



IIWARI TRACKING SOLUTIONS

iiwari API Specification

23.11.2023

Copyright © Iiwari Tracking Solutions Oy

Teemu Lätti teemu.latti@iiwari.com

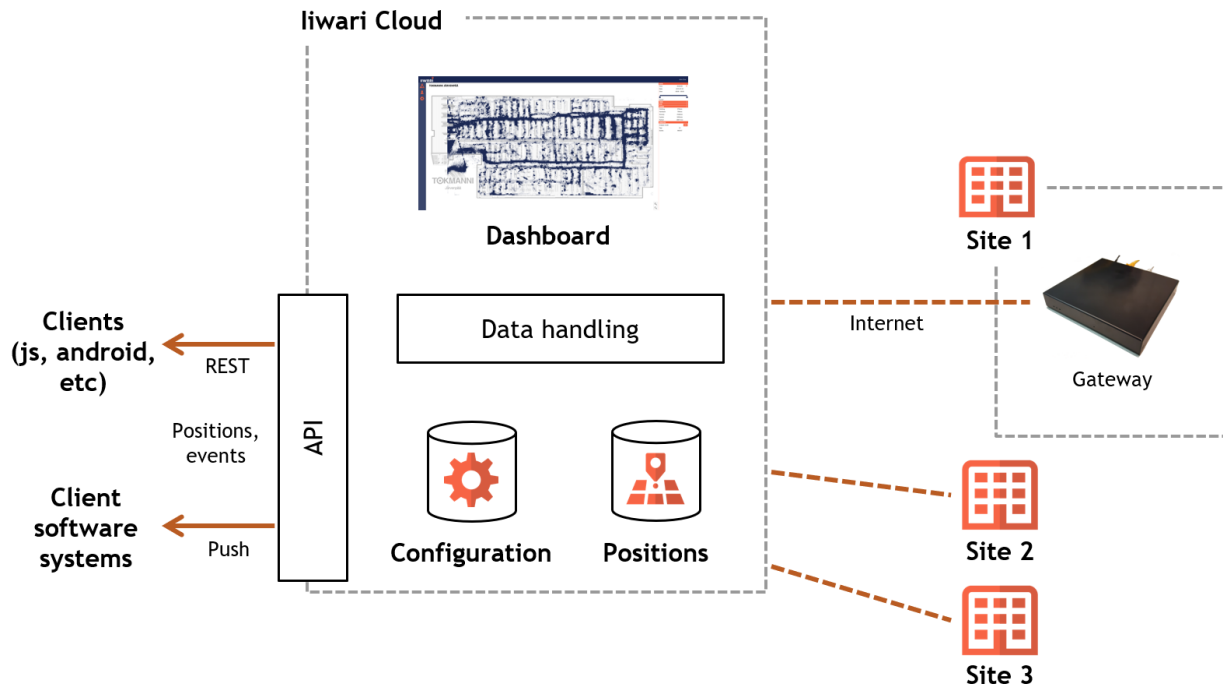
Table of contents

1. Introduction	3
2. liwari API	4
2.1. Authentication	5
2.2. Get sites	6
2.3. Get site	6
2.4. Locations stream	8
2.5. Locations	10
2.6. Events stream	10
2.7. Events	12
2.8. Tag status	13
2.9. Search	14
3. CURL examples	15

1. Introduction

This document describes how to use the liwari API. The liwari API is exposed from the liwari cloud and allows you to use liwari positioning system data and features in your own software and applications.

The following figure shows the high-level overview of the liwari positioning software system. Each site has a local device “liwari Gateway” which manages the local installation and connects to the liwari cloud. The liwari cloud manages all sites, stores positioning data and exposes interfaces to external systems. External systems can either use the liwari API (exposed as a REST interface) or the cloud can push data to them (when such arrangement is agreed). User interface (UI) “liwari Dashboard” is available for visualization of the data. Dash UI is also used to configure sites, zones, assets, tags, users etc.



The user of liwari API is a software program (backend, web app, mobile app, etc) which here is referred to as “client”. The liwari side system serving this API is here referred to as “server”.

