Invoke-WebRequest

Reference

Feedback

Module: Microsoft.PowerShell.Utility

In this article

Gets content from a web page on the internet.

Syntax

PowerShell Copy

```
Invoke-WebRequest
      [-UseBasicParsing]
      [-Uri] <Uri>
      [-HttpVersion <Version>]
      [-WebSession <WebRequestSession>]
      [-SessionVariable <String>]
      [-AllowUnencryptedAuthentication]
      [-Authentication <WebAuthenticationType>]
      [-Credential <PSCredential>]
      [-UseDefaultCredentials]
      [-CertificateThumbprint <String>]
      [-Certificate <X509Certificate>]
      [-SkipCertificateCheck]
      [-SslProtocol <WebSslProtocol>]
      [-Token <SecureString>]
      [-UserAgent <String>]
      [-DisableKeepAlive]
      [-ConnectionTimeoutSeconds <Int32>]
      [-OperationTimeoutSeconds <Int32>]
      [-Headers <IDictionary>]
      [-SkipHeaderValidation]
      [-AllowInsecureRedirect]
      [-MaximumRedirection <Int32>]
      [-MaximumRetryCount <Int32>]
      [-PreserveAuthorizationOnRedirect]
      [-RetryIntervalSec <Int32>]
      [-Method <WebRequestMethod>]
      [-PreserveHttpMethodOnRedirect]
      [-UnixSocket <UnixDomainSocketEndPoint>]
      [-Proxy <Uri>]
      [-ProxyCredential <PSCredential>]
      [-ProxyUseDefaultCredentials]
      [-Body <Object>]
      [-Form <IDictionary>]
      [-ContentType <String>]
      [-TransferEncoding <String>]
      [-InFile <String>]
      [-OutFile <String>]
      [-PassThru]
      [-Resume]
      [-SkipHttpErrorCheck]
      [<CommonParameters>]
```

PowerShell Copy

```
Invoke-WebRequest
      [-UseBasicParsing]
      [-Uri] <Uri>
      [-HttpVersion <Version>]
      [-WebSession <WebRequestSession>]
      [-SessionVariable <String>]
      [-AllowUnencryptedAuthentication]
      [-Authentication <WebAuthenticationType>]
      [-Credential <PSCredential>]
      [-UseDefaultCredentials]
      [-CertificateThumbprint <String>]
      [-Certificate <X509Certificate>]
      [-SkipCertificateCheck]
      [-SslProtocol <WebSslProtocol>]
      [-Token <SecureString>]
      [-UserAgent <String>]
      [-DisableKeepAlive]
      [-ConnectionTimeoutSeconds <Int32>]
      [-OperationTimeoutSeconds <Int32>]
      [-Headers <IDictionary>]
      [-SkipHeaderValidation]
      [-AllowInsecureRedirect]
      [-MaximumRedirection <Int32>]
      [-MaximumRetryCount <Int32>]
      [-PreserveAuthorizationOnRedirect]
      [-RetryIntervalSec <Int32>]
      [-Method <WebRequestMethod>]
      [-PreserveHttpMethodOnRedirect]
      [-UnixSocket <UnixDomainSocketEndPoint>]
      [-NoProxy]
      [-Body <Object>]
      [-Form <IDictionary>]
      [-ContentType <String>]
      [-TransferEncoding <String>]
      [-InFile <String>]
      [-OutFile <String>]
      [-PassThru]
      [-Resume]
      [-SkipHttpErrorCheck]
      [<CommonParameters>]
```

