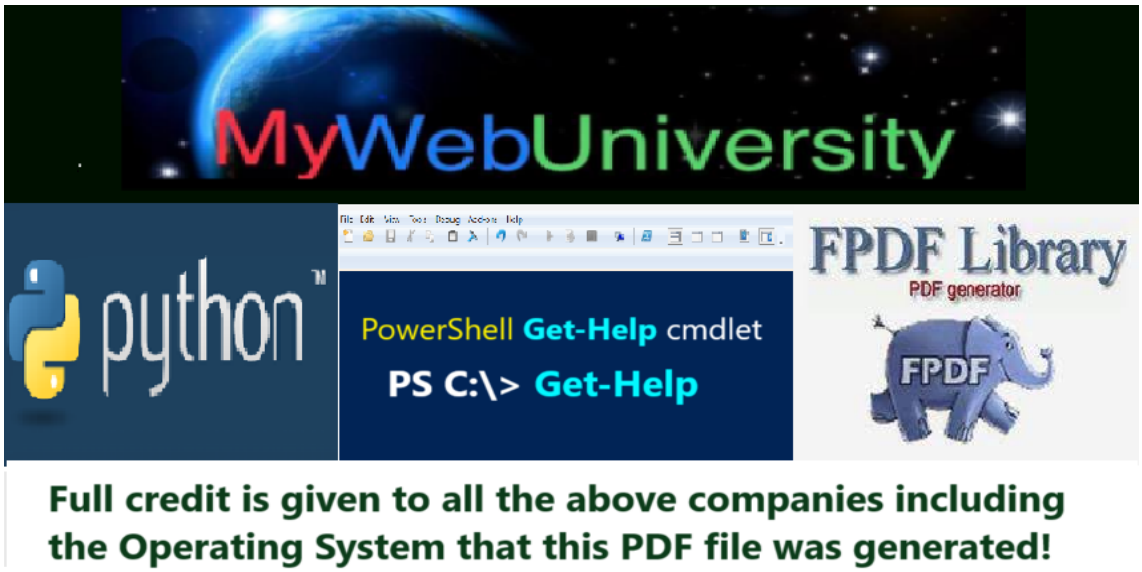

Windows PowerShell Get-Help on Cmdlet 'Invoke-WebRequest'

PS:\>Get-HELP Invoke-WebRequest -Full

NAME

Invoke-WebRequest

SYNOPSIS

Gets content from a web page on the internet.

SYNTAX

Invoke-WebRequest [-Uri] <System.Uri> [-Body <System.Object>] [-Certificate <System.Security.Cryptography.X509Certificates.X509Certificate>] [-CertificateThumbprint <System.String>] [-ContentType <System.String>] [-Credential <System.Management.Automation.PSCredential>] [-DisableKeepAlive] [-Headers <System.Collections.IDictionary>] [-InFile <System.String>] [-MaximumRedirection <System.Int32>] [-Method {Default | Get | Head | Post | Put | Delete | Trace | Options | Merge | Patch}] [-OutFile <System.String>] [-PassThru] [-Proxy <System.Uri>] [-ProxyCredential <System.Management.Automation.PSCredential>] [-ProxyUseDefaultCredentials] [-SessionVariable <System.String>] [-TimeoutSec <System.Int32>] [-TransferEncoding {chunked | compress | deflate | gzip | identity}] [-UseBasicParsing] [-UseDefaultCredentials] [-UserAgent <System.String>] [-WebSession <Microsoft.PowerShell.Commands.WebRequestSession>] [<CommonParameters>]

DESCRIPTION

The `Invoke-WebRequest` cmdlet sends HTTP, HTTPS, FTP, and FILE requests to a web page or web service. It parses the response and returns collections of forms, links,

images, and other significant HTML elements.

This cmdlet was introduced in Windows PowerShell 3.0.

> [!NOTE] > By default, script code in the web page may be run when the page is being parsed to populate the > `ParsedHtml` property. Use the `-UseBasicParsing`

switch to suppress this.

> [!IMPORTANT] > The examples in this article reference hosts in the `contoso.com` domain. This is a fictitious > domain used by Microsoft for examples. The examples

are designed to show how to use the cmdlets. > However, since the `contoso.com` sites don't exist, the examples don't work. Adapt the examples > to hosts in your

environment.

PARAMETERS

-Body <System.Object>

Specifies the body of the request. The body is the content of the request that follows the headers. You can also pipe a body value to `Invoke-WebRequest`.

The Body parameter can be used to specify a list of query parameters or specify the content of the response.

When the input is a GET request and the body is an IDictionary (typically, a hash table), the body is added to the URI as query parameters. For other request

types (such as POST), the body is set as the value of the request body in the standard `name=value` format.