All use of liwari API should be first agreed with liwari Tracking Solutions. In case of any questions, please contact the authors.

2. Iiwari API

You can programmatically access the Iiwari API from the following URL:

<https://dash.iiwari.cloud/api/v1>

In the rest of this document, the base URL is omitted when specifying the URL for operations.

You can issue operations using REST requests. Operations may support multiple HTTP methods (GET, POST, PATCH, DELETE) for different purposes as specified for each operation. The request URL may include parameters such as target entity, path parameters and query parameters. Example with operation and parameters:

<https://dash.iiwari.cloud/api/v1/sites/00000000-0000-0000-0000-000000000000/operation?name=foo&title=bar>

Most requests operate with JSON data. Depending on your client you may need to include a header to specify the content type:

Content-Type: application/json

In this document, JSON objects are specified for relevant content and fields only. You should parse all valid JSON from the response body and access only specified fields and ignore others.

Most database entities are identified by UUID which has the format "00000000-0000-0000-0000-000000000000". Devices (like positioning tags) are identified by HWID (hardware ID) which has the format "0000-0000-0000-0000".

Database entities identified with UUID always have a field "rev" which is an automatically increasing number whenever the database entity is changed. Whenever you make a PATCH request to update entity fields you must provide the most recent rev value, otherwise the update request fails. This mechanism is in place to avoid unintended overwriting of entity fields.

All timestamps are presented in UTC with millisecond precision in the format "2023-01-01T00:00:00.000Z".

Any request which is made using somehow incomplete or invalid payload data, results in a "400 Bad request" response.

Some methods use the WebSocket protocol. Note that in this case request URL has different protocol:

<wss://dash.iiwari.cloud/api/v1>



2.1. Authentication

All requests must be authenticated (except login request). You must specify a valid authentication token in each request you make. The token can be passed as a header:

```
Authorization: Bearer abcdef
```

Alternatively if header cannot be used, you can pass the token as a query parameter:

```
operation?token=abcdef
```

To get an authentication token from your login credentials, you should use the login request. Note that you do not need to make this request every time your application starts, but you can store and reuse the same token once obtained. However, if the token becomes invalid for some reason, your application must be able to request the token again.

Login request is made with the same credentials (email + password) as used for the Dash UI. During development you can use your own credentials, but for production application development we recommend you request from us new specific user credentials for that purpose with only required permissions.

Method:	POST
URL:	/users/login
Body:	{ "email": "foo@bar.com", "password": "foobar" }
Response codes:	200 OK 400 Bad Request 401 Unauthorized
Response:	{ "user": { "id": "00000000-0000-0000-0000-000000000000", "email": "foo@bar.com", "name": "Foo bar", }, "token": "abcdef" }

You can test whether a previously received token is still valid by making a login request with the attached token and no payload. Response 200 OK means the token is still valid.

Allowed operations for a specific user are based on their permissions and site. Attempting to execute an unallowed operation results in a “403 Forbidden” response.

2.2. Get sites

You can get a list of sites using this call:

Method:	GET
URL:	/sites
Response:	<pre>[{ "id": "00000000-0000-0000-0000-000000000000", "name": "Foobar" }]</pre>

The response is a JSON array of sites which you are allowed to see. You can use the site ID to refer to a site in other operations which require it. Instead of making this call, you can get a site ID from Dash UI.

2.3. Get site

You can get detailed data for a specific site using this call:

