

Building a Cisco Connect Flow

In this lab, you'll learn how to create and configure a flow using Cisco Connect's Flow Builder. The lab will guide you through the process of setting up a webhook, parsing data from an API request, evaluating and processing that data, and finally making an outbound HTTP request based on the result of the evaluation. By the end of this lab, you'll have a clear understanding of how to create a flow from scratch, configure its components, and understand the interactions between these components. This will be a hands-on experience in integrating webhooks, parsing data, executing conditional logic, and communicating with external services through HTTP requests.

Activity Objective

In this exercise, you will create a Cisco Connect flow that is triggered via a request to a webhook.

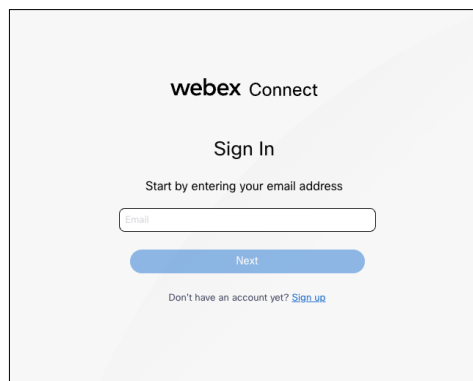
- Create a Cisco Connect service and flow
- Configure a Cisco Connect flow
- Test Cisco Connect flow functionality by making an API request

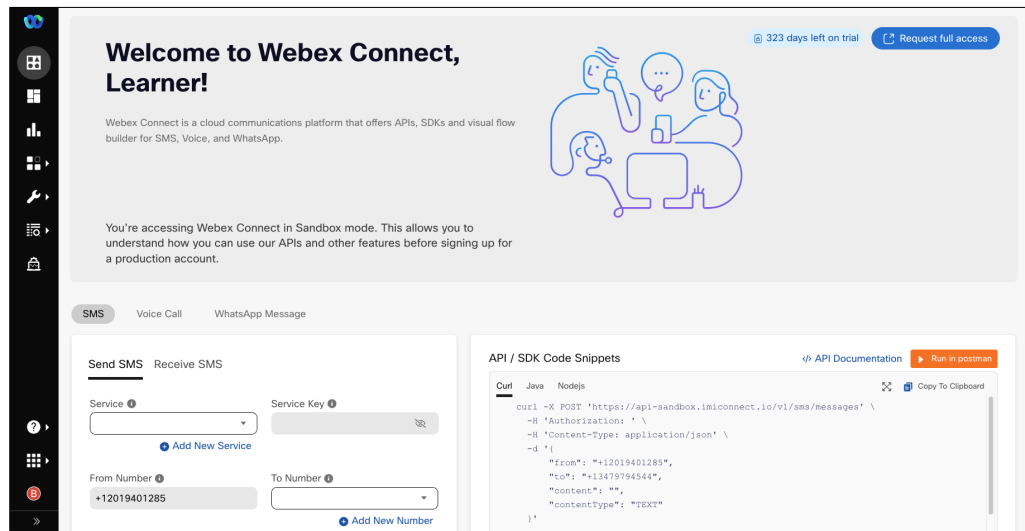
Task 1: (task1)

Activity Procedure

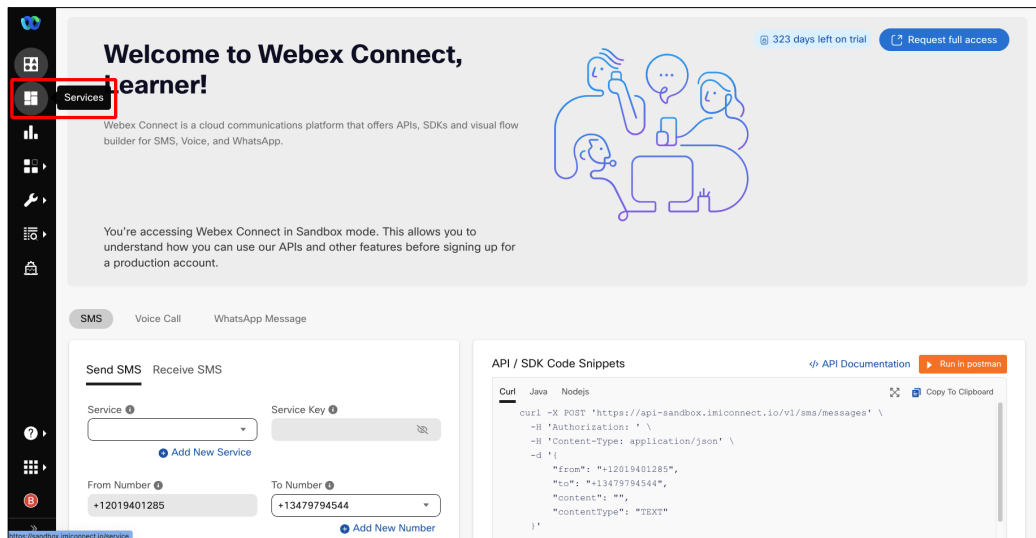
Step 1: Navigate to <https://sandbox.imiconnect.io/> and log in with your Cisco ID.

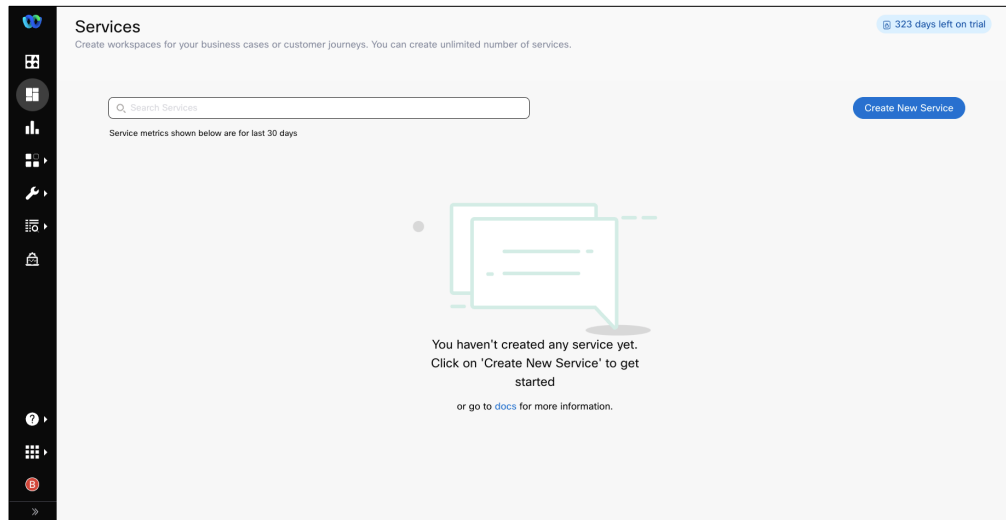
Note: If this fails, contact your administrator to ensure your account has access to the developer sandbox. Alternatively, if you are allowed, you can head over to <https://help.imiconnect.io/docs/getting-started-with-sandbox> and sign up for free.



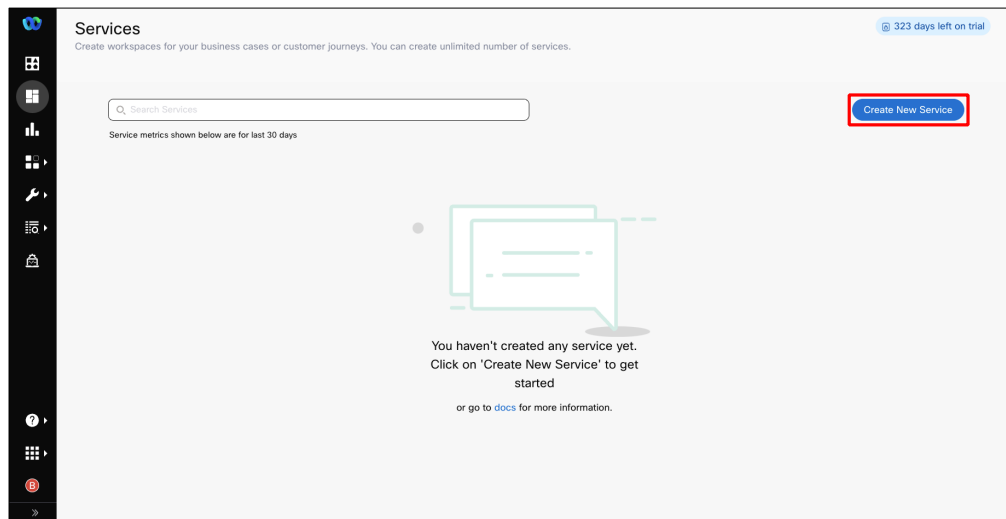


Step 2: Once logged in, locate “Services” in the sidebar and click on it. This will take you to the area where you can manage and create new services.





Step 3: In the Services section, find and click the button labeled “Create New Service”. This action will initiate the process of setting up a new service in your environment.



Create New Service

Service Name ⓘ

CancelCreate

Step 4: When prompted to name your new service, type “My First Service” into the designated field. This name will help you identify and manage the service in future sessions. Click the “Create” button when you are done.

Create New Service

Service Name ⓘ

My First Service

Cancel Create

Step 5: After naming your service, select the option “Create Blank Flow”. This allows you to start building a flow from scratch, giving you full control over its configuration.

Services - My First Service 323 days left on trial

Click to edit service description. E.g., "This service is for appointment reminders".

Dashboard </> API Flows Rules Settings

My First Service
Welcome, Brandon!

This is your service dashboard.

A service is a named workspace for managing all configurations related to a customer interaction/journey. Each service provides you with a unique service key and JWT tokens (refer to the API tab) that are required for authentication when using messaging APIs, voice APIs, custom event API, and/or inbound webhooks.

Services can be locked from within the 'Settings' section to limit edit access.

Service ID:50180

Flows
Start quickly with one of our pre-built templates or start with a blank canvas.

Parcel Alerts
Use This Template

Auto Responder
Use This Template

Appointment Reminders
Use This Template

Create Blank Flow View My Flows Buy Numbers Configure Apps View Documentation

Traffic ⓘ

SMS	0	0%
Voice	0	0%
WhatsApp	0	0%

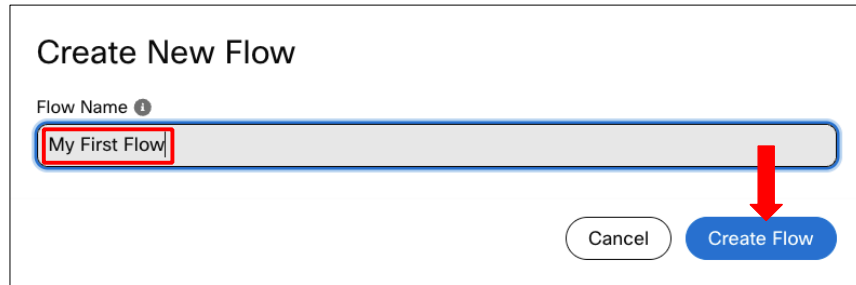
Create New Flow

Flow Name ⓘ

e.g., My New Flow

Cancel Create Flow

Step 6: You will now be prompted to name your new flow. Enter “My First Flow” as the name. This identifies your workflow within the service you just created. Click the “Create Flow” button when you’re done.



The 'Create New Flow' modal features a title bar at the top. Below it, the 'Flow Name' field is highlighted with a red box and contains the text 'My First Flow'. To the right of the input field is a red arrow pointing down towards the 'Create Flow' button. At the bottom right, there are two buttons: 'Cancel' and 'Create Flow'.

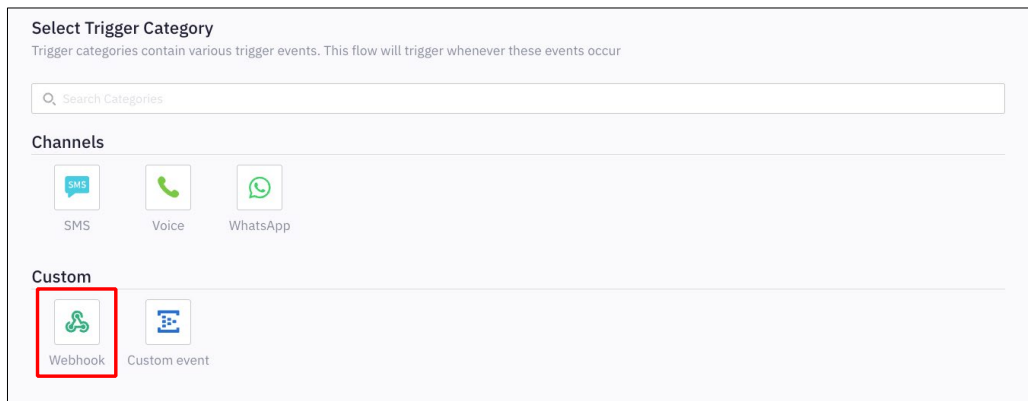
Create New Flow

Flow Name ⓘ

My First Flow

Cancel Create Flow

Step 7: For the trigger of this flow, select “Webhook”. This choice sets up the flow to be initiated by an incoming webhook, which is a key component of this lab.



The 'Select Trigger Category' modal has a title and a subtitle. Below the subtitle is a search bar. Under the 'Channels' section, there are three icons: SMS, Voice, and WhatsApp. Under the 'Custom' section, there are two icons: Webhook (highlighted with a red box) and Custom event.

Select Trigger Category

Trigger categories contain various trigger events. This flow will trigger whenever these events occur

Search Categories

Channels

SMS Voice WhatsApp

Custom

Webhook Custom event

Step 8: Before proceeding further with configuring this flow, take note of the “HELP” link located in the top right of the modal. This link is an important resource, as it directs you to documentation specific to each node you will be working with in the flow builder.