When the body is a form, or it is the output of an `Invoke-WebRequest` call, PowerShell sets the request content to the form fields. For example:

`$r = Invoke-WebRequest https://website.com/login.aspx` `$r.Forms[0].Name = "MyName"` `$r.Forms[0].Password = "MyPassword"` `Invoke-RestMethod

https://website.com/service.aspx -Body $r`

- or -

`Invoke-RestMethod https://website.com/service.aspx -Body $r.Forms[0]`

Required? false

Position? named

Default value None

Accept pipeline input? True (ByValue)

Accept wildcard characters? false

-Certificate <System.Security.Cryptography.X509Certificates.X509Certificate>

Specifies the client certificate that's used for a secure web request. Enter a variable that contains a certificate or a command or expression that gets the

certificate.

To find a certificate, use `Get-PfxCertificate` or use the `Get-ChildItem` cmdlet in the Certificate (`Cert:`) drive. If the certificate isn't valid or doesn't

have sufficient authority, the command fails.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-CertificateThumbprint <System.String>

Specifies the digital public key certificate (X509) of a user account that has permission to send the request. Enter the certificate thumbprint of the certificate.

Certificates are used in client certificate-based authentication. Certificates can only be mapped only to local user accounts, not domain accounts.

To see the certificate thumbprint, use the `Get-Item` or `Get-ChildItem` command to find the certificate in `Cert:\CurrentUser\My`.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-ContentType <System.String>

Specifies the content type of the web request.

If this parameter is omitted and the request method is POST, `Invoke-WebRequest` sets the content type to `application/x-www-form-urlencoded`. Otherwise, the

content type isn't specified in the call.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-Credential <System.Management.Automation.PSCredential>

Specifies a user account that has permission to send the request. The default is the current user.

Type a user name, such as User01 or Domain01\User01 , or enter a PSCredential object generated by the `Get-Credential` cmdlet.

Credentials are stored in a PSCredential (/dotnet/api/system.management.automation.pscredential)object and the

password is stored as a SecureString

(/dotnet/api/system.security.securestring).

> [!NOTE] > For more information about SecureString data protection, see > How secure is SecureString?

(/dotnet/api/system.security.securestring#how-secure-is-securestring).

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-DisableKeepAlive <System.Management.Automation.SwitchParameter>

Indicates that the cmdlet sets the KeepAlive value in the HTTP header to False . By default, KeepAlive is True . KeepAlive establishes a persistent connection to

the server to facilitate subsequent requests.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-Headers <System.Collections.IDictionary>

Specifies the headers of the web request. Enter a hash table or dictionary.

To set UserAgent headers, use the UserAgent parameter. You cannot use this parameter to specify UserAgent or cookie headers.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-InFile <System.String>

Gets the content of the web request from a file.

Enter a path and file name. If you omit the path, the default is the current location.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-MaximumRedirection <System.Int32>

Specifies how many times PowerShell redirects a connection to an alternate Uniform Resource Identifier (URI) before the connection fails. The default value is 5.

A value of 0 (zero) prevents all redirection.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-Method <Microsoft.PowerShell.Commands.WebRequestMethod>

Specifies the method used for the web request. The acceptable values for this parameter are:

- `Default`

- `Delete`

- `Get`

- `Head`

- `Merge`

- `Options`

- `Patch`

- `Post`

- `Put`

- `Trace`

Required? false
Position? named
Default value None
Accept pipeline input? False
Accept wildcard characters? false

-OutFile <System.String>

Specifies the output file for which this cmdlet saves the response body. Enter a path and file name. If you omit the path, the default is the current location.

By default, `Invoke-WebRequest` returns the results to the pipeline. To send the results to a file and to the pipeline, use the Passthru parameter.

Required? false
Position? named
Default value None
Accept pipeline input? False
Accept wildcard characters? false

-PassThru <System.Management.Automation.SwitchParameter>

Indicates that the cmdlet returns the results, in addition to writing them to a file. This parameter is valid only when the OutFile parameter is also used in the

command.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-Proxy <System.Uri>

Specifies a proxy server for the request, rather than connecting directly to the Internet resource. Enter the URI of a network proxy server.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-ProxyCredential <System.Management.Automation.PSCredential>

Specifies a user account that has permission to use the proxy server specified by the Proxy parameter. The default is the current user.

Type a user name, such as `User01` or `Domain01\User01`, or enter a PSCredential object, such as one generated by the `Get-Credential` cmdlet.

