file into
objects while it reads.

The default is "\n", the end-of-line character.
Therefore, by default, when reading a text file, Get-Content returns a
collection of string objects, each of which ends with an end-of-line
character.

When you enter a delimiter that does not exist in the file, Get-Content
returns the entire file as a single, undelimited object.

You can use this parameter to split a large file into smaller files by
specifying a file separator, such as "End of Example", as the delimiter.
The delimiter is preserved (not discarded) and becomes the last item in
each file section.

Troubleshooting Note: Currently, when the value of the Delimiter parameter
is an empty string, Get-Content does not return anything. This is a known
issue. To force Get-Content to return the entire file as a single,
undelimited string, enter a value that does not exist in the file.

Cmdlets Supported: Get-Content

-Wait <System.Management.Automation.SwitchParameter>
    Waits for content to be appended to the file. If content is appended, it
    returns the appended content. If the content has changed, it returns the
    entire file.

    When waiting, Get-Content checks the file once each second until you
    interrupt it, such as by pressing CTRL+C.

    Cmdlets Supported: Get-Content

-Attributes <FlagsExpression[System.IO.FileAttributes]>
    Gets files and folders with the specified attributes. This parameter
    supports all attributes and lets you specify complex combinations of
    attributes.

    The Attributes parameter was introduced in Windows PowerShell 3.0.

    The Attributes parameter supports the following attributes: Archive,
    Compressed, Device, Directory, Encrypted, Hidden, Normal,
    NotContentIndexed, Offline, ReadOnly, ReparsePoint, SparseFile, System,
    and Temporary. For a description of these attributes, see the
    FileAttributes enumeration at
    http://go.microsoft.com/fwlink/?LinkId=201508.

    Use the following operators to combine attributes.
    ! NOT
    + AND
    , OR

    No spaces are permitted between an operator and its attribute. However,
    spaces are permitted before commas.

    Cmdlets Supported: Get-ChildItem

-Directory <System.Management.Automation.SwitchParameter>
    Gets directories (folders).

    The Directory parameter was introduced in Windows PowerShell 3.0.

    To get only directories, use the Directory parameter and omit the File
    parameter. To exclude directories, use the File parameter and omit the
    Directory parameter, or use the Attributes parameter.

    Cmdlets Supported: Get-ChildItem

-File <System.Management.Automation.SwitchParameter>
    Gets files.

    The File parameter was introduced in Windows PowerShell 3.0.

    To get only files, use the File parameter and omit the Directory
    parameter. To exclude files, use the Directory parameter and omit the File
    parameter, or use the Attributes parameter.

    Cmdlets Supported: Get-ChildItem

-Hidden <System.Management.Automation.SwitchParameter>
    Gets only hidden files and directories (folders). By default,
    Get-ChildItem gets only non-hidden items.

    The Hidden parameter was introduced in Windows PowerShell 3.0.

    To get only hidden items, use the Hidden parameter, its "h" or "ah"
    aliases, or the Hidden value of the Attributes parameter. To exclude

hidden items, omit the Hidden parameter or use the Attributes parameter.

Cmdlets Supported: Get-ChildItem

-ReadOnly <System.Management.Automation.SwitchParameter>
Gets only read-only files and directories (folders).

The ReadOnly parameter was introduced in Windows PowerShell 3.0.

To get only read-only items, use the ReadOnly parameter, its "ar" alias, or the ReadOnly value of the Attributes parameter. To exclude read-only items, use the Attributes parameter.

Cmdlets Supported: Get-ChildItem

-System <System.Management.Automation.SwitchParameter>
Gets only system files and directories (folders).

The System parameter was introduced in Windows PowerShell 3.0.

To get only system files and folders, use the System parameter, its "as" alias, or the System value of the Attributes parameter. To exclude system files and folders, use the Attributes parameter.

Cmdlets Supported: Get-ChildItem

-NewerThan <System.DateTime>
Returns "True" ($True) when the LastWriteTime value of a file is greater than the specified date. Otherwise, it returns "False" ($False).

Enter a DateTime object, such as one that the Get-Date cmdlet returns, or a string that can be converted to a DateTime object, such as "August 10, 2011 2:00 PM".

Cmdlets Supported: Test-Path

-OlderThan <System.DateTime>
Returns "True" ($True) when the LastWriteTime value of a file is less than the specified date. Otherwise, it returns "False" ($False).

Enter a DateTime object, such as one that the Get-Date cmdlet returns, or a string that can be converted to a DateTime object, such as "August 10, 2011 2:00 PM".

Cmdlets Supported: Test-Path

-Stream <System.String []>
Manages alternate data streams. Enter the stream name. Wildcards are valid only in Get-Item and Remove-Item commands.

Cmdlets Supported: Add-Content, Clear-Content, Get-Item, Get-Content, Remove-Item, Set-Content

```
    -Raw <SwitchParameter>
        Ignores newline characters. Returns contents as a single item.




        Cmdlets Supported: Get-Content

NOTES


RELATED LINKS
    about_Providers
```