PowerShell Copy

```
Invoke-WebRequest
      [-UseBasicParsing]
      [-Uri] <Uri>
      [-HttpVersion <Version>]
      [-WebSession <WebRequestSession>]
      [-SessionVariable <String>]
      [-AllowUnencryptedAuthentication]
      [-Authentication <WebAuthenticationType>]
      [-Credential <PSCredential>]
      [-UseDefaultCredentials]
      [-CertificateThumbprint <String>]
      [-Certificate <X509Certificate>]
      [-SkipCertificateCheck]
      [-SslProtocol <WebSslProtocol>]
      [-Token <SecureString>]
      [-UserAgent <String>]
      [-DisableKeepAlive]
      [-ConnectionTimeoutSeconds <Int32>]
      [-OperationTimeoutSeconds <Int32>]
      [-Headers <IDictionary>]
      [-SkipHeaderValidation]
      [-AllowInsecureRedirect]
      [-MaximumRedirection <Int32>]
      [-MaximumRetryCount <Int32>]
      [-PreserveAuthorizationOnRedirect]
      [-RetryIntervalSec <Int32>]
      -CustomMethod <String>
      [-PreserveHttpMethodOnRedirect]
      [-UnixSocket <UnixDomainSocketEndPoint>]
      [-Proxy <Uri>]
      [-ProxyCredential <PSCredential>]
```

```
      [-ProxyUseDefaultCredentials]
      [-Body <Object>]
      [-Form <IDictionary>]
      [-ContentType <String>]
      [-TransferEncoding <String>]
      [-InFile <String>]
      [-OutFile <String>]
      [-PassThru]
      [-Resume]
      [-SkipHttpErrorCheck]
      [<CommonParameters>]
```

PowerShell Copy

```
Invoke-WebRequest
      [-UseBasicParsing]
      [-Uri] <Uri>
      [-HttpVersion <Version>]
      [-WebSession <WebRequestSession>]
      [-SessionVariable <String>]
      [-AllowUnencryptedAuthentication]
      [-Authentication <WebAuthenticationType>]
      [-Credential <PSCredential>]
      [-UseDefaultCredentials]
      [-CertificateThumbprint <String>]
      [-Certificate <X509Certificate>]
      [-SkipCertificateCheck]
      [-SslProtocol <WebSslProtocol>]
      [-Token <SecureString>]
      [-UserAgent <String>]
      [-DisableKeepAlive]
      [-ConnectionTimeoutSeconds <Int32>]
      [-OperationTimeoutSeconds <Int32>]
      [-Headers <IDictionary>]
      [-SkipHeaderValidation]
      [-AllowInsecureRedirect]
      [-MaximumRedirection <Int32>]
      [-MaximumRetryCount <Int32>]
      [-PreserveAuthorizationOnRedirect]
      [-RetryIntervalSec <Int32>]
      -CustomMethod <String>
      [-PreserveHttpMethodOnRedirect]
      [-UnixSocket <UnixDomainSocketEndPoint>]
      [-NoProxy]
      [-Body <Object>]
      [-Form <IDictionary>]
      [-ContentType <String>]
      [-TransferEncoding <String>]
      [-InFile <String>]
      [-OutFile <String>]
      [-PassThru]
      [-Resume]
      [-SkipHttpErrorCheck]
      [<CommonParameters>]
```

Description

The `Invoke-WebRequest` cmdlet sends HTTP and HTTPS requests to a web page or web service. It parses the response and returns collections of links, images, and other significant HTML elements.

This cmdlet was introduced in PowerShell 3.0.

Beginning in PowerShell 7.0, `Invoke-WebRequest` supports proxy configuration defined by environment variables. See the Notes section of this article.

> **ⓘ 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.

Beginning in PowerShell 7.4, character encoding for requests defaults to UTF-8 instead of ASCII. If you need a different encoding, you must set the `charset` attribute in the `Content-Type` header.

Examples

Example 1: Send a web request

This example uses the `Invoke-WebRequest` cmdlet to send a web request to the Bing.com site.

PowerShell Copy

```
$Response = Invoke-WebRequest -URI https://www.bing.com/search?q=how+many+feet+i
$Response.InputFields | Where-Object {
    $_.name -like "* Value*"
} | Select-Object Name, Value

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

The first command issues the request and saves the response in the `$Response` variable.

The second command gets any **InputField** 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

This example shows how to use the `Invoke-WebRequest` cmdlet with a stateful web service.

PowerShell Copy

```
$LoginParameters = @{
    Uri             = 'https://www.contoso.com/login/'
    SessionVariable = 'Session'
    Method          = 'POST'
    Body            = @{
        User     = 'jdoe'
        Password = 'P@S$w0rd!'
    }
}
$LoginResponse = Invoke-WebRequest @LoginParameters
$ProfileResponse = Invoke-WebRequest 'https://www.contoso.com/profile/' -WebSess
```

The first call to `Invoke-WebRequest` sends a sign-in request. The command specifies a value of `Session` for the value of the **SessionVariable** parameter. When the command completes, the `$LoginResponse` variable contains an **BasicHtmlWebResponseObject** and the `$Session` variable contains a `WebRequestSession` object. This logs the user into the site.

The second call to `Invoke-WebRequest` fetches the user's profile, which requires the user be signed into the site. The session data stored in the `$Session` variable provides session cookies to the site created during the login.

Example 3: Get links from a web page

This example gets the links in a web page. It uses the `Invoke-WebRequest` cmdlet to get the web page content. Then it uses the **Links** property of the **BasicHtmlWebResponseObject** that

`Invoke-WebRequest` returns, and the **Href** property of each link.

PowerShell Copy

