Webhook Connector

Download the PDF of this article.

In this Article

Related Articles

Overview

FormAssembly's Webhook Connector is a powerful tool that allows you to send data from your forms to any web service in real-time. The flexibility it offers can be transformative for your workflows, enabling seamless data transfers and automation that enhance the efficiency of your processes.

Send workflow response data directly to any endpoint you control access to. Customize your payloads to meet the needs of your endpoints using a user-friendly UI and an open-ended JSON editor. This connector supports the following authentication types: API key, basic (username and password), and OAuth 2.

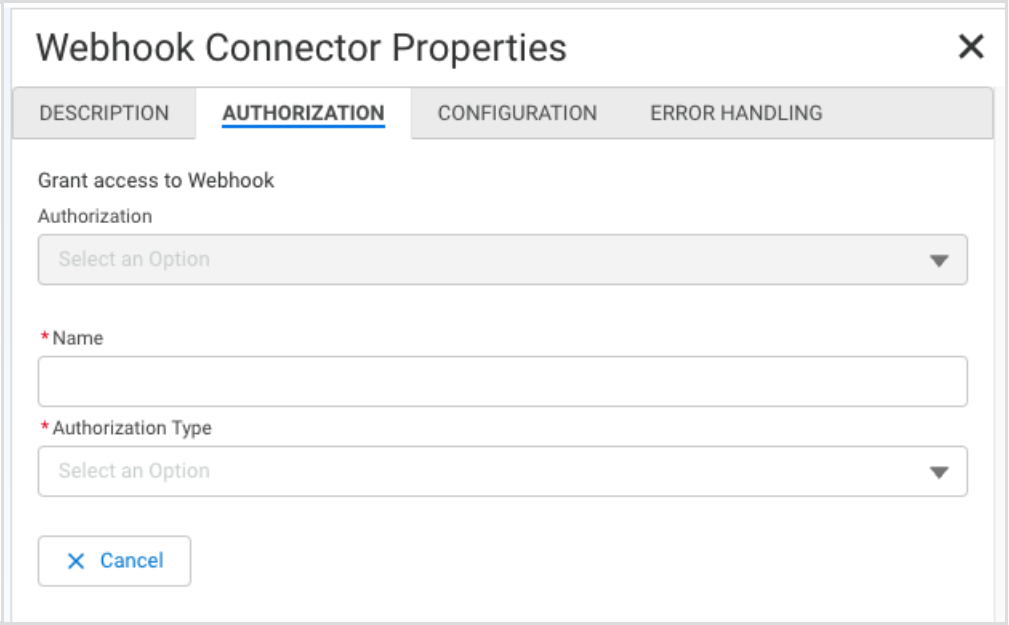
Connector Configuration

Refer to our Workflow-native Connectors article for general configuration information.

Authorization Tab

- Create a New Authorization
- Edit an Existing Authorization
- Manage or Delete Existing Authorizations

The **Authorization** dropdown lists all saved Webhook Connector authorizations from all workflows owned by your user account. Select an existing authorization from the list or scroll to the bottom to create a **New Authorization**.



Create a New Authorization

- Select **New Authorization** from the Authorization dropdown list
- Enter a **Name** for the authorization
- Select the **Authorization Type** required by the destination endpoint the connector is being configured for. When the Authorization Type is selected, fields required for the selected authorization type will display

OAuth Authorization Steps

- Enter the details for each required field
    - **Client ID**: The client identifier issued during the application registration process
    - **Client Secret**: The client secret issued during the application registration process
    - **Authorization URL**: The URL of the authorization server
    - **Access Token URL**: The URL to retrieve the access token
    - **Scope**: The scope of the access request. Separate multiple values with a space.
    - **(optional) Refresh Token URL**: The URL to retrieve the refresh token
- A **Redirect URL** is provided for you to input into your Oauth service's settings.
- Click **Authorize** to view the Oauth verification screen of your Oauth service and complete any additional steps required

Note: If the verification screen does not display properly, verify your Authorization URL and try again, or contact your Oauth system administrator.

