Remote by Journey

Webhook Integration Guide

Overview

When events occur on the Remote by Journey platform, a message can be configured to be sent to a provided URL. This document outlines how to configure and receive that message.

In the account settings of your account, you're able to enter a webhook url to receive event messages. When an event occurs, a `POST` request will be made to that url with event details structured as JSON in the request body.

Encryption

Also in the account settings is an area to input a 4096-bit RSA public key. If a key is provided, the JSON payload will be sent encrypted to the webhook. The following commands can be used to generate the key in the correct format. If at any time you need to roll the keys, simply generate a new one and enter the public key into the account settings. The change will immediately take effect.

```
openssl genrsa -des3 -out private.pem 4096
openssl rsa -in private.pem -outform PEM -pubout -out public.pem
cat public.pem
```

Test webhook requests can be sent from the admin panel to make integration a smooth process. They will have the payload below and will be encrypted if a public key is saved to the account.

Payload structure

This is an example of the structure of the body of a webhook request.

```
{
  // A unique identifier for this event.
  "id": "5fd6842cb2e36360c6cdda4a",

  // The type of event that initiated the webhook request.
  //   Event types are listed below.
  "type": "test_event",

  // The IP address of the client initiating the event
  "ip_address": "127.0.0.1",

  // A location may (but may not) be specified for the event, details below.
  //   The location precision is not a fixed number of digits.
  "location": {
    "latitude": 35.0395,
    "longitude": 135.7285
  },

  // This is the timestamp when the event was recorded.
  //   Rely on this timestamp rather than the time you received the webhook.
  "timestamp": "2020-12-13T21:14:20.3167Z",

  // User details for who initiated the event
  "user": {

    // This is Journey's internal identifier for the user.
    "id": "5fd6842cb2e36360c6cdda4b",

    // The unique identifier was provided when a user is added.
    //   Any string, likely an email or internal identifier.
    "unique_id": "user@example.com"
  }
}
```

Event Types

These are the types that can be passed in the `type` field of the webhook payload.

```
enrollment: A user does their initial enrollment.

active_auth: A user actively authenticates when they start a work session. This can be thought of as logging in at the beginning of the day or after lunch.

passive_auth: A user has been passively authenticated while they are in a work session.

session_ended: The user clicked the sign-out button in the web interface. They are unauthenticated and there will be no unable_to_passively_auth event.

new_location: We have detected a significant change in location. A `new_location` event type will always be accompanied by a `location` key with coordinates.

unable_to_passively_auth: If we're unable to authenticate the user within the duration set in account settings, we'll send a webhook with this type.

multiple_faces: Multiple faces were detected.

test_event: This is the event type if you send yourself a test webhook request from the admin panel.
```

Note: new event types will be added as needed.

Miscellaneous notes

Location

The `location` field may or may not be present, depending on the event type:

```
enrollment: if available
active_auth: always
passive_auth: always
session_ended: never
new_location: always
unable_to_passively_auth: never
multiple_faces: if available
```

Multiple face detection

Once the user is initially authenticated and is then in the continuous authentication mode, the app begins to monitor for multiple faces in the camera frame. If multiple faces are detected, one image will be sent to the server to record a "`multiple_faces`" event. The application will generate a maximum of one "`multiple_faces`" event every 30 seconds.

Passive authentications will not occur as long as there are multiple faces detected in the image.

Browser sessions

Once a user authenticates, either by active or by passive authentication, they will have an authenticated session until one of two things happen:

1) The user clicks on the top right sign-out icon. The camera will turn off immediately and they will be returned to the initial authentication screen. This is recorded as a "`session_ended`" event.
2) The time threshold set in the settings under "Passive authentication threshold" expires without a passive authentication. This is recorded as an "`unable_to_passively_auth`" event.

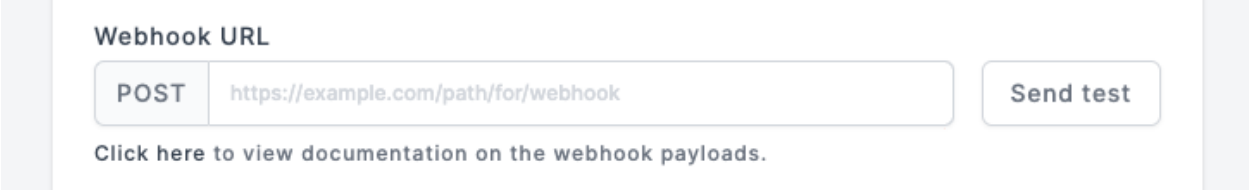
If the user closes the browser tab, but re-opens it before the passive authentication window expires, they will still be authenticated. If they reopen the tab after the deadline is reached, they will be required to perform an active authentication.

IP Addresses

An IP address will be included in the webhook payload only if the event was triggered directly from a browser. "`unable_to_passively_auth`" doesn't have an IP address attached because it's not the result of a user action.

Webhook Setup

To configure webhook notifications, go to the **Account** tab in the navigation bar. Find **Webhook URL** in the form and fill in a complete url. You can send a test webhook to confirm the url is working properly.



Encryption

Optionally, you may add a 4096-bit RSA public key which will be used to encrypt the JSON payload of each webhook.

You can use the following commands to generate a properly formatted key pair:

```
openssl genrsa -des3 -out private.pem 4096
openssl rsa -in private.pem -outform PEM -pubout -out public.pem
cat public.pem
```