```
(Invoke-WebRequest -Uri "https://aka.ms/pscore6-docs").Links.Href
```

Example 4: Write response content to a file using the encoding defined in the requested page

This example uses the `Invoke-WebRequest` cmdlet to retrieve the web page content of a PowerShell documentation page.

PowerShell Copy

```
$Response = Invoke-WebRequest -Uri "https://aka.ms/pscore6-docs"
$Stream = [System.IO.StreamWriter]::new('.\docspage.html', $false, $Response.Enc
try {
    $Stream.Write($Response.Content)
} finally {
    $Stream.Dispose()
}
```

The first command retrieves the page and saves the response object in the `$Response` variable.

The second command creates a **StreamWriter** to use to write the response content to a file. The **Encoding** property of the response object is used to set the encoding for the file.

The final few commands write the **Content** property to the file then disposes the **StreamWriter**.

Note that the **Encoding** property is null if the web request doesn't return text content.

Example 5: Submit a multipart/form-data file

This example uses the `Invoke-WebRequest` cmdlet upload a file as a `multipart/form-data` submission. The file `c:\document.txt` is submitted as the form field `document` with the `Content-Type` of `text/plain`.

PowerShell Copy

```
$FilePath = 'c:\document.txt'
$FieldName = 'document'
$ContentType = 'text/plain'

$FileStream = [System.IO.FileStream]::new($filePath, [System.IO.FileMode]::Open)
$FileHeader = [System.Net.Http.Headers.ContentDispositionHeaderValue]::new('form
$FileHeader.Name = $FieldName
$FileHeader.FileName = Split-Path -leaf $FilePath
$FileContent = [System.Net.Http.StreamContent]::new($FileStream)
$FileContent.Headers.ContentDisposition = $FileHeader
$FileContent.Headers.ContentType = [System.Net.Http.Headers.MediaTypeHeaderValue

$MultipartContent = [System.Net.Http.MultipartFormDataContent]::new()
$MultipartContent.Add($FileContent)

$Response = Invoke-WebRequest -Body $MultipartContent -Method 'POST' -Uri 'https
```

Example 6: Simplified Multipart/Form-Data Submission

Some APIs require `multipart/form-data` submissions to upload files and mixed content. This example demonstrates updating a user profile.

PowerShell Copy

```
$Uri = 'https://api.contoso.com/v2/profile'
$Form = @{
    firstName  = 'John'
    lastName   = 'Doe'
    email      = 'john.doe@contoso.com'
    avatar     = Get-Item -Path 'c:\Pictures\jdoe.png'
    birthday   = '1980-10-15'
    hobbies    = 'Hiking','Fishing','Jogging'
}
$Result = Invoke-WebRequest -Uri $Uri -Method Post -Form $Form
```

The profile form requires these fields: `firstName`, `lastName`, `email`, `avatar`, `birthday`, and `hobbies`. The API is expecting an image for the user profile pic to be supplied in the `avatar` field. The API also accepts multiple `hobbies` entries to be submitted in the same form.

When creating the `$Form` HashTable, the key names are used as form field names. By default, the values of the HashTable are converted to strings. If a **System.IO.FileInfo** value is present, the file contents are submitted. If a collection such as arrays or lists are present, the form field is submitted multiple times.

Using `Get-Item` on the `avatar` key, the `FileInfo` object is set as the value. The result is that the image data for `jdoe.png` is submitted.

By supplying a list to the `hobbies` key, the `hobbies` field is present in the submissions once for each list item.

Example 7: Catch non success messages from Invoke-WebRequest

When `Invoke-WebRequest` encounters a non-success HTTP message (404, 500, etc.), it returns no output and throws a terminating error. To catch the error and view the **StatusCode** you can enclose execution in a `try/catch` block.

PowerShell Copy

```
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

The `Invoke-WebRequest` cmdlet can only download one file at a time. The following example uses `Start-ThreadJob` to create multiple thread jobs to download multiple files at the same time.

