drone harmony
infrastructure digitalization

# Application Programming Interface

**LAST UPDATE: 3 April 2025**

# Table of contents

# API Overview

## Terms Used in this Document

- **DH API** – The RESTful API outlined in this document.
- **Client** – A company or organization that integrates with the DH API.
- **DH Web** – The Drone Harmony web application.
- **Client Backend** – The server-side component of the Client application that interacts with the DH API.
- **Client Web App** – The web-based front end of the Client application.
- **Client Mobile App** – The mobile application of the Client (if applicable).
- **DH User** – A user account within Drone Harmony.
- **Client User** – A user account within the Client's system. The term `client_user_id` is used in this document to refer to the unique identifier of this user.

## URL Base

The base URL for all requests is (depending on the app flavour):

https://{server_subdomain}.droneharmony.com/v1/

Where ***{server_subdomain}*** is of: *app, dock, usa or asia*

## Swagger Documentation

https://{server_subdomain}.droneharmony.com/v1/swagger-ui/index.html?urls.primaryName=dh_external_api#

Where ***{server_subdomain}*** is of: *app, dock, usa or asia*

## Api Key

### API Key Requirements

- Every request must include an API key in the HTTP header:
  `X-API-KEY:your_api_key_here`
- Requests without a valid API key will be denied with an **HTTP 403 error**.

### Security Note

- The **DH API** is meant to be called **only from the Client Backend**.
- The API key is unique to each Client and **must not be shared** with the **Client Web App** or **Client Mobile App**.

---

# Authentication

## Authentication and Single Sign-On (SSO)

Certain API functions require a **user context** to execute actions on behalf of a specific user. For example, when a Client posts a scene in the **DH Web App**, the system must associate a **Client User account** with a **DH User account**. This association is referred to as the **SSO-Link**. The **DH server** manages SSO-Links within its database.

The process for establishing an **SSO-Link** is outlined below:

**Initiating Authentication:**

- The Client Backend communicates with the DH API through the /auth endpoint, providing a unique id for the Client User (refer to the note below for id requisites).
- The **DH API** responds with an **Auth Token**, which remains valid for **one hour**.

**Token Utilization:**

- With the acquired token, the Client Web App can now launch a browser tab (or an iFrame) encapsulating the DH Web App, appending the Auth Token to the URL parameters, possibly along with other parameters.

**SSO-Link Establishment:**

- New SSO-Link:
    - Should the SSO-Link not have been established previously, the DH Web App will present a standard DH login page to the user.
    - Upon successful login to the DH Web App, the SSO-Link gets recorded in the DH database, thereby establishing the link for subsequent requests. Consequently, the user gets logged into the DH Web App.

    - Existing SSO-Link:
        - If the SSO-Link had been established in the past, the user gets logged into the DH Web App instantly.

**Requisites for Client User ID:**

- **Uniqueness** – Each **Client User ID** must be unique within the Client's system. Since these IDs are assigned per Client, there is no risk of duplication across different Clients.
- **Retrievability** – The **Client Backend** must be able to retrieve user information based on this ID. Using a **hashed database ID** alone is insufficient unless the hash value is persistently stored and accessible in the database.

**Viable Examples of Client User IDs:**

- Actual account ID in the database.
- A GUID or a hash (that remains unchanged) stored alongside the account.
- AES encrypted account ID of the user.
- User email, provided it's assured to remain unchanged.

The Client User IDs are confined to the DH Backend and will not be shared to the DH web and mobile clients, nor will they be visible to DH users. The api provides an option to erase the SSO-Link.

## Getting an Auth Token for a Client User / Establishing the SSO link

**Request:** GET `/auth/{client_user_id}`

**Description:** Retrieves an **auth token** for a **Client User**. The token remains valid for **one hour**.

- For **Client User ID requirements**, refer to the section above.
- The **ID** must not exceed **256 characters**.

- If the **ID** contains symbols that are not valid in a URL, it must be **URL-encoded** before transmission.