Method:	GET
URL:	/sites/00000000-0000-0000-0000-000000000000
Response:	<pre>{ "id": "00000000-0000-0000-0000-000000000000", "name": "Iiwari Kuopio", "floors": [{ "id": "00000000-0000-0000-0000-000000000000", "name": "3rd floor", "z_min": 500, "z_max": 1000, "floormaps": [{ "id": "00000000-0000-0000-0000-000000000000",</pre>



```

    "position0":{"x":0,"y":0},
    "position1":{"x":100,"y":100},
    "pixel0x":0,
    "pixel0y":0,
    "pixel1x":4000,
    "pixel1y":2000,
    "image_hash":"00000000"
  }],
  "zones":[{
    "id":"00000000-0000-0000-0000-000000000000",
    "name":"Zone 1",
    "type":104,
    "corners":[
      {"x":100,"y":0},
      {"x":200,"y":0},
      {"x":200,"y":100},
      {"x":200,"y":100}
    ]
  }],
  "pois":[{
    "id":"00000000-0000-0000-0000-000000000000",
    "x":600,
    "y":400,
    "z":100,
    "text":"Poi 1",
    "type":0,
    "state":0,
    "images":[]
  }],
  "assets":[{
    "id":"00000000-0000-0000-0000-000000000000",
    "type":17,
    "tag_hwid":"0000-0000-0000-0000",
    "name":"Asset 1",
    "icon":null,
    "data":{"foo":"bar"}
  }
]
}

```

	}
--	---

2.4. Locations stream

You can continuously receive tag positions from a site using this call:

Method:	WSS: GET -> WEBSOCKET
URL:	<pre> /sites/siteID/locations/stream /sites/siteID/locations/stream?startAt=2023-01-01T00:00:00.000Z&endAt=2023-01-01T00:10:00.000Z /sites/siteID/locations/stream?filter=kalman /sites/siteID/locations/stream?filter=kalman&maxMovementSpeed=500&processVariance=0.1&locationVariance=1&velocityVariance=0.5&accelerationVariance=0.2 /sites/siteID/locations/stream?filter=kalman&format=geo </pre>

Time range (UTC) can be specified but is optional. If “startAt” is specified, the server fetches positions from the database starting at that time. If “endAt” is not specified, the server continues sending real time positions as they arrive. When the stream has finished fetching positions from the database and switches to real time or only specifies realtime, a special “mark” message is sent to indicate that real time streaming has started.

Filter selection is optional and if omitted unfiltered data will be returned. Filter parameters are optional. The defaults are the same as what Dash UI shows in the filter configuration dialog. It is recommended that you enable the default Kalman filter, because the positioning system is calibrated to work best with the default parameters.

The request is always upgraded to a websocket connection. You can keep the websocket open as long as needed. You need to be able to reconnect if the connection breaks up for some reason.

The message body format is one of the following:

Tag position (local)	{
----------------------	---

coordinates)	<pre>"ts": "2023-01-01T00:00:00.000Z", "node": "0000-0000-0000-0000", "x": 100, "y": 100, "z": 100 }</pre>
Tag position (geo)	<pre>{ "ts": "2023-01-01T00:00:00.000Z", "node": "0000-0000-0000-0000", "latitude": 64.140161, "longitude": 28.268121 }</pre>
Tag position in privacy zone	<pre>{ "ts": "2023-01-01T00:00:00.000Z", "node": "0000-0000-0000-0000" }</pre>
Real time stream starts	<pre>{ "mark": 1 }</pre>

You should attempt to verify that the received message is in the expected format and ignore any other messages.

Field “ts” is the timestamp. The time is in UTC as the format implies.

Field “node” is the tag HWID which can be used to identify the tag.

Position fields (x, y, z) are in cm relative to the site origin position. Z is cm relative to the floor level of the first floor. When the client specifies “format=geo” in the call, positions are converted to geo positions (latitude,longitude) assuming the site has been properly configured.

When the position is inside a privacy zone, no coordinates are given (only timestamp). You can use this as an indication that the tag’s exact position should not be shown. This message can still be used to indicate that the tag is alive and somewhere within the site.

2.5. Locations

You can get tag positions from a site using this call:

Method:	GET
URL:	<code>/sites/siteID/locations?startAt=2023-01-01T00:00:00.000Z&endAt=2023-01-01T00:10:00.000Z</code>

This returns positions as a single JSON array in the response body instead of websocket stream. Request parameters and the format of response messages are exactly the same as in the stream request.

2.6. Events stream

You can continuously receive events from a site using this call:

Method:	WSS: GET -> WEBSOCKET
URL:	<code>/sites/siteID/events/stream</code> <code>/sites/siteID/events/stream?startAt=2023-01-01T00:00:00.000Z&endAt=2023-01-01T00:10:00.000Z</code> <code>/sites/siteID/events/stream?events=20,21</code>

Time range (UTC) can be specified but is optional. If “startAt” is specified, the server fetches events from the database starting at that time. If “endAt” is not specified, the server continues sending real time events as they arrive.

