



IIWARI TRACKING SOLUTIONS

IIWARI API SPECIFICATION

4.1.2024

Copyright © Iiwari Tracking Solutions Oy

Teemu Lätti teemu.latti@iiwari.com

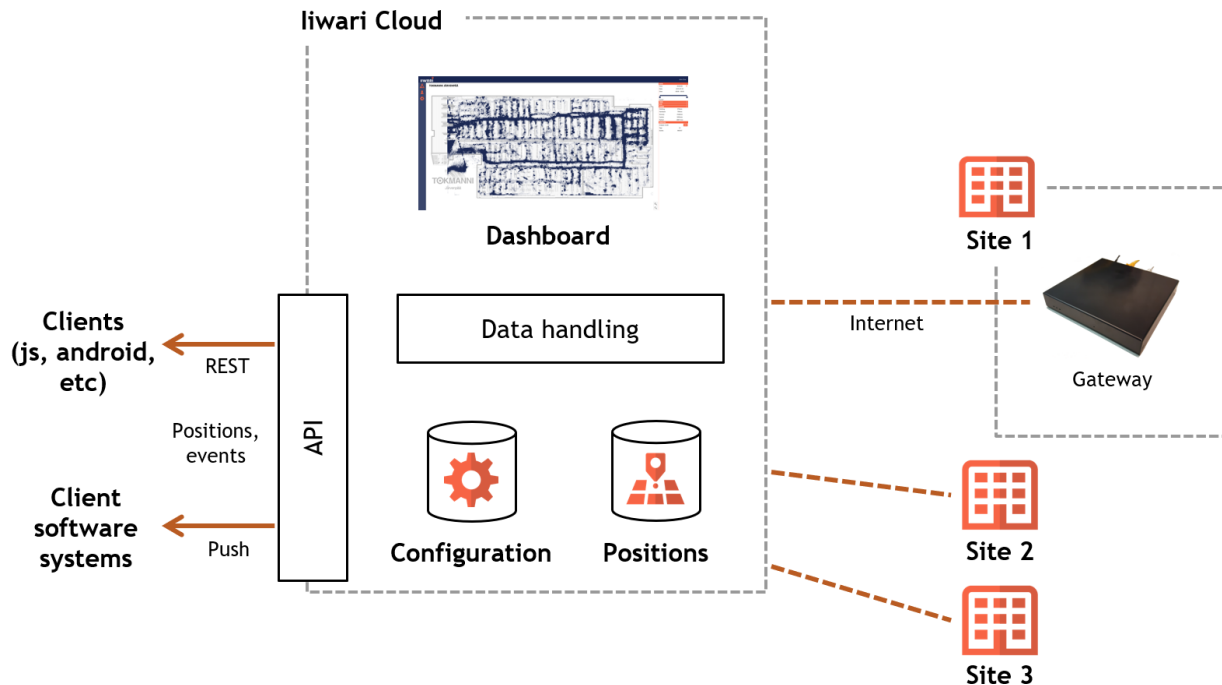
Table of contents

1. Introduction	3
2. Iiwari API	4
2.1. Authentication	5
2.2. Get sites	6
2.3. Get site	6
2.4. Stream	8
2.5. History	13
2.6. Locations stream	14
2.7. Locations	14
2.8. Events stream	14
2.9. Events	14
2.10. Tags in site	15
2.11. Tags in floor	16
2.12. Tags in zone	16
2.13. Zones with tags	17
2.14. Tag status	18
2.15. Search	19
3. Dash UI	21
4. CURL examples	22

1. Introduction

This document describes how to use the liwari API. The liwari API is exposed from the liwari cloud (or standalone system) 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 “Iiwari Gateway” which manages the local positioning infrastructure, calculates positions 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 liwari API (REST interface) or the cloud can push data to them (when such arrangement is agreed). User interface (UI) “Iiwari Dashboard” is available for data visualization. Dash UI is also used to configure sites, floors, 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 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 author.

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.

Operations are issued using REST requests. Operations may support multiple HTTP methods (GET, POST, PATCH, DELETE) for different purposes as specified. The request URL may include parameters such as target entity, path parameters and query parameters. For example:

<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 valid JSON from the response body and access only specified fields and ignore others (they may be changed or omitted in the future).

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 with old data.

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). 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 can 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. However, if the token becomes invalid for some reason, your application must be able to request the token again. Alternatively you can get the token from Dash UI, see chapter “CURL examples”.