**Response:**

```
{
 "authToken": "auth_token",
 "ssoLinkPresent": "true"
}
```

**Response status codes:** refer to swagger.

**Example request:** GET https://{server_subdomain}.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://{server_subdomain}.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.

## Getting the SSO Link status

**Request:** GET `/auth/sso/{client_user_id}`

```
{
 "email": String,
}
```

## Deleting the SSO Link

**Request:** DELETE `/auth/sso/{client_user_id}`

---

# Media

## Upload

**Request:** POST `/v1/media/{client_user_id}/upload`

Headers: 'X-API-KEY: {X-API-KEY}',

'Content-Type: multipart/form-data'

Body: file - string($binary)

**Description:** Uploads single or several media files to DH server. Returns information about the media such as location of the image extracted from the Exif data. If the image was created with DH Mobile - will also match the image to mission log and return additional information about the mission.

**Example response**

Response schema corresponds to https://geojson.org/ data structure with DH (custom) properties:


```
{
 "type": "FeatureCollection",
 "features": [
   {
     "type": "Feature",
     "properties": {
       "createTime": 1698157172000,
       "panoramaId": null,
       "name": "7ac14fe4b646fc03e75b04fa8b922e2394f9a5ea.jpeg",
       "missionName": null,
       "missionGuid": null,
       "logId": null,
       "id": 202,
       "timestampStartMission": null
     },
     "geometry": {
       "type": "Point",
       "coordinates": [
```

```
          9.28006491666667,
          47.3923667777778,
          827.82
        ]
      }
    }
  ]
}
```

## List media using a filter

**Request:** POST /v1/media/{client_user_id}/filtered-medias

Headers: 'X-API-KEY: {X-API-KEY}',

**Description:** Contains all available metadata supplemented by links, mission guides, image sensor type etc. Supports paging.

### Example request

```
{
  "taskId": "43a8dcb6-8fec-4fd3-8e1b-2fa6ca1183c7",
  "startTime": 1694091556000,
  "endTime": 1694091556000,
  "imageSources": [
    "WideCamera",
    "ZoomCamera",
    "InfraredCamera"
  ],
  "fromIndex": 0,
  "pageSize": 25
}
```

### Example response

Response schema corresponds to https://geojson.org/ data structure with metadata as well as the point where the media has been produced

```
{
  "type": "FeatureCollection",
  "features": [
    {
```

```
      "type": "Feature",
      "properties": {
        "createTime": 1699634984000,
        "panoramaId": null,
        "downloadMediaThumbnailLink":
"https://{server_subdomain}.droneharmony.com/v1/download?key=5c0ddeeaf21d98e...", //with
original content type
        "name": "DJI_20231110164944_0001_W.jpeg",
        "missionName": "Manual",
        "missionGuid": "cb09a6e1-606e-4c91-ba18-0606aa91e35b",
        "logId": null,
        "viewMediaLink":
"https://{server_subdomain}.droneharmony.com/v1/download?key=5c0ddeeaf21d98e4ec5f2ab7b8da89b
34a6552d9dae888930dc75b...", //with original content type
        "id": 6041,
        "downloadMediaLink":
"https://{server_subdomain}.droneharmony.com/v1/download?key=5c0ddeeaf21d98e4ec5f2ab7b8da89b
34a6552d9dae888930dc7...", //always OCTET_STREAM
        "timestampStartMission": 1699606146282,
        "taskId": "cb09a6e1-606e-4c91-ba18-0606aa91e35b"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          0.0000305555555555556,
          -0.0000175277777777778
        ]
      }
    }
  ]
}
```