Configure Webhook

HELP

Configuration

Transition Actions (Optional)

Configure webhook settings to trigger this flow

☒ Select existing webhook
 ☐ Create new event

WEBHOOK NAME

Select a webhook

We've generated a new endpoint for you to send requests. [Learn more about using webhooks.](#)

WEBHOOK URL

https://hooks-sandbox.imconnect.io/events/

☐ Service key or JWT needs to be passed in request header if this option is selected

☐ Validate signature

Connect can validate JSON signatures generated using SHA1,SHA256 or SHA512 passed in the header. Invalid request will return a HTTP 400 error

Example Data

Paste JSON

Paste XML

Send Via Hook

PROVIDE SAMPLE INPUT

1

SELECT OUTPUT VARIABLES

PARSED VARIABLES(0)

Start

Node ID: 2

CANCEL

SAVE

Input Variables

List of variables available as input for this node

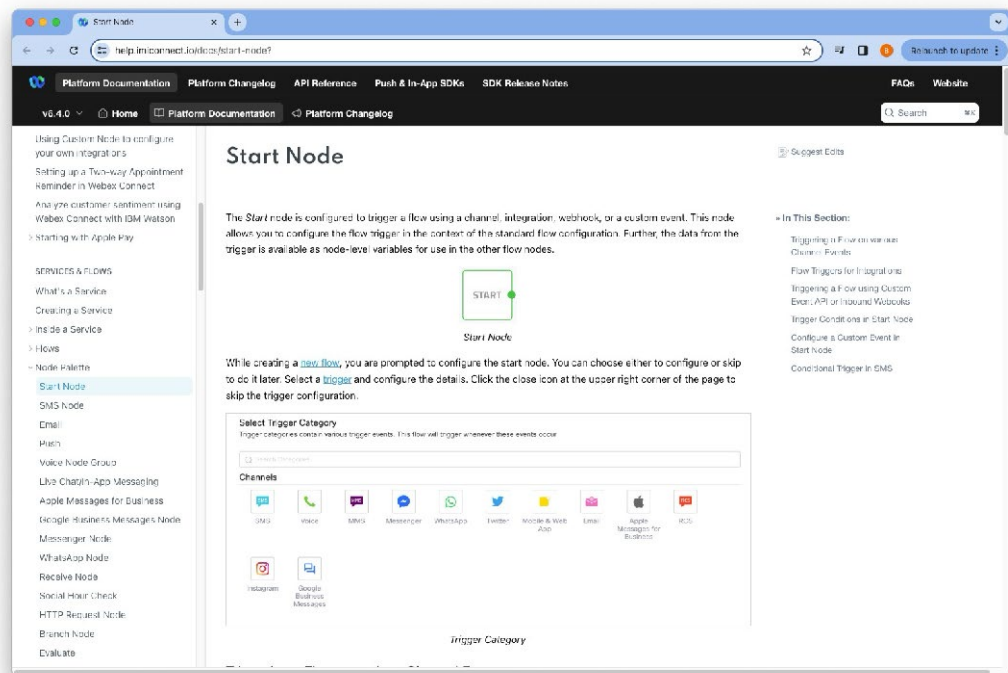
Search

Custom variables

[F23272]

Output Variables

Node Outcomes



Step 9: Click on the “Create new event” button to start setting up the inbound webhook that will trigger this flow. Observe that Cisco Connect automatically generates a URL for this webhook, which is essential for its operation.

Configure Webhook HELP

Configuration **Transition Actions (Optional)**

Configure webhook settings to trigger this flow

☐ Select existing webhook ☒ Create new event

WEBHOOK URL

<https://hooks-sandbox.imiconnect.io/events/Y0ZYV3EBX3>

We've generated a new endpoint for you to send requests. [Learn more about using webhooks.](#)

NAME ⓘ

☐ Service key or JWT needs to be passed in request header if this option is selected

☐ Validate signature

Connect can validate JSON signatures generated using SHA1, SHA256 or SHA512 passed in the header. Invalid request will return a HTTP 400 error

Example Data

[Paste JSON](#) [Paste XML](#) [Send Via Hook](#)

PROVIDE SAMPLE INPUT ⓘ

1

SELECT OUTPUT VARIABLES ⓘ

PARSED VARIABLES(0)

Start
Node ID: 2

[CANCEL](#) [SAVE](#)

Input Variables
List of variables available as input for this node

Custom variables [F23272]

Output Variables

Node Outcomes

Step 10: Name the webhook “My First Webhook”. This naming convention helps in identifying and referencing the webhook within your flow.

Configure Webhook HELP

Configuration **Transition Actions (Optional)**

Configure webhook settings to trigger this flow

☐ Select existing webhook ☒ Create new event

WEBHOOK URL

<https://hooks-sandbox.imiconnect.io/events/Y0ZYV3EBX3>

We've generated a new endpoint for you to send requests. [Learn more about using webhooks.](#)

NAME ⓘ

☐ Service key or JWT needs to be passed in request header if this option is selected

☐ Validate signature

Connect can validate JSON signatures generated using SHA1, SHA256 or SHA512 passed in the header. Invalid request will return a HTTP 400 error

Example Data

[Paste JSON](#) [Paste XML](#) [Send Via Hook](#)

PROVIDE SAMPLE INPUT ⓘ

1

SELECT OUTPUT VARIABLES ⓘ

PARSED VARIABLES(0)

Start
Node ID: 2

[CANCEL](#) [SAVE](#)

Input Variables
List of variables available as input for this node

Custom variables [F23272]

Output Variables

Node Outcomes

Step 11: In the section titled “Provide Sample Input”, type the following sample JSON

```
{  
  "x": 1,  
  "y": 2  
}
```

After entering this data, click the “Parse” button. Notice how the variables `x` and `y` appear as output variables to the right of the JSON sample. This indicates that subsequent steps in the flow can utilize these values using their variable names.

Example Data

Paste Json Paste XML Send Via Hook

PROVIDE SAMPLE INPUT ⓘ

1 {

2 "x": 1,

3 "y": 2

4 }

PARSE

Example Data

Paste Json Paste XML Send Via Hook

PROVIDE SAMPLE INPUT ⓘ

1 {

2 "x": 1,

3 "y": 2

4 }

↓

PARSE

PROVIDE SAMPLE INPUT ⓘ

```

1 {
2   "x": 1,
3   "y": 2
4 }

```

SELECT OUTPUT VARIABLES ⓘ

PARSED VARIABLES(2)

x:1

y:2

Step 12: To finalize your webhook configuration, click the “SAVE” button. This action confirms the settings and inputs you have entered for the webhook.

Configure Webhook

HELP

Configuration

Transition Actions (Optional)

HTTP 400 error

Example Data

Paste JSON

Paste XML

Send Via Hook

PROVIDE SAMPLE INPUT ⓘ

```

1 {
2   "x": 1,
3   "y": 2
4 }

```

SELECT OUTPUT VARIABLES ⓘ

PARSED VARIABLES(2)

x:1

y:2

PARSE

Conditions

Flow will invoke only when these conditions are met

Start

Node ID: 2

CANCEL

SAVE

Input Variables

List of variables available as input for this node

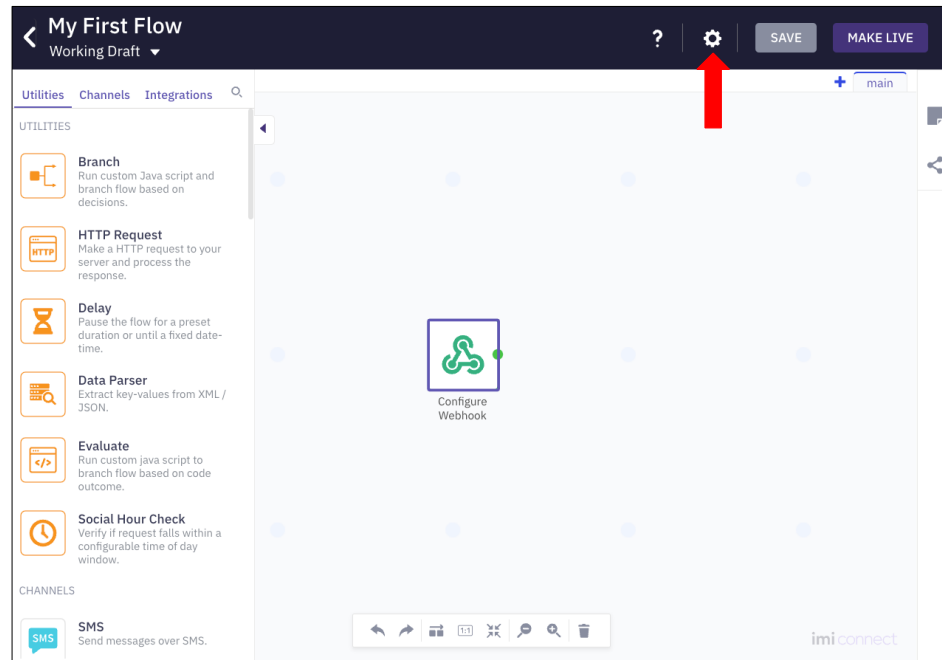
Search

Custom variables [F23272]

Output Variables

Node Outcomes

Step 13: Before moving on to the next configuration step, it’s important to ensure the flow builder is optimized for debugging. To do this, click the gear icon at the top of the flow builder screen. This will open the flow settings. Here, enable “Descriptive Logs” by clicking the toggle next to it. This setting allows for detailed outputs, including data, to be included in the flow log. Once you have enabled this feature, click the “Save” button to apply these settings.



Flow Settings

HELP

General

Flow Outcomes

Custom Variables

FLOW NAME

My First Flow

DESCRIPTION (OPTIONAL)

Enter Description here

Advanced settings (optional)

Set flow behaviour when running multiple instances of flows in parallel.

CORRELATION ID

\$(corrid)

Enabled

1

Descriptive logs

Activate to capture information required for Developers when trying to debug a problem, helpful when testing your service. Descriptive logs capture complete transaction details including sequence of all activities within flows, send/receive message payload, HTTP request & response entities which may include sensitive PII data of your customers.

Disabled

2

Prevent duplicate flow runs

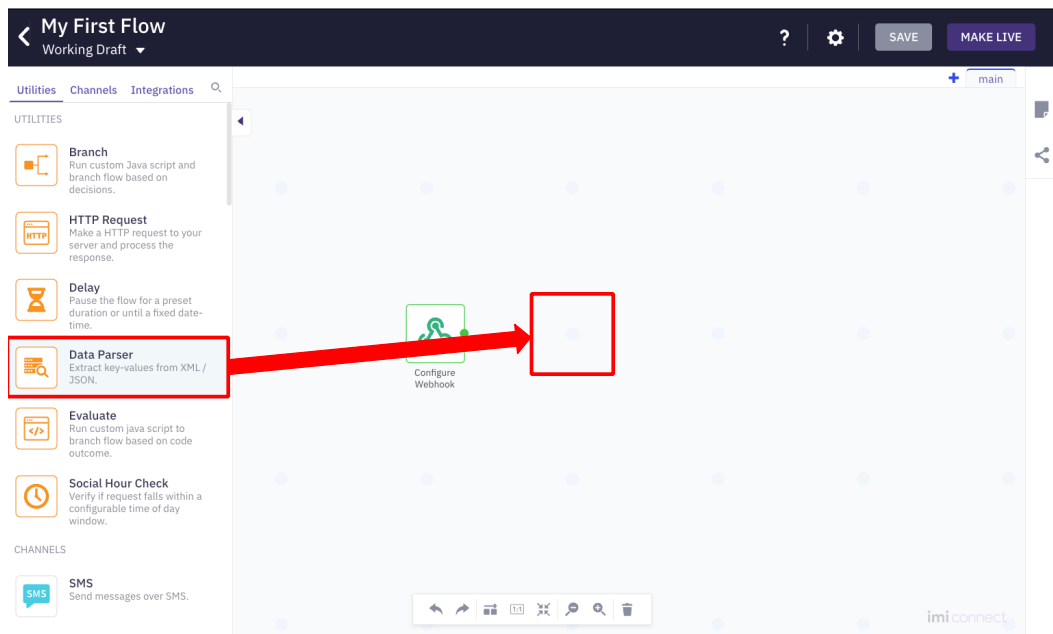
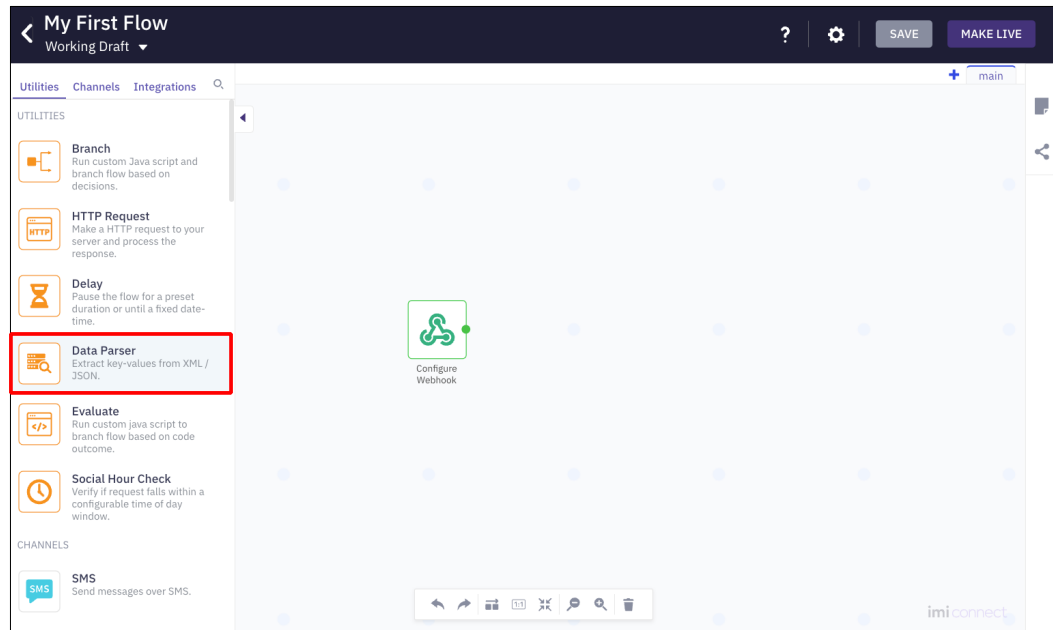
Enable to set a unique identifier for each run of the flow. Duplicate requests with the same set of identifiers will be discarded. Comma separate multiple identifiers.

