Full credit is given to the above companies including the OS that this PDF file was generated!

PowerShell Get-Help on command 'Select-Xml'

PS C:\Users\wahid> Get-Help Select-Xml

NAME

Select-Xml

SYNOPSIS

Finds text in an XML string or document.

SYNTAX

Select-Xml [-XPath] <System.String> -Content <System.String[]> [-Namespace <System.Collections.Hashtable>] [<CommonParameters>]

Select-Xml [-XPath] <System.String> -LiteralPath <System.String[]> [-Namespace <System.Collections.Hashtable>] [<CommonParameters>]

Select-Xml [-XPath] <System.String> [-Path] <System.String[]> [-Namespace <System.Collections.Hashtable>] [<CommonParameters>]

Select-Xml [-XPath] <System.String> [-Xml] <System.Xml.XmlNode[]> [-Namespace <System.Collections.Hashtable>] [<CommonParameters>]

DESCRIPTION

The `Select-Xml` cmdlet lets you use XPath queries to search for text in XML strings and documents. Enter an XPath query, and use the Content , Path , or Xml parameter to specify the XML to be searched.

PARAMETERS

-Content <System.String[]>

Specifies a string that contains the XML to search. You can also pipe strings to `Select-Xml`.

-LiteralPath <System.String[]>

Specifies the paths and file names of the XML files to search. Unlike Path , the value of the LiteralPath parameter is used exactly as it is typed. No characters are interpreted as wildcards. If the path includes escape characters, enclose it in single quotation marks. Single quotation marks tell PowerShell not to interpret any characters as escape sequences.

-Namespace <System.Collections.Hashtable>

Specifies a hash table of the namespaces used in the XML. Use the format`@{<namespaceName> = <namespaceValue>}`.

When the XML uses the default namespace, which begins with xmlns, use an arbitrary key for the namespace name. You cannot use xmlns. In the XPath statement, prefix each node name with the namespace name and a colon, such as `//namespaceName:Node`.

-Path <System.String[]>

Specifies the path and file names of the XML files to search. Wildcard characters are permitted.

-Xml <System.Xml.XmlNode[]>

Specifies one or more XML nodes.

An XML document will be processed as a collection of XML nodes. If you pipe an XML document to `Select-Xml`, each document node will be searched separately as it comes through the pipeline.

-XPath <System.String>

Specifies an XPath search query. The query language is case-sensitive. This parameter is required.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

---------- Example 1: Select AliasProperty nodes ----------

```
$Path = "$Pshome\Types.ps1xml"
$XPath = "/Types/Type/Members/AliasProperty"
Select-Xml -Path $Path -XPath $Xpath | Select-Object -ExpandProperty Node

Name               ReferencedMemberName
----               --------------------
Count              Length
Name               Key
Name               ServiceName
RequiredServices   ServicesDependedOn
ProcessName        Name
Handles            Handlecount
VM                 VirtualSize
WS                 WorkingSetSize
Name               ProcessName
Handles            Handlecount
```

VM VirtualMemorySize

WS WorkingSet

PM PagedMemorySize

NPM NonpagedSystemMemorySize

Name __Class

Namespace ModuleName

The result shows the Name and ReferencedMemberName of each alias property in the `Types.ps1xml` file. For example, there is a Count property that is an alias of the Length property.

---------- Example 2: Input an XML document ----------

```
[xml]$Types = Get-Content $Pshome\Types.ps1xml
Select-Xml -Xml $Types -XPath "//MethodName"
```

---------- Example 3: Search PowerShell Help files ----------

```
$Namespace = @{
   command = "http://schemas.microsoft.com/maml/dev/command/2004/10"
   maml = "http://schemas.microsoft.com/maml/2004/10"
   dev = "http://schemas.microsoft.com/maml/dev/2004/10"
}

$Path = "$Pshome\en-us\*dll-Help.xml"
$Xml = Select-Xml -Path $Path -Namespace $Namespace -XPath "//command:name"
$Xml | Format-Table @{Label="Name"; Expression= {($_.node.innerxml).trim()}},
Path -AutoSize
```

Name Path

---- ----

Export-Counter C:\Windows\system32\WindowsPowerShell\v1.0\en-us\Micros

oft.PowerShell.Commands.Diagnostics.dll-Help.xml

```
Get-Counter             C:\Windows\system32\WindowsPowerShell\v1.0\en-us\Micros
oft.PowerShell.Commands.Diagnostics.dll-Help.xml
Get-WinEvent            C:\Windows\system32\WindowsPowerShell\v1.0\en-us\Micros
oft.PowerShell.Commands.Diagnostics.dll-Help.xml
Import-Counter          C:\Windows\system32\WindowsPowerShell\v1.0\en-us\Micros
oft.PowerShell.Commands.Diagnostics.dll-Help.xml
Add-Computer            C:\Windows\system32\WindowsPowerShell\v1.0\en-us\Micros
oft.PowerShell.Commands.Management.dll-Help.xml
Add-Content             C:\Windows\system32\WindowsPowerShell\v1.0\en-us\Micros
oft.PowerShell.Commands.Management.dll-Help.xml
Checkpoint-Computer     C:\Windows\system32\WindowsPowerShell\v1.0\en-us\Micros
oft.PowerShell.Commands.Management.dll-Help.xml
...



---------- Example 4: Different ways to input XML ----------


$Xml = @"
<?xml version="1.0" encoding="utf-8"?>
<Book>
  <projects>
    <project name="Book1" date="2009-01-20">
      <editions>
        <edition language="English">En.Book1.com</edition>
        <edition language="German">Ge.Book1.Com</edition>
        <edition language="French">Fr.Book1.com</edition>
        <edition language="Polish">Pl.Book1.com</edition>
      </editions>
    </project>
  </projects>
</Book>
"@
```

```
Select-Xml -Content $Xml -XPath "//edition" | foreach {$_.node.InnerXML}

En.Book1.com
Ge.Book1.Com
Fr.Book1.com
Pl.Book1.com

$Xml | Select-Xml -XPath "//edition" | foreach {$_.node.InnerXML}

En.Book1.com
Ge.Book1.Com
Fr.Book1.com
Pl.Book1.com


---------- Example 5: Use the default xmlns namespace ----------

$SnippetNamespace = @{snip =
"http://schemas.microsoft.com/PowerShell/Snippets"}

Select-Xml -Path $HOME\Documents\WindowsPowerShell\Snippets -Namespace
$SnippetNamespace -XPath "//snip:Title" |
    ForEach-Object {$_.Node.Innerxml}
```

REMARKS

To see the examples, type: "get-help Select-Xml -examples".

For more information, type: "get-help Select-Xml -detailed".

For technical information, type: "get-help Select-Xml -full".

For online help, type: "get-help Select-Xml -online"