The parameter “events” is a list of event types. If specified, only those events are returned.

The request is always upgraded to a websocket connection. You can keep the websocket open as long as needed. You need to be able to reconnect if the connection breaks up for some reason.

The message body format is one of the following:

Tag button press	<pre>{ "ts": "2023-01-01T00:00:00.000Z",</pre>
------------------	--



	<pre>"node" : "0000-0000-0000-0000", "type" : 10 }</pre>
Tag switch on	<pre>{ "ts" : "2023-01-01T00:00:00.000Z", "node" : "0000-0000-0000-0000", "type" : 14 }</pre>
Tag switch off	<pre>{ "ts" : "2023-01-01T00:00:00.000Z", "node" : "0000-0000-0000-0000", "type" : 13 }</pre>
Device battery voltage	<pre>{ "ts" : "2023-01-01T00:00:00.000Z", "node" : "0000-0000-0000-0000", "type" : 1, "value" : 4632 }</pre>
Device temperature	<pre>{ "ts" : "2023-01-01T00:00:00.000Z", "node" : "0000-0000-0000-0000", "type" : 2, "value" : 2245 }</pre>
Zone enter	<pre>{ "ts" : "2023-01-01T00:00:00.000Z", "node" : "0000-0000-0000-0000", "type" : 20, "zone" : "00000000-0000-0000-0000-000000000000" }</pre>
Zone leave	<pre>{ "ts" : "2023-01-01T00:00:00.000Z", "node" : "0000-0000-0000-0000", }</pre>

	<pre>"type":21, "zone":"00000000-0000-0000-0000-000000000000" }</pre>
Site enter	<pre>{ "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000", "type":22 }</pre>
Site leave	<pre>{ "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000", "type":23 }</pre>

You should attempt to verify that the received message is in the expected format and ignore any other messages.

Field "type" identifies the event type and should be checked first. There will be more events with different payloads in the future.

Field "ts" is the timestamp. The time is in UTC as the format implies.

Field "node" is the tag HWID which can be used to identify the tag (or other device).

Device voltage is internal battery voltage. Divide the field "value" with 1000 to get battery voltage in V.

Device temperature is ambient temperature. Divide the field "value" with 100 to get temperature in Celsius degrees.

Zone events are triggered when a tag enters or leaves a zone which is configured in a site in the Dash UI.

2.7. Events

You can get events from a site using this call:

Method:	GET
---------	-----

URL:	/sites/siteID/events?startAt=2023-01-01T00:00:00.000Z&endAt=2023-01-01T00:10:00.000Z
------	--

This returns events as a single JSON array in the response body instead of websocket stream. Request parameters and the format of response messages are exactly the same as in the stream request.

2.8. Tag status

You can get the current status of a tag using this call:

Method:	GET
URL:	/tags/hwid/0000-0000-0000-0000/status

The returned data is a JSON object containing tag information. The current position is identified by fields "site_id" and "floor_id" and coordinates. List of zones is returned in which the tag is currently. For each zone, the time when the tag has entered the zone is given in the field "in_time".

Response:	<pre>{ "hwid": "0000-0000-0000-0000", "site_id": "00000000-0000-0000-0000-000000000000", "floor_id": "00000000-0000-0000-0000-000000000000", "position": { "ts": "2023-01-01T00:00:00.000Z", "x": 100, "y": 100, "z": 100 }, "zones": [{ "id": "00000000-0000-0000-0000-000000000000", "name": "Zone", "type": 101, "in_time": "2023-01-01T00:00:00.000Z" }], "status_ts": "2023-01-01T00:00:00.000Z", "firmware": "5.6.2", }</pre>
-----------	---



```
    "voltage":3000
  }
```

2.9. Search

You can search assets (tags) from a site using this call:

Method:	GET
URL:	<pre>/sites/siteID/assets/search /sites/siteID/assets/search?asset_types=1,3 /sites/siteID/assets/search?hwid=0000-0000-0000-0000 /sites/siteID/assets/search?name=Foo /sites/siteID/assets/search?data=field:value</pre>

This makes a search to all site assets and any global assets which are currently in the site. You can specify any number of parameters specified above. The search is a filtering search where all specified parameters must match for any given asset. If no parameters are given, the result is all assets.

