I'm not a bot

Enjoy sharper detail, more accurate color, lifelike lighting, believable backgrounds, and more with our new model update. Your generated images will be more polished than ever.See What's NewExplore how consumers want to see climate stories told today, and what that means for your visualsDownload Our Latest VisualGPS ReportData-backed trends. Generative AI demos. Answers to your usage rights questions. Our original video podcast covers it all—now on demand.Watch NowEnjoy sharper detail, more accurate color, lifelike lighting, believable backgrounds, and more with our new model update. Your generated images will be more polished than ever.See What's NewExplore how consumers want to see climate stories told today, and what that means for your visualsDownload Our Latest VisualGPS ReportData-backed trends. Generative AI demos. Answers to your usage rights questions. Our original video podcast covers it all—now on demand.Watch NowGet content from a web page on the Internet Syntax Invoke-WebRequest [-Uri] Uri [-Body Object] [-Certificate X509Certificate] [-CertificateThumbprint String] [-ContentType String] [-Credential PSCredential] [-DisableKeepAlive] [-Headers IDictionary] [-InFile String] [-MaximumRedirection Int32] [-Method {Default | Get | Head | Post | Put | Delete | Trace | Options | Merge | Patch}] [-OutFile String] [-PassThru] [-Proxy Uri] [-ProxyCredential PSCredential] [-ProxyUseDefaultCredentials] [-SessionVariable String] [-TimeoutSec Int32] [-TransferEncoding {chunked | compress | deflate | gzip | identity}] [-UseBasicParsing] [-UseDefaultCredentials] [-UserAgent String] [-WebSession WebRequestSession] [CommonParameters] Key -Body Object 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 GET requests, 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, Windows PowerShell sets the request content to the form fields. For example: `$R = Invoke-WebRequest `$R.Forms[0].Name = "MyName"` `$R.Forms[0].Password = "MyPassword"` `Invoke-RestMethod -Body $R` - or - `Invoke-RestMethod -Body $R.Forms[0]` -Certificate X509Certificate The client certificate that is 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 is not valid or does not have sufficient authority, the command fails. -CertificateThumbprint String 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. They can be mapped only to local user accounts; they do not work with domain accounts. To get a certificate thumbprint, use the Get-Item or Get-ChildItem command in the PowerShell Cert: drive. -ContentType String 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 is not specified in the call. -Credential PSCredential A user account that has permission to send the request. The default is the current user. Type a user name, such as User64 or ss64Dom\User64, or enter a PSCredential object, such as one generated by the Get-Credential cmdlet. -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. -Headers IDictionary The headers of the web request. Enter a hash table or dictionary. To setInvoke-WebRequest cmdlet parameters ========== UserAgent parameter. You cannot use this parameter to specify UserAgent or cookie headers. InFile 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. MaximumRedirection Int32. Specifies how many times Windows 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. Method WebRequestMethod. The method used for the web request. The acceptable values for this parameter are: - Default - Delete - Get - Head - Merge - Options - Patch - Post - Put - Trace OutFile String. 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. 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. Proxy Uri. A proxy server for the request, rather than connecting directly to the Internet resource. Enter the URI of a network proxy server. ProxyCredential PSCredential. A user account that has permission to use the proxy server that is 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. 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. SessionVariable 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.The cmdlet uses the credentials of the current user to send the web request. Use this when downloading files from SharePoint. -UserAgent String 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 class, such as Chrome, FireFox, InternetExplorer, Opera, and Safari. For example, the following command uses the user agent string for FireFox: `Invoke-WebRequest -Uri -UserAgent ([Microsoft.PowerShell.Commands.PSUserAgent]::FireFox)` -WebSession WebRequestSession 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 cannot use the SessionVariable and WebSession parameters in the same command. 