This parameter is valid only when the Proxy parameter is also used in the command. You can't use the ProxyCredential and ProxyUseDefaultCredentials parameters in

the same command.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-ProxyUseDefaultCredentials <System.Management.Automation.SwitchParameter>

Indicates that the cmdlet uses the credentials of the current user to access the proxy server that is specified by the Proxy parameter.

This parameter is valid only when the Proxy parameter is also used in the command. You can't use the ProxyCredential and ProxyUseDefaultCredentials parameters in

the same command.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-SessionVariable <System.String>

Specifies a variable for which this cmdlet creates a web request session and saves it in the value. Enter a variable name without the dollar sign (`$`) symbol.

When you specify a session variable, `Invoke-WebRequest` creates a web request session object and assigns it to a variable with the specified name in your

PowerShell session. You can use the variable in your session as soon as the command completes.

Unlike a remote session, the web request session isn't a persistent connection. It's an object that contains information about the connection and the request,

including cookies, credentials, the maximum redirection value, and the user agent string. You can use it to share state and data among web requests.

To use the web request session in subsequent web requests, specify the session variable in the value of the

WebSession parameter. PowerShell uses the data in the

web request session object when establishing the new connection. To override a value in the web request session, use a cmdlet parameter, such as UserAgent or

Credential . Parameter values take precedence over values in the web request session.

You can't use the SessionVariable and WebSession parameters in the same command.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-TimeoutSec <System.Int32>

Specifies how long the request can be pending before it times out. Enter a value in seconds. The default value, 0, specifies an indefinite time-out.

A Domain Name System (DNS) query can take up to 15 seconds to return or time out. If your request contains a host name that requires resolution, and you set

TimeoutSec to a value greater than zero, but less than 15 seconds, it can take 15 seconds or more before a WebException is thrown, and your request times out.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-TransferEncoding <System.String>

Specifies a value for the transfer-encoding HTTP response header. The acceptable values for this parameter are:

- `Chunked`

- `Compress`

- `Deflate`

- `GZip`

- `Identity`

Required? false
Position? named
Default value None
Accept pipeline input? False
Accept wildcard characters? false

-Uri <System.Uri>

Specifies the Uniform Resource Identifier (URI) of the Internet resource to which the web request is sent. Enter a URI. This parameter supports HTTP, HTTPS, FTP,

and FILE values.

This parameter is required. The parameter name Uri is optional.

Required? true
Position? 0
Default value None
Accept pipeline input? False
Accept wildcard characters? false

-UseBasicParsing <System.Management.Automation.SwitchParameter>

Indicates that the cmdlet uses the response object for HTML content without Document Object Model (DOM) parsing. This parameter is required when Internet Explorer

is not installed on the computers, such as on a Server Core installation of a Windows Server operating system.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-UseDefaultCredentials <System.Management.Automation.SwitchParameter>

Indicates that the cmdlet uses the credentials of the current user to send the web request.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-UserAgent <System.String>

Specifies a user agent string for the web request. The default user agent is similar to `Mozilla/5.0 (Windows NT; Windows NT 6.1; en-US) WindowsPowerShell/3.0`

with slight variations for each operating system and platform.

To test a website with the standard user agent string that is used by most Internet browsers, use the properties of the PSUserAgent

(/dotnet/api/microsoft.powershell.commands.psuseragent)class, such as Chrome, FireFox, InternetExplorer, Opera, and Safari. For example, the following command

uses the user agent string for Internet Explorer: `Invoke-WebRequest -Uri https://website.com/ -UserAgent ([Microsoft.PowerShell.Commands.PSUserAgent]::InternetExplorer)`

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-WebSession <Microsoft.PowerShell.Commands.WebRequestSession>

Specifies a web request session. Enter the variable name, including the dollar sign (`$`).

To override a value in the web request session, use a cmdlet parameter, such as UserAgent or Credential . Parameter values take precedence over values in the web

request session.

Unlike a remote session, the web request session is not a persistent connection. It is an object that contains information about the connection and the request,

including cookies, credentials, the maximum redirection value, and the user agent string. You can use it to share state and data among web requests.

To create a web request session, enter a variable name, without a dollar sign, in the value of the SessionVariable parameter of an `Invoke-WebRequest` command.

`Invoke-WebRequest` creates the session and saves it in the variable. In subsequent commands, use the variable as the value of the WebSession parameter.

You can't use the SessionVariable and WebSession parameters in the same command.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

<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).

INPUTS

System.Object

You can pipe the body of a web request to this cmdlet.

OUTPUTS

Microsoft.PowerShell.Commands.HtmlWebResponseObject

This cmdlet returns the response object representing the result of the web request.

