drone harmony
infrastructure digitalization

Application Programming Interface (API) Guide

LAST UPDATE: 10 OCTOBER 2021

Contents

API Overview

Terms Used in this Document

- **DH** - Drone Harmmory.
- **DH API** - The RESTful API described in this document.
- **Client** - company or organization that integrates with the DH API.
- **DH Web** - the Drone Harmony web application.
- **DH Mobile** - the Drone Harmony mobile application (Android or iOS).
- **Client Backend** - the server side of the Client application that calls the DH API.
- **Client Web App** - the web front end of the Client application.
- **Client Mobile App** - mobile application of the Client (if present).
- **DH User** - a specific user account in DH (DH Web and DH Mobile use the same accounts).
- **Client User** - a specific user account inside the system of the Client. The term `client_user_id` is used in this document to denote the id of this user.

All altitudes / distances / areas are accepted and returned in metric units.

URL Base

The base URL for all requests is:

`https://api.droneharmony.com/v1/`

Api Key

All requests require an api key that must be passed on each HTTP request header, like this:

`X-API-KEY: your_api_key_here`

Requests that do not provide a valid api key will be rejected with HTTP code 403.

Important security note: DH API is intended to be called from the Client Backend. The api key is unique to the Client that received it and should at no time be made public or

shared with a Client Web App or the Client Mobile App. Client Web App can open DH Web App in separate browser tabs or iFrames according to this spec, but all REST calls are designed to be made from the Client Backend.

Authentication and Single Sign-On (SSO)

Parts of the API require a user context under which the actions are performed. Example of this is when Client posts a scene inside the DH Web app. This requires establishing a link between a Client User account and a DH User account. We will call this link - the **SSO-Link**. SSO-Links are managed by the DH server, inside the DH database. The outline of the process of establishing the SSO-Link is:

1. Client Backend calls DH API `/auth` call providing the Client User unique id (see note below about the requirements from such an id).
2. In the response to the call above, DH API returns an Auth Token that is valid for 1 hour.
3. Using this token the Client Web App can open a browser tab (or an iFrame) with DH Web App, providing in the URL parameters the Auth Token (possibly along with some other parameters).
4. If the SSO-Link has not been established in the past:
    a. DH Web App will present the user with a standard DH login page
    b. Upon successful login into the DH Web App, the SSO-Link will be saved into the DH database, thus establishing the SSO-Link for future requests.
    c. The user will be logged into the DH Web App.
5. Else (SSO-Link is already established in the past):
    a. The user will be immediately logged into the DH Web App.

Requirements from the Client User ID:

- Must be unique for the Client user account. Client user ids are scoped per Client and there is no risk of them coinciding with other Client user ids.

- Client Backend should be able to find the user using this id. So, for example, Hash(databse_id) will not work unless the result of the hash is saved in the database.

Examples of Client User ids that will work:

- Actual account id in the database
- Guid or a hash stored along the account that does not change
- AES encrypted account id of the user
- User email in case it is guaranteed not to change

Client User ids will not leave DH Backend (will not be shared with the DH web and mobile clients) and will not be visible to DH users. DH Web App will allow the user to erase the SSO-Link.

Getting an Auth Token for a Client User

Request: GET /auth/{client_user_id}

Description: Get auth token for a Client User. Auth token is valid for 1 hour. For requirements from the Client User ids - read the section above. Id should not be longer than 256 characters. Note that If the ids can contain symbols that are not valid in a URL- the string must be URL-encoded.

Response:

```
{
    authToken: auth_token,     // string, will not require URL-encoding
    ssoLinkPresent: boolean,   // true when SSO-Link is present
}
```

Response status codes:

HTTP Code	Meaning	Returned when
200	Successful	
403	Forbidden	Bad or missing API Key

Example request: GET https://api.droneharmony.com/v1/auth/100

To open DH Web, the Client Web App will need to open the following URL in a new browser tab:

https://app.droneharmony.com?at=auth_token

DH Web will:

- In case the SSO-Link was not yet established - redirect to DH login page. After the user signs in, SSO-Link will be established inside the DH database between the client_user_id provided in the URL and the DH User that has signed in. After this the DH Web App will open as usual after login (unless other parameters were specified, see below).
- If the SSO-Link is already present the user will not be required to provide additional credentials to sign in.

Posting a site into DH Web

Request: POST `/site/{client_user_id}`

Description: Creates a site (See section “Site Storage” in DH Web App main menu) inside DH for a specific user. After creating the site it will be accessible from the “Load Site” menu option.

Request entity body: JSON with the following format:

```
{
    siteName: string,    // must
    siteTags: string[],  // optional
    externalId: string,  // optional
    scene: geo_json,     // must, single item or collection
    noFlyZones: geo_json,   // optional, polygon(s) NOT YET RELEASED
    geoFences: geo_json,   // optional, polygon(s) NOT YET RELEASED
}
```

Scene parameter explained: The scene GeoJSON can be a single element (for example a polygon) or a GeoJSON collection. The GeoJSON objects can contain custom properties with additional information. GeoJSON elements are mapped in the following way:

- GeoJSON Polygon is mapped into DH polygons
- GeoJSON LineString is mapped into DH lines
- GeoJSON Points -> mapped into DH points of interest

Example request:

https://droneharmony.com/api/samples/post_site_request.json

Custom DH **properties** on the GeoJSON elements are case sensitive. Possible properties:

- name [optional] - name of the area. Limited by 256 symbols (default “Area”).
- height [optional] - height in meters of the structure for area polygons. Default is 0.
- color [optional] - area colors are encoded with values between 0 and 11. Reference table:

Requirements from the scene: at least one valid geographic element must be present (one polygon, one line, one point of interest), otherwise the scene will be considered as

empty (see 400 error code below). The scene should be contained in a bounding box which is less than 10 kilometers in diagonal distance. Polygons that are not simple (self-crossing) or are larger than 5 square kilometers in area - will be ignored. Max line length is 10 km. Height range is [0-500] m.

