PRD

## **v1. PRD Information**

* **Feature Name**: API Blocks in Journeys
* **Owner**: [Balaji Vanjinathan](mailto:balaji.vanjinathan@capillarytech.com); [Prajwal Patil](mailto:prajwal.patil@capillarytech.com)
* **Contributors**: [Prasenjit Saha](mailto:prasenjit.saha@capillarytech.com), [Prashu Chaudhary](mailto:prashu.chaudhary@capillarytech.com), [Priya Pai](mailto:priya.pai@capillarytech.com), [Amogh Rane](mailto:amogh.rane@capillarytech.com)
* **Approvers:** <Product Leads>, <Engineering Leads>, <Design Leads>
* **Version History**:

| **Version** | **Creation Date** | **Owner** | **Contributors** | **Changelog** | **Signed off by** |
| --- | --- | --- | --- | --- | --- |
| v1 | Mar 3, 2025 | [Balaji Vanjinathan](mailto:balaji.vanjinathan@capillarytech.com)[Prajwal Patil](mailto:prajwal.patil@capillarytech.com) | Person  Person  Person | First draft of PRD | Person - Date  Person - Date  Person - Date |

## **2. Overview**

### **2.1 Objective**

The objective of introducing API blocks in journeys is to expand their capabilities to integrate with any system using an API endpoint, enabling real-time data exchange and actions during the customer journey lifecycle.

### **2.2 Background**

API blocks help make journeys extensible and provides value to brands where they might want to call external APIs to take certain actions

* **GET calls:** Ability to call an external offer engine, trigger a push notification/email that displays the offer to the customer.
* **POST/PUT calls**: Ability to call a third party messaging service to trigger messages or update external systems with certain data-points

### **2.3 Use Cases**

1. Passing data to other third-party systems
2. Sending messages to your customers via channels not directly supported by Braze
3. Retrieving real-time data such as offers that can be used in creatives
4. Retrieving inventory listings to use in creatives for offers
5. **Trigger Actions in External Systems**Enable real-time communication between the journey and other platforms in our ecosystem to drive timely, personalised experiences.
6. **Real-Time Reporting & Tracking**Send journey milestones and actions to our analytics systems to allow live monitoring of campaign performance and customer progression.
7. **Cross-Platform Orchestration**Coordinate and trigger actions across multiple systems in sync with journey events to ensure a consistent customer experience across channels.

## **3. Scope**

### **3.1 In-Scope**

* HTTP Methods: POST, PUT
* Request Configuration
* Dynamic Data Handling
* Response Handling
* Rate Limiting & Throttling

### **3.2 Out of Scope**

* Bulk API operations
* DELETE calls
* Complex data transformations requiring custom coding
* Direct database connections

## **4. Product Requirements**

### **4.1 Epics**

**Epic 1**: Request Configuration

* **Description**: Enable marketing managers to configure the API request details including method, URL, headers, and body.

**Epic 2**: Dynamic Data Handling

* **Description**: Enable the use of journey context variables and customer profile data as dynamic inputs for API requests.

**Epic 3**: Response Handling

* **Description**: Process API responses and make them available for subsequent journey blocks.

**Epic 4**: Rate Limiting & Throttling

* **Description**: Implement rate limiting and throttling mechanisms to prevent API abuse and ensure stable performance.

### **4.2 User Stories**

#### **Epic 1: Request Configuration**

* **User Story 1.1**: As a marketing manager, I want to configure GET calls, so that I can retrieve information from an external system and use it in my journeys.
  + **Acceptance Criteria**:

**Scenario: Creating a new API block with GET method**

Given I am in the journey builder

When I add a new API block to my journey

And I select "GET" as the HTTP method

And I enter a valid API endpoint URL

And I add required headers to the request

Then I should be able to save the API block configuration

And the API block should appear in my journey workflow

**Scenario: Adding dynamic parameters to GET URL**

Given I am configuring a GET API block

When I click on the URL parameter section

And I select a journey variable to insert

Then the variable should appear in the URL with proper syntax

And the preview should show the dynamic URL structure

**Scenario: Testing a GET API configuration**

Given I have configured a GET API block

When I click on "Test Request"

Then the system should make a sample API call

And display the response data

And highlight any errors if the request fails

* **User Story 1.2**: As a marketing manager, I want to configure POST/PUT calls, so that I can send information to an external system during a journey.
  + **Acceptance Criteria**:

**Scenario: Creating a POST API block**

Given I am in the journey builder

When I add a new API block to my journey

And I select "POST" as the HTTP method

And I enter a valid API endpoint URL

And I configure the request body

Then I should be able to save the API block configuration

**Scenario: Building JSON request body**

Given I am configuring a POST API block

When I select "JSON" as the body format

And I use the JSON editor to create my request body

And I add journey variables to the JSON structure

Then the system should validate the JSON format

And display a formatted preview of the request body

**Scenario: Building form data request body**

Given I am configuring a PUT API block

When I select "Form Data" as the body format

And I add key-value pairs to the form

Then the system should format the request as form data

And display a preview of the form submission

* **User Story 1.3**: As a marketing manager, I want to configure Authorization for my API blocks, so that I can securely access protected API endpoints.
  + **Acceptance Criteria**:

**Scenario: Configuring Basic Authentication**

Given I am configuring an API block

When I select "Basic Auth" as the authentication method

And I enter a valid username and password

Then the system should encrypt the credentials

And add the appropriate Authorization header to requests

**Scenario: Configuring API Key Authentication**

Given I am configuring an API block

When I select "API Key" as the authentication method

And I specify whether the key should be in header or query parameter

And I enter the key name and value

Then the system should add the API key to all requests

And mask the key value in the UI after saving

**Scenario: Configuring OAuth 2.0 Authentication**

Given I am configuring an API block

When I select "OAuth 2.0" as the authentication method

And I enter the client ID and client secret

And I specify the token URL

Then the system should handle OAuth token acquisition

And automatically refresh expired tokens

And use the token for all API requests

#### **Epic 2: Dynamic Data Handling**

* **User Story 2.1**: As a marketing manager, I want to use journey context variables in my API requests, so that I can make dynamic, personalized API calls.
  + **Acceptance Criteria**:

**Scenario: Adding journey variables to URL parameters**

Given I am configuring an API block URL

When I click on the variable selector

And I select a journey variable from the list

Then the variable should be inserted into the URL with the syntax {{variable\_name}}

And the UI should highlight the variable in a different color

**Scenario: Adding journey variables to request headers**

Given I am configuring API request headers

When I add a new header or edit an existing one

And I select a journey variable for the header value

Then the variable should be inserted with proper syntax

And the header preview should show the variable placeholder

**Scenario: Adding journey variables to request body**

Given I am configuring a request body

When I click on the variable selector while editing the body

And I select a journey variable

Then the variable should be inserted at the cursor position

And the preview should show how the variable will be rendered

**Scenario: Validating journey variables**

Given I have added journey variables to my API request

When I save the configuration

Then the system should validate that all variables exist in the journey context

And warn me about any unavailable variables

* **User Story 2.2**: As a marketing manager, I want to use customer profile attributes in my API requests, so that I can personalize API calls based on customer data.
  + **Acceptance Criteria**:

**Scenario: Selecting customer attributes**

Given I am configuring an API block

When I open the variable selector

And I navigate to the "Customer Profile" section

Then I should see a list of available customer attributes

And be able to select attributes to use in my request

**Scenario: Using customer attributes in URL parameters**

Given I am configuring the API endpoint URL

When I add a URL parameter

And select a customer attribute for the parameter value

Then the attribute should be inserted with the syntax {{profile.attribute\_name}}

And the UI should highlight it as a customer attribute

**Scenario: Applying transformations to customer attributes**

Given I have selected a customer attribute

When I click on the transformation options

And select a transformation type (uppercase, lowercase, encode)

Then the attribute syntax should include the transformation

And the preview should show an example of the transformed value

**Scenario: Handling missing customer attributes**

Given I have configured an API block with customer attributes

When I configure the fallback behavior for missing attributes

And select "Use default value" as the fallback strategy

And provide a default value

Then the system should use the default when the attribute is missing

And log the fallback action in the journey history

#### **Epic 3: Response Handling**

* **User Story 3.1**: As a marketing manager, I want to extract data from API responses, so that I can use the information in subsequent journey blocks.
  + **Acceptance Criteria**:

**Scenario: Mapping simple response fields to journey variables**

Given I am configuring the response handling section of an API block

When I select a field from the response structure

And I create a new journey variable to store the value

Then the system should create the mapping between the response field and journey variable

And show the mapping in the response handling interface

**Scenario: Handling nested JSON structures**

Given I am viewing a sample JSON response with nested objects

When I use the dot notation to specify a nested field path

And map it to a journey variable

Then the system should correctly extract the nested value at runtime

And store it in the specified journey variable

**Scenario: Previewing response mapping with sample data**

Given I have configured response field mappings

When I click on "Test with Sample Data"

And the system receives a sample response

Then I should see how each field is extracted

And what values would be stored in journey variables

**Scenario: Handling array responses**

Given I receive an API response with an array of items

When I specify an array index in my field mapping

Or I choose to map the entire array to a variable

Then the system should correctly extract either the specific element

Or store the complete array for later use

* **User Story 3.2**: As a marketing manager, I want to define branching logic based on API responses, so that I can create different journey paths based on API results.
  + **Acceptance Criteria**:

**Scenario: Creating status code based branches**

Given I have configured an API block

When I add a branch condition based on status code

And I select status code 200 for the condition

And I connect this branch to a success path in the journey

Then the journey should follow this path when the API returns a 200 status

**Scenario: Creating content-based branches**

Given I have configured an API block

When I add a branch condition based on response content

And I specify a JSON path to evaluate

And I set a comparison operator and value

And I connect this branch to a specific path in the journey

Then the journey should follow this path when the condition evaluates to true

**Scenario: Defining a default path**

Given I have configured branches for specific API responses

When I add a default branch

And connect it to a fallback path in the journey

Then the journey should follow this path when no other conditions match

**Scenario: Handling API timeouts**

Given I have configured an API block

When I add a specific branch for timeout errors

And set the timeout threshold to 10 seconds

And connect this branch to an error handling path

Then the journey should follow this path when the API call times out

And the timeout event should be logged

**Scenario: Handling connection errors**

Given I have configured an API block

When I add a branch for connection errors

And connect it to an error notification block

Then the journey should follow this path when the API is unreachable

And record the connection failure details

#### **Epic 4: Rate Limiting & Throttling**

* **User Story 4.1**: As a marketing manager, I want to configure API call limits, so that I can prevent overloading external systems.
  + **Acceptance Criteria**:

**Scenario: Setting maximum requests per time period**

Given I am configuring an API block's rate limiting settings

When I set the maximum requests to 60

And I select "per minute" as the time period

Then the system should enforce this limit across all instances of the journey

And queue additional requests when the limit is reached

**Scenario: Configuring queue behavior**

Given I have set rate limits for my API block

When I set the queue behavior to "Wait in queue for up to 5 minutes"

Then requests beyond the rate limit should be placed in a queue

And executed when capacity becomes available

And abandoned if they exceed the 5 minute queue time

**Scenario: Setting up rate limit notifications**

Given I have configured rate limits

When I enable rate limit notifications

And set the threshold to 80%

And provide notification recipients

Then the system should send an alert when usage reaches 80% of the limit

And include usage statistics in the notification

**Scenario: Viewing API usage metrics**

Given I have an active journey with API blocks

When I navigate to the journey analytics dashboard

Then I should see metrics on API calls made

And rate limit utilization over time

And any rate limit breaches or queuing events

* **User Story 4.2**: As a marketing manager, I want to configure retry logic, so that I can handle temporary API failures.
  + **Acceptance Criteria**:

**Scenario: Enabling automatic retries**

Given I am configuring an API block

When I enable the automatic retry option

And set the number of retry attempts to 3

Then the system should automatically retry failed API calls

And only consider the request failed after all retries are exhausted

**Scenario: Setting retry interval strategy**

Given I have enabled automatic retries

When I select "Exponential backoff" as the retry strategy

And set the initial interval to 5 seconds

Then the system should wait 5 seconds before the first retry

And double the wait time for each subsequent retry

**Scenario: Configuring retry conditions**

Given I have enabled automatic retries

When I specify to only retry on status codes 429 and 503

Then the system should only initiate retries for these specific error codes

And immediately fail for other error codes

**Scenario: Monitoring retry attempts**

Given a journey with API blocks is running

When an API call triggers retry attempts

Then each retry should be recorded in the journey logs

And the final outcome should be clearly indicated

And retry metrics should be available in the analytics dashboard

## **5. UI/UX Design**

* **https://www.figma.com/design/clmHWxDsK3nGNDaWsa5iVy/Webhook-block-%7C-Journeys?node-id=91-76327&t=KcC2kKoJNn74PQAr-0**

## **6. Technical Requirements**

### **6.1 Fair Usage**

* Milestone configuration limits (e.g., maximum number of milestones per brand).

### **6.2 Security & Compliance**

* **Permissions**: Ensure only authorized users can create or edit milestones.
* **API Access**: Restrict API access for milestone data to specific user groups.
* **Risk**: Milestones may lead to increased database load due to tracking requirements.
  + **Mitigation**: Implement usage limits and conduct load testing before release.

## **7. Dependencies**

* **Bulk Milestones Creation Using File Upload**: Dependency on the **Apps Infra Team** to enable bulk creation through file upload functionality.
* **Reporting on Milestones**: Dependency on the **Insights Team** to develop and support reporting features for milestone tracking and analytics.

## **8. Release Plan**

### **8.1 Timelines**

| **Phase** | **Epics** | **Release Date** |
| --- | --- | --- |
| Phase 1 | Epic 1  Epic 2 | 5th Jan 2025 |
| Phase 2 | Epic 3 | 20nd Feb 2025 |
| Phase 3 | Epic 4 | 18th Mar 2025 |

### **8.2 Release Tasks**

**Phase 1**

* Release:
  + Beta release
  + Hypercare period
  + Final release
* Documentation:
  + Product documentation draft
  + Release notes draft
  + API doc confluence link
  + Status
* Academy:
  + Product training format and date
  + Quizzes
  + Status
* Product Marketing:
  + GTM kit draft
  + Status
* Pricing:
  + Pricing details
  + Status
* Customer onboarding:
  + Customer onboarded
  + Feedback/tickets/challenges
  + Status -

**9. Metrics**

Metrics that need to be tracked:

* Adoption metrics:
  + Total no of brands using milestones by Brand, verticals and regions
* Usage and Usability metrics:
  + No of milestones created in last 6 months
  + Total no active milestones
  + Average time to create end-to-end milestone based loyalty promotions
  + No of tickets created around Milestones
    - As designed
    - Tasks
    - Enhancements
* Performance and Errors metrics:
  + Time from milestones achieved to rewards issued
  + No of errors in milestone tracking and rewards issual
  + Milestone UI page loading time
  + Average time to bulk upload milestone target values
  + No of bug tickets
* Documentation metrics
  + Top doc bot queries related to milestones
  + Per cent queries answered
* Academy metrics
  + Milestone loyalty course completions
* Marketing metrics
  + No of prospects pitched
  + No of views of GTM kit

