

Introducing HAM v1.0.4.

reinhardt1010.id – 9 May 2024

From <https://reinhardt1010.id/blog/2024/05/09/introducing-ham-v1-0-4>.

Scan the QR Code to view the article on your device or web browser.



Content may subject to copyright. Visit the original website to view copyright and licensing information about this content. QR Code is a registered trademark of DENSO WAVE, Inc. in Japan and other countries. Generated on 2025-05-23 07:34:39.

[HAM](#) is a simple Jekyll framework that allows you to build static wiki sites. And today, we are introducing a maintenance update with the following changes.

First, the Bootstrap Icons dependency was updated from v1.11.1 to v1.11.3. There are no significant changes from the [100+ new icons](#) introduced since v1.10. These icons are directly built into HAM, and you can simply use them in your Markdown source files as `<i>` HTML tags. (Just make sure you have reviewed their [Web Accessibility recommendations](#) on placing icons.)

Next, we have also fixed a bug on v1.0.0 where users are redirected to the wrong YouTube URL when their web browsers do not support `<iframe>`. Well now, all modern web browsers supports that feature, and we're making this change to ensure that .

HAM is not going anyway sooner.

We still love HAM, and still use them on some of our internal projects. In fact, HAM still looks very nice on building another API documentation website like this:

On This Page

- Request Parameters
- ↳ Declaring Multiple Values on Form ...
- Request Responses
- Multiprocessing
- Naming Conventions
- Activity Check
- ↳ GET /ping
- ↳ GET /v1/test/ping
- Authentication
- ↳ POST /v1/test/access-token
- ↳ POST /v1/test/otp
- Assignments
- ↳ Model Structure
- ↳ GET /v1/assignments
- ↳ GET /v1/assignments/id
- ↳ GET /v1/assignments/feed
- ↳ GET /v1/assignments/statistics
- ↳ POST /v1/assignments/claim
- ↳ POST /v1/assignments/submit

Authentication

There are two main ways to authenticate into the REST API:

Authentication Type	Usage	How to obtain/change
Time-based OTP (TOTP)	Administrative tasks, such as configuring workers.	Change the <code>TOTP_SECRET</code> environment variable. The variable will be rehashed into Base32 to be compatible for Google Authenticator.
Access Token	Receive and submit assigned tasks from worker.	Create a new worker with <code>POST /v1/workers/create</code> , or refresh the token in <code>POST /v1/workers/refresh-token</code> . TOTP is still required to perform these tasks.

POST /v1/test/access-token

Check whether the server accepts the given access token.

This endpoint has no parameters, because the Worker ID and access token has been encoded into the `Authorization` request header, with the format `Authorization: Basic`, with `Base64` being a Base64-encoded data containing the value `:` to comply with [RFC7617](#).

This endpoint may return the following errors in [Standard Response](#) format.

HTTP Status Code	Standard Response Error Code	Description
400	<code>BAD_REQUEST</code>	The <code>Authorization</code> request header is not present or contain corrupt Base64 data.
400	<code>INVALID_AUTH_TYPE</code>	The <code>Authorization</code> request header contains value other than <code>Basic</code> .
404	<code>WORKER_NOT_FOUND</code>	The worker ID is invalid and not found in the database.
401	<code>INVALID_TOKEN</code>	The access token is verified to be invalid.

Or else, the endpoint will return a "OK" message in [Standard Response](#) format.

Of course, there are still some work to do. At least, readjusting the heading texts and adding [more Liquid tags](#), and making these ugly tables more beautiful just what [we recently did](#) on our main website.

So don't worry, we're still dogfooding HAM on our own.