- Click **Save** to store and apply this authorization to the Webhook Connector

Basic Authorization Steps

- Enter the details for each required field
    - **Username**: The username required by the endpoint
    - **Password**: The password required by the endpoint
        - **Note**: This password value is masked in the UI for security
- Click **Save** to store and apply this authorization to the Webhook Connector

API Key Authorization Steps

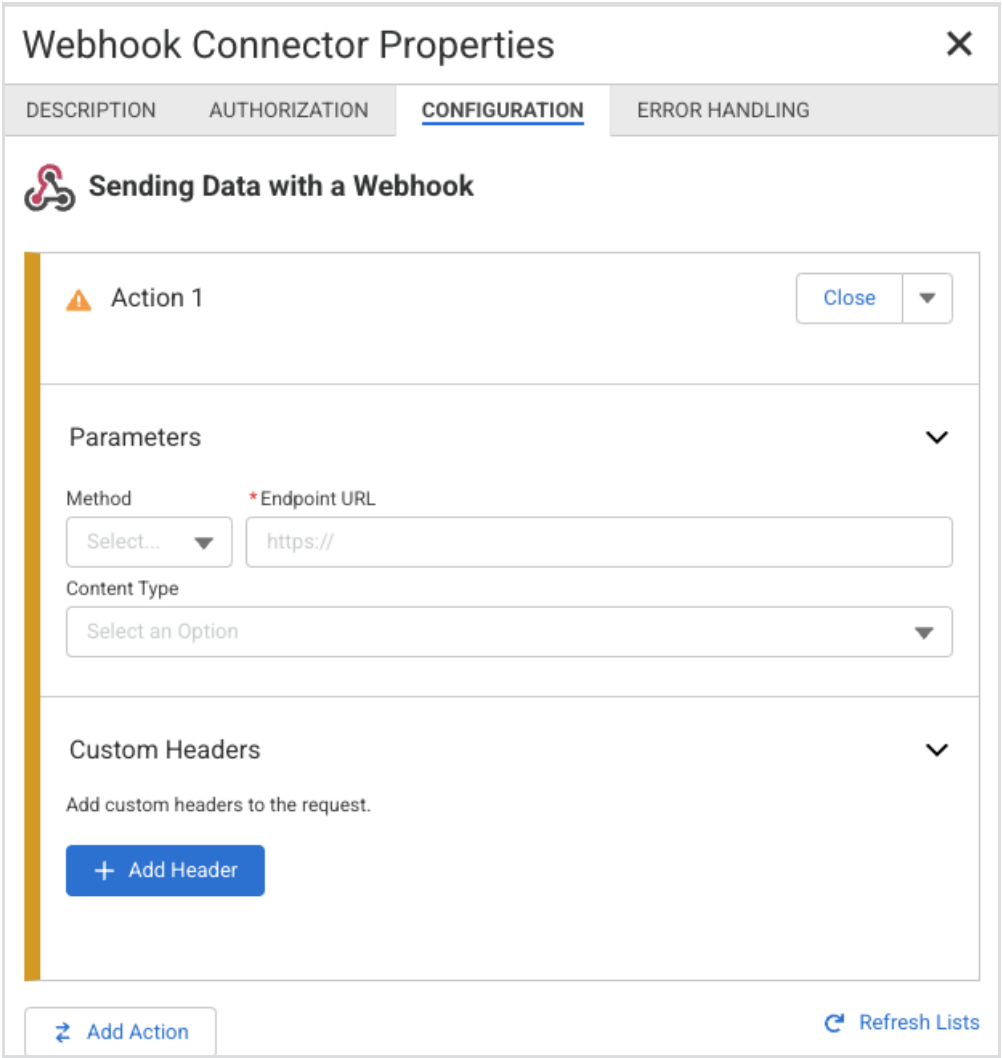
- Enter the details for each required field
    - **API Key Name**: The API Key Name, to use as a header or query parameter
    - **API Key Value**: The API Key or Token required by the endpoint

- Select an option from the **Add to** dropdown list that is appropriate for your endpoint
  - **Header**: Pass the API Key in the header
  - **Query**: Pass the API Key as a query parameter
- Click **Save** to store and apply this authorization to the Webhook Connector

Configuration Tab

An **Action** is used to set the Parameters, Custom Headers, and Body of your request to your endpoint. The connector may be configured to send multiple requests or send data to more than one destination by using multiple Actions.

Note: All Actions will use the same authorization information entered in the Authorization tab. If different authorizations need to be used, additional Webhook Connectors may be added to the workflow.