Login request is made with the same credentials (email + password) as 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 sites and operations for a specific user are based on their permissions. 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:	[{ "id": "00000000-0000-0000-0000-000000000000", "name": "Foobar" }]

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. Alternatively you can get the 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/SITEID
Response:	{ "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",
    "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. Stream

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

Method:	WSS: GET -> WEBSOCKET
URL:	<pre>/sites/SITEID/stream?filter=kalman /sites/SITEID/stream?startAt=2023-01-01T00:00:00.000Z&endAt=2023-01-01T00:10:00.000Z /sites/SITEID/stream?filter=kalman&maxMovementSpeed=500&processVariance=0.1&locationVariance=1&velocityVariance=0.5&accelerationVariance=0.2 /sites/SITEID/stream?filter=kalman&format=geo /sites/SITEID/stream?events=20,21</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 this. Applications can use this information to know when history has loaded.

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 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.

Possible messages which you will receive from the websockets are specified in the table below. You should verify that the received message is in the expected format and ignore 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.

Field “node” is the tag HWID which can be used to identify the tag (or other device) when relevant.

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. If you specify “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 returned (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.

Zone, floor and site events are triggered when a tag enters/leaves a zone/floor/site which is configured in a site in the Dash UI.

Events which are notifications for real time events (such as site change) are not returned when receiving from history.

Real time stream starts	<pre>{ "mark":1 }</pre> <p>Streaming events from history has completed and messages after this are real time.</p>
Tag position (local coordinates)	<pre>{ "type":0, "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000", "x":100, "y":100, "z":100 }</pre> <p>Tag position in local coordinates as cm (this is the default). Tag is identified by HWID as the field “node”.</p>
Tag position (geo)	<pre>{</pre>

	<pre> "type":0, "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000", "latitude":64.140161, "longitude":28.268121 } </pre> <p>Tag position in geo coordinates (only if requested).</p>
Tag position in privacy zone	<pre> { "type":0, "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000" } </pre> <p>Tag moves in a privacy zone. No exact position.</p>
Tag button press	<pre> { "type":10, "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000" } </pre> <p>User presses the tag button.</p>
Tag switch on	<pre> { "type":14, "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000" } </pre> <p>User switches the tag on.</p>
Tag switch off	<pre> { "type":13, "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000" } </pre> <p>User switches the tag off.</p>

Device status	<pre>{ "type":9, "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000", "value":17 }</pre> <p>Devices on site periodically send this “I’m alive” message. Value is product type which have one of the following values:</p> <p>1 = GW 16 - 18 = MBS 32 - 35 = BS 48 - 53 = TAG</p>
Device battery voltage	<pre>{ "type":1, "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000", "value":4632 }</pre> <p>Device voltage is its internal battery voltage. Divide the field “value” with 1000 to get battery voltage in V.</p>
Device temperature	<pre>{ "type":2, "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000", "value":2245 }</pre> <p>Device temperature is the ambient temperature around the device. Divide the field “value” with 100 to get temperature in Celsius degrees.</p>
Zone enter	<pre>{ "type":20, "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000", "zone":"00000000-0000-0000-0000-000000000000" }</pre>

	<pre>} Tag enters a zone.</pre>
Zone leave	<pre>{ "type":21, "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000", "zone":"00000000-0000-0000-0000-000000000000" }</pre> <p>Tag leaves a zone.</p>
Floor enter	<pre>{ "type":24, "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000", "floor":"00000000-0000-0000-0000-000000000000" }</pre> <p>Tag enters a floor.</p>
Floor leave	<pre>{ "type":25, "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000", "floor":"00000000-0000-0000-0000-000000000000" }</pre> <p>Tag leaves a floor.</p>
Site enter	<pre>{ "type":22, "ts":"2023-01-01T00:00:00.000Z", "node":"0000-0000-0000-0000" }</pre> <p>Tag enters the site.</p>
Site leave	<pre>{ "type":23,</pre>

	<pre>"ts": "2023-01-01T00:00:00.000Z", "node": "0000-0000-0000-0000" }</pre> <p>Tag leaves the site. Happens when the tag has not been seen for 2 minutes 30 seconds.</p>
Zones status	<pre>{ "type": 29, "ts": "2023-01-01T00:00:00.000Z", "node": "0000-0000-0000-0000", "zone": "00000000-0000-0000-0000-000000000000", "in_time": "2023-01-01T00:00:00.000Z", "in_duration": 15000 }</pre> <p>Optional message which is sent periodically. This message is sent separately for each tag and zone and indicates that the tag is currently in the zone. The interval is specified as seconds in the query parameter, for example "followZones=5". If not specified or has value zero, this message is not sent. This message is not sent from history.</p>
Site change	<pre>{ "type": 99, "ts": "2023-01-01T00:00:00.000Z" }</pre> <p>Site has been changed. Triggered by a change to floors, floormaps, zones, pois or assets.</p>

2.5. History

You can get tag positions and events from the history for a site using this call:

Method:	GET
URL:	/sites/SITEID/history?filter=kalman&startAt=2023-01-01T00:00:00.000Z&endAt=2023-01-01T00:10:00.000Z

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

2.6. Locations stream

If you want to receive only positions from a site, you can use this call. Request parameters and response messages are the same as for the stream.