PowerShell Copy

```
$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
}
```

Example 9: Skipping Header Validation

By default, the `Invoke-WebRequest` cmdlet validates the values of well-known headers that have a standards-defined value format. The following example shows how this validation can raise an error and how you can use the **SkipHeaderValidation** parameter to avoid validating values for endpoints that tolerate invalidly formatted values.

PowerShell Copy

```
$Uri = 'https://httpbin.org/headers'
$InvalidHeaders = @{
    'If-Match' = '12345'
}

Invoke-WebRequest -Uri $Uri -Headers $InvalidHeaders

Invoke-WebRequest -Uri $Uri -Headers $InvalidHeaders -SkipHeaderValidation

Invoke-WebRequest: The format of value '12345' is invalid.

StatusCode        : 200
StatusDescription : OK
Content           : {
                      "headers": {
                        "Host": "httpbin.org",
                        "If-Match": "12345",
                        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Microsoft W
                        "X-Amzn-Trace-Id": 
RawContent        : HTTP/1.1 200 OK
                    Date: Mon, 08 Aug 2022 16:24:24 GMT
                    Connection: keep-alive
                    Server: gunicorn/19.9.0
                    Access-Control-Allow-Origin: *
                    Access-Control-Allow-Credentials: true
                    Content-Type: application
Headers           : {[Date, System.String[]], [Connection, System.String[]], [Se
Images            : {}
InputFields       : {}
Links             : {}
```

```
RawContentLength  : 249
RelationLink      : {}
```

httpbin.org is a service that returns information about web requests and responses for troubleshooting. The `$Uri` variable is assigned to the `/headers` endpoint of the service, which returns a request's headers as the content in its response.

The `If-Match` request header is defined in RFC-7232 section 3.1 and requires the value for that header to be defined with surrounding quotes. The `$InvalidHeaders` variable is assigned a hash table where the value of `If-Match` is invalid because it's defined as `12345` instead of `"12345"`.

Calling `Invoke-WebRequest` with the invalid headers returns an error reporting that the formatted value is invalid. The request is not sent to the endpoint.

Calling `Invoke-WebRequest` with the **SkipHeaderValidation** parameter ignores the validation failure and sends the request to the endpoint. Because the endpoint tolerates non-compliant header values, the cmdlet returns the response object without error.

Example 10: Send a request using HTTP 2.0

This example gets the links in a web page using the HTTP 2.0 protocol. It uses the `Invoke-WebRequest` cmdlet to get the web page content. Then it uses the **Links** property of the **BasicHtmlWebResponseObject** that `Invoke-WebRequest` returns, and the **Href** property of each link.

PowerShell Copy

```
(Invoke-WebRequest -Uri 'https://aka.ms/pscore6-docs' -HttpVersion 2.0).Links.Hr
```

Example 11: Send a request to a Unix socket application

Some applications, such as Docker, expose a Unix socket for communication. This example queries for a list of Docker images using the Docker API. The cmdlet connects to the Docker daemon using the Unix socket.

PowerShell Copy

```
Invoke-WebRequest -Uri "http://localhost/v1.40/images/json/" -UnixSocket "/var/r
```

Parameters

-AllowInsecureRedirect

Allows redirecting from HTTPS to HTTP. By default, any request that is redirected from HTTPS to HTTP results in an error and the request is aborted to prevent unintentionally communicating in plain text over unencrypted connections. To override this behavior at your own risk, use the **AllowInsecureRedirect** parameter.

This parameter was added in PowerShell 7.4.

Expand table

Type:	SwitchParameter
Position:	Named

Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-AllowUnencryptedAuthentication

Allows sending of credentials and secrets over unencrypted connections. By default, supplying **Credential** or any **Authentication** option with a **Uri** that doesn't begin with `https://` results in an error and the request is aborted to prevent unintentionally communicating secrets in plain text over unencrypted connections. To override this behavior at your own risk, supply the **AllowUnencryptedAuthentication** parameter.

