

# Debugging RESTful API

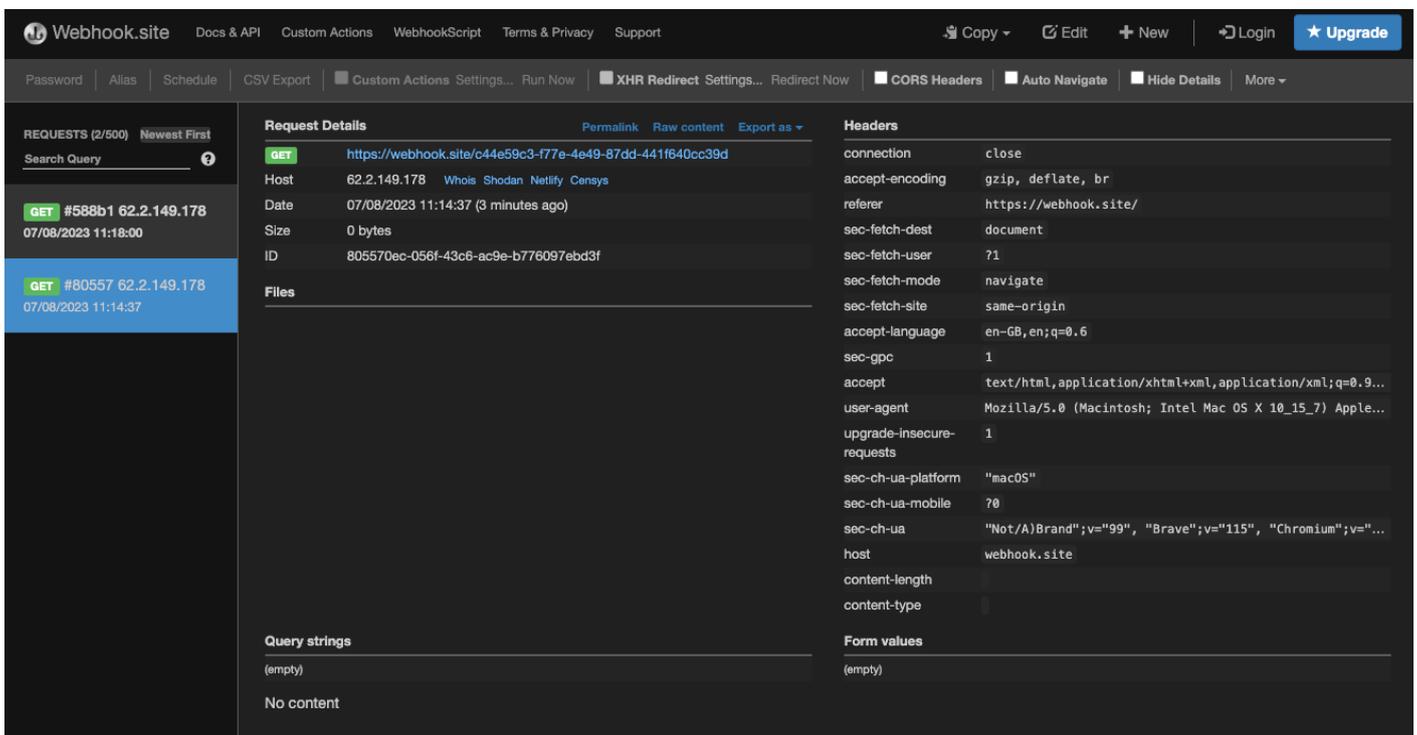
If you need to integrate for example IoT devices, you need maybe to check why the device is not able to connect to the server processing your location data or other data from a LoRaWAN Tracker or any other IoT device.

For this there's a very good service to see what kind of requests the IoT device or a middleware is making during an uplink of the data.

First the Selfhosted tools:

## webhook.site

With [Webhook.site](https://webhook.site), you instantly get a unique, random URL that you can use to test and debug Webhooks and HTTP requests, as well as to create your own workflows using the Custom Actions graphical editor or WebhookScript, a simple scripting language, to transform, validate and process HTTP requests.