Response status codes:

HTTP Code	Meaning	Returned when
200	Successful	Site saved
400	Bad request	- Scene was empty or too large or the format was wrong. - Id parameter was not valid (requires URL encoding or is longer than 256 characters).
403	Forbidden	- Bad or missing API Key - Scene with a given external id already exists. In this case the message in the body will be “Already exists”.

Response entity body:

```
{
        siteId: string          // string, no need to URL encode.
        externalId?: string,    // will be present in case was given
}
```

Reading a site from DH

Request: GET `/site/{client_user_id}/{site_id OR externalId}`

Description: Read a site. SSO-Link must be present.

Response entity body:

```
{
      siteId: string,
      externanId: string,  // optional
      siteName: string,
      siteTags: string[],
      lastModified: Unix_time_millis,
      scene: geo_json,
      geoFences: geo_json,   // polygons, NOT YET RELEASED
      noFlyZones: geo_json,   // polygons, NOT YET RELEASED

      Missions: [geo_json]
}
```

Site can contain missions. Missions will be returned as GeoJSON FeatureCollection, with a single line and one point for each waypoint. Example:

https://droneharmony.com/api/samples/get_site_response.json

Mission properties (present on the single LineString geo-json feature):

- guid - unique identifier of the mission (will be useful in the future for tying with captured results)
- missionName - user given mission name
- droneProfile - unique id of drone profile in DH system, example “mavic-2-pro”
- cameraProfile - unique id of camera profile in DH system, example “mavic-2-pro-camera”

Waypoint properties (present on each Point geo-json feature):

- azimuth - direction of the camera (for most drones also corresponds to the direction of the drone). North is 0, valid range is [0, 360], direction is clockwise.
- type: one of: liftoff / regular / landing. Liftoff is always the first waypoint, landing always the last. DH missions have at least one regular waypoint.

Response status codes:

HTTP Code	Meaning	Returned when
200	Successful	Site found and SSO link is present.
400	Bad request	SSO-link is not present for the user or the user has manually removed the SSO-Link.
403	Forbidden	Bad or missing API Key
404	Site not found.	Site id provided was found. User could have deleted the site manually or it was never present.

Opening a site inside DH Web

After successful posting of a site inside DH, it is possible to open the site inside DH Web. The site can be opened in a stand alone tab of the browser as well as embedded inside an iFrame.

To open the site, the Client Web App will need to open the following URL:

`https://app.droneharmony.com?at=auth_token&site=site_id&auto-save=[true / false]`

Url parameters explanation:

- at - auth token returned by DH server to the Client server
- site - [optional] DH id of the site to open as returned by POST (create) or GET (read) call from Client server to DH server.
- auto-save - [optional] if this flag is given, the opened site will be auto saved on each user change, without the need from the user to explicitly save the site (go into Menu / Sites / Save site). It is very recommended to use this option when using DH ui embedded inside another application using an iFrame.

Note that since the Auth Token is valid for a limited time it needs to be retrieved every time, the typically sequence of calls will be:

1. Call from the Client Backend to `/auth` to get the auth token.
2. Call from the Client Backend to `/site/client_user_id` to post a new site OR call to `/site/client_user_id/{site_id OR externalId}` to get a DH site id of an existing site. It's a valid flow to call POST to create a site, receive 403 / "Already exists" and then use GET to get the site. There is no requirement from the API implementor to store in its own database that a specific external id site was already created inside DH.
3. Open the DH Web in a new tab or an iFrame from the Client Web App using the above url.

Listing all Client User site ids in DH

Request: GET `/site/{client_user_id}?filter=sub_string`

Description: Reads all the client site ids from DH. SSO-Link must be present. Filter parameter is used to filter over substrings in the name or match tags precisely. Filtering is case insensitive. Paging is currently not supported. Maximum number of returned items is 100.

Response:

```
[
      {
            siteId: string,
            externalId: string,   // optional
            siteName: string,
            siteTags: string[],
            lastModified: Unix_time_millis,  // example 1623258716294
      },
      ...
]
```

Response status codes:

HTTP Code	Meaning	Returned when
200	Successful	
400	Bad request	SSO-link is not present for the user or the user has manually removed the SSO-Link.
403	Forbidden	Bad or missing API Key

Deleting a site in DH

Request: DELETE `/site/{client_user_id}/{site_id OR externalId}`

Description: Deletes a specific site from DH. SSO-Link must be present.

Response:

```
{
    siteId: site_id,  // string
    externalId: string,   // optional
    siteName: string,
    siteTags: [string],
    lastModified: Unix_time_millis
}
```

Response status codes:

HTTP Code	Meaning	Returned when
200	Successful	
400	Bad request	SSO-link is not present for the user or the user has manually removed the SSO-Link.
403	Forbidden	Bad or missing API Key

Sharing a site between users

Request: POST `/site/{client_user_id}/{site_id OR externalId}/share`

Description: Share a site between multiple users. SSO-Link must be present for all users. A shared site can be read and modified by all the users that have access to it. Only the owner (original user who created the site) can delete the site using the web ui.

Response:

```
{
    sharedUsers: [client_user_id1, ...],  // list of client user ids
}
```

Response status codes:

HTTP Code	Meaning	Returned when
200	Successful	
400	Bad request	SSO-link is not present for the user or the user has manually removed the SSO-Link.
403	Forbidden	Bad or missing API Key.
404	Not found	The site with the given ID was not found.