> ⚠ **Warning**
>
> Using this parameter isn't secure and isn't recommended. It is provided only for compatibility with legacy systems that can't provide encrypted connections. Use at your own risk.

This feature was added in PowerShell 6.0.0.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Authentication

Specifies the explicit authentication type to use for the request. The default is **None**. The **Authentication** parameter can't be used with the **UseDefaultCredentials** parameter.

Available Authentication Options:

- `None`: This is the default option when **Authentication** isn't supplied. No explicit authentication is used.
- `Basic`: Requires **Credential**. The credentials are sent as an RFC 7617 Basic Authentication `Authorization: Basic` header in the format of `base64(user:password)`.
- `Bearer`: Requires the **Token** parameter. Sends an RFC 6750 `Authorization: Bearer` header with the supplied token.
- `OAuth`: Requires the **Token** parameter. Sends an RFC 6750 `Authorization: Bearer` header with the supplied token.

Supplying **Authentication** overrides any `Authorization` headers supplied to **Headers** or included in **WebSession**.

This feature was added in PowerShell 6.0.0.

Expand table

Type:	WebAuthenticationType

Accepted values:	None, Basic, Bearer, OAuth
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Body

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. For query parameters, the cmdlet uses the **System.Net.WebUtility.UrlEncode** method method to encode the key-value pairs. For more information about encoding strings for URLs, see the UrlEncode() method reference.

When the input is a POST request and the body is a **String**, the value to the left of the first equals sign (`=`) is set as a key in the form data and the remaining text is set as the value. To specify multiple keys, use an **IDictionary** object, such as a hash table, for the **Body**.

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 PATCH), the body is set as the value of the request body in the standard `name=value` format with the values URL-encoded.

When the input is a **System.Xml.XmlNode** object and the XML declaration specifies an encoding, that encoding is used for the data in the request unless overridden by the **ContentType** parameter.

The **Body** parameter also accepts a `System.Net.Http.MultipartFormDataContent` object. This facilitates `multipart/form-data` requests. When a **MultipartFormDataContent** object is supplied for **Body**, any Content related headers supplied to the **ContentType**, **Headers**, or **WebSession** parameters is overridden by the Content headers of the **MultipartFormDataContent** object. This feature was added in PowerShell 6.0.0.

Expand table

Type:	Object
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	True
Accept wildcard characters:	False

-Certificate

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.

Expand table

Type:	X509Certificate
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-CertificateThumbprint

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`.

> **Note**
>
> This feature is only supported on Windows OS platforms.

Expand table

Type:	String
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-ConnectionTimeoutSeconds

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 **ConnectionTimeoutSeconds** 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.

This parameter replaced the **TimeoutSec** parameter in PowerShell 7.4. You can use **TimeoutSec** as an alias for **ConnectionTimeoutSeconds**.

Expand table

Type:	Int32
Aliases:	TimeoutSec
Position:	Named
Default value:	None

Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-ContentType

Specifies the content type of the web request.

If the value for **ContentType** contains the encoding format (as `charset`), the cmdlet uses that format to encode the body of the web request. If the **ContentType** doesn't specify an encoding format, the default encoding format is used instead. An example of a **ContentType** with an encoding format is `text/plain; charset=iso-8859-5`, which specifies the Latin/Cyrillic alphabet.

If this parameter is omitted and the request method is POST or PUT, `Invoke-WebRequest` sets the content type to `application/x-www-form-urlencoded`. Otherwise, the content type isn't specified in the call.

ContentType is overridden when a **MultipartFormDataContent** object is supplied for **Body**.

Starting in PowerShell 7.4, if you use this both this parameter and the **Headers** parameter to define the `Content-Type` header, the value specified in the **ContentType** parameter is used.

Expand table

Type:	String
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Credential

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.

Credential can be used alone or in conjunction with certain **Authentication** parameter options. When used alone, it only supplies credentials to the remote server if the remote server sends an authentication challenge request. When used with **Authentication** options, the credentials are explicitly sent.

Credentials are stored in a PSCredential object and the password is stored as a SecureString.

> **Note**
>
> For more information about **SecureString** data protection, see How secure is SecureString?.

Expand table

Type:	PSCredential
Position:	Named
Default value:	Current user
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-CustomMethod

Specifies a custom method used for the web request. This can be used if the Request Method required by the endpoint isn't an available option on the **Method**. **Method** and **CustomMethod** can't be used together.