## **10. Appendix**

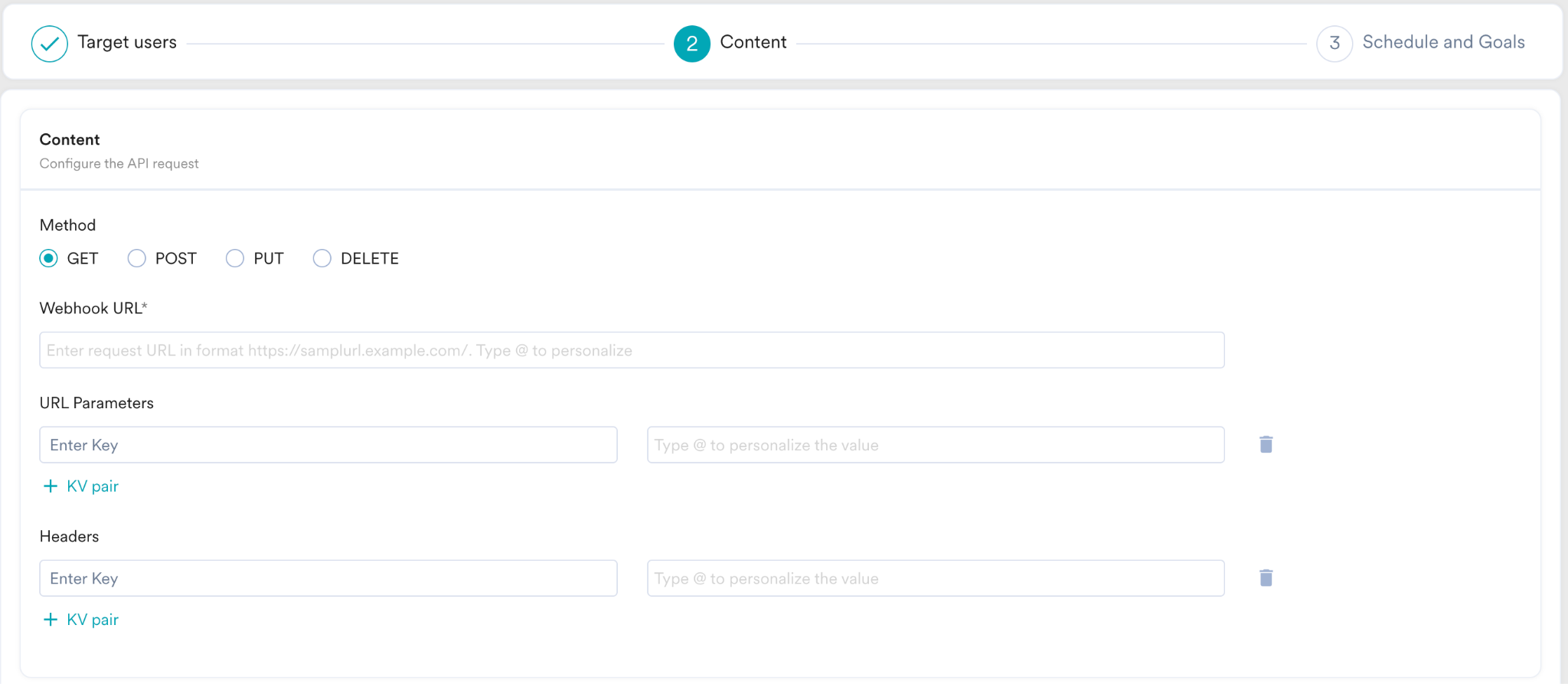
### **10.1 Research**

* **Competitive Analysis**:

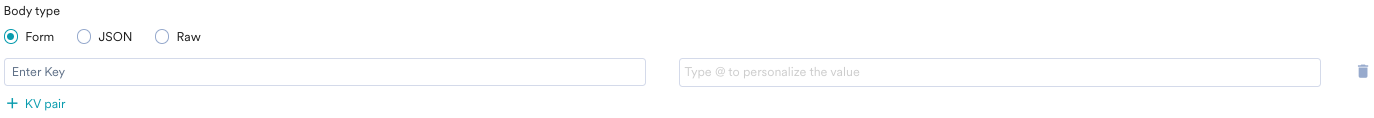
MoEngage:

Currently

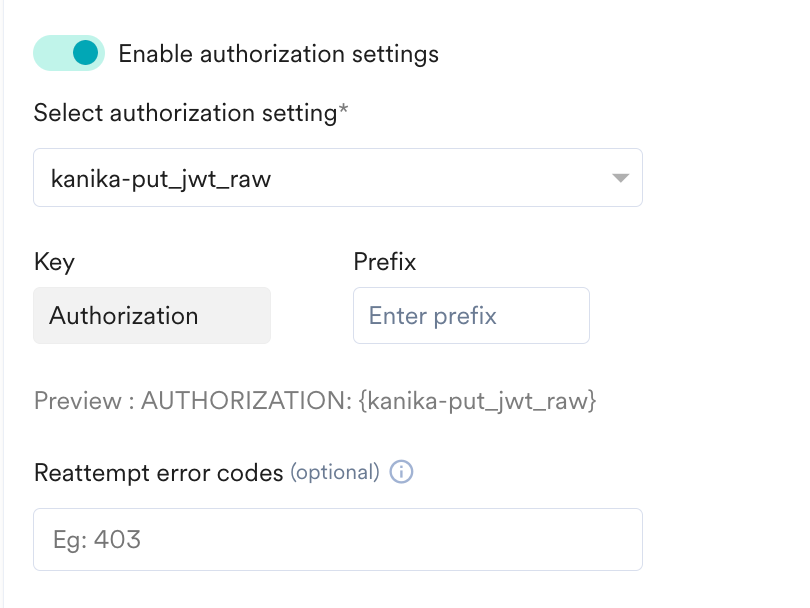
**GET Method:**



**PUT, POST, DELETE Method:**



**Authorization:**



### **10.2 Key Decisions**

* Decision to focus milestone rewards on transactional KPIs in the first release.