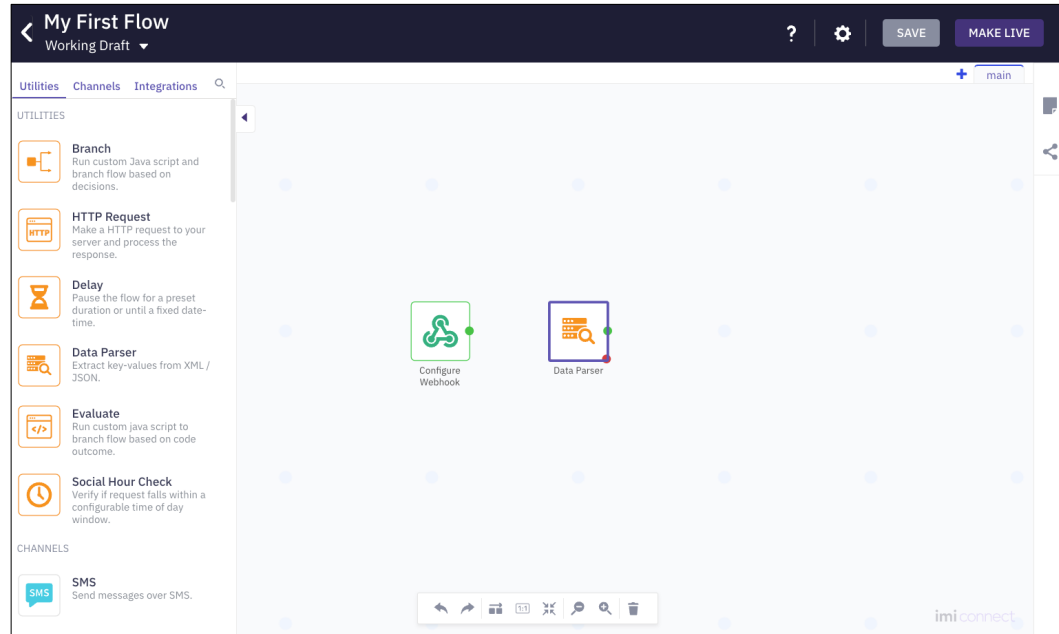
Flow id: 23272

CANCEL

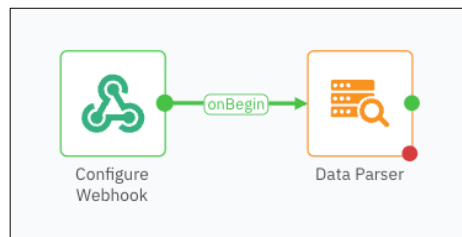
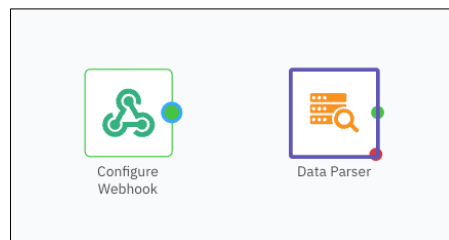
SAVE

Step 14: Next, incorporate the “Data Parser” node into your flow. This node is vital for processing incoming data. To add it, drag the “Data Parser” node from the sidebar into the flow builder workspace, positioning it to the right of the webhook start node. This node’s role is to locate data in the incoming JSON request and assign the located values to custom variables, which can be used later in the flow.



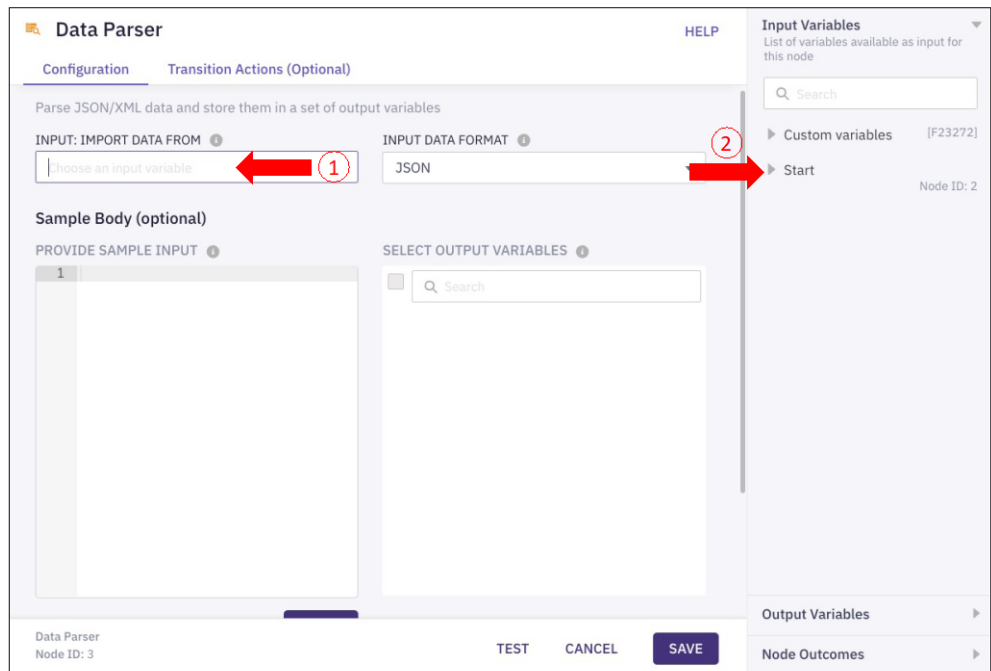
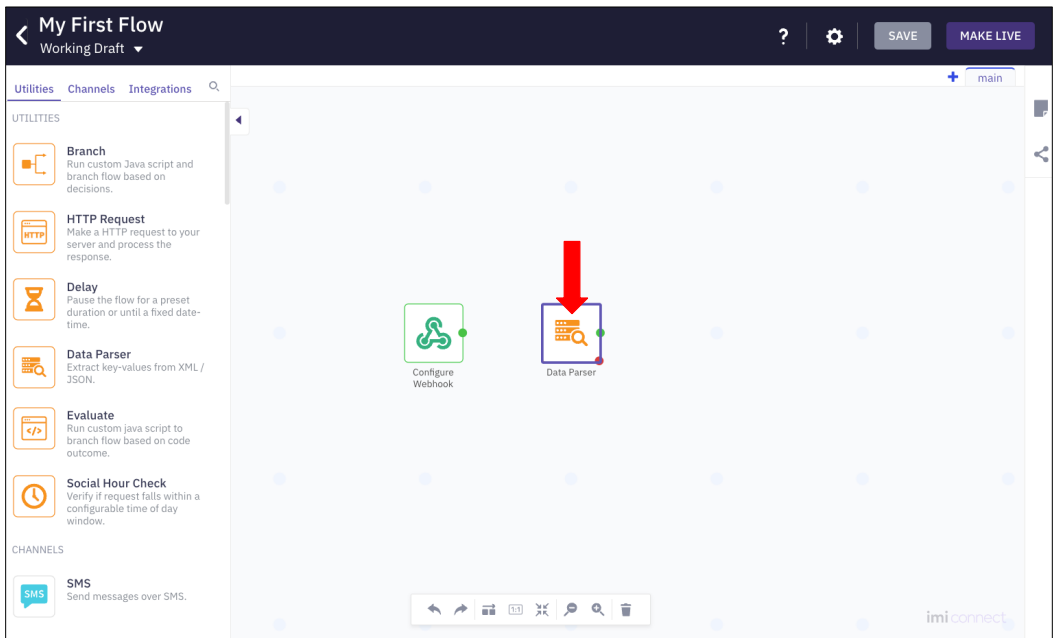


Step 15: To define the sequence of operation between nodes, use your cursor to create a connection. Click on the green dot on the “Configure Webhook” node, drag it to the “Data Parser” node, and release. This action connects these nodes, causing them to run in a defined order. Consequently, after the webhook receives an HTTP request, the output variables are assigned values, which are then processed by the “Data Parser” node to parse data and define custom variables for use in subsequent nodes of this flow.



Step 16: Double-click on the Data Parser node to start editing its functionality. You’ll need to specify the source of the input data for this node. Place your cursor in the text field labeled “Input: Input Data From”. Then, using the sidebar on the right,

expand the “Start” section and select “n2.inboundWebhook.payload”. This action will autofill the variable in the field. It’s important to note that the variable name is enclosed within ‘\$()’, a syntax used within the flow builder to reference variable values in text and other input fields.



Input Variables
List of variables available as input for this node

Search

Custom variables [F23272]

Start
Node ID: 2

- inboundWebhook.timestamp
- inboundWebhook.transId
- inboundWebhook.payload**
- inboundWebhook.x
- inboundWebhook.y

INPUT: IMPORT DATA FROM ⓘ

\$(n2.inboundWebhook.payload)

Step 17: In the “Sample Body” input field, type the same JSON you used earlier:

```
{
  "x": 1,
  "y": 2
}
```

After entering this data, click the “Parse” button. This will cause variables to appear in the “Select Output Variables” section of the modal.

Sample Body (optional)

PROVIDE SAMPLE INPUT ⓘ

```
1 {
2   "x": 1,
3   "y": 2
4 }
```

PARSE

Sample Body (optional)

PROVIDE SAMPLE INPUT ⓘ

```
1 {
2   "x": 1,
3   "y": 2
4 }
```

↓

PARSE

SELECT OUTPUT VARIABLES ⓘ

☐

☐ \$.x
1

☐ \$.y
2

Step 18: In the “Select Output Variables” section, click the box to the left of each variable option. This action selects these variables for use in the flow. Next click the “Import” button. Note the “Parsed data output” section is populated partially.

SELECT OUTPUT VARIABLES ⓘ

☐

☒ \$.x
1

☒ \$.y
2

IMPORT

SELECT OUTPUT VARIABLES ⓘ

☐

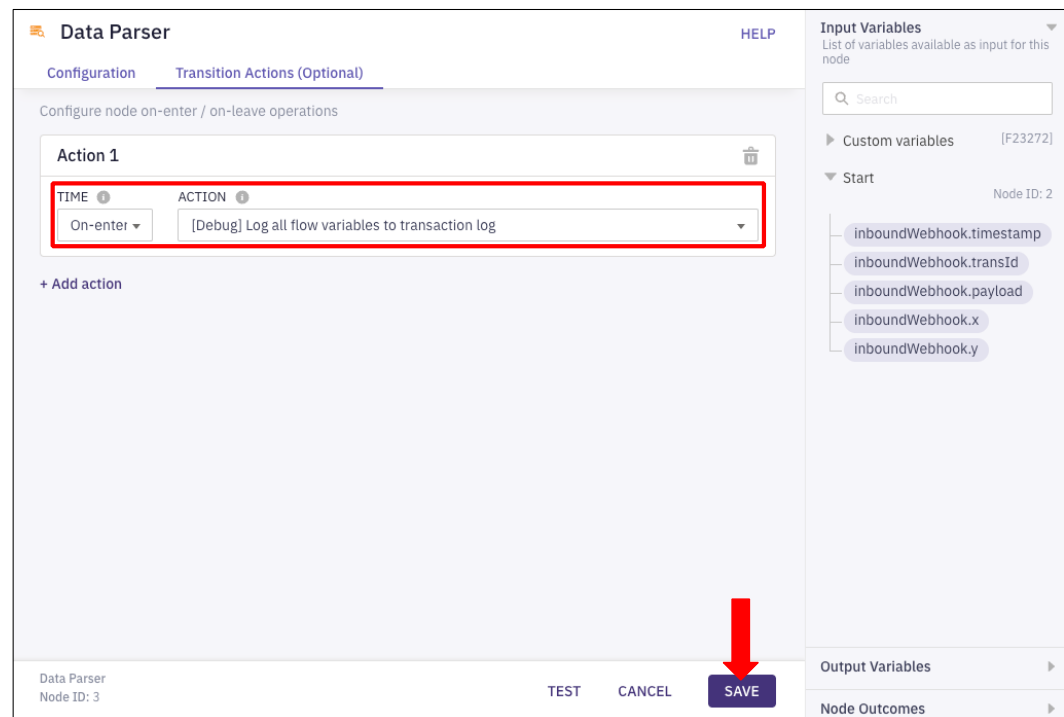
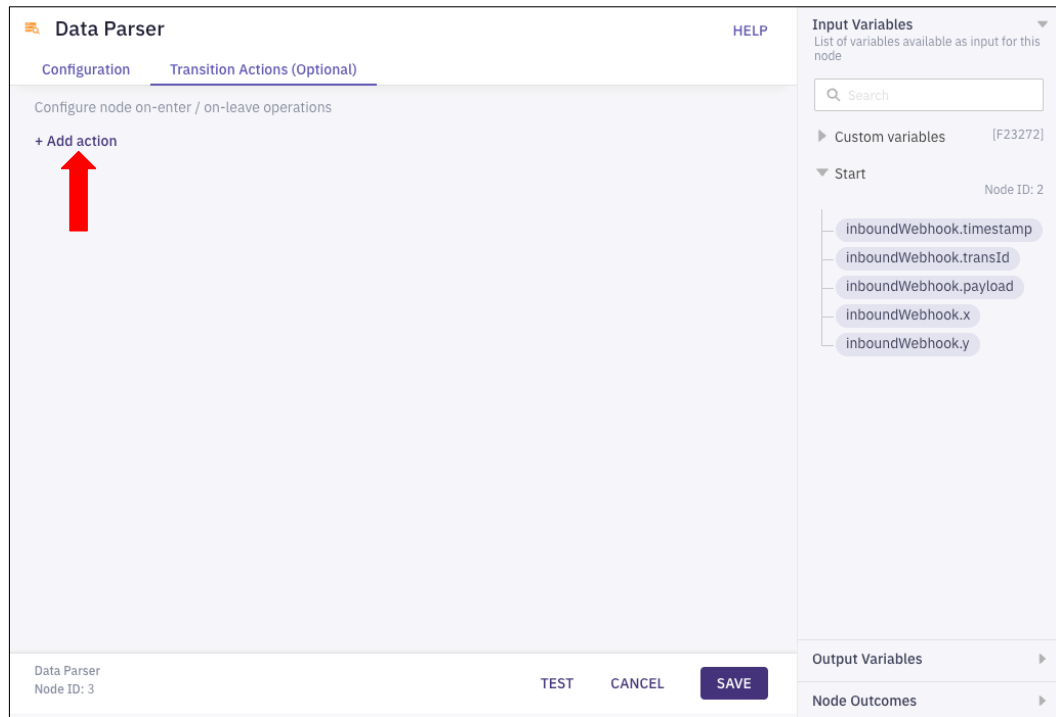
☒ \$.x
1