This example makes a `TEST` HTTP request to the API:

```
Invoke-WebRequest -uri 'https://api.contoso.com/widget/' -CustomMethod 'TEST'
```

This feature was added in PowerShell 6.0.0.

Expand table

Type:	String
Aliases:	CM
Position:	Named
Default value:	None
Required:	True
Accept pipeline input:	False
Accept wildcard characters:	False

-DisableKeepAlive

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.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Form

Converts a dictionary to a `multipart/form-data` submission. **Form** may not be used with **Body**. If **ContentType** is used, it's ignored.

The keys of the dictionary are used as the form field names. By default, form values are converted to string values.

If the value is a **System.IO.FileInfo** object, then the binary file contents are submitted. The name of the file is submitted as the **filename** property. The MIME type is set as `application/octet-stream`. `Get-Item` can be used to simplify supplying the **System.IO.FileInfo** object.

$Form = @{ resume = Get-Item 'c:\Users\jdoe\Documents\John Doe.pdf' }

If the value is a collection type, such Arrays or Lists, the for field are submitted multiple times. The values of the list are treated as strings by default. If the value is a **System.IO.FileInfo** object, then the binary file contents are submitted. Nested collections aren't supported.

$Form = @{ tags = 'Vacation', 'Italy', '2017' pictures = Get-ChildItem 'c:\Users\jdoe\Pictures\2017-Italy' }

In the above example the `tags` field are supplied three times in the form, once for each of `Vacation`, `Italy`, and `2017`. The `pictures` field is also submitted once for each file in the `2017-Italy` folder. The binary contents of the files in that folder are submitted as the values.

This feature was added in PowerShell 6.1.0.

Expand table

Type:	IDictionary
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Headers

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

Content related headers, such as `Content-Type` are overridden when a **MultipartFormDataContent** object is supplied for **Body**.

Starting in PowerShell 7.4, if you use this parameter to define the `Content-Type` header and use **ContentType** parameter, the value specified in the **ContentType** parameter is used.

Expand table

Type:	IDictionary
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-HttpVersion

Specifies the HTTP version used for the request. The default is `1.1`.

Valid values are:

- 1.0
- 1.1
- 2.0
- 3.0

Expand table

Type:	Version
Position:	Named
Default value:	1.1
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-InFile

Gets the content of the web request from a file. Enter a path and filename. If you omit the path, the default is the current location.

Expand table

Type:	String
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-MaximumRedirection

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.

Expand table

Type:	Int32
Position:	Named
Default value:	5
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-MaximumRetryCount

Specifies how many times PowerShell retries a connection when a failure code between 400 and 599, inclusive or 304 is received. Also see **RetryIntervalSec** parameter for specifying number of retries.

Expand table

Type:	Int32
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Method

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`

The **CustomMethod** parameter can be used for Request Methods not listed above.

Expand table

Type:	WebRequestMethod
Accepted values:	Default, Get, Head, Post, Put, Delete, Trace, Options, Merge, Patch
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-NoProxy

Indicates that the cmdlet shouldn't use a proxy to reach the destination. When you need to bypass the proxy configured in the environment, use this switch. This feature was added in PowerShell 6.0.0.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	True
Accept pipeline input:	False
Accept wildcard characters:	False

-OperationTimeoutSeconds

This timeout applies to data reads within a stream, not to the stream time as a whole. The default value, 0, specifies an indefinite timeout.

Setting the value to 30 seconds means that any delay of longer than 30 seconds between data in the stream terminates the request. A large file that takes several minutes to download won't terminate unless the stream stalls for more than 30 seconds.

Expand table

Type:	Int32
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-OutFile

By default, `Invoke-WebRequest` returns the results to the pipeline. When you use the **OutFile** parameter, the results are saved to the specified file and not returned to the pipeline. Enter a path and filename. To send the results to a file and to the pipeline, add the **Passthru** parameter.

If you omit the path, the default is the current location. The name is treated as a literal path. Names that contain brackets (`[]`) must be enclosed in single quotes (`'`).

Starting in PowerShell 7.4, you can specify a folder path without the filename. When you do, the command uses the filename from the last segment of the resolved URI after any redirections. When you specify a folder path for **OutFile**, you can't use the **Resume** parameter.