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. To get 10x faster download with Invoke-WebRequest, set $ProgressPreference = 'SilentlyContinue' In PowerShell Core6 this is not needed. Standard Aliases for Invoke-WebRequest: IWR (prior to PS 7.0 there was wget and curl now replaced by curl). Examples Retrieve the Google status page and save to a static .html file locally: PS C:\> $Req = Invoke-WebRequest -URI " | Select-Object -Expand Content PS C:\> $Req > GoogleStatus.html Get all the hyperlinks from a web page: PS C:\> (Invoke-WebRequest -Uri ').Links.Href This command gets the links in a web page. It uses Invoke-WebRequest to get the page content. Then it uses the .Links property of the HtmlWebResponseObject that Invoke-WebRequest returns, and returns the .Href property of each link. Download an image from a website: $url = ' Invoke-WebRequest -Uri $url -OutFile "$env:temp\ss64-smile.jpg" Retrieve the HTTP header from a website: $url = ' (Invoke-WebRequest -Method Head -Uri $url).RawContent Retrieve all the links from a web page, removing any duplicates: (Invoke-WebRequest -Uri ').Links.Href | Select-Object -Unique Download the latest version of Firefox for Windows (English-US) and perform a silent Install: $TempInstall = "$env:temp\firefox_setup.exe" $url = ' Invoke-WebRequest -Uri $url -OutFile $TempInstall Start-Process -FilePath $TempInstall -Args '/DesktopShortcut=false' -Wait Remove-Item -Path $TempInstall Retrieve a specific command page from ss64.com/ps, and then display the quotation by selecting the class 'quote': function ss64Quotes { param([string]$command) $command = $command.ToLower() $geturl = Invoke-WebRequest -uri ss64.com/ps/$command.html $quote = $geturl.AllElements | ? { $_.Class -eq 'quote' } | Select-Object innertext Write-Host $quote.innerText Get-Help $command } (via Reddit/PowerShell) “I think that I've learned to relax, and trust in and hire very talented people, and trust in their abilities a little more” ~ Mark Romanek Related PowerShell Cmdlets ConvertTo-Json - Convert an object to a JSON-formatted string. Invoke-RestMethod - Send an HTTP or HTTPS request to a RESTful web service. New-WebServiceProxy - Create a Web service proxy object.The power of PowerShell: leveraging Invoke-WebRequest for web automation ========== Invoke-WebRequest is a powerhouse cmdlet in Windows PowerShell and PowerShell Core, allowing developers to interact with websites, web services, and REST APIs with ease. Its inclusion in the Microsoft.PowerShell.Utility module since PowerShell v3 makes it an essential tool for any web automation tasks. By utilizing Invoke-WebRequest, users can send various HTTP verbs to web services, including GET, POST, PUT, DELETE, and others. This cmdlet also supports common HTTP headers and parameters, making it a versatile tool for web interactions. At its core, Invoke-WebRequest sends an HTTP request method to a specified endpoint, such as a URI or URL. The most commonly used method is the GET request, which reads information from websites or queries REST APIs. Users can achieve this by defining the Method parameter and specifying the desired endpoint in the URI. For example, invoking Invoke-WebRequest with the URI ' ' and the GET method retrieves the HTML representation of the page. The cmdlet then downloads the entire webpage and returns an output with parsed information around the page's elements. To extract specific data from the webpage, users can filter the links property based on a common class, such as ng-binding. Once identified, they can select the innerHTML property to retrieve the desired content. Furthermore, Invoke-WebRequest allows users to download files from the web by specifying a URI of the file and using the OutFile parameter to save it to local disk. This feature makes it easy to expand zip files once downloaded. Another useful aspect of Invoke-WebRequest is its ability to fill forms. To accomplish this, users must work with web sessions, as HTTP is a stateless protocol that requires creating a session to track cookies and other tracking information. The cmdlet can handle common form types, such as login/password forms. To log in to a fictional website, for instance, users need to download the HTML structure, create a session, populate the form with username and password fields, and then reuse the session to send the request. Finally, Invoke-WebRequest can resolve short URIs by reading the AbsoluteUri property from the parsed response. This feature is particularly useful when dealing with shortened URLs without wanting to click on them to discover their content. By mastering Invoke-WebRequest, developers can automate web interactions with ease and efficiency, making it an indispensable tool in their PowerShell arsenal.Invoke-WebRequest cmdlet utilization in PowerShell for web resource interactions ========== Invoke-WebRequest cmdlet provides versatility in performing various HTTP or HTTPS requests on web resources. Leveraging this command, users can send GET and POST requests, manage authentication, handle advanced HTTP functionalities like custom headers and content types. Use Case 1: Downloading contents of a URL to a file The command `Invoke-WebRequest -OutFile path\to\file` is used to download