☒ \$.y
2

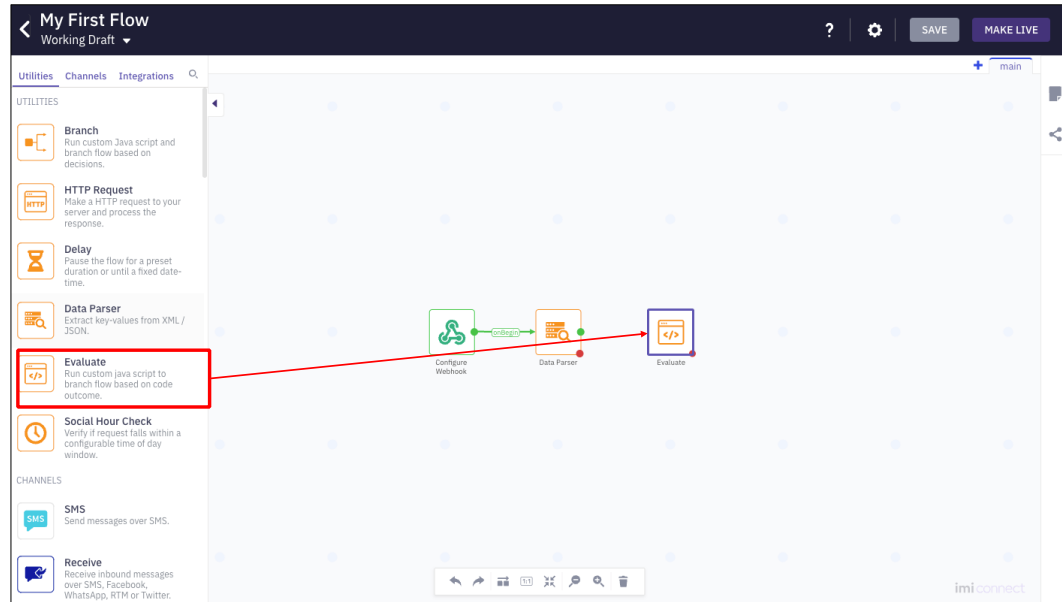
IMPORT

Step 19: In the modal's "Parsed data input" section, edit the "Output Variable Name" text input field for `$.x` and `$.y` to be `x` and `y` respectively. This will make it so the local variables `$.x` and `$.y` are accessible as custom variables by all following nodes by the names `x` and `y`. Click the checkbox labeled "Mandatory" at the end of each row. This will lead to an error being raised if the keyword "x" or "y" is not present in the incoming JSON payload.

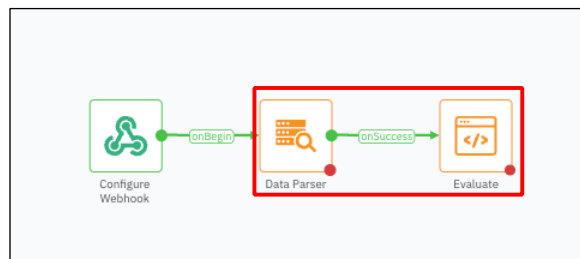
Step 20: To facilitate debugging, click on the "Transaction Actions" tab at the top of the modal. Click "+ Add action" to begin configuring an action that will run at some point during this node's execution. Set the value of "Time" to be "On-enter" if it is not already so. Next, select the action dropdown menu and choose "[Debug] Log all flow variables to transaction log". This setting is crucial as it aids in identifying issues if errors arise when developing your flow. Once these settings are configured, click the "Save" button to complete editing this node.



Step 21: Now, add the “Evaluate” node to your flow. Drag this node from the sidebar into the flow builder, positioning it to the right of the “Data Parser” node.



Step 22: Establish a connection between the “Data Parser” and “Evaluate” nodes. Using your cursor, click on the green dot on the “Data Parser” node, drag to the “Evaluate” node, and release.



Step 23: Double-click the “Evaluate” node to begin its configuration. After your webhook receives an HTTP request, the output variables are assigned values, which are then utilized by the “Evaluate” node to perform JavaScript logic. In this node, you will input JavaScript code that adds the values of two variables from the API request that will be used to initiate this flow. This code checks if the variables `x` and `y`, which were defined in the previous node, are numeric in value or contain a number.

Note: It’s important to understand the difference between numeric and string representations of numbers. For example, `1` is a number, whereas `"1"` is a string containing a number. The JavaScript function `isNaN()` (is Not a Number) will evaluate to `False` for both of these values. Adding an exclamation point (!) before the function negates this value, turning it to `True`.

If both variables are numbers or contain numbers, the variable `sum` is created with its value set to the sum of `x` and `y` after they have been coerced into the

Number data type. Following the creation of the **sum** variable, the number **1** is returned. The returned value determines the branching that will occur after this node. In this case, **1** will represent success.

If one of the variables is not a number or does not contain a number, the variable **sum** is still created, but it is assigned the value **0**, and then the number **2** is returned. In this instance, **2** will represent failure due to invalid data.

Enter the following code in the code entry input field:

```
if (!isNaN(x) && !isNaN(y)) {  
    var sum = Number(x) + Number(y);  
    1;  
} else {  
    var sum = -1;  
    2;  
}
```

Evaluate HELP

Configuration **Transition Actions (Optional)**

Enter JavaScript to perform logic and return values as output. Configured script outputs will be available as node outcome

```
1 if (!isNaN(x) && !isNaN(y)) {  
2   var sum = Number(x) + Number(y);  
3   1;  
4 } else {  
5   var sum = -1;  
6   2;  
7 }  
8
```

Step 24: In the “Script Output” text box, type **1**. Then, in the “Branch Name” input box, type **onSuccess**. This configures the branch for successful outcomes. Click “+ ADD NEW” to create another branch to handle error outcomes. In the “Script Output” text box for this new branch, type **2**. Then, in the “Branch Name” input box, type **onError**. This configures the branch for error outcomes. When you are done, click the “Save” button to finalize the configuration of the “Evaluate” node.

Configure Script Output

SCRIPT OUTPUT ⓘ	BRANCH NAME ⓘ
1	onSuccess
2	onError

+ ADD NEW

Step 25: Click on the “Transaction Actions” tab at the top of the modal. Click “+ Add action”. Set the value of “Time” to be “On-enter” if it is not already so. Next, select the action dropdown menu and choose “[Debug] Log all flow variables to transaction log”. Once these settings are configured, click the “Save” button to complete editing this node.

Evaluate

HELP

Configuration

Transition Actions (Optional)

Configure node on-enter / on-leave operations

Action 1

TIME ⓘ

On-enter

ACTION ⓘ

[Debug] Log all flow variables to transaction log

+ Add action

Evaluate

Node ID: 5

TEST

CANCEL

SAVE

Input Variables

List of variables available as input for this node

Search

Custom variables

[F23272]

Start

Node ID: 2

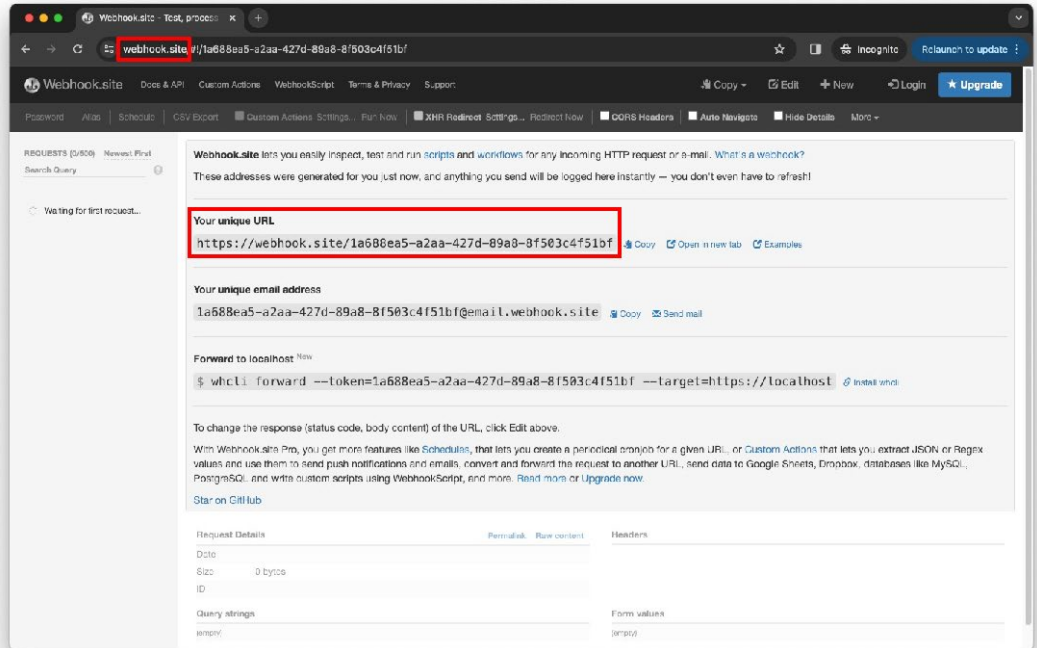
Data Parser

Node ID: 3

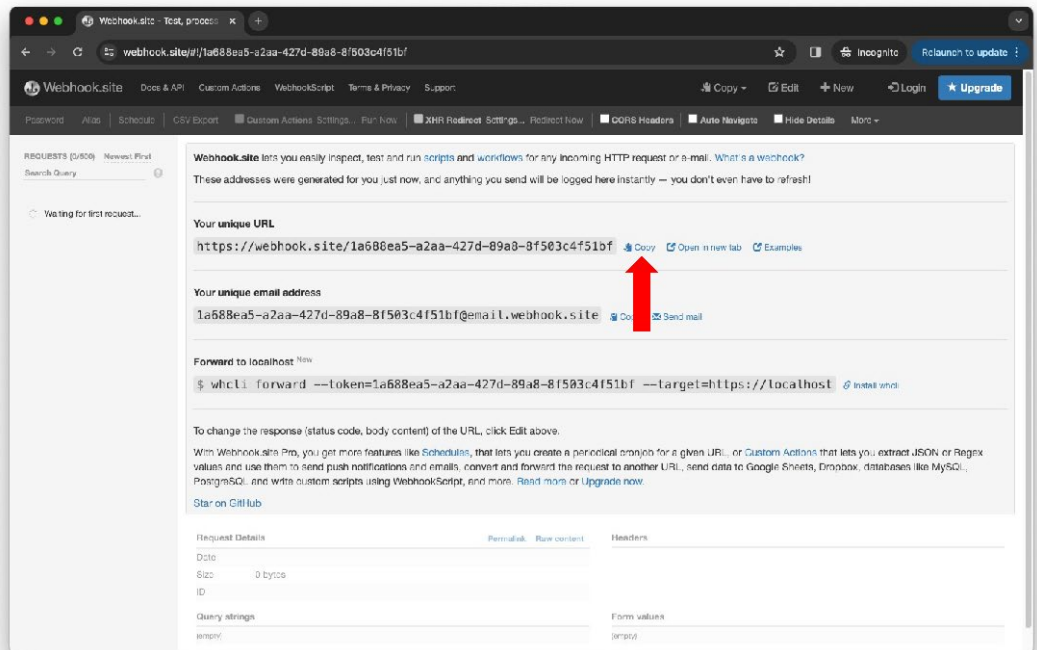
Output Variables

Node Outcomes

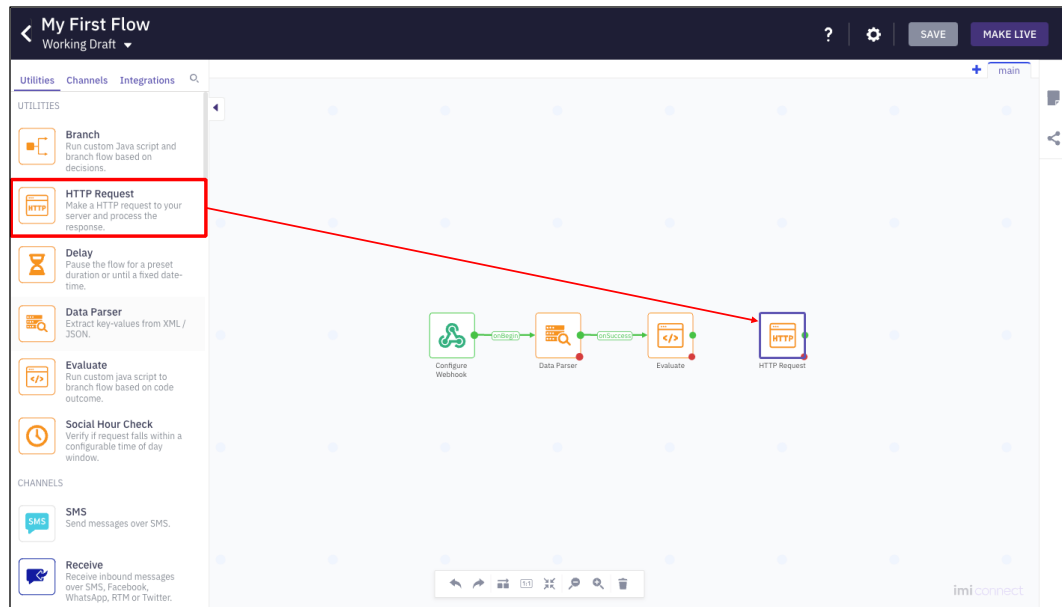
Step 26: Open a new tab in your web browser and navigate to <https://webhook.site/>. This step is preparatory for defining the completion logic of your flow. For any given event, whether it results in failure or success, you can make an HTTP request with data to another webhook. This could potentially trigger another programmatic process or simply log the results.