Expand table

Type:	String
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-PassThru

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.

> **Note**
>
> When you use the **PassThru** parameter, the output is written to the pipeline but the file isn't created. This is fixed in PowerShell 7.5-preview.4. For more information, see PowerShell Issue #15409.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-PreserveAuthorizationOnRedirect

Indicates the cmdlet should preserve the `Authorization` header, when present, across redirections.

By default, the cmdlet strips the `Authorization` header before redirecting. Specifying this parameter disables this logic for cases where the header needs to be sent to the redirection location.

This feature was added in PowerShell 6.0.0.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-PreserveHttpMethodOnRedirect

Indicates the cmdlet should preserve the method of the request across redirections.

By default, the cmdlet changes the method to `GET` when redirected. Specifying this parameter disables this logic to ensure that the intended method can be used with redirection.

This feature was added in PowerShell 7.4.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Proxy

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

Expand table

Type:	Uri
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-ProxyCredential

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.

Expand table

Type:	PSCredential
Position:	Named
Default value:	Current user
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-ProxyUseDefaultCredentials

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.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Resume

Performs a best effort attempt to resume downloading a partial file. **Resume** requires **OutFile**.

Resume only operates on the size of the local file and remote file and performs no other validation that the local file and the remote file are the same.

If the local file size is smaller than the remote file size, then the cmdlet attempts to resume downloading the file and append the remaining bytes to the end of the file.

If the local file size is the same as the remote file size, then no action is taken and the cmdlet assumes the download already complete.

If the local file size is larger than the remote file size, then the local file is overwritten and the entire remote file is re-downloaded. This behavior is the same as using **OutFile** without **Resume**.

If the remote server doesn't support download resuming, then the local file is overwritten and the entire remote file is re-downloaded. This behavior is the same as using **OutFile** without **Resume**.

If the local file doesn't exist, then the local file is created and the entire remote file is downloaded. This behavior is the same as using **OutFile** without **Resume**.

This feature was added in PowerShell 6.1.0.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-RetryIntervalSec

Specifies the interval between retries for the connection when a failure code between 400 and 599, inclusive or 304 is received. Also see **MaximumRetryCount** parameter for specifying number of retries. The value must be between `1` and `[int]::MaxValue`.

When the failure code is 429 and the response includes the **Retry-After** property in its headers, the cmdlet uses that value for the retry interval, even if this parameter is specified.

Expand table

Type:	Int32
Position:	Named
Default value:	5
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-SessionVariable

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.

Before PowerShell 7.4, 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.

Starting in PowerShell 7.4, the web request session is persistent as long as the properties of the session aren't overridden in a subsequent request. When they are, the cmdlet recreates the session with the new values. The persistent sessions reduce the overhead for repeated requests, making them much faster.

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.

Expand table

Type:	String
Aliases:	SV
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-SkipCertificateCheck

Skips certificate validation checks. This includes all validations such as expiration, revocation, trusted root authority, etc.

> ⚠ **Warning**
>
> Using this parameter isn't secure and isn't recommended. This switch is only intended to be used against known hosts using a self-signed certificate for testing purposes. Use at your own risk.

This feature was added in PowerShell 6.0.0.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-SkipHeaderValidation

Indicates the cmdlet should add headers to the request without validation.

This switch should be used for sites that require header values that don't conform to standards. Specifying this switch disables validation to allow the value to be passed unchecked. When specified, all headers are added without validation.

This switch disables validation for values passed to the **ContentType**, **Headers** and **UserAgent** parameters.

This feature was added in PowerShell 6.0.0.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-SkipHttpErrorCheck

This parameter causes the cmdlet to ignore HTTP error statuses and continue to process responses. The error responses are written to the pipeline just as if they were successful.

This parameter was introduced in PowerShell 7.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-SslProtocol

Sets the SSL/TLS protocols that are permissible for the web request. By default all, SSL/TLS protocols supported by the system are allowed. **SslProtocol** allows for limiting to specific protocols for compliance purposes.

These values are defined as a flag-based enumeration. You can combine multiple values together to set multiple flags using this parameter. The values can be passed to the **SslProtocol** parameter as an array of values or as a comma-separated string of those values. The cmdlet combines the values using a binary-OR operation. Passing values as an array is the simplest option and also allows you to use tab-completion on the values. You may not be able to define multiple options on all platforms.