The screenshot displays the Webhook.site interface. At the top, there's a navigation bar with 'Webhook.site' and various utility links like 'Docs & API', 'Custom Actions', 'WebhookScript', 'Terms & Privacy', and 'Support'. On the right, there are buttons for 'Copy', 'Edit', 'New', 'Login', and 'Upgrade'. Below the navigation bar, there's a toolbar with options like 'Password', 'Alias', 'Schedule', 'CSV Export', 'Custom Actions Settings...', 'Run Now', 'XHR Redirect Settings...', 'Redirect Now', 'CORS Headers', 'Auto Navigate', 'Hide Details', and 'More'. The main content area is divided into three columns. The left column shows a list of requests, with the most recent one highlighted in blue. The middle column, titled 'Request Details', shows the following information: Method: GET, URL: https://webhook.site/c44e59c3-f77e-4e49-87dd-441f640cc39d, Host: 62.2.149.178, Date: 07/08/2023 11:14:37 (3 minutes ago), Size: 0 bytes, ID: 805570ec-056f-43c6-ac9e-b776097ebd3f. Below this, there are sections for 'Files' (empty), 'Query strings' (empty), and 'No content'. The right column, titled 'Headers', lists various headers and their values, including 'connection: close', 'accept-encoding: gzip, deflate, br', 'referer: https://webhook.site/', 'sec-fetch-dest: document', 'sec-fetch-user: ?1', 'sec-fetch-mode: navigate', 'sec-fetch-site: same-origin', 'accept-language: en-GB,en;q=0.6', 'sec-gpc: 1', 'accept: text/html,application/xhtml+xml,application/xml;q=0.9...', 'user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_15\_7) Apple...', 'upgrade-insecure-requests: 1', 'sec-ch-ua-platform: "macOS"', 'sec-ch-ua-mobile: ?0', 'sec-ch-ua: "Not(A)Brand";v="99", "Brave";v="115", "Chromium";v="...", 'host: webhook.site', 'content-length', and 'content-type'. At the bottom of the headers section, there is a 'Form values' section which is also empty.

Github Development Page: <https://github.com/webhooksite/webhook.site>

Docker Image: <https://hub.docker.com/r/webhooksite/webhook.site>

# Request Baskets

Request Baskets is a web service to collect arbitrary HTTP requests and inspect them via RESTful API or simple web UI.

The screenshot shows the Request Baskets web interface. At the top, there's a header with the title "Request Baskets" and several utility icons (refresh, reload, settings, back, search, delete). Below the header, the main content area is titled "Basket: demo" and shows "Requests: 3 (3)".

The first request is a **[DELETE]** to `/demo/123`, captured at 4:09:56 PM on 8/15/2016. It shows a "Headers" section.

The second request is a **[POST]** to `/demo`, captured at 4:09:23 PM on 8/15/2016. It shows "Headers" and a "Body" section containing a JSON object:

```
{
  "id": 45902,
  "type": "image",
  "size": {
    "x": 1200,
    "y": 800
  }
}
```

The third request is a **[GET]** to `/demo?name=Peter+Pan&value=923831`, captured at 4:07:51 PM on 8/15/2016. It shows "Headers" and a "Query Params" section.

There's also a docker image here: <https://hub.docker.com/r/darklynx/request-baskets>

## Beeceptor

With beeceptor you Build mock APIs in a few seconds, Inspect & Intercept HTTP requests.

With the free plan you can create 50 requests/day, see the pricing here:

<https://beeceptor.com/pricing>

Very interesting is the inspection of HTTP request, check it out: <https://beeceptor.com/docs/inspect-http-request-payloads/>

#endpoint.free.beeceptor.com 2

Rules enabled

<https://endpoint.free.beeceptor.com> → {nowhere}[Mocking Rules \(15\)](#) [Proxy Setup](#)

## Looks Awesome!

The following endpoint is all set up. Use it in your code as base URL and send a request. You can inspect these requests here and build rules to mock responses.

<https://endpoint.free.beeceptor.com>

For example, run the following command in shell/terminal to get started.

```
curl -v -X POST 'https://endpoint.free.beeceptor.com/my/api/path' -H 'Content-Type: application/json' -d '{"data":"Hello Beeceptor"}'
```

(or [click here](#) to simulate in web-browser)

## PostBin

Programmatically Test your API Clients or Webhooks.

<https://postb.in/>

## RequestBin

Inspect webhooks and HTTP requests.

Get a URL to collect HTTP or webhook requests and inspect them in a human-friendly way. Optionally connect APIs, run code and return a custom response on each request.

<https://requestbin.com>

## More Tools

Link	Description
<a href="https://transfer.symbiose.com/download/info/">https://transfer.symbiose.com/download/info/</a>	Info about request headers

Link	Description
<a href="https://httpbin.org/">https://httpbin.org/</a>	Analysis for request and responses

---

Revision #8

Created 19 July 2023 09:08:35 by Peter Baumann

Updated 17 April 2024 14:40:48 by Peter Baumann