## Get media metadata

**Request:** GET `/v1/media/{client_user_id}/{media_id}/info`

Headers: ‘X-API-KEY: {X-API-KEY}’,

**Description:** Gets all extracted Exif metadata map for selected mediaId. SSO-Link must be present.

### Example request

```
{
  "fileName": "DJI_0608_W.JPG",
  "iSO": 200,
  "whiteBalance": "0",
```

```
  "gPSAltitude": 109.326,
  "absoluteAltitude": 109.33,
  "saturation": 0,
  "hyperfocalDistance": 1.28373473983553,
  "model": "MAVIC2-ENTERPRISE-ADVANCED",
  "gPSLatitude": 47.1435263333333,
  "sharpness": 0,
  "imageWidth": 8000,
  "meteringMode": "1",
  "megapixels": 48,
  "relativeAltitude": 10.01,
  "shutterSpeed": "0.06666666667",
  "fileSize": 8291434,
  "fNumber": 2.8,
  "flightPitchDegree": -7.3,
  "contrast": 0,
  "gimbalYawDegree": -73.9,
  "exposureTime": "0.06666666667",
  "gPSLongitude": 8.50589677777778,
  "focalLengthIn35mmFormat": "24",
  "fOV": "73.7398575770812",
  "focalLength35efl": 24,
  "scaleFactor35efl": 5.33333333333333,
  "createDate": 1664471582000,
  "exposureCompensation": "1",
  "imageHeight": 6000,
  "lightValue": 5.87774424987687,
  "aperture": 2.8,
  "gimbalPitchDegree": 0,
  "fileType": "JPEG",
  "flightRollDegree": -4.5,
  "focalLength": "4.5"
}
```


## List media per day count

**Request:** GET `/v1/media/{client_user_id}/grouped-medias`

Headers: 'X-API-KEY: {X-API-KEY}',

**Description:** Gets media count per day. Received start/end date can be used as a time range for other API. SSO-Link must be present.

### Example response


```
[
  {
```

```
    "endDate": 1664496000000,
    "startDate": 1664409600000,
    "count": 27
  },
  {
    "endDate": 1666224000000,
    "startDate": 1666137600000,
    "count": 41
  },
  {
    "endDate": 1668816000000,
    "startDate": 1668729600000,
    "count": 501
  },
  {
    "endDate": 1664323200000,
    "startDate": 1664236800000,
    "count": 7
  }
]
```


## List media in a time range

**Request:** GET `/v1/media/{client_user_id}/medias`

Headers: 'X-API-KEY: {X-API-KEY}'

Query params:
start_time - Long

end_time - Long

**Description:** Gets all the media in time range. Timestamps must be provided as UTC timestamps in milliseconds. A maximum of 100 media will be returned. Medias will be ordered by time when flight was executed (latest first). Flight data will include the metadata as well as the point where the media has been produced. SSO-Link must be present.

### Example response


```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
```

```
      "properties": {
        "fileName": "d2786a9c3eb7f5e93ba0645f946ba0bdd2b33cf6.JPG",
        "createTime": 1668783582000,
        "panoramaId": null,
        "name": "DJI_0147.JPG",
        "missionName": null,
        "pathBlob": "media/2022-11-18/",
        "missionGuid": null,
        "containerBlobName": "user01234",
        "logId": "Nov_18_2022_14_54",
        "id": 395,
        "timestampStartMission": null,
        "downloadMediaLink":
"https://{server_subdomain}.droneharmony.com/v1/download/?key=5c0dde..", //always
OCTET_STREAM
        "downloadMediaThumbnailLink":
"https://{server_subdomain}.droneharmony.com/v1/download/?key=5c0dde..", //with original
content type
        "viewMediaLink":
"https://{server_subdomain}.droneharmony.com/v1/download/?key=5c0ddeea.." //with original
content type
      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          8.13353511111111,
          47.245297
        ]
      }
    },
    ..
}
```

## Delete

**Request:** DELETE `/v1/media/{client_user_id}/{media_id}/delete`

Headers: ‘X-API-KEY: {X-API-KEY}’

**Description:** Delete single media file. SSO-Link must be present.

### Example response

```
File deleted
```

## Media API usage scenarios

### Download media generated during a Dock mission

**Long polling:** Initiate a List media by filter call with the known mission time range or **taskId** to filter the relevant media and use downloadMediaLink to get octet-stream.