### **10.3 Open Questions**

* Should milestone notifications be customizable by brand?

Tech Doc

Doc

# Documentation

Doc link: <https://docs.capillarytech.com/docs/action-building-block#/webhook>

# **Webhook**

The webhook block allows your Journeys to communicate and send real-time information to external, third-party systems. This allows you to automate tasks in external systems like your customer relationship management (CRM) system , support desk, or advertising platforms. When a customer reaches this point in a journey, it automatically sends data to a webhook URL you specify.

For example, when a customer gives a low satisfaction score in a survey, a Journey can automatically start. A Webhook block within that journey can instantly connect to your support system, like Zendesk, and create a ticket. The webhook sends the customer's details and their poor feedback directly into the ticket, alerting your support team to resolve the issue immediately.

## **Prerequisites**

Before configuring the webhook block, obtain the following from the external system you want to connect to.

* Endpoint URL: The specific URL that will receive the data.
* HTTP Method: The required method for the webhook.
* Authentication Details: The credentials needed to securely access the webhook.
* JSON Data Structure: The JSON payload for the webhook.

## **Webhook request structure**

### **Query parameters**

Query parameters are simple key-value pairs appended to the end of a URL. They provide visible, top-level context about the request.

Use query parameters to send simple, non-sensitive metadata. For example, parameters to track the source of a request (such as source=journey-automation), specify an API version, or provide a simple identifier for filtering.

### **Headers**

Headers provide technical instructions and metadata for the receiving server on how to process an incoming request.

Use headers for authentication and for specifying the data format. For example, use the Authorization header to securely pass an API key or bearer token. Use the Content-Type header with a value like application/json to inform the server that the payload is structured as JSON.