> **Note**
>
> On non-Windows platforms it may not be possible to supply `Tls` or `Tls12` as an option. Support for `Tls13` isn't available on all operating systems and will need to be verified on a per operating system basis.

This feature was added in PowerShell 6.0.0 and support for `Tls13` was added in PowerShell 7.1.

Expand table

Type:	WebSslProtocol
Accepted values:	Default, Tls, Tls11, Tls12
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Token

The OAuth or Bearer token to include in the request. **Token** is required by certain **Authentication** options. It can't be used independently.

Token takes a `SecureString` containing the token. To supply the token manually use the following:

```
Invoke-WebRequest -Uri $uri -Authentication OAuth -Token (Read-Host -AsSecureString)
```

This parameter was introduced in PowerShell 6.0.

Expand table

Type:	SecureString
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-TransferEncoding

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

- `Chunked`
- `Compress`
- `Deflate`
- `GZip`
- `Identity`

Expand table

Type:	String
Accepted values:	chunked, compress, deflate, gzip, identity
Position:	Named
Default value:	None

Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-UnixSocket

Specifies the name of the Unix socket to connect to. This parameter is supported on Unix-based systems and Windows version 1803 and later. For more information about Windows support of Unix sockets, see the Windows/WSL Interop with AF_UNIX blog post.

This parameter was added in PowerShell 7.4.

Expand table

Type:	UnixDomainSocketEndPoint
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-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 or HTTPS only.

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

Expand table

Type:	Uri
Position:	0
Default value:	None
Required:	True
Accept pipeline input:	False
Accept wildcard characters:	False

-UseBasicParsing

This parameter has been deprecated. Beginning with PowerShell 6.0.0, all Web requests use basic parsing only. This parameter is included for backwards compatibility only and any use of it has no effect on the operation of the cmdlet.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-UseDefaultCredentials

Indicates that the cmdlet uses the credentials of the current user to send the web request. This can't be used with **Authentication** or **Credential** and may not be supported on all platforms.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-UserAgent

Specifies a user agent string for the web request.

The default user agent is similar to `Mozilla/5.0 (Windows NT 10.0; Microsoft Windows 10.0.15063; en-US) PowerShell/6.0.0` with slight variations for each operating system and platform.

To test a website with the standard user agent string that's used by most internet browsers, use the properties of the 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)`

Expand table

Type:	String
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-WebSession

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. Content related headers, such as `Content-Type`, are also be overridden when a **MultipartFormDataContent** object is supplied for **Body**.

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 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.

Expand table

Type:	WebRequestSession
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

Inputs

Object

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

Outputs

BasicHtmlWebResponseObject

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

Notes

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

- All platforms:
  - `iwr`

Beginning with PowerShell 6.0.0 `Invoke-WebRequest` supports basic parsing only.

For more information, see BasicHtmlWebResponseObject.

Because of changes in .NET Core 3.1, PowerShell 7.0 and higher use the HttpClient.DefaultProxy property to determine the proxy configuration.

The value of this property is determined by your platform:

- **For Windows**: Reads proxy configuration from environment variables. If those variables aren't defined the property is derived from the user's proxy settings.
- **For macOS**: Reads proxy configuration from environment variables. If those variables aren't defined the property is derived from the system's proxy settings.
- **For Linux**: Reads proxy configuration from environment variables. If those variables aren't defined the property initializes a non-configured instance that bypasses all addresses.

The environment variables used for `DefaultProxy` initialization on Windows and Unix-based platforms are:

- `HTTP_PROXY`: the hostname or IP address of the proxy server used on HTTP requests.
- `HTTPS_PROXY`: the hostname or IP address of the proxy server used on HTTPS requests.
- `ALL_PROXY`: the hostname or IP address of the proxy server used on HTTP and HTTPS requests in case `HTTP_PROXY` or `HTTPS_PROXY` aren't defined.

- `NO_PROXY`: a comma-separated list of hostnames that should be excluded from proxying.

PowerShell 7.4 added support for the Brotli compression algorithm.

Related Links

- Invoke-RestMethod
- ConvertFrom-Json
- ConvertTo-Json

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

PowerShell feedback

PowerShell is an open source project. Select a link to provide feedback:

Open a documentation issue

Provide product feedback