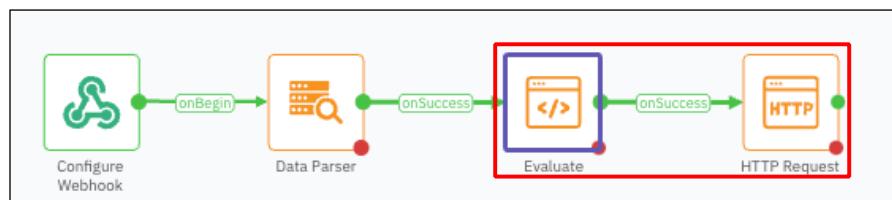
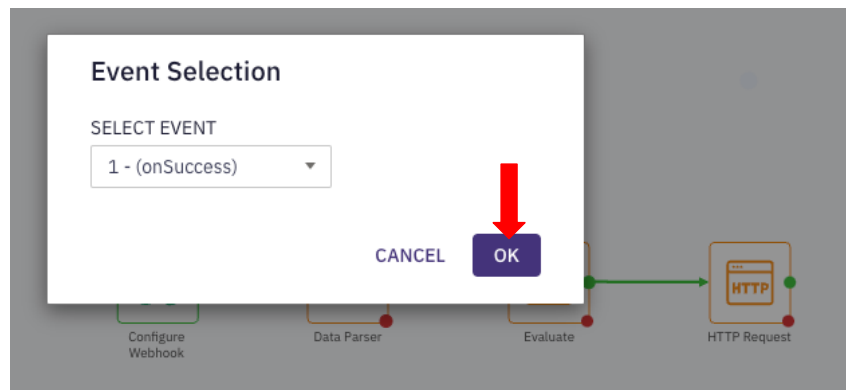
Step 27: Copy the value of “Your unique URL” from https://webhook.site for use in the following steps. Keep this tab open and return to the tab with the flow builder.



Step 28: Return to the Cisco Connect flow builder tab and add the “HTTP Request” node to your flow. To do this, drag the “HTTP Request” node from the sidebar and position it to the right of the “Evaluate” node.



Step 29: Now, establish the workflow connection between the “Evaluate” node and the “HTTP Request” node. Note that because there are numerous branches possible after the “Evaluate” node, you are now prompted to choose which branch should lead to the “HTTP Request” node. Ensure **onSuccess** is selected and click save.



Step 30: Double-click the “HTTP Request” node to begin configuration. Change the method to “POST” and paste the webhook URL you copied from <https://webhook.site/> into the “Endpoint URL” input field.

HTTP Request

HELP

Configuration

Transition Actions (Optional)

Make GET, POST, PUT, PATCH and DELETE requests to your application

METHOD

GET

ENDPOINT URL

e.g., https://api.myendpoint.com/v1/resources/

HEADER

e.g., Authorization

VALUE

e.g., Bearer ya29.c.Elo4BQQH_IS2AmoDSaIn_3-3hQ-kOSYDHPav

+ ADD ANOTHER HEADER

CONNECTION TIMEOUT

e.g., 1000 ms

REQUEST TIMEOUT

e.g., 1000 ms

PROXY ADDRESS(OPTIONAL)

e.g., 192.128.19.243:80

Response

The response body will be stored in `n6.http.responseBody` variable.

HTTP Request

Node ID: 6

TEST

CANCEL

SAVE

Input Variables

List of variables available as input for this node

Q Search

Custom variables

[F23272]

Start

Node ID: 2

Data Parser

Node ID: 3

Evaluate

Node ID: 5

Output Variables

Node Outcomes

HTTP Request

HELP

Configuration

Transition Actions (Optional)

Make GET, POST, PUT, PATCH and DELETE requests to your application

METHOD

POST

ENDPOINT URL

https://webhook.site/1a688ea5-a2aa-427d-89a8-8f503c4f51bf

HEADER

e.g., Authorization

VALUE

e.g., Bearer ya29.c.Elo4BQQH_IS2AmoDSaIn_3-3hQ-kOSYDHPav

+ ADD ANOTHER HEADER

BODY

CONNECTION TIMEOUT

e.g., 1000 ms

REQUEST TIMEOUT

e.g., 1000 ms

PROXY ADDRESS(OPTIONAL)

e.g., 192.128.19.243:80

Response

The response body will be stored in `n6.http.responseBody` variable.

HTTP Request

Node ID: 6

TEST

CANCEL

SAVE

Input Variables

List of variables available as input for this node

Q Search

Custom variables

[F23272]

Start

Node ID: 2

Data Parser

Node ID: 3

Evaluate

Node ID: 5

Output Variables

Node Outcomes

Step 31: In the “BODY” input field of the “HTTP Request” node, input the following JSON structure:

```
{
  "x": $(n2.inboundWebhook.x),
  "y": $(n2.inboundWebhook.y),
  "sum": $(sum)
}
```

This format ensures that the values of `x`, `y`, and `sum` from the previous nodes are correctly passed as parameters in the HTTP request. When you are done, click the “SAVE” button to finalize the configuration of your “HTTP Request” node.

HTTP Request HELP

Configuration **Transition Actions (Optional)**

Make GET, POST, PUT, PATCH and DELETE requests to your application

METHOD POST **ENDPOINT URL** https://webhook.site/1a688ea5-a2aa-427d-89a8-8f503c4f51bf

HEADER e.g., Authorization **VALUE** e.g., Bearer ya29.c.Eio4BQQH_J52AmoD5aIn_J-3hQ-k05YDPav

+ ADD ANOTHER HEADER

BODY

```
{
  "x": "${n2.inboundWebhook.x}",
  "y": "${n2.inboundWebhook.y}",
  "sum": "${sum}"
}
```

CONNECTION TIMEOUT e.g., 1000 ms **REQUEST TIMEOUT** e.g., 1000 ms

PROXY ADDRESS(OPTIONAL) e.g., 192.128.19.243:80

Response
The response body will be stored in `n6.http.responseBody` variable.

TEST CANCEL **SAVE**

Input Variables
List of variables available as input for this node

Search

Custom variables [F23272]

Start Node ID: 2

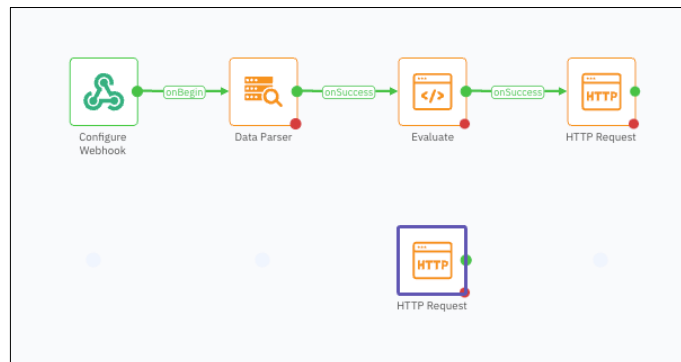
Data Parser Node ID: 3

Evaluate Node ID: 5

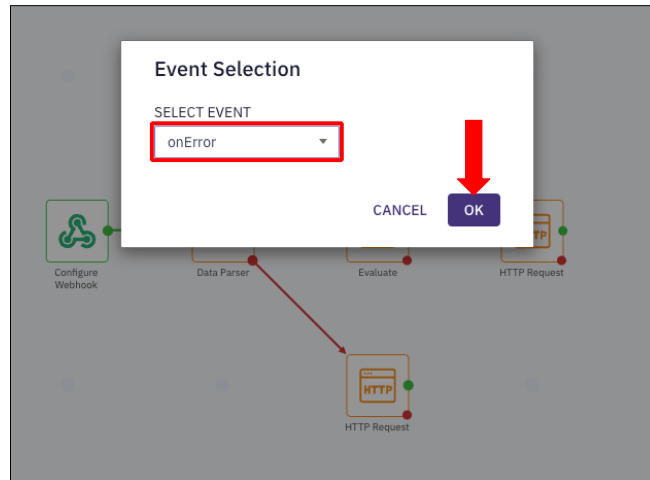
Output Variables

Node Outcomes

Step 32: Your flow could be deployed at this point, but if anything were to go wrong, there is no error handling to gracefully handle failures. Let's configure some more HTTP requests to the same external webhook to log errors. Drag another “HTTP Request” node to the flow builder, positioned below the “Evaluate” node.



Step 33: Drag a line connecting the red dot on the “Data Parser” node to the new “HTTP Request” node. When prompted for the branch that should lead to this node, select `onError` and click “OK”.



Step 34: Double-click the new “HTTP Request” node to begin configuring it. Select **POST** as the method, paste the webhook URL from <https://webhook.site>, and input the JSON below into the BODY text input field. When you are done, click Save to finish editing this node.

```
{
  "error": true,
  "message": "An error has occurred while trying to parse
request JSON data"
}
```

HTTP Request HELP

Configuration Transition Actions (Optional)

Make GET, POST, PUT, PATCH and DELETE requests to your application

METHOD POST **ENDPOINT URL** https://webhook.site/1a688ea5-a2aa-427d-89a8-8f503c4f51bf

HEADER e.g., Authorization **VALUE** e.g., Bearer ya29.c.Elo4BQQHtJS2AmoDSaIn_3-nQ-k0SYDHPav

BODY

```
{
  "error": true,
  "message": "An error has occurred while trying to parse request JSON data"
}
```

CONNECTION TIMEOUT e.g., 1000 ms **REQUEST TIMEOUT** e.g., 1000 ms

PROXY ADDRESS(OPTIONAL) e.g., 192.128.19.243:80

Response
The response body will be stored in `n7.http.responseBody` variable.

HTTP Request
Node ID: 7

TEST CANCEL **SAVE**

Input Variables
List of variables available as input for this node

Search

Custom variables [F23272]

Start Node ID: 2

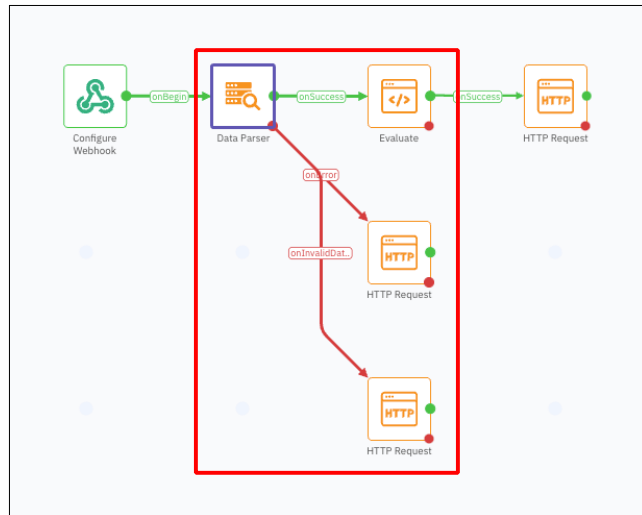
Data Parser Node ID: 3

Input Variables

Output Variables

Node Outcomes

Step 35: Drag another “HTTP Request” node and drop it below the node you just configured. Drag another line from the red dot on the “Data Parser” node to the new “HTTP Request” node. Note that you are not prompted to select a branch, as there is only one branch remaining that could be selected, `onInvalidData`.



Step 36: Double-click the new “HTTP Request” node to begin configuring it. Select `POST` as the method, paste the webhook URL from <https://webhook.site>, and input the JSON below into the BODY text input field. When you are done, click Save to finish editing this node.

```
{
  "error": true,
  "message": "Invalid Input. Data may be missing from JSON request body"
}
```

HTTP Request HELP

Configuration **Transition Actions (Optional)**

Make GET, POST, PUT, PATCH and DELETE requests to your application

METHOD **ENDPOINT URL**

POST

HEADER **VALUE**

e.g., Authorization e.g., Bearer ya29...Elo4BQOH...S2AmoDSaIn...3-3HQ-k0SYDHPav

+ ADD ANOTHER HEADER

BODY

```
{
  "error": true,
  "message": "Invalid Input. Data may be missing from JSON request body"
}
```

CONNECTION TIMEOUT **REQUEST TIMEOUT**

e.g., 1000 ms e.g., 1000 ms

PROXY ADDRESS(OPTIONAL)

e.g., 192.128.19.243:80

Response

The response body will be stored in `n8.http.responseBody` variable.

HTTP Request
Node ID: 8

TEST CANCEL **SAVE**

Input Variables
List of variables available as input for this node

Search

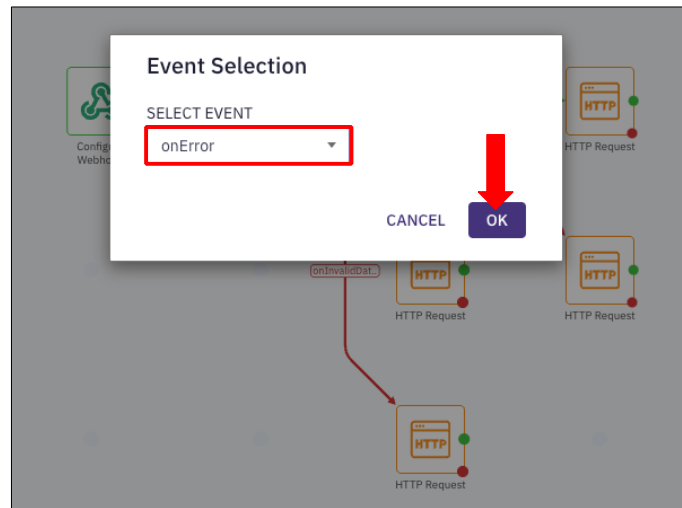
Custom variables [F23272]

Start Node ID: 2

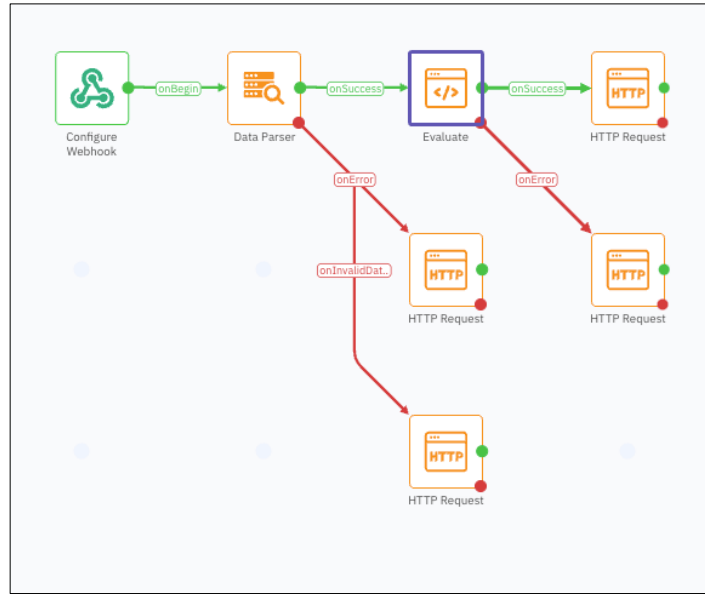
Data Parser Node ID: 3

Output Variables

Node Outcomes



Step 37: Click the **onError** “HTTP Request” node and use a keyboard shortcut to copy this node and its configuration (**Ctrl + C** on windows, **Cmd + C** on mac). Paste the node into the flow builder and drag it to be positioned below the top “HTTP Request” node. Once positioned, drag a line from the red dot on the “Evaluate” node to the new “HTTP Request” node. When prompted for a branch, select **onError** and click OK.