NOTES

Windows PowerShell includes the following aliases for `Invoke-WebRequest`:

- `iwr`

---------- Example 1: Send a web request ----------

```
$Response = Invoke-WebRequest -UseBasicParsing -URI https://www.bing.com?q=how+many+feet+in+a+mile
$Response.InputFields |
    Where-Object name -like "* Value" |
    Select-Object name, value

name       value
----       -----
From Value 1
To Value   5280
```

The data returned by `Invoke-WebRequest` is stored in the `$Response` variable. The InputFields property of the response contains the form fields. `Where-Object` is

used to filter the form fields to those where the name property is like "* Value". The filtered results are piped to `Select-Object` to select the name and value

properties.

---------- Example 2: Use a stateful web service ----------

```
$R = Invoke-WebRequest https://www.facebook.com/login.php -SessionVariable fb
# This command stores the first form in the Forms property of the $R variable in the $Form variable.
$Form = $R.Forms[0]
# This command shows the fields available in the Form.
$Form.fields

Key                 Value
---                 -----
...
email
pass
...

# These commands populate the username and password of the respective Form fields.
$Form.Fields["email"]="User01@Fabrikam.com"
$Form.Fields["pass"]="P@ssw0rd"
# This command creates the Uri that will be used to log in to facebook.
# The value of the Uri parameter is the value of the Action property of the form.
$Uri = "https://www.facebook.com" + $Form.Action
# Now the Invoke-WebRequest cmdlet is used to sign into the Facebook web service.
# The WebRequestSession object in the $FB variable is passed as the value of the WebSession parameter.
# The value of the Body parameter is the hash table in the Fields property of the form.
# The value of the *Method* parameter is POST. The command saves the output in the $R variable.
$R = Invoke-WebRequest -Uri $Uri -WebSession $FB -Method POST -Body $Form.Fields
$R.StatusDescription
```

The first command uses the `Invoke-WebRequest` cmdlet to send a sign-in request. The command specifies a value of "FB" for the value of the SessionVariable parameter,

and saves the result in the `$R` variable. When the command completes, the `$R` variable contains an HtmlWebResponseObject and the `$FB` variable contains a

WebRequestSession object.

After the `Invoke-WebRequest` cmdlet signs in to facebook, the StatusDescription property of the web response object in

the `$R` variable indicates that the user is

signed in successfully.

---------- Example 3: Get links from a web page ----------

```
(Invoke-WebRequest -Uri "https://devblogs.microsoft.com/powershell/").Links.Href
```

The `Invoke-WebRequest` cmdlet gets the web page content. Then the Links property of the returned HtmlWebResponseObject is used to display the Href property of each

link.

- Example 4: Catch non success messages from Invoke-WebRequest -

```
try
{
    $Response = Invoke-WebRequest -Uri "www.microsoft.com/unkownhost"
    # This will only execute if the Invoke-WebRequest is successful.
    $StatusCode = $Response.StatusCode
}
catch
{
    $StatusCode = $_.Exception.Response.StatusCode.value__
}
$StatusCode
```

```
404
```

The terminating error is caught by the `catch` block, which retrieves the StatusCode from the Exception object.

----- Example 8: Download multiple files at the same time -----

```
$baseUri = 'https://github.com/PowerShell/PowerShell/releases/download'
$files = @(
    @{
        Uri = "$baseUri/v7.3.0-preview.5/PowerShell-7.3.0-preview.5-win-x64.msi"
        OutFile = 'PowerShell-7.3.0-preview.5-win-x64.msi'
```

```
    },
    @{
        Uri = "$baseUri/v7.3.0-preview.5/PowerShell-7.3.0-preview.5-win-x64.zip"
        OutFile = 'PowerShell-7.3.0-preview.5-win-x64.zip'
    },
    @{
        Uri = "$baseUri/v7.2.5/PowerShell-7.2.5-win-x64.msi"
        OutFile = 'PowerShell-7.2.5-win-x64.msi'
    },
    @{
        Uri = "$baseUri/v7.2.5/PowerShell-7.2.5-win-x64.zip"
        OutFile = 'PowerShell-7.2.5-win-x64.zip'
    }
)

$jobs = @()

foreach ($file in $files) {
    $jobs += Start-ThreadJob -Name $file.OutFile -ScriptBlock {
        $params = $using:file
        Invoke-WebRequest @params
    }
}

Write-Host "Downloads started..."
Wait-Job -Job $jobs

foreach ($job in $jobs) {
    Receive-Job -Job $job
}
```

> [!NOTE] > To use the `Start-ThreadJob` cmdlet you must install the ThreadJob module from the PowerShell > Gallery.

RELATED LINKS

Online Version:

https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/invoke-webrequest?view=powershell-5.1&WT.mc_id=ps-gethelp

Invoke-RestMethod

ConvertFrom-Json

ConvertTo-Json