

HTTP request



HTTP request is a default Switch app that for each incoming job executes an HTTP or HTTPS request. Thus the incoming job is a request trigger and it can be used to define the request specification: URL, authentication data, request parameters, file to upload, etc. The protocol type (HTTP or HTTPS) is automatically detected from the URL: if the URL starts with “https://”, HTTPS is used, otherwise HTTP is used.

The server responds to the request by sending something back: an HTML source, JSON or XML data, or a file that is downloaded. The tool provides properties to manipulate this server response. The server response is always saved as a file and the tool can inject this file into the flow as a new job, attach it as metadata dataset to the incoming job or assemble it into a job folder together with the incoming job.



Important: Please be aware that in version 18 of this app automatic redirects are not supported!

Keywords

If you enter one of the following keywords in the Search field at the top of the Flow Elements pane, the **HTTP request** element will be shown in the list:

- HTTP
- HTTPS
- POST
- PUT
- GET
- HEAD
- DELETE
- download
- upload
- request


Connections

HTTP request supports outgoing traffic light connections:

- The 'Data' connections transfer the server response file as well as the input job (depending on the property values specified for the flow element).
- The 'Log' connections transfer a generated text log file containing the request URL, the finished status, the status code, the status description, the last error and other relevant information. The log file name is generated from the input job file name by replacing the input job extension with 'log'.

- The 'Success out' connections are used in case the request finished status is 'Ok' and the HTTP request status code is in the range 100-299.
- The 'Error out' connections are used in case the request finished status is not 'Ok' or the HTTP request status code is in the range 300-599. The input job may fail in case of some internal tool error or invalid tool settings.


Properties

Property	Description
Element type	The flow element type: HTTP request. This property is useful to identify renamed flow elements. It cannot be changed.
Name	The name of the flow element displayed in the canvas.
Description	A description of the flow element displayed in the canvas. This description is also shown in the tooltip that appears when moving your cursor over the flow element.
URL	<p>The URL to fetch. The URL string must be URI-encoded (in a URI-encoded string a space is shown as %20).</p> <p>The tool detects the protocol to use for the request automatically from the URL: if the URL starts with 'https://', the tool will use HTTPS, otherwise HTTP will be used.</p> <p>Example 1 (entered using the 'Single-line text with variables' editor); assumes the job name can contain only ASCII alphabetic characters and digits: <code>https://api-content.dropbox.com/1/files_put/auto/[Job.Name]</code></p> <p>Example 2 (entered using the 'Script expression' editor): <code>HTTP.encodeURI("https://api-content.dropbox.com/1/files_put/auto/" + job.getName());</code></p>
Request type	<p>The type of the request.</p> <p>The supported request types are:</p> <ul style="list-style-type: none"> • HEAD (used for retrieving response headers) • GET (used for downloading data) • PUT (used for uploading data) • PATCH (used for modifying data) • POST (used for uploading with extra parameters) • POST a body (used for uploading, with the option to define the source of the content of the body of the POST request). • DELETE (used for removing resources on external services) <p> Note: The value for the "Content-Type" header in the request is generated automatically depending on the request type and other settings like MIME</p>

Property	Description
	<p>encoding and defined parameters. In the HTTP request tool, it is not possible to define the custom header value. However in some cases it is possible by writing a script that uses the HTTP class from the Switch scripting API. More details about the automatically generated value for the "Content-Type" header and the ways to specify a custom value for it, can be found in Switch scripting reference.</p>
<i>Attached file</i>	<p>This property is available only if Request type is <i>POST</i> or <i>PUT</i>.</p> <p>A file to append to request if the POST or PUT requests are used. The file will be uploaded to the server.</p>
<i>Use MIME encoding</i>	<p>This property is available only if Request type is <i>POST</i>.</p> <p>To use MIME encoding, choose Yes; otherwise choose No (default).</p>
<i>File variable</i>	<p>This property is available only if Request type is <i>POST</i> and Use MIME encoding is Yes.</p> <p>The name of the HTTP form data variable that is used by the receiving HTTP server to identify the correct file part of the uploaded MIME package.</p>
<i>Body content</i>	<p>This property is available only if Request type is <i>POST</i> <i>a body</i>.</p> <p>The content of the body of the POST request.</p> <p>Options are:</p> <ul style="list-style-type: none"> • <i>Input job</i>: The body of the POST will be the contents of the input job. • <i>Dataset</i>: The body of the POST will be the contents of the dataset. The dataset name has to be specified in the subordinate property. • <i>Custom</i>: The body of the POST will be the contents of the text defined in the subordinate property. You can generate a multi-line text with variables to structure a JSON file, an XML, or a similar format that is needed by the server for the POST request body.
Authentication scheme	<p>The authentication scheme to use when server authorization is required:</p> <ul style="list-style-type: none"> • <i>None</i> (default): No authentication is performed. • <i>Basic</i>: Basic authentication based on a user name and password.