Method:	WSS: GET -> WEBSOCKET
URL:	/sites/SITEID/locations/stream?filter=kalman

2.7. Locations

If you want to get only positions from the history for a site, you can use this call. Request parameters and response messages are the same as for the stream.

Method:	GET
URL:	/sites/SITEID/locations?filter=kalman&startAt=2023-01-01T00:00:00.000Z&endAt=2023-01-01T00:10:00.000Z

2.8. Events stream

If you want to receive only events from a site, you can use this call. Request parameters and response messages are the same as for the stream.

Method:	WSS: GET -> WEBSOCKET
URL:	/sites/SITEID/events/stream

2.9. Events

If you want to get only events from the history for a site, you can use this call. Request parameters and response messages are the same as for the stream.



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

2.10. Tags in site

You can get a list of tags currently in the site using this call:

Method:	GET
URL:	/sites/SITEID/tags

The returned data is a JSON array containing information for each tag on site including zones where the tag is currently located.

Response:	<pre>[{ "hwid": "0000-0000-0000-0000", "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", "in_duration": 15000 }], "status_ts": "2023-01-01T00:00:00.000Z", "firmware": "5.6.2", "voltage": 3000 }]</pre>
-----------	--

2.11. Tags in floor

You can get a list of tags currently in a specified floor using this call:

Method:	GET
URL:	/sites/SITEID/floors/FL00RID/tags

The returned data is a JSON array containing information for each tag in the floor including zones where the tag is currently located.

Response:	<pre>[{ "hwid": "0000-0000-0000-0000", "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", "in_duration": 15000 }], "status_ts": "2023-01-01T00:00:00.000Z", "firmware": "5.6.2", "voltage": 3000 }]</pre>
-----------	--

2.12. Tags in zone

You can get a list of tags currently in a specified zone using this call:

Method:	GET
URL:	/sites/SITEID/zones/ZONEID/tags



The returned data is a JSON array containing information for each tag in the zone.

Response:	<pre>[{ "hwid": "0000-0000-0000-0000", "position": { "ts": "2023-01-01T00:00:00.000Z", "x": 100, "y": 100, "z": 100 }, "status_ts": "2023-01-01T00:00:00.000Z", "firmware": "5.6.2", "voltage": 3000 }]</pre>
-----------	---

2.13. Zones with tags

You can get a list of site zones including tags currently in the zones using this call:

Method:	GET
URL:	/sites/SITEID/zones/tags

The returned data is a JSON array containing information for each zone on site and a list of tags currently in the zone. Field "in_time" tells when the tag has entered the zone. Field "in_duration" tells the number of milliseconds the tag has been in the zone.

Response:	<pre>[{ "id": "00000000-0000-0000-0000-000000000000", "name": "Zone", "type": 101, "tags": [{ "hwid": "0000-0000-0000-0000", "in_time": "2023-01-01T00:00:00.000Z",</pre>
-----------	---

```

        "in_duration":15000
    } ]
} ]

```

2.14. Tag status

You can get the current status of an individual 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". Field "in_duration" tells the number of milliseconds the tag has been in the zone.

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", "in_duration": 15000 }], "status_ts": "2023-01-01T00:00:00.000Z", "firmware": "5.6.2", </pre>
-----------	---

	<pre> "voltage":3000 } </pre>
--	-------------------------------

2.15. 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:

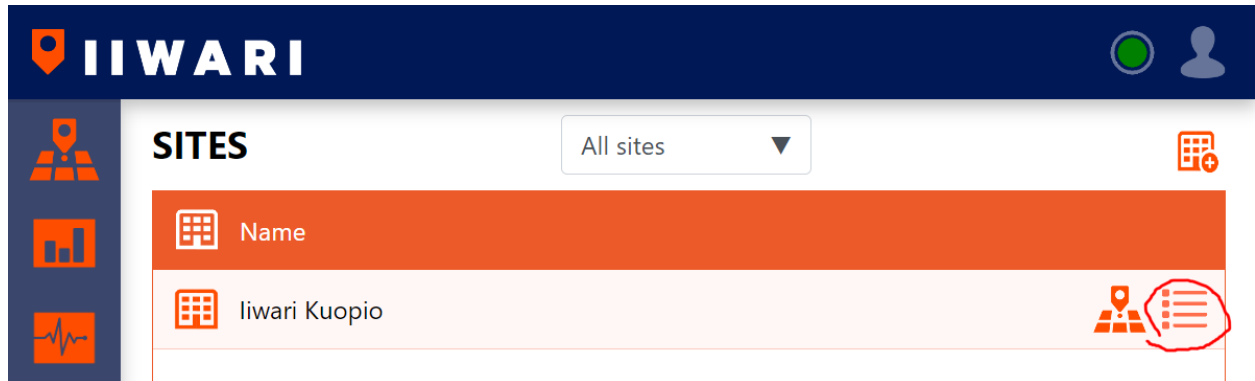
Response:	<pre> [{ "id": "00000000-0000-0000-0000-000000000000", "site_id": "00000000-0000-0000-0000-000000000000", "type": 1, "tag_hwid": "0000-0000-0000-0000", "name": "Foo", "icon": "...", "data": { "field": "value" }, "tag": { "id": "00000000-0000-0000-0000-000000000000", "hwid": "0000-0000-0000-0000",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

To explore and debug events, you can use the Dash UI which shows the same messages which you would programmatically receive from the websocket. In Dash, click the right side icon to open the events page (instead of clicking the site name which opens the map).



The events page can be used to view events in real time or you can select a time range from history by clicking the time range on top of the page.

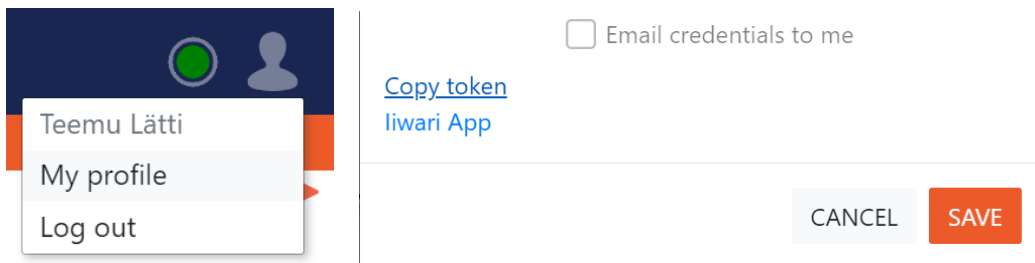
The screenshot shows the IIWARI Dash UI for the 'IIWARI KUOPIO' site. The header is dark blue with the IIWARI logo and status/user icons. The sidebar has the same three orange icons. The main content area has a title 'IIWARI KUOPIO' and a timestamp '2023-11-30 12:03:32'. Below the timestamp are two checked checkboxes: 'Kalman filter' and 'Restrictions'. To the right of these is the text '83 events'. Below this is a table with four columns: 'TIME', 'HWID', 'PRODUCT', and 'NAME'. The table contains four rows of event data.

TIME	HWID	PRODUCT	NAME
2023-11-30 11:57:35.372	1047-3932-1592-4086	Tag	Teemu Lätti
2023-11-30 11:57:36.396	1047-3932-1592-4086	Tag	Teemu Lätti
2023-11-30 11:57:36.396	1047-3932-1592-4086	Tag	Teemu Lätti
2023-11-30 11:57:36.396	1047-3932-1592-4086	Tag	Teemu Lätti

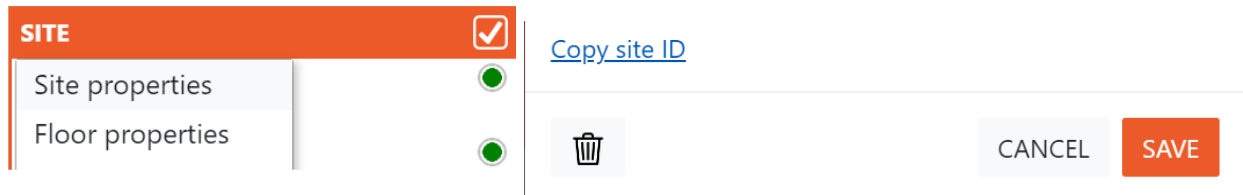
4. CURL examples

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



For example, to start receiving locations and events, you can use the following command to open a websocket connection. Replace TOKEN with the token copied above, and replace SITEID 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/SITEID/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/SITEID/history?filter=kalman&  
startAt=2023-01-01T12:00:00Z&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.