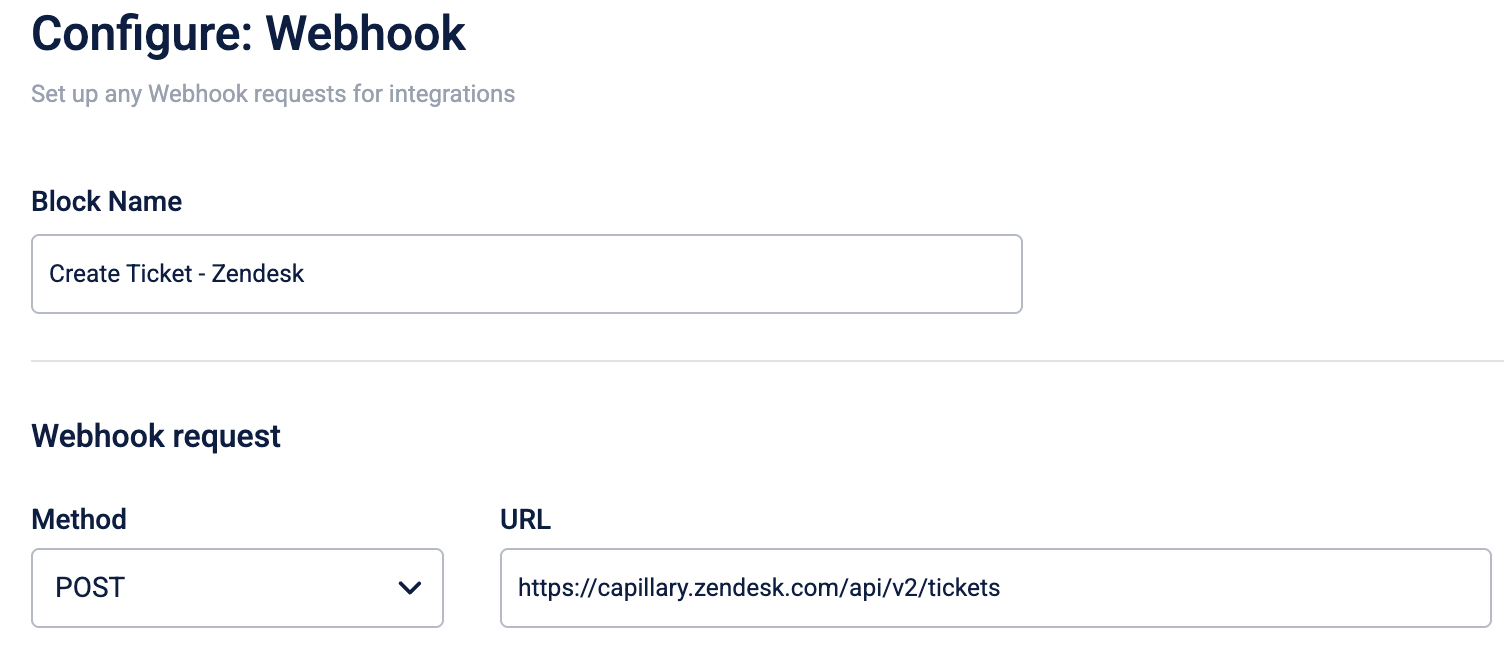
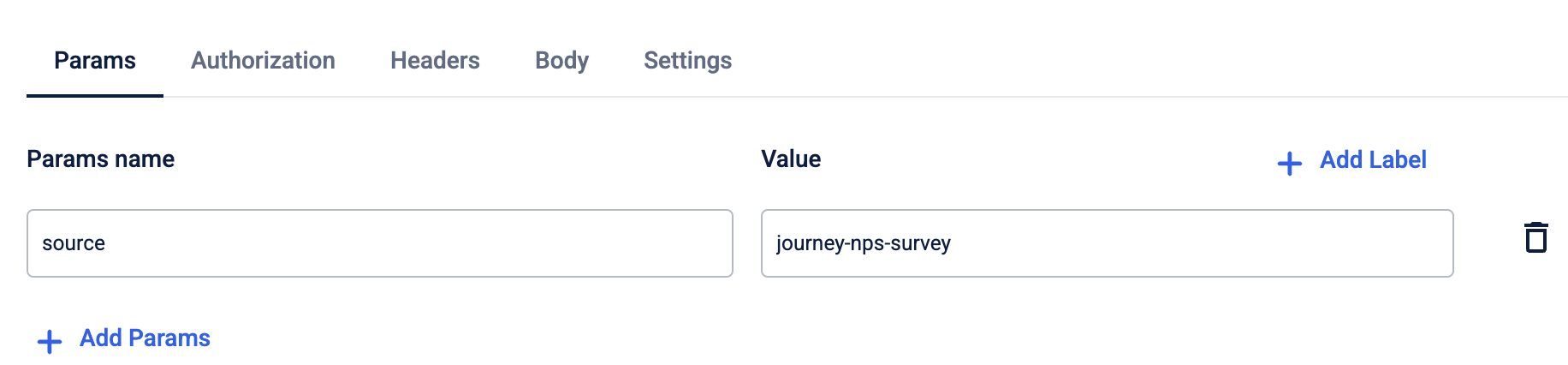
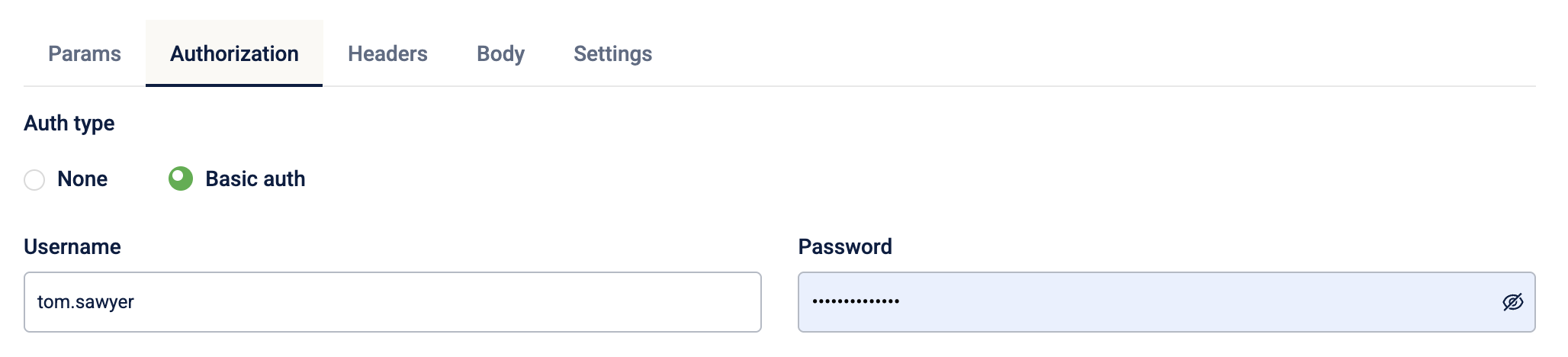
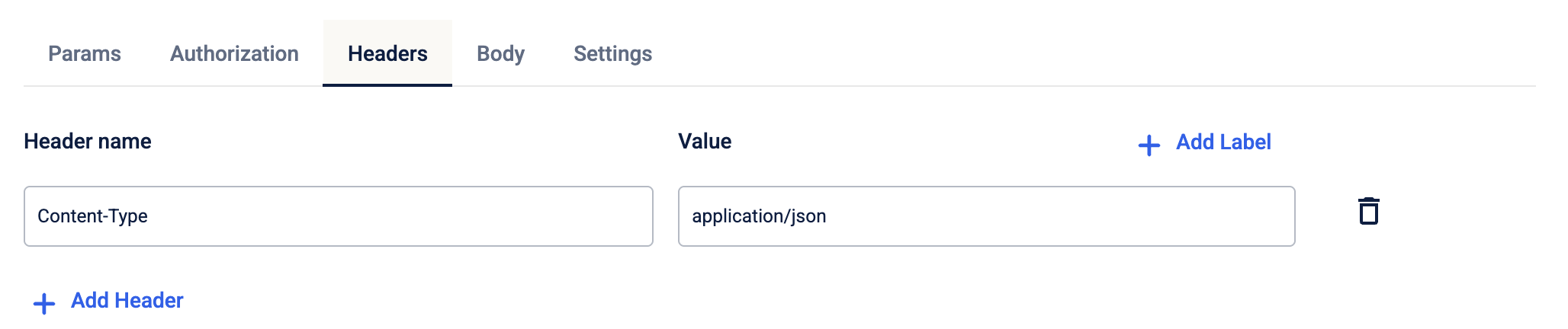
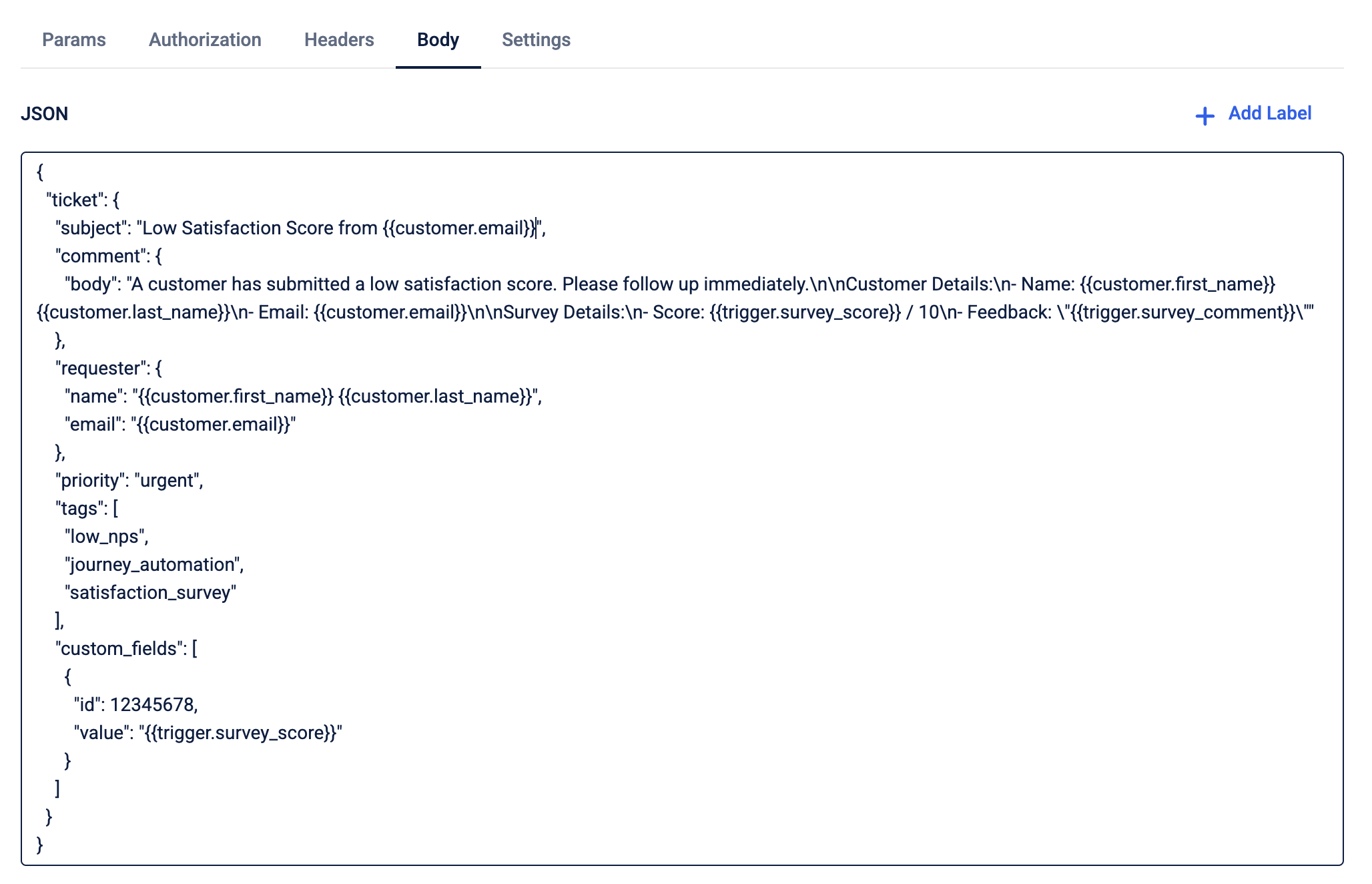
### **Payload (request body)**