The returned data is a JSON array of matching assets:

Response:	<pre>[{ "id": "00000000-0000-0000-0000-000000000000", "site_id": "00000000-0000-0000-0000-000000000000", "type": 1, "tag_hwid": "0000-0000-0000-0000", "name": "Foo", "icon": "...", "data": { "field": "value" }, "tag": { "id": "00000000-0000-0000-0000-000000000000", "hwid": "0000-0000-0000-0000",</pre>
-----------	---

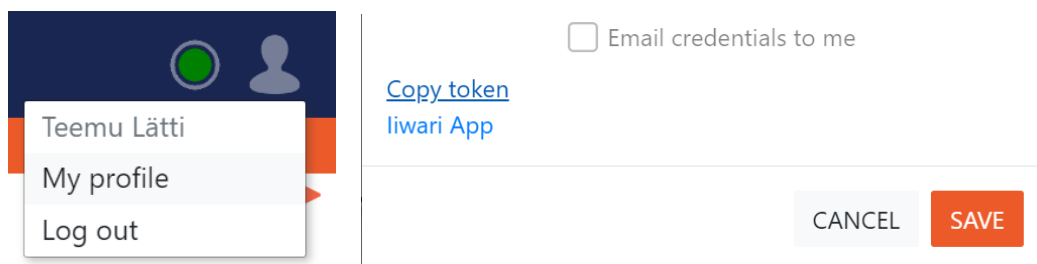
```
"product":50,
"fw_version":"1.0.0",
"status_at":"2023-01-01T00:00:00.000Z",
"voltage":4632,
"z":100
},
"position":{
  "ts":"2023-01-01T00:00:00.000Z",
  "x":100,
  "y":100,
  "z":100
}
}]
```

The position field may be null if the tag is not currently detected on the site. Make sure to validate the position timestamp since the position can be old (last known position).

3. CURL examples

You can test the liwari API if you have code or a tool which is capable of making HTTP REST requests. This chapter describes how to use Linux command prompt tool CURL in your own computer to access the liwari API.


To access the API, you first need a valid authorization token. You can use your personal token which is connected with your login credentials. Login to Dash at <https://dash.iiwari.cloud/> and click the user icon top-right and select "My Profile" from the menu. Then in the dialog, select "Copy token".



Most methods require a site ID which refers to a site which the methods are run against. Open Dash, open your site, right-click on the "SITE" drawer on the right or the site title in top-left and select "Site properties" from the menu. Then in the dialog, select "Copy site ID".

SITE	<input checked="" type="checkbox"/>
Site properties	<input type="checkbox"/>
Floor properties	<input type="checkbox"/>

[Copy site ID](#)


CANCEL
SAVE

For example, to start receiving live locations from tags, you can use the following command to open a websocket connection. Replace TOKEN with the token copied above, and replace SITE with the site ID of a site that you have access permission for.

```
curl -H "Authorization: Bearer TOKEN" -H "Connection: Upgrade" -H
"Upgrade: websocket" -H "Sec-WebSocket-Key: MDEyMzQ1Njc4OUFCQ0RFRg=="
-H "Sec-WebSocket-Version: 13" --no-buffer --http1.1 -s -o -
"https://dash.iiwari.cloud/api/v1/sites/SITE/locations/stream?filter=
kalman"
```

The following example shows how to access a method which is a simple GET call with parameters as part of the URL.

```
curl -H "Authorization: Bearer TOKEN" -s -o -
"https://dash.iiwari.cloud/api/v1/sites/SITE/locations?startAt=2023-0
1-01T12:00:00Z&endAt=2023-01-01T12:01:00Z"
```

Using the format of these examples, you can make a request for any of the methods specified in this document.