Property	Description
	<ul style="list-style-type: none"> • <i>Digest</i>: Digest authentication (requires a user name and password). • <i>NTLM</i>: NTLM authentication (requires a user name and password). • <i>OAuth</i>: OAuth ("Open Authorization") authentication. The authorization string must be supplied through the Authorization property. • <i>OAuth2.0</i>: OAuth 2.0 ("Open Authorization 2.0") authentication. The authorization string must be supplied through the OAuth2.0 authorization property.
<i>User name</i>	<p>This property is available only if Authentication scheme is <i>Basic</i>, <i>Digest</i> or <i>NTLM</i>.</p> <p>A user name if authentication is to be used.</p>
<i>Password</i>	<p>This property is available only if Authentication scheme is <i>Basic</i>, <i>Digest</i> or <i>NTLM</i>.</p> <p>A password if authentication is to be used.</p>
<i>Authorization</i>	<p>This property is available only if Authentication scheme is <i>OAuth</i>.</p> <p>The authorization string to be sent to the server.</p>
<i>OAuth2.0 Authorization</i>	<p>This property is available only if Authentication scheme is <i>OAuth2.0</i>.</p> <p>The authorization string to be sent to the server. Click the three dots to open the dialog to enter the parameters necessary to initiate the OAuth2.0 authorization:</p> <ul style="list-style-type: none"> • Application ID • Application password • Redirection port (or select the 'Automatic' checkbox) • Authorization URL • Token URL • Scope
Parameters	<p>The parameters to attach to the request. Each parameter should be specified in a separate line by the string 'key=value' (without quotes).</p> <p>The parameters are URI-encoded automatically. For the POST and PUT requests, the parameters are included in the HTTP request after the HTTP headers. For the HEAD and GET requests, the parameters are appended to the URL after the question mark ('?').</p> <p>Example:</p>

Property	Description												
	<p>root=auto</p> <p>path=[Job.JobState]</p>												
Headers	<p>The headers to attach to the request. Each header should be specified on a separate line by the string key:value.</p> <p>Example: Content-Type : text/plain</p>												
Response	<p>The response received from the server is always saved to a file. This property defines how to handle this file:</p> <ul style="list-style-type: none"> • <i>Inject as new job</i>: the response is sent to the output data connections as a new job. • <i>Attach as dataset</i>: the response is attached to the input job as an opaque dataset with the name specified in the 'Dataset name' property ('HTTPResponse' by default) and then the input job is sent to the output data connections. • <i>Assemble in jobfolder</i>: the response is assembled into a new jobfolder together with the input job and the jobfolder is sent to the output data connections. The jobfolder name is the same as the name of the input job (without the extension). • <i>Discard</i>: the response is discarded. In this case the input job is sent to the output data connections. 												
<i>Dataset name</i>	<p>This property is a subordinate property of Response if it is set to <i>Attach as dataset</i>; it allows you to specify a custom name for the dataset.</p>												
<i>Dataset model</i>	<p>This property subordinate property of Response if it is set to <i>Attach as dataset</i>; it allows you to define the dataset model of the response. Options are:</p> <ul style="list-style-type: none"> • <i>Automatic</i> (default). In this case, the dataset model is determined by the Content-type key in the header. <table border="1" data-bbox="774 1523 1436 1814"> <thead> <tr> <th>Content-type key</th> <th>datamodel</th> </tr> </thead> <tbody> <tr> <td>application/json</td> <td>JSON</td> </tr> <tr> <td>application/xml</td> <td>XML</td> </tr> <tr> <td>application/vnd.cip4-jdf+xml</td> <td>JDF</td> </tr> <tr> <td>application/vnd.cip4-jmf+xml</td> <td>JDF</td> </tr> <tr> <td>anything else</td> <td>Opaque</td> </tr> </tbody> </table> <ul style="list-style-type: none"> • <i>Opaque</i> • <i>JSON</i> 	Content-type key	datamodel	application/json	JSON	application/xml	XML	application/vnd.cip4-jdf+xml	JDF	application/vnd.cip4-jmf+xml	JDF	anything else	Opaque
Content-type key	datamodel												
application/json	JSON												
application/xml	XML												
application/vnd.cip4-jdf+xml	JDF												
application/vnd.cip4-jmf+xml	JDF												
anything else	Opaque												