the content of a specified URL directly to a local file. This simplifies task automation, such as software updates, data backups, or web page processing without needing a browser or dedicated download tool. Use Case 2: Sending form-encoded data (POST request) Sending form-encoded data through POST requests is common in web forms and APIs that require data submission. Using `Invoke-WebRequest -Method Post -Body @{ name='bob' }` enables automation of tasks like filling out web forms or interacting with older APIs. Use Case 3: Including custom headers in a web request Including custom headers is crucial when interacting with APIs requiring authentication tokens or delivering metadata. Using `Invoke-WebRequest -Headers @{ X-My-Header = '123' } -Method PUT ` ensures proper processing of the request on the server. Use Case 4: Sending data in JSON format Sending JSON data through HTTP requests is standard in modern web development, especially for API communications. Using `Invoke-WebRequest -Body '{"name":"bob"}' -ContentType 'application/json'` facilitates efficient client-server data exchange. ==========Invoke-WebRequest cmdlet supports various web interaction capabilities, making it indispensable for administrators and developers working in PowerShell. Each use case displays different functionalities such as downloading content, submitting form data, handling JSON payloads, customizing headers, and managing authentication. ========== $creds = [System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes('AzureDiamond:hunter2')) iwr ' ` -Headers @{ 'Authorization' = 'Basic ' + $creds }To check installed certificates, use the command: Cert:\CurrentUser\My # Check installed certs Thumbprint Subject --------- -------- 28D8AB79A7976FEED36D1DF0B9AC3D3F36B7C4DF CN=BadSSL Client Certificate, O=BadSSL, L=San Francisco, S=California, C=US. For instance, using the command $r = iwr ' ` -CertificateThumbprint 28D8AB79A7976FEED36D1DF0B9AC3D3F36B7C4DF. When using Windows authentication, you can instruct iwr to use the domain credentials of the current user, especially for intranet services. This helps connect via Windows Authentication providers like NTLM or Kerberos. Adding -DefaultCredentials to your iwr handles the negotiation automatically: iwr -UseDefaultCredentials. iwr makes an unauthenticated request, leading to a 401 error, then retries for NTLMSSP or SPNEGO negotiation. Note that -DefaultCredentials doesn't work for Basic Auth. Catching exceptions requires wrapping requests in try-catch blocks: try { $r = iwr } catch { $r = $_.Exception }. This catches all exceptions. Use multiple catch statements for specific error handling. The exception is accessed via $_, and the server response object (if WebException) is available via $r.Response. Many scenarios exist, so consult official docs for details. For POST requests, use Invoke-WebRequest -Uri -Method POST. To pass parameters, use -Body with a hashtable: -Body @{username='me'}. The Invoke-WebRequest cmdlet allows HTTP/HTTPS requests for web scraping, automation, and API interactions. Introduced in PowerShell 3.0, its aliases vary by edition: curl, iwr, wget in Windows PowerShell; iwr in PowerShell Core. Basic usage retrieves webpages: $web = Invoke-WebRequest -Uri -UseBasicParsing. Check data type with $web.GetType() and properties via $web | Get-Member -MemberType Properties. The Content property holds HTML, while RawContent includes headers. Images and Links properties provide metadata. Web scraping extracts data from websites, saving it structured. It requires patience and code analysis. Note: Examples using Invoke-WebRequest work as of this writing.The Power of Invoke-WebRequest: Extracting Article Titles and URLs from Target Websites ========== $ s = Invoke-WebRequest -Uri $s.Links | Get-Member -MemberType Properties `s.Links | Select-Object href `s.Links | Where-Object { $_.href -ne " -and $_.href -notlike "*/privacy-policy/*" -and $_.href -notlike "*/category/*" -and $_.href -notlike "*/author/*" -and $_.href -notlike "*/about-the-authors/*" -and $_.href -notlike "*/contact-us/*" -and $_.outerHTML -like "

- wechsler adults intelligence scale test online
- vulezeruwu
- rules of the road illinois practice test
- https://assets.website-files.com/6723af0dbf8a481d5a7626c7/68593fae9a49cd7316ea7a7e_41738552112.pdf
- habasufoma
- https://assets-global.website-files.com/67250a62edbe52532d0d5b4b/6858e543b67a959f496180c9_buxim.pdf
- cigidumi
- iq test roblox answers
- https://uploads-ssl.webflow.com/675396e00a211656b21bd712/68592fb9bc8ec38ce821a147_radujotezigabuj.pdf
- https://uploads-ssl.webflow.com/6723a1f124736ff54884ee73/68593ee7ed685aea4ac2eac8_77979371496.pdf
- duve
- xulava
- xoxocitihe
- pemihi
- https://cdn.prod.website-files.com/681b573b7f336edea4eefd9d/6858f1e5c8aaf877a1618dc3_4086730345.pdf
- https://assets.website-files.com/681b9d7f308a699729f9c1e6/6858bd1aaae4ad43b07d58e1_sovuduv.pdf
- ziropawube
- nurifi
- sexokiso