Step 38: Double-click the new “HTTP Request” node to edit, then change the value of “message” in the Body to be **“An error has occurred while trying to evaluate data”**. Click Save when you are done.

HTTP Request HELP

Configuration Transition Actions (Optional)

Make GET, POST, PUT, PATCH and DELETE requests to your application

METHOD: POST ENDPOINT URL: https://webhook.site/1a688ea5-a2aa-427d-89a8-8f503c4f51bf

HEADER: e.g., Authorization VALUE: e.g., Bearer ya29.c.Elo4BQOH_iS2AmoDSaIn_3-3hQ-kOSYDHPav

+ ADD ANOTHER HEADER

BODY

```

{
  "error": true,
  "message": "An error has occurred while trying to evaluate data"
}

```

CONNECTION TIMEOUT: e.g., 1000 ms REQUEST TIMEOUT: e.g., 1000 ms

PROXY ADDRESS(OPTIONAL): e.g., 192.128.19.243:80

Response

The response body will be stored in `n10.http.responseBody` variable.

HTTP Request Node ID: 10 TEST CANCEL **SAVE**

Input Variables

List of variables available as input for this node

Search

Custom variables [F23272]

Start Node ID: 2

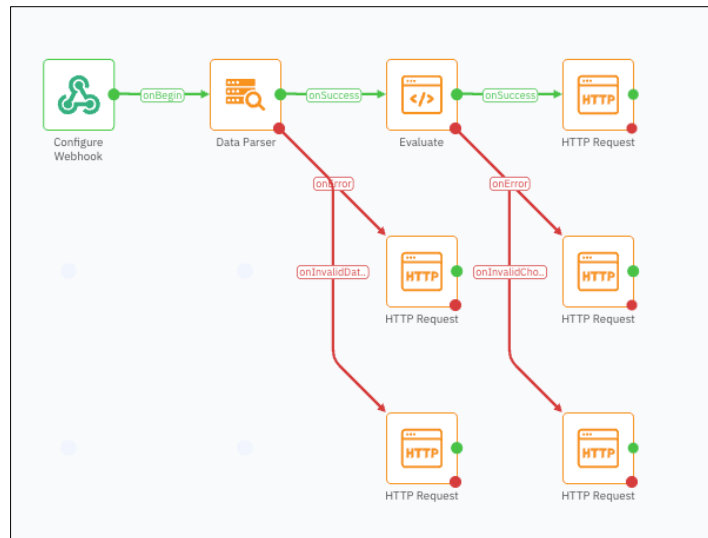
Data Parser Node ID: 3

Evaluate Node ID: 5

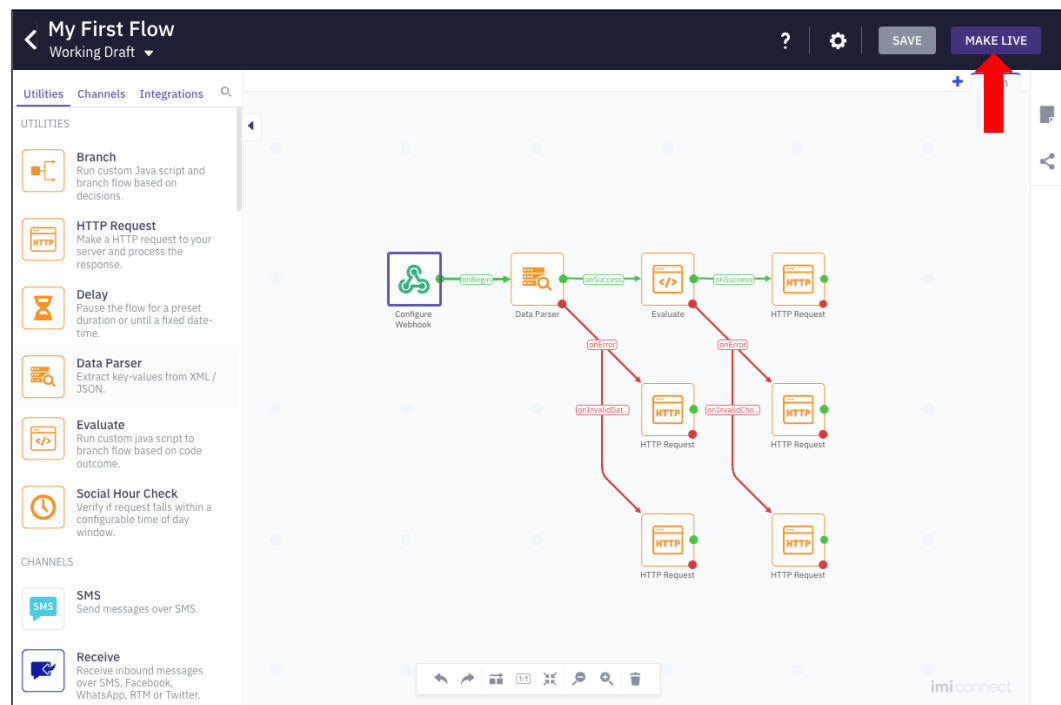
Output Variables

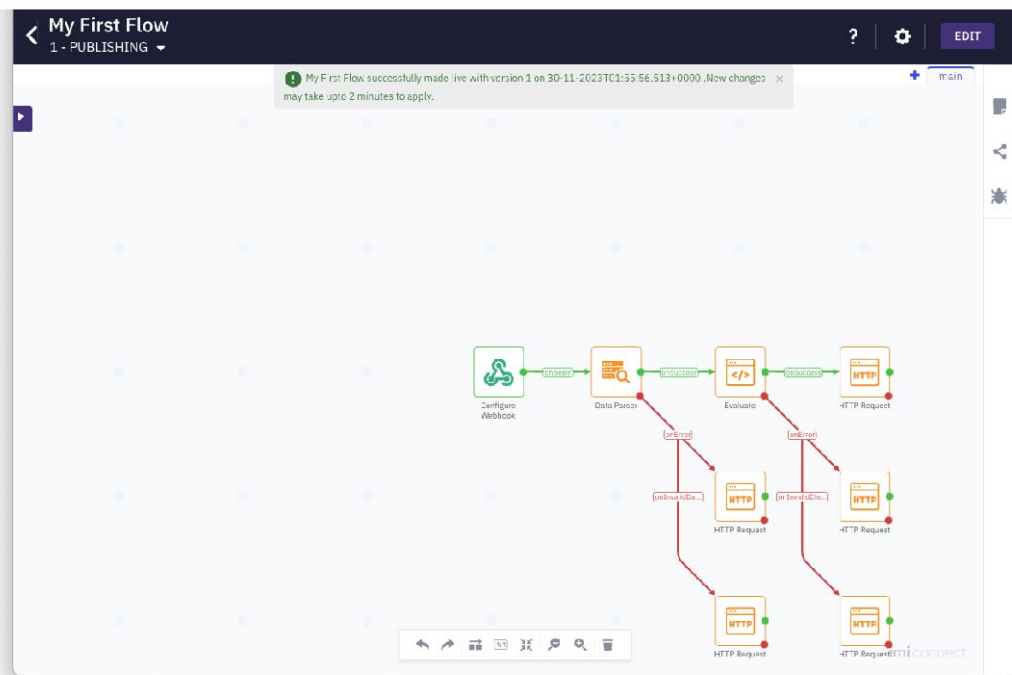
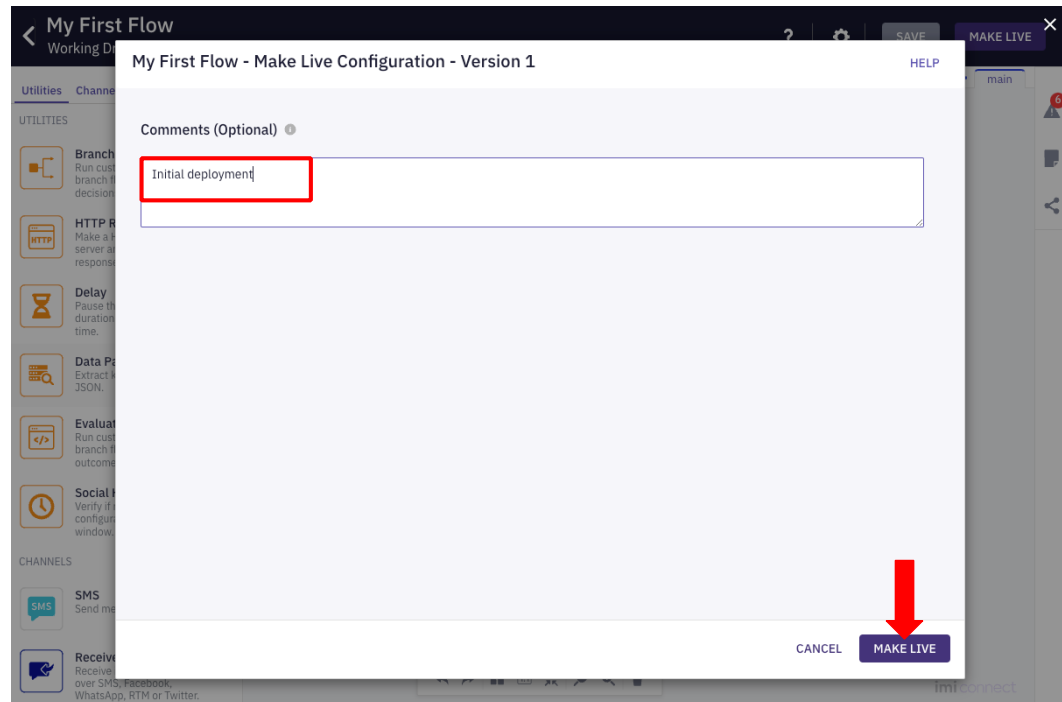
Node Outcomes

Step 39: Click the **onInvalidData** “HTTP Request” node, copy, and paste the node into the flow builder and drag it to be positioned below the last “HTTP Request” node you configured. Once positioned, drag a line from the red dot on the “Evaluate” node to the new “HTTP Request” node. This node will handle the **onInvalidData** node event that can be raised by the “Evaluate” node.

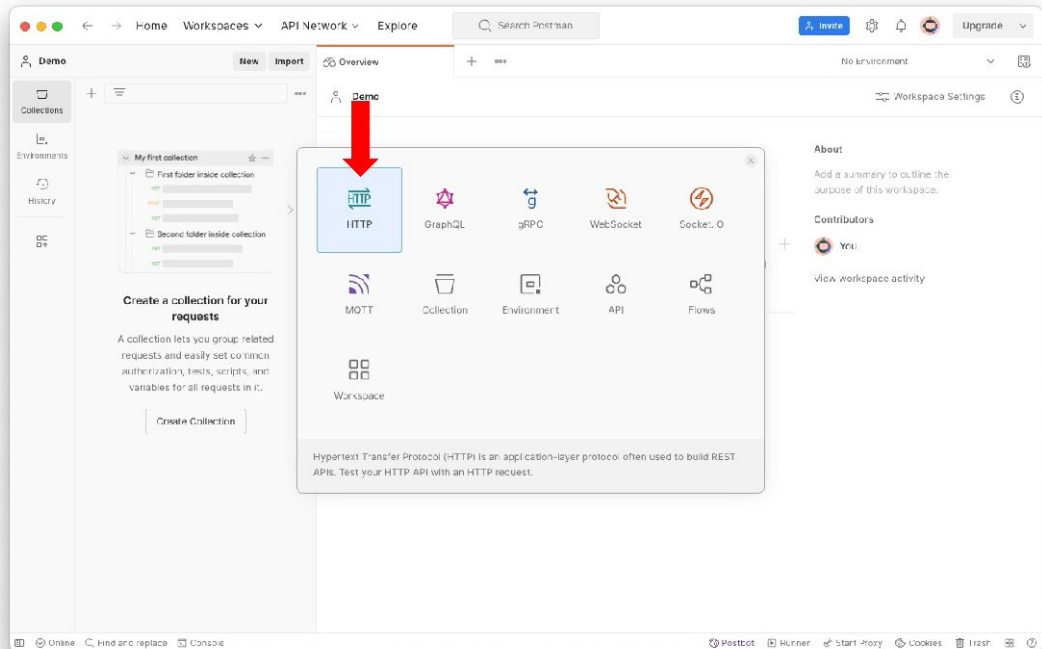
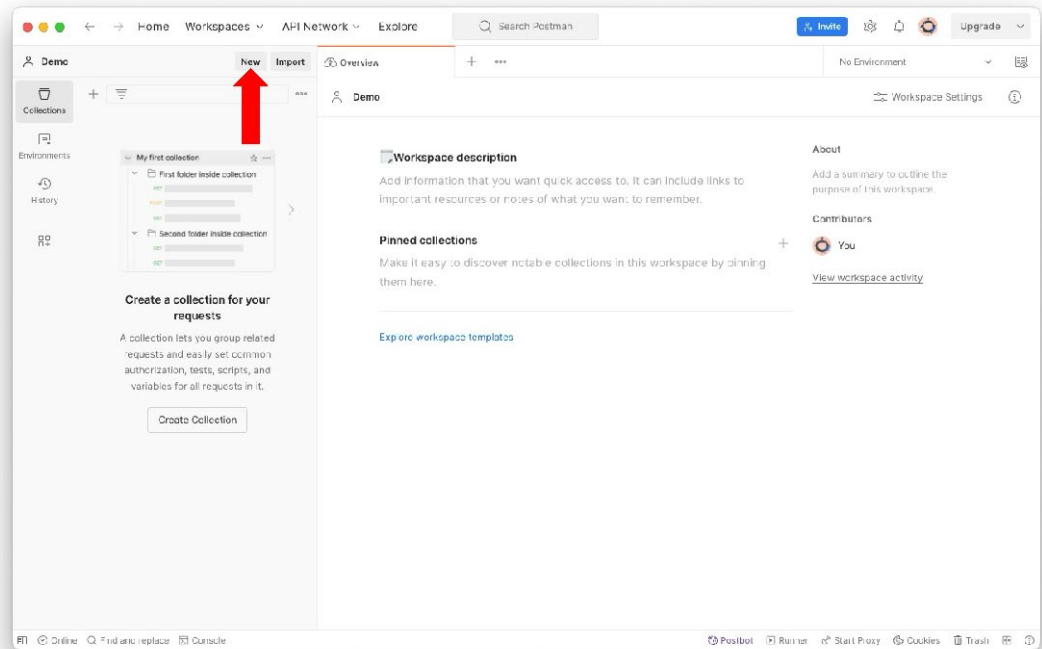