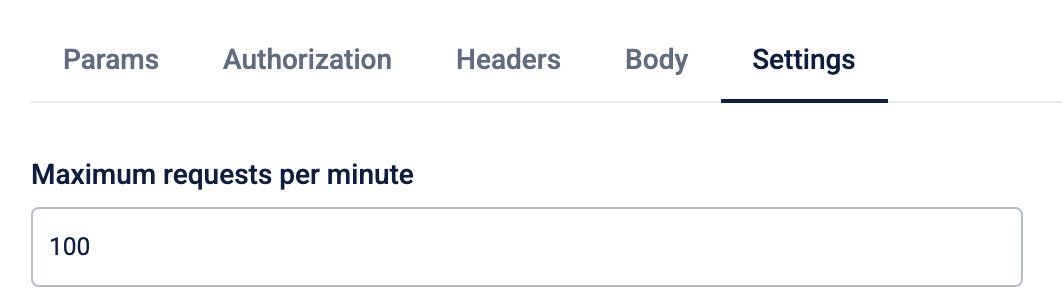
The payload, also known as the request body, is the main content of the message. It contains the detailed, structured information for the external system to process, especially when using POST or PUT methods.

Use the payload to send complex, multi-field, or sensitive data. For example, when creating a support ticket, the customer's email, the ticket subject, and the comment are structured within the JSON payload. Because the payload is not visible in the URL, it is the secure and appropriate place for detailed customer or event information.

