

Application Knowledge

During my work with different applications I found some tricks and hints how to work with them. This is the collection I put together with the knowledge about it.

- Collaboration
 - Zimbra LDAP
- Debugging
 - Debugging RESTful API

Collaboration

Collaboration Apps

Zimbra LDAP

Connect to Zimbra LDAP Server

```
zimbra @ mail: ~ $ zmlocalconfig -s zimbra_ldap_password ldap_master_url
```

the result will be something like

```
zimbra_ldap_password = thepassword  
ldap_master_url = ldap://server:389
```

Connect to the LDAP Server with

```
LDAP Server address: ldap://server  
LDAP Server port: 389  
Binddn LDAP: uid=zimbra,cn=admins,cn=zimbra  
LDAP password: thepassword  
LDAP base: dc=yourdomain_zimbra,dc=com
```

Example LDAP Filter for Zimbra

```
(& (uid =% s)) for authentication user without @domain  
(& (mail =% s)) for the authentication with user@domain.com
```

Debugging

The different debugging technics also in the cloud

Debugging RESTful API

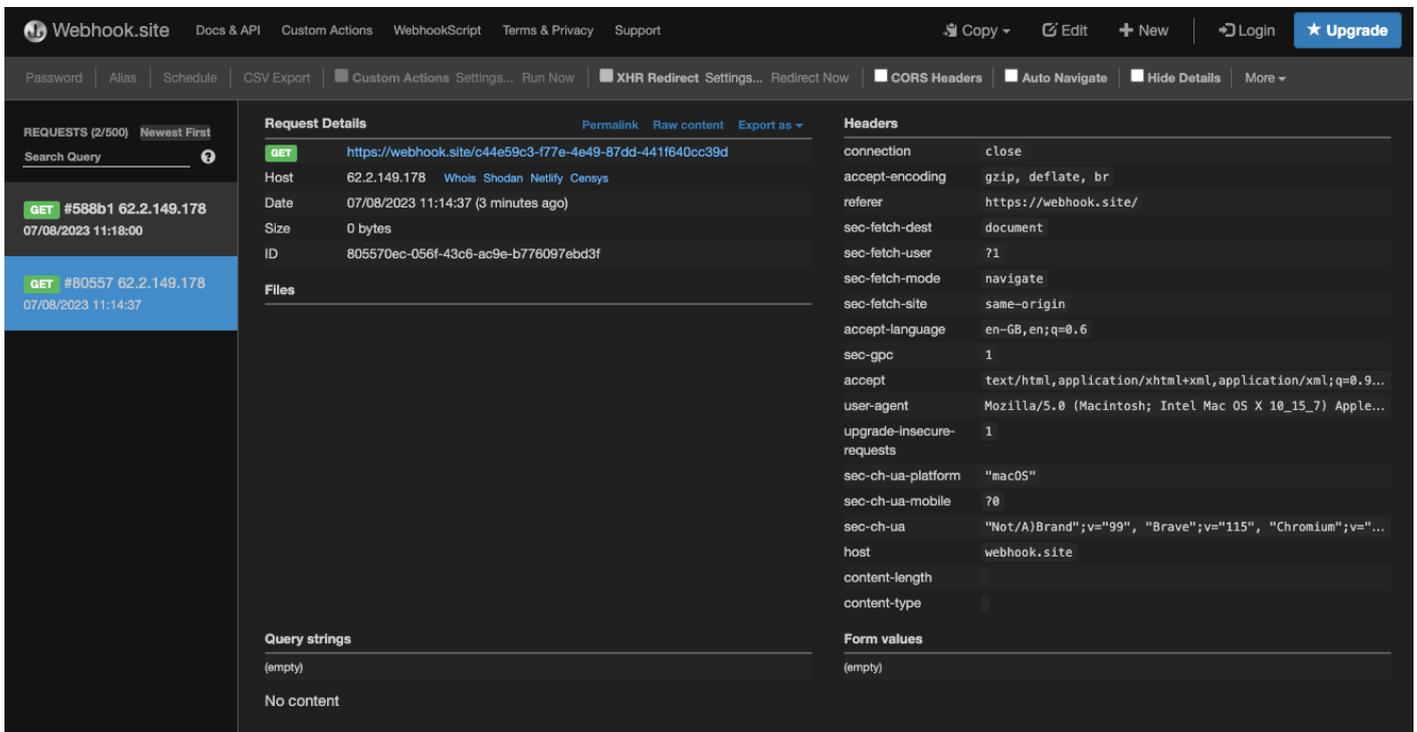
If you need to integrate for example IoT devices, you need maybe to check why the device is not able to connect to the server processing your location data or other data from a LoRaWAN Tracker or any other IoT device.

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 links for 'Docs & API', 'Custom Actions', 'WebhookScript', 'Terms & Privacy', and 'Support'. On the right, there are utility buttons for 'Copy', 'Edit', 'New', 'Login', and 'Upgrade'. Below this is a secondary bar with various settings 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 sections: 'REQUESTS (2/500) Newest First' with a search bar, 'Request Details' for a selected GET request, and 'Headers'. The 'Request Details' section shows a GET request to 'https://webhook.site/c44e59c3-f77e-4e49-87dd-441f640cc39d' with host '62.2.149.178', date '07/08/2023 11:14:37 (3 minutes ago)', size '0 bytes', and ID '805570ec-056f-43c6-ac9e-b776097ebd3f'. The 'Headers' section lists various headers such as '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', 'content-type', 'Form values', and 'Query strings' (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] request to `/demo/123`, timestamped 4:09:56 PM on 8/15/2016. It has a "Headers" section.

The second request is a [POST] request to `/demo`, timestamped 4:09:23 PM on 8/15/2016. It has "Headers" and "Body" sections. The body contains a JSON object:

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

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

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

Beeceptor

With beceptor 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
https://transfer.symbiose.com/download/info/	Info about request headers

Link	Description
https://httpbin.org/	Analysis for request and responses