Make HTTP Request with Callback action

Documentation

The **Make HTTP Request with Callback** action in Zenphi allows you to make calls to external APIs and other servers.

The action performs an HTTP request to a specified endpoint and *immediately continues execution* as soon as a response is received. Use this action when you need to make a call and handle the response *without waiting* for further external triggers.

The 'HTTP Action with Callback' in Zenphi allows your workflows to:

- **Pause and Wait**: After making a call to an external system's API, the workflow will pause.
- **Resume on Response**: It will then resume based on the response received via a provided callback URL.
- **Callback URL**: The action itself provides this callback URL. You can access it via the "x-zenphi-callback-url" header.
- **Supported Methods**: The callback API supports both GET and POST methods.
- **Payload Support**: It can handle body contents, files or forms in the callback. The maximum payload size is 30MB.

Here's how it works:

Connectivity: It enables interaction with systems beyond Zenphi's built-in actions and can even call other Zenphi flows.
HTTP Methods: It supports various HTTP request types such as GET (retrieve data), POST (create new data), PUT (replace existing data), PATCH (update data), and DELETE (remove data).
Configuration: You need to provide the URL of the API you want to access. You can also specify headers to send with the request.
Authentication: It offers several authentication methods, including 'no auth', 'API Key', 'bearer token', and 'Basic Auth'.
Request Body: You can configure the body of the request with different types like 'none', 'form-data', and 'x-www-form-urlencoded'. For 'form-data', you can define if the input is text or a file.
Certificate Validation: You have the option to ignore server certificate validation for trusted endpoints with self-signed certificates.
Response: The action will retrieve data from the specified API and typically return it as a JSON payload. You can enable an option to receive the API response as a file (document or PDF).

To **configure** the action you have the following tabs available:

- The **Settings** tab is where you configure the action:
  - **API Url** (required): This is the URL of the API endpoint that you want to call.
  - **Authorization** (default: No Auth): Select the type of authorization that your request needs. Often this one is left empty and authorization is instead declared in the Headers field. Your options here are:
    - **No Auth** - No authorization / Leave empty

        - **Basic Auth** - Basic authentication is a simple HTTP authentication scheme where clients provide their credentials (username and password) in a base64-encoded format within the Authorization header of their requests
        - **API Key** - API key authentication uses a unique, randomly generated string (the API key) to identify and authenticate an application or user making requests to an API, providing a simple and popular method for securing API access. You commonly request the key with the API provider.
        - **Bearer token** - A bearer token is a string used in HTTP authentication, typically included in the "Authorization" header, that grants access to protected resources, often used in OAuth 2.0 and other token-based authentication systems.
    - **Headers** (optional): Key-value pairs that provide metadata about an API request or response, such as content type, authorization details, and caching instructions, helping the client and server understand and process the data effectively. Stated as `[collection key:value pairs]`. Add one header item per line.
    - **Http Request Method** (default:get): Also known as HTTP verbs, are a set of instructions that tell a server what action to perform on a resource, such as retrieving, creating, updating, or deleting data. Your options are:
        - **Get** - Requests using GET should only be used to request data and shouldn't contain a body, instead use the API Url `[domain.api/?token=parameter]` . *The semantics of sending a message body in GET requests are undefined. Some servers may reject the request with a 4XX client error response*.
        - **Post** - Used to send the request data to a server to create a new resource or update an existing one.
        - **Put** - Creates a new resource or replaces a representation of the target resource with the request content. *The difference between PUT and POST is that PUT is idempotent: calling it once is no different from calling it several times successively (there are no side effects)*.
        - **Head** - Similar to GET, but it only retrieves the response headers (metadata) of a resource without the actual content (the body). *Useful for checking resource existence, size, modification date, and cached validity without downloading the resource*.
        - **Patch** - the HTTP PATCH method is used to apply partial modifications to an existing resource, unlike PUT which replaces the entire resource.
        - **Delete** - Used to delete a specific resource in a REST API. *It is performed by sending a DELETE request to the URL of the resource to be deleted, and the API will remove the resource if it exists.*
    - **Request body type** (default:text): refers to the data sent from a client to a server. Specify Content Type in the *Request Body Content Type* field. *This is particularly relevant for HTTP methods like POST, PUT, and PATCH, which are used to send data to the server to create or modify resources*. Your options are:
        - **Text** - Default, this is used for sending simple text data.
        - **Binary** - This is used for sending binary data, such as images, videos, or other files.
        - **None** - Essentially means that you are sending an HTTP request without any data in the body of the request.