For aggregating all generated media once per day: use List media per day count to obtain the last day's time range and then use it in List media by filter.

**Webhook:** Subscribe for DOCK_TASK_COMPLETED_COPY_TO_S3 trigger providing your S3 bucket configs. After the task completion, Dock uploads the media to the DH server. Subsequently, the media is copied to the specified S3 bucket, automating the retrieval process.

---

# Webhooks

## Subscribe

**Request:** POST v1/webhook

Headers: X-API-KEY

**NOTE: currently webhooks are only relevant if you are using the DH Dock solution.**

**Description:** Webhooks are a mechanism by which the DH server calls the client's server when some action completes. **Note that at the moment, webhooks are only triggered if one of the Team Admins on which the dock is connected has an SSO link created (contact us if you need help with this).**

Webhooks are unique per **clientDestinationUrl** and X-API-KEY. So basically the trigger URL is also the unique identifier by which the webhook is managed in context of a single api user. If you have 2 environments (dev, prod for example) with 2 separate api keys from us - then you can have 2 equal destination urls under each api key.

For simplifying the development there is a manual way to call a webhooks. The webhook in this case will contain some "dummy" data, but in correct json format. See section "Test Webhook Trigger" for more information.

Available triggers:

- FLY_TO_COMPLETED - on dock's fly-to-point flight completed (/v1/dock/flight/fly-to-point);
- TAKEOFF_COMPLETED - on dock's takeoff flight completed (/v1/dock/flight/takeoff);
- DOCK_TASK_COMPLETED - on dock mission is completed;
- DOCK_TASK_COMPLETED_MEDIA_UPLOADED - on dock mission is completed and all media uploaded by dock to DH server;

- DOCK_TASK_COMPLETED_COPY_TO_S3 - on dock mission is completed, all images uploaded by dock to DH server and copied to the client's S3 bucket. Requires S3 trigger configs to be provided upon subscription (see Appendix A1).

  Images will appear in the provided bucket at `{path}/taskId/` folder.

  Straightforward way to generate S3 configs:

  1. **Create dedicated bucket (recommended)**
  2. **Navigate to IAM Dashboard**
  3. **Policy creation:**
     Policies -> Create policy -> Service S3 -> Action allowed: PutObject ->
     -> Resources: Specific, Add ARNs to restrict access, type dedicated bucket (recommended) and path (optional) ->
     -> give some meaningful policy name (e.g. dh_s3_put) and create
  4. **IAM user creation:**
     Users -> Create user -> give some meaningful username (e.g. dh_uploader) ->
     -> attach policies directly, find created policy and attach
  5. **Access key generation:**
     Users -> select created user -> Create access key -> Third-party service -> copy generated keys

### Example request

```
{
  "clientDestinationUrl": "https://example.com/hook",
  "triggers": [
    {
      "triggerType": "DOCK_TASK_COMPLETED_COPY_TO_S3",
      "triggerConfigs": {
        "bucket": "example-bucket",
        "region": "us-west-2",
        "IAMUserAccessKey": "AKIAIOSFODNN7EXAMPLE",
        "IAMUserSecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "path": "root/path",
        "uploadMediaTypes": ["image", "other", "video"]
      }
    },
    {
      "triggerType": "FLY_TO_COMPLETED"
    },
    {
      "triggerType": "TAKEOFF_COMPLETED"
```

```
    },
    {
      "triggerType": "DOCK_TASK_COMPLETED_MEDIA_UPLOADED"
    },
    {
      "triggerType": "DOCK_TASK_COMPLETED"
    }
  ]
}
```

*uploadMediaTypes* - filters copied to S3 media files by type:

- image:  formats like .JPEG etc.
- other: .OBS, .RTK, .MRK, .NAV etc.
- video: .MP4

**Example webhook body**

See webhook BODY examples at Appendix A1.