Parameters

Define the parameters for your Webhook Connector.

- **Endpoint URL**: This is where your form data will be sent. The Endpoint URL field should represent the full URL and path of the endpoint receiving the Webhook Connector request.
- **Method**: Choose from POST, PATCH, PUT, or GET, depending on the nature of your web service interaction.

- **POST**: Send data to the server to create or update a resource; ideal for submitting forms or creating new records.
- **PATCH**: Partially update a resource on the server; use when only specific fields need to be changed.
- **PUT**: Replace or update an entire resource on the server; use when you need to completely overwrite existing data.
- **GET**: Retrieve data from the server; typically used for reading information without making changes. In this case, it will return whether the connector successfully interacted with the integration or not. If the GET request encounters an error, the service is unavailable, and the workflow can use an alternate path to avoid relying on a service that is currently down.

> **Note**: FormAssembly does not support retrieval of data coming back from GET requests from the Webhook Connector at this time.

Custom Headers

Additional metadata needed for your request can be added to the Customer Headers section. Choose to add fields, values, or aliases as part of the request headers. This optional section is not required to configure or run the Webhook Connector.

- Click **Add Header** to add a new custom header
- Click the **Source** dropdown (the file icon) and select "**Form Field**" or "**Value or Formula**"
  - **Form Field**: Use the list to search for and select a form field from the forms included in the workflow.
  - **Value or Formula**: Enter a value as text or open the formula editor to use the formula builder to add workflow aliases.
- **Header Name**: Enter a name for the header.

> **Note**: If a value has been added to the Source field, the Header Name field requires a value, or the configuration will display as incomplete.

Body

The Body section determines how data is structured and sent to the endpoint. Select the content type (Form Data, URL-Encoded, JSON, or Raw) that best meets the data format needs of your endpoint. This section is required for the Webhook Connector to be fully configured unless you are using a GET method in the Parameters section.

- Select a **Content Type** from the dropdown list
  - **URL Encoded**: Send data in key-value pairs encoded within the URL
  - **Form Data**: Send data in key-value pairs
  - **JSON**: Send a JSON object
  - **Raw**: Send raw text

URL Encoded and Form Data Steps

- Select **Add Another** to add a line item to the body of the request
  - Selecting "**Add all standard fields**" will automatically add all form fields in the order that they appear on the form to the body section. Note that this option leaves out file upload fields.
- Click the **Source** dropdown (the file icon) and select "**Form Field**" or "**Value or Formula**"
  - **Form Field**: Use the list to search for and select a form field from the forms included in the workflow.
    - **Note**: While files uploaded via file upload fields cannot be selected and submitted to the endpoint, the file name, file size, mime type, path, URL, and file content can be sent and are selectable as Form Fields
  - **Value or Formula**: Enter a value as text or open the formula editor to use the formula builder to add

workflow aliases.

- **Field Name**: Enter a name for the Field.

JSON Steps

When JSON is selected, you are presented with a JSON editor where you can manually enter and configure JSON objects that refer to form data. This is a flexible and customizable option that allows you to craft a JSON object that matches the data model that the endpoint system expects to receive.

- Enter text manually into the JSON editor
- Use the bar above the text editor to add FormAssembly workflow values.
    - Click the **Source** dropdown (the file icon) and select **Form Field** or **Repeatable Element**.
        - When **Form Field** is selected, click into the search bar and search for form fields and form aliases to add to the JSON object.
            - **Note**: Items added in this method should be wrapped by quotation marks to adhere to JSON syntax requirements.
        - When **Repeatable Elements** is selected, view a list of repeatable elements (fields and sections) in workflow forms to add repeating sections to the JSON object. Clicking on a repeatable element adds a BEGIN and END repeatable element tag to the JSON editor which allows you to identify exactly which data should be sent on each repeating element – data will be sent between the BEGIN and END tags for each repeated item.

Note:

- Ensure JSON syntax is correct; otherwise, the connector may encounter an error.
- Certain workflow aliases may break JSON formatting and will be unable to be parsed if JSON is expected. Aliases such as Response Text and Response HTML may be unappeasable when inserted into a JSON object.