  - **Form-Data** - A more versatile encoding scheme designed to handle complex form submissions, including file uploads. It divides data into distinct parts, each with its own headers, separated by a defined boundary. This format supports both text and binary data, making it essential for forms with mixed data types and larger payloads. *While more complex than x-www-form-urlencoded, it enables the transfer of files and diverse data within a single HTTP request.*
  - **x-www-form-urlencoded** - a simple encoding scheme for submitting form data. It encodes data as key-value pairs, separated by '=' and '&', with spaces replaced by '+' and special characters percent-encoded. This format is ideal for basic text-based data like login credentials or search queries, and is efficient for small amounts of data. *However, it cannot handle file uploads or binary data, limiting its use to simple text inputs.*

- **Request Body** (required): The body contains the data being sent to the API server. Its content varies based on the API's requirements and the "Content-Type" header, such as Form-Data or plain text. Essentially, it's the payload of information sent to the server for processing.
- **Request Body Content Type** (required): Specific content type of the request. ex. [`application/json`], complements the *Request body type* selection field.
- **Omit charset from content-type header** (optional, default:off): If you turn it on, the charset will be removed from the content-type header.
- **Ignore Server Certificate Validation** (optional, default:off): Turning on this option will disable all certificate validations. **Only turn on this option if you trust the destination server**.

The **Output** from the action will present the following information:

- Request Headers
    - Access-Control-Allow-Origin - Indicates whether the response can be shared with requesting code from the given origin
    - Allow - Lists the set of methods supported by the resource
    - Cache-Control - Directives for caching mechanisms in requests and responses
    - Content-Disposition - Indicates if the content is expected to be displayed inline in the browser or as an attachment
    - Content-Encoding - The type of encoding used on the data
    - Content-Language - Describes the language(s) intended for the audience
    - Content-Length - Indicates the length of the response body in bytes
    - Content-Range - Where in a full body message a partial message belongs
    - Content-Security-Policy- Used to specify browser security policies
    - Content-Type - Specifies the media type of the response body
    - Date - The date and time at which the message was originated
    - ETag - Entity tag for the requested resource
    - Expires - The date/time after which the response is considered stale
    - Last-Modified - The date and time the resource was last modified

  - Link - Used to express a typed relationship with another resource
  - Location - Used in redirection or when a new resource has been created
  - Pragma - Implementation-specific header that may have various effects
  - Referrer-Policy - Governs which referrer information to include with requests
  - Refresh - Used in redirection, or when a new resource has been created
  - Retry-After - Indicates how long the user agent should wait before making a follow-up request
  - Server - A name for the server
  - Set-Cookie - Used to send a cookie from the server to the user agent
  - Strict-Transport-Security - A security header to enforce the use of HTTPS
  - Transfer-Encoding- The form of encoding used to safely transfer the entity to the user
  - Upgrade-Insecure-Requests - Asks for the user agent to upgrade to a more secure protocol
  - Vary - Tells that the output is subject to change based on certain request headers
  - WWW-Authenticate - Indicates the authentication method required to access a resource
  - X-Content-Type-Options - Prevents the browser from MIME-sniffing a response away from the declared content-type
  - X-Frame-Options - Indicates whether or not a browser should be allowed to render a page in a <frame>, <iframe>, <embed> or <object>
  - X-XSS-Protection - Enables cross-site scripting filtering
- Response Headers Collection - A collection of HTTP headers associated with the request
  - Header Name - The name of the HTTP header, e.g., Content-Type or Authorization.
  - Header Value - The value assigned to the HTTP header, e.g., application/json or Bearer token.
- Query Strings
  - Name - Name of the query parameter.
  - Value - Value of the query parameter.
- Request Body - Text body content of the request. This will be populated if the submitted request body to the resume API is text content.
- Body File - Request body as a file. This will be populated if submitted request body to the resume API is binary.
- Form Data - Request form data. This will be populated if the submitted request to the resume API is form or multipart.
  - Form Parameters
    - Key - Name of the form field.
    - Value - Value of the form field.
  - Form Files
    - Form Field Name - Name of the file field.
    - Form File Name - Name of the uploaded file.
    - Form File Content Type - MIME type of the file.
    - Form File Size - Size of the file in bytes.

        - Form File Content - Reference to the file content submitted in the request form.
- Correlation ID - A unique identifier for the request issued by Zenphi, used for tracking and debugging.
- Duration - The time it took to complete this action
    - In Seconds - The total number of seconds it took to complete this action
    - In Minutes - The total number of minutes it took to complete this action
    - In Hours - The total number of hours it took to complete this action
    - In Days - The total number of days it took to complete this action

Having an action like the Make HTTP Request allows your Zenphi workflows to interact with systems and services beyond the pre-built actions available. A seamless connection with other business tools and platforms, even those not directly provided through Zenphi's standard integrations. If you need to perform a specific operation or integrate with a niche service, the HTTP Request action lets you implement custom logic by interacting directly with APIs. It allows your workflows to send data to external systems and receive data back, enabling more complex and integrated automation scenarios. And it manages situations like working with trusted systems that have their own security certificates

Alternative action*: The 'Make HTTP Request' action allows your workflow to execute immediately after making an API call.*

Mikael Klambro
Egoiste Zenphi Application Consultant