## **Configure the webhook block**

To configure the webhook block, follow these steps:

1. In the Block Name box, enter a descriptive name for the webhook block. For example: Create Ticket - Zendesk or Update Lead - Salesforce.
2. From the Method list, choose the request type. The available types are [POST](https://www.postman.com/api-evangelist/design/request/ok1rqps/post-request) and [PUT](https://www.postman.com/api-evangelist/design/documentation/4x6fo24/postman-echo?entity=request-a571dbe1-a5d8-48bb-9153-7ff05b3ab8c9).
3. In the URL box, enter the endpoint URL.  
   
4. On the Params tab, enter the query parameters. Query parameters are optional key-value pairs added to the URL for tracking or filtering. You can add up to 10 parameters. You can add a label, such as "Customer Email", as a variable value. For more information, refer to the documentation on [Labels](https://docs.capillarytech.com/docs/labels#/).  
   
5. On the Authorization tab, select the authentication type. For more information, refer to the section on [Configuring authorization](https://docs.capillarytech.com/docs/action-building-block#configuring-authorization).  
   
6. On the Headers tab, in the Header name and Value boxes, enter the headers and values. A common header is Content-Type with the value application/json. You can add up to 10 headers. You can add a label, such as "Organisation ID", as a variable value. For more information, refer to the documentation on [Labels](https://docs.capillarytech.com/docs/labels#/).  
   
7. In the JSON box, enter the JSON request body. You can add a label, such as "First Name", as a variable value. For more information, refer to the documentation on [Labels](https://docs.capillarytech.com/docs/labels#/).  
   
8. On the Settings tab, enter the rate limit (requests per minute).



🚧

### Note

The webhook block cannot be edited once the journey has been sent for approval. To make changes, reject the approval request and make the changes.

1. Select Done to save your changes.

🚧

### Note

The webhook block cannot be edited once the journey has been sent for approval. To make changes, reject the approval request and make the changes.

## **Configuring authorization**

The Authorization tab allows you to specify the credentials the webhook will use. The correct method depends on the requirements of the third-party service. The following types are supported:

* None: No authorisation.
* Basic Auth: A simple username and password combination.
* API Key: A unique key that you pass in either the headers or parameters. For example, a header named X-API-Key with your key as the value.

## **Use Cases**

### **1. Add a High-Intent Lead to Your CRM**

Scenario: A user visits the pricing page and then downloads a "Buyer's Guide." This high-intent behaviour triggers a journey. You want to create a new lead in your CRM for immediate sales follow-up.

Configuration:

* Method: POST
* URL: https://your-company.crm.com/services/data/Lead/
* JSON body:
* JSON

{

"Company": "{{profile.company\_name}}",

"LastName": "{{profile.last\_name}}",

"FirstName": "{{profile.first\_name}}",

"Email": "{{profile.email}}",

"LeadSource": "Journey: Buyer's Guide Download"

* }

Outcome: A new lead is instantly created in the CRM with the customer's details, attributed correctly to your journey.

### **2. Send a Real-Time Alert to an Internal Slack Channel**

Scenario: A customer makes a purchase of ₹50,000. To celebrate the win and ensure premium order fulfilment, you want to post an alert to your internal #sales-wins Slack channel.

Configuration:

* Method: POST
* URL: https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXXXXXXXXXXXXXX (Your Slack Incoming Webhook URL)
* JSON body:
* JSON

{

"text": "🎉 Big Sale! Customer {{profile.email}} just made a purchase of ₹{{trigger.purchase\_amount}}!"

* }

Outcome: Your team sees a real-time notification in Slack the moment the high-value purchase is made.

### **3. Update Customer Preferences in Your CRM**

Scenario: A customer is in a re-engagement journey and clicks a button to update their email frequency preference to "weekly." You need to update their profile in your CRM to reflect this change.

Configuration:

* Method: PUT
* URL: https://your-company.crm.com/members/{memberId}updateEmail
* JSON body:
* JSON

{

"tags": {

"frequency": "weekly"

},

"status\_if\_new": "subscribed"

* }

Outcome: The customer's profile in the CRM is updated, ensuring they receive emails at their preferred frequency.

### **4. Sync a Customer's Tier Status to Your Support Desk**

Scenario: Based on loyalty points accumulated, a customer is promoted to "Gold Tier" within a journey. To ensure they receive priority support, you need to update their existing user profile in your Zendesk support desk.

Configuration:

* Method: PUT
* URL: https://your-domain.zendesk.com/api/v2/users/{zendesk\_user\_id}
* JSON body:
* JSON

{

"user": {

"user\_fields": {

"loyalty\_tier": "Gold"

}

}

* }

Outcome: The customer's user record in Zendesk is updated with their new tier, and any support agent who views their profile will see they are a Gold Tier member.

## **FAQs**

1. Which HTTP methods are supported? The webhook block supports POST and PUT methods.
2. What authentication methods are supported? The webhook block supports basic authorization
3. How many headers or params can I add? You can add up to 10 headers and 10 params per request.
4. What happens if I add unsupported tags in the request? The request returns 400, and the UI shows an error stating that the tags are not supported.
5. Does editing a webhook block in a live journey create a new version? Yes. Editing a block in a live journey creates a new version that goes into the approval flow.
6. Can I edit the configuration once the journey has been sent for approval? The webhook block cannot be edited once the journey has been sent for approval. To make changes, reject the approval request and make the changes.