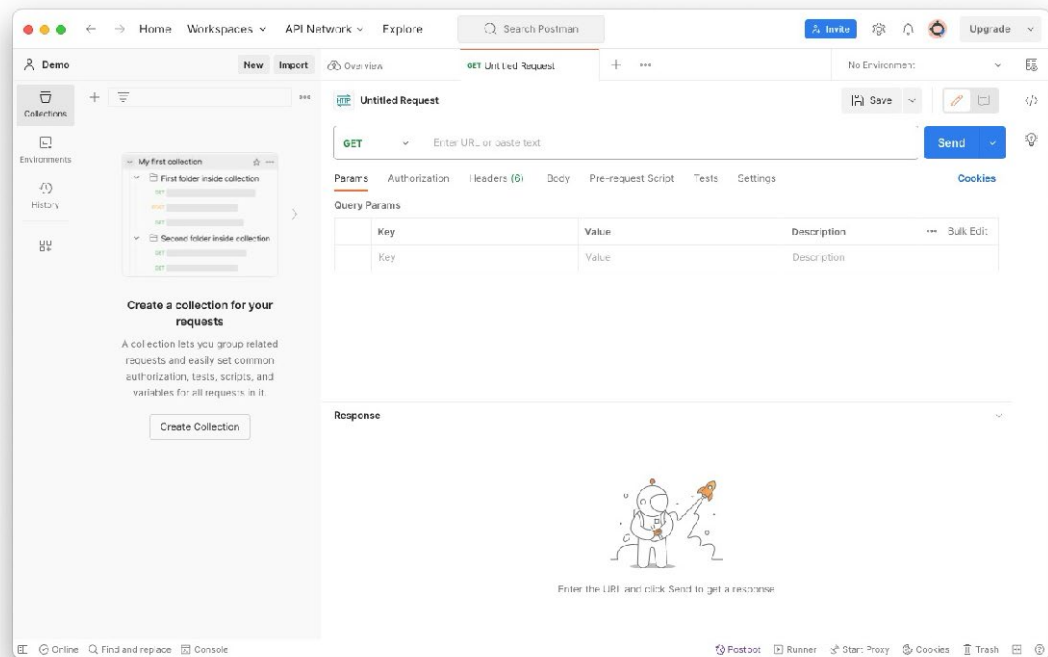
Step 40: You are now finished editing this flow. We could go on to add error handling for the outgoing HTTP Requests, or we could even configure one node that all HTTP Requests branch to on failure, but we should be in a good place to deploy and test our flow now. Click “Make Live” to deploy your flow. In the next screen, type “Initial deployment” in the text input field and click Save. This action changes the status of the flow from a draft or development state to live, making it operational. Once this is done, your flow is active and ready to respond to incoming webhook events.



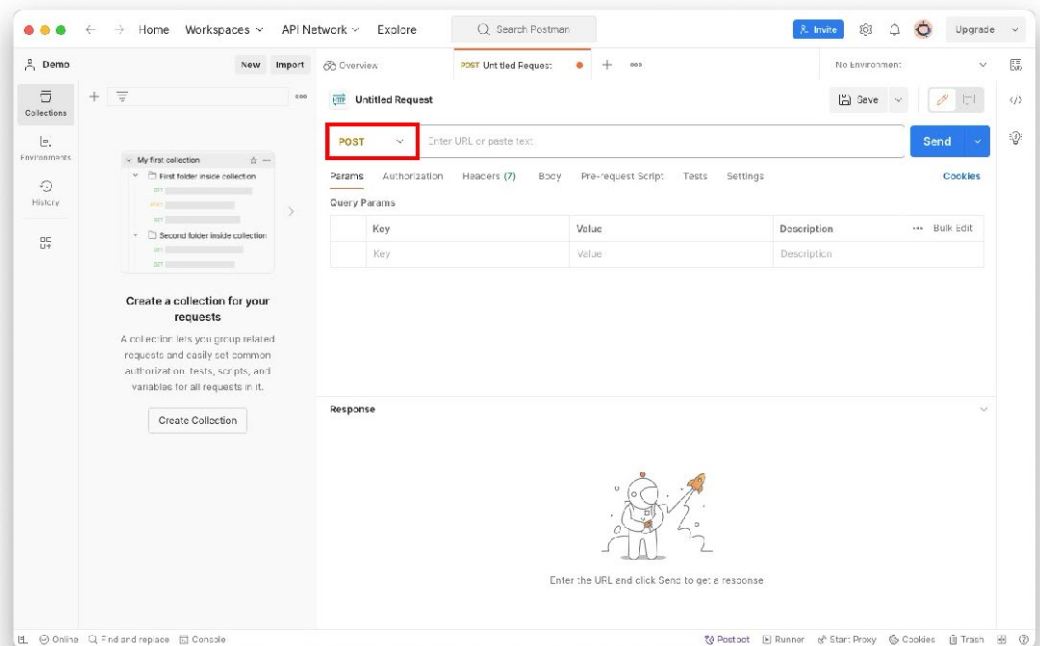


Step 41: Open Postman, a popular tool for testing and interacting with APIs. Once open, click "New", then click "HTTP" to create a new HTTP request.





Step 42: Change the method to POST, as this is the method your webhook is configured to accept. Return to the flow builder tab, double-click the “Configure Webhook” node, and copy the webhook URL. Back in Postman, paste the copied webhook URL in the address field of the HTTP request you are building. This directs the request to the specific endpoint you have set up in your flow.



Configure Webhook

[Configuration](#)
[Transition Actions \(Optional\)](#)

Configure webhook settings to trigger this flow

☒ Select existing webhook
 ☐ Create new event

WEBHOOK NAME

My First Webhook

We've generated a new endpoint for you to send requests. [Learn more about using webhooks.](#)

WEBHOOK URL

<https://hooks-sandbox.imconnect.io/events/Y0ZYV3EBX3>

☐ Service key or JWT needs to be passed in request header if this option is selected

☐ Validate signature
 Connect can validate JSON signatures generated using SHA1, SHA256 or SHA512 passed in the header. Invalid request will return a HTTP 400 error

Example Data

[Paste JSON](#)
[Paste XML](#)
[Send Via Hook](#)

PROVIDE SAMPLE INPUT

```

1 {
2   "x": 1,
3   "y": 2
4 }
          
```

SELECT OUTPUT VARIABLES

PARSED VARIABLES(2)

x:1

y:2

Start

Node ID: 2

CANCEL

Input Variables

List of variables available as input for this node

Search

Custom variables [F23272]

Output Variables

Node Outcomes

Home

Workspaces

API Network

Explore

[Invite](#)
[Upgrade](#)

Demo

[New](#)
[Import](#)

[Overview](#)

POST <https://hooks-sandbox.imconnect.io/events/Y0ZYV3EBX3>

[Save](#)
[Send](#)

[Parameters](#)
[Authorization](#)
[Headers \(7\)](#)
[Body](#)
[Pre-request Script](#)
[Tests](#)
[Settings](#)

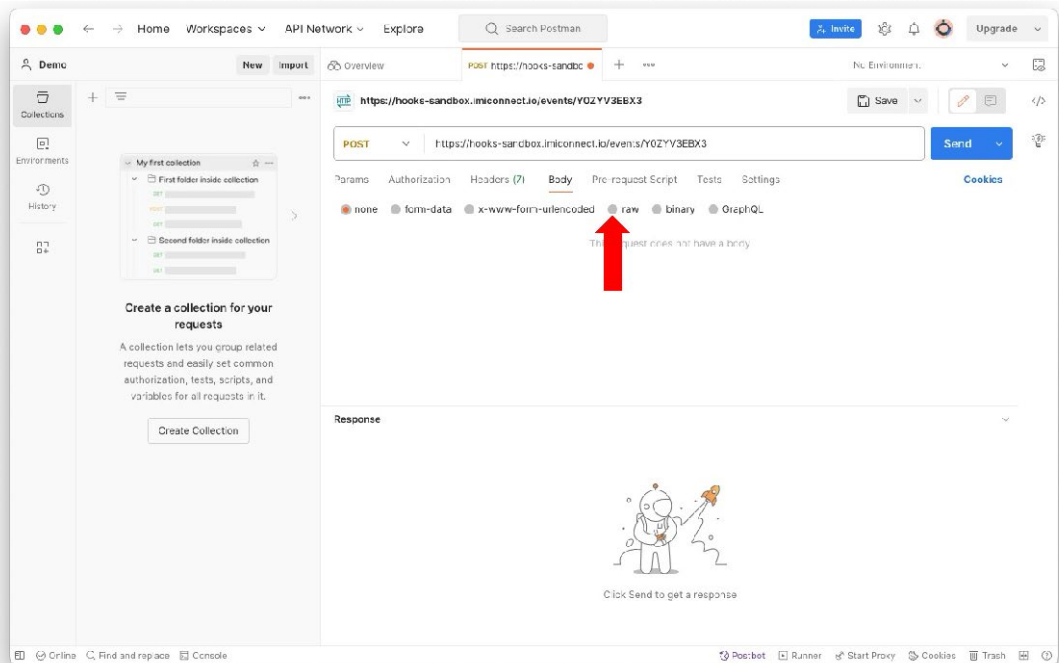
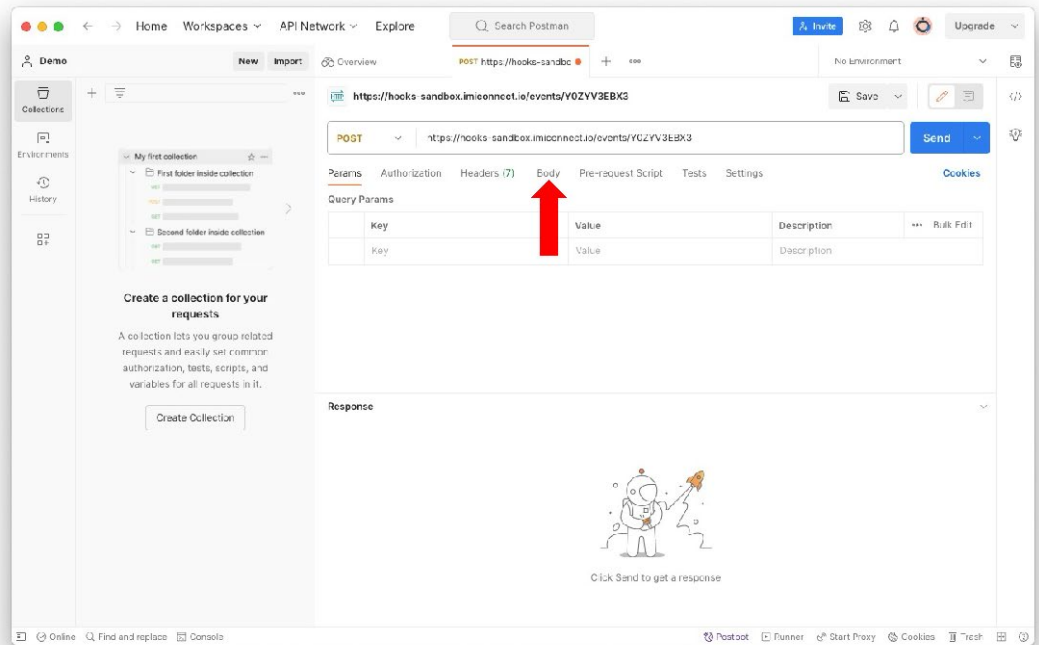
Query Params

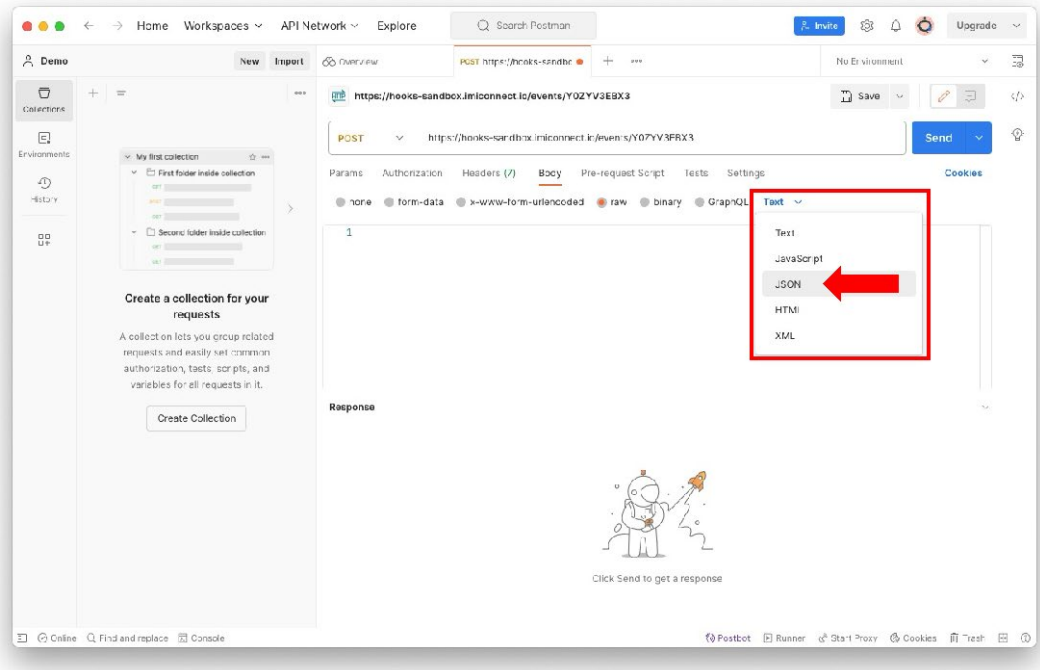
Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Response

Click Send to get a response

Step 43: Click body, select raw, then select JSON from the text type dropdown. JSON is the format in which your webhook expects to receive data.