Property	Description
	<ul style="list-style-type: none"> • <i>XML</i> • <i>JDF</i>
<i>File Name</i>	<p>This property is available only if Response is <i>Inject as new job</i> or <i>Assemble in jobfolder</i>.</p> <p>The file name (with an extension) to use for the response file.</p> <p><i>Automatic:</i> The file name is taken from the Content-Disposition header of the server response. If the Content-Disposition header is missing or does not contain a valid file name, the extension is taken from the response Content-type and the file name will become [JobNameProper].extension.</p>
<i>Input job</i>	<p>This property is available only if Response is <i>Inject as new job</i>.</p> <p>Defines what to do with the input job in case the response file is injected as a new job:</p> <ul style="list-style-type: none"> • <i>Send to output:</i> the input job is sent to the output data connections as a separate job. In other words, the input job and the response file are now two separate jobs. • <i>Attach as dataset:</i> if the input job is a file, it is attached as an opaque dataset with the name specified in the 'Dataset name' property ('HTTPInput' by default) to the new job representing the response file. If the input job is a folder then it is discarded and an error message is logged. <hr/> <p> Note: If "Attach as dataset" is chosen, you'll get two extra subordinate properties: <i>Dataset name</i>, allowing you to specify a custom name for the dataset, and <i>Dataset model</i>, allowing you to specify the dataset model of the response.</p> <hr/> <ul style="list-style-type: none"> • <i>Discard:</i> the input job is discarded.
Response headers	<p>This property defines how to handle the response headers:</p> <ul style="list-style-type: none"> • <i>Attach as dataset:</i> The response headers are saved to a text file (each header on a separate line in format 'key:value') and the file is attached to the output job as an opaque dataset with the name specified in the 'Dataset name' property ('HTTPResponseHeaders' by default) and the dataset model specified in the 'Dataset model' property (JSON (default) or XML). • <i>Discard:</i> The response headers are discarded.

Property	Description
<i>Dataset name</i>	This property is a subordinate property of Response headers if it is set to <i>Attach as dataset</i> ; it allows to specify a custom name for the dataset.
<i>Dataset model</i>	This property is a subordinate property of Response headers if it is set to <i>Attach as dataset</i> ; it defines the response headers dataset model with options JSON(default) and XML.
Retry after failed connection	If enabled, Switch will try again to process jobs that failed due to a network connection failure. This way, you don't have to restart your flow manually.
<i>Retry count</i>	This property is available only if Retry after failed connection is set to Yes. The number of retry attempts in case of a network connection error. The default value is 0 (= unlimited retries), meaning that Switch will keep on trying.
<i>Retry delay</i>	This property is available only if Retry after failed connection is set to Yes. The delay (in minutes) between each retry attempt. The default value is 10.
Ignore server certificate errors	If set to Yes, server certificate errors will be ignored when connecting to an HTTPS server. Examples of certificate errors: <ul style="list-style-type: none">• Self-signed certificate• Invalid host name• Expired certificate