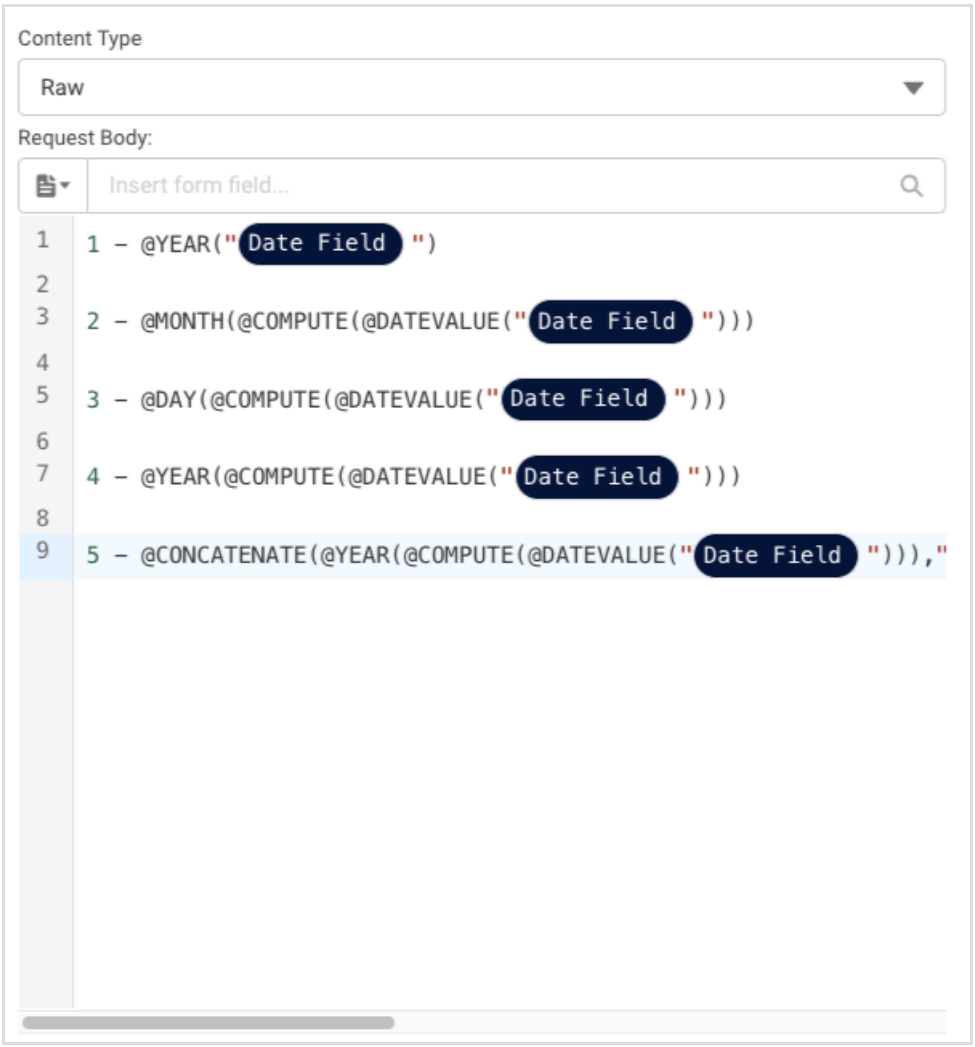
Raw Steps

Raw, similar to JSON, uses a text editor but sends raw text rather than a JSON object. Form fields, values, and aliases can be added to the Raw text editor in the same method as the JSON editor. One distinction, though, is that Raw can support aliases that would break JSON syntax, such as Response Text and Response HTML.

Note: When using calculated formulas inside Raw content bodies, wrap form fields with quotation marks to ensure values are interpreted correctly. This can be especially important for date values.
Example: To return the year of a date input, you could add the following calculated formula:

```
@YEAR("%%form.api._step_1:tfa_1%%")
```

Note the use of quotation marks wrapping the form field value.



Error Logs

Logs for this connector indicate if something is a “client error” (400-level response) or a “server error” (500-level response). It states the request method and URL endpoint, as well as the status code and status code description. Errors may be further investigated on your endpoint system.

Example Error Log:

```
Connector error: Client error: `POST https://example-url.events/abc123` resulted in a `401 Unauthorized` response
```

Error Log Breakdown:

```
Connector error: [Error Type (Client/Server)]: '[Method] [Endoint URL]' resulted in a '[Status Code] [Status Code Description]' response
```