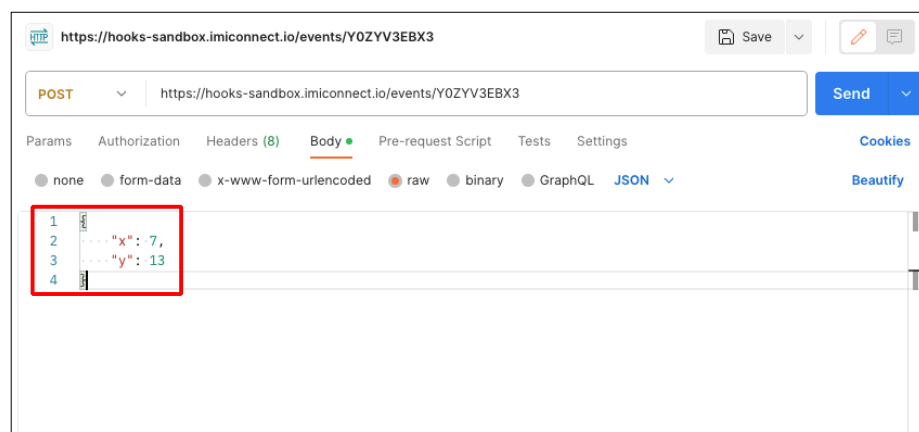




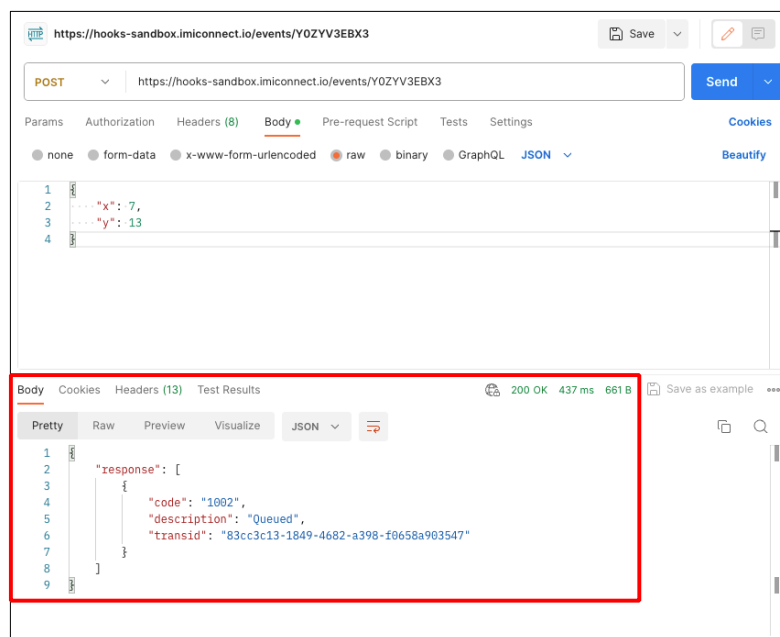
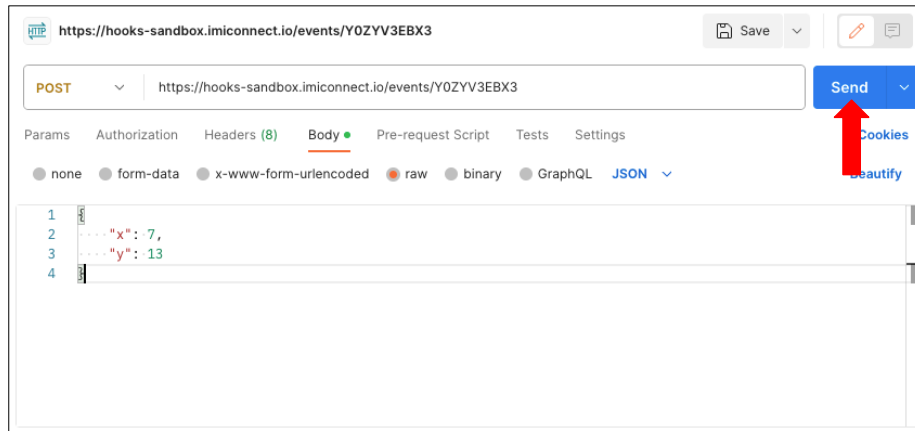
Step 44: In the body input field, type the following JSON. You can use the provided numbers or choose others:

```
{
  "x": 7,
  "y": 13
}
```

This JSON payload simulates the data that your webhook will process.



Step 45: Click the send button in Postman to execute the request. This action sends your test data to the webhook URL, triggering the flow you've just configured.



Step 46: Return to the <https://webhook.site> tab and now you should see a message from your outgoing “HTTP Request” node. If you see JSON data which contains the “sum” keyword and the value is the sum of the values of `x` and `y` like in the screenshot below you have succeeded! If you see a different message in the webhook log, that message should indicate where the error has occurred, pointing you in the correct direction to begin debugging.

The screenshot shows the Webhook.site interface. On the left, a sidebar lists requests, with the selected one being a POST request from 54.188.138.211. The main panel displays the details of this request. The 'Raw Content' tab is highlighted with a red box, showing the following JSON body:

```
{
  "x": 7,
  "y": 13,
  "sum": 20
}
```

The interface also shows headers, files, query strings, and form values for the request.

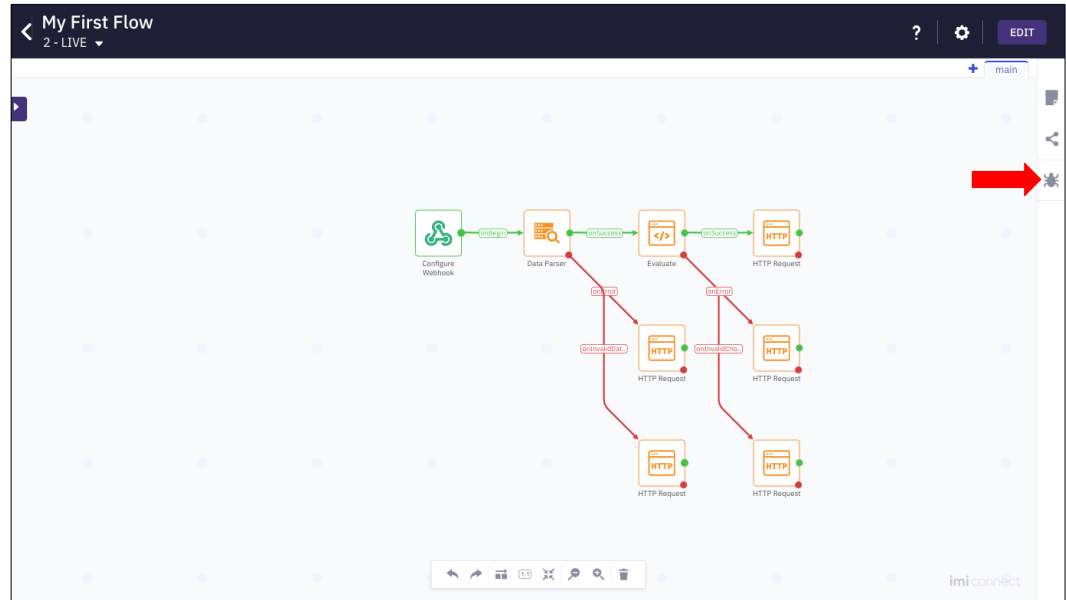
Activity Verification

You have completed this task when you attain these results:

- Send an API request to the Cisco Connect flow webhook URL and receive the same message you sent, but with an additional attribute, the **sum** of **x** and **y** from the body of your request.

DEBUGGING

To debug this flow, you can start by reading the message in the JSON body in the webhook on <https://webhook.site>. In production, this log would be more detailed and make it easier for developers to debug. For a more granular inspection, you can use the flow debugger which empowered developers to inspect the values of variables and the result of each node in the flow. To open the debugger, click the bug icon in the right sidebar while viewing your flow.



TRANSACTION LOGS					DECRYPT LOGS	HELP	ⓘ	×
Nov 29th, 2023 00:00:00 to Nov 29th, 2023 23:59:59					Transaction ID	SEARCH		
TIME STAMP	T1	TRANSACTION ID	T1	INVOKED BY	T1	TIME TAKEN (HH:mm:ss.ms)	T1	LAST NODE
30-11-2023T02:20:22.010Z		13b46906-e5de-4192-820b-9b22c5c619f5_17111_41967		Rule 17111		00:00:00.879		HTTP Request - oncomplete

In the flow debugger, you can inspect the individual transactions that have run. Click the most recent link in the transaction log to begin inspecting that run of the flow. Currently, the log data is encrypted. You need to click “Decrypt Logs” in order to view the raw data values in each stage of the flow.

TRANSACTION LOGS				DECRYPT LOGS	HELP	🔍	✖
Nov 29th, 2023 00:00:00 to Nov 29th, 2023 23:59:59			Transaction ID	<div>SEARCH</div>			
TIME STAMP	⌚ TRANSACTION ID	⌚ INVOKED BY	⌚ TIME TAKEN (H:mm:ss.ms)	⌚ LAST NODE			
30-11-2023T02:20:22.010Z	13b46906-e5de-4192-820b-9b22c5c619f5_17111_41967	Rule 17111	00:00:00.879	HTTP Request - oncomplete			

TRANSACTION LOGS				13b46906-e5de-4192-820b-9b22c5c619f5_17111_41967		DECRYPT LOGS		HELP		X	
Trans ID : 13b46906-e5de-4192-820b-9b22c5c619f5_17111_41967				Time Elapsed (H:M:ss.ms) : 00 : 00 : 00 . 833				Invoked by : Rule 17111			
NODE ID	NODE	OUTCOME	TIME TAKEN(H:M:ss.ms)	DETAILS							
2	Configure Webhook	onBegin	00 : 00 : 00 . 001	Node Trans ID : 5894b586-f1b4-4ff1-b8ba-1c91c612ec35							
3	Data Parser	onSuccess	00 : 00 : 00 . 000	Description : success							
5	Evaluate	onSuccess	00 : 00 : 00 . 001	Encrypted data : GWJPhqX9gddaELbylcePF8LOBKtETXyeZmEC2nWlOWCJ9 ...							
6	HTTP Request	oncomplete	00 : 00 : 00 . 831								

TRANSACTION LOGS					13b46906-e5de-4192-820b-9b22c5c619f5_17111_41967		DECRYPT LOGS		HELP	X
Trans ID : 13b46906-e5de-4192-820b-9b22c5c619f5_17111_41967					Time Elapsed (H:M:ss.ms) : 00 : 00 : 00 . 833			Invoked by : Rule 17111		
NODE ID	NODE	OUTCOME	TIME TAKEN(H:M:ss.ms)	DETAILS						
2	Configure Webhook	onBegin	00 : 00 : 00 . 001	Node Trans ID : 5894b586-f1b4-4ff1-b8ba-1c91c612ec35						
3	Data Parser	onSuccess	00 : 00 : 00 . 000	Description : success						
5	Evaluate	onSuccess	00 : 00 : 00 . 001	Encrypted data : QWJPhqX9gddaELbylcePF8LOBKtETXyeZmEC2nWlOWCJ9 ...						
6	HTTP Request	oncomplete	00 : 00 : 00 . 831							

TRANSACTION LOGS				13b46906-e5de-4192-820b-9b22c5c619f5_17111_41967		HELP	
Trans ID : 13b46906-e5de-4192-820b-9b22c5c619f5_17111_41967				Time Elapsed (H:mm:ss.ms) : 00 : 00 : 00.833		Invoked by : Rule 17111	
NODE ID	NODE	OUTCOME	TIME TAKEN(H:mm:ss.ms)	DETAILS			
2	Configure Webhook	onBegin	00 : 00 : 00.001	Node Trans ID : 5894b586-f1b4-4ff1-b8ba-1c91c612ec35			
3	Data Parser	onSuccess	00 : 00 : 00.000	Description : success			
5	Evaluate	onSuccess	00 : 00 : 00.001	Channel : Custom Event			
6	HTTP Request	oncomplete	00 : 00 : 00.831	Data : <div><pre>{ "inboundWebhook.payload": "in \"x\" 7,in \"y\" 13\n", "inboundWebhook.y": "13", "inboundWebhook.x": "7", "inboundWebhook.transId": "13b46906-e5de-4192-820b-9b22c5c619f5", "inboundWebhook.timestamp": "2023-11-30T02:20:20.863Z" }</pre></div>			

If any node had an error or failure, the “Outcome” column would indicate what the result of the execution was for that node. You can click that line item in the log to further inspect the data values at that node. If the output in the log is not detailed enough, you can return to the node, navigate to the Transaction Actions tab, and configure all available data to be written to the log on entering and exiting that node. By combining these approaches, you should be able to determine where your flow is failing, which should give you a good start on deciding what to do next to attempt to fix the error.