## List Subscriptions

**Request:** GET `v1/webhook`

Headers: X-API-KEY

**Description:** Returns list of company subscriptions info.

**Example response**

```
[
  {
    "id": 1,
    "webhookSubscriptionId": 1,
    "destination": "http://localhost:8081/wh",
    "triggerType": "DOCK_TASK_COMPLETED_COPY_TO_S3",
    "payload":
"{\"trigger\":\"DOCK_TASK_COMPLETED_COPY_TO_S3\",\"triggerTimestamp\":1708419244911,\"client
UserId\":\"41\",\"status\":\"SUCCESS\",\"payload\":{}}",
    "notificationTitle": "DOCK_TASK_COMPLETED_COPY_TO_S3 - 2024-02-20T08:54:05.244994",
    "createdOn": "2024-02-20 08:54:05",
    "lastAttemptOn": "2024-02-20 08:54:24",
```

```
    "finalResult": "SUCCESS"
  },
  {
    "id": 2,
    "webhookSubscriptionId": 1,
    "destination": "http://localhost:8081/wh",
    "triggerType": "DOCK_TASK_COMPLETED_COPY_TO_S3",
    "payload":
"{\"trigger\":\"DOCK_TASK_COMPLETED_COPY_TO_S3\",\"triggerTimestamp\":1708419880159,\"client
UserId\":\"41\",\"status\":\"SUCCESS\",\"payload\":{}}",
    "notificationTitle": "DOCK_TASK_COMPLETED_COPY_TO_S3 - 2024-02-20T09:04:40.474637",
    "createdOn": "2024-02-20 09:04:40",
    "lastAttemptOn": "2024-02-20 09:04:59",
    "finalResult": "SUCCESS"
  }
]
```


## Delete Subscription

**Request:** DELETE `/v1/webhook/{subscription_id}`

Headers: X-API-KEY - String

Path parameters: subscription_id - integer

**Description:** Deletes subscription by ID

## List Webhook Events

**Request:** POST `/v1/webhook/events`

Headers: X-API-KEY - String

**Description:** Returns list of filtered webhook events. Filter provided in body.

### Example request


```
{
  "webhookSubscriptionIds": [
    2,
    7
  ],
  "from": 1694091556000,
  "to": 1694091556000,
```

```
  "result": [
    "SUCCESS",
    "FAILURE"
  ],
  "pageNumber": 0,
  "pageSize": 25
}
```

### Example response

```
[
  {
    "id": 1,
    "webhookSubscriptionId": 1,
    "destination": "http://localhost:8081/wh",
    "triggerType": "DOCK_TASK_COMPLETED_COPY_TO_S3",
    "payload":
"{\"trigger\":\"DOCK_TASK_COMPLETED_COPY_TO_S3\",\"triggerTimestamp\":1708419244911,\"client
UserId\":\"41\",\"status\":\"SUCCESS\",\"payload\":{}}",
    "notificationTitle": "DOCK_TASK_COMPLETED_COPY_TO_S3 - 2024-02-20T08:54:05.244994",
    "createdOn": "2024-02-20 08:54:05",
    "lastAttemptOn": "2024-02-20 08:54:24",
    "finalResult": "SUCCESS"
  },
  {
    "id": 2,
    "webhookSubscriptionId": 1,
    "destination": "http://localhost:8081/wh",
    "triggerType": "DOCK_TASK_COMPLETED_COPY_TO_S3",
    "payload":
"{\"trigger\":\"DOCK_TASK_COMPLETED_COPY_TO_S3\",\"triggerTimestamp\":1708419880159,\"client
UserId\":\"41\",\"status\":\"SUCCESS\",\"payload\":{}}",
    "notificationTitle": "DOCK_TASK_COMPLETED_COPY_TO_S3 - 2024-02-20T09:04:40.474637",
    "createdOn": "2024-02-20 09:04:40",
    "lastAttemptOn": "2024-02-20 09:04:59",
    "finalResult": "SUCCESS"
  }
]
```

## Test Webhook Trigger

**Request:** POST `/v1/webhook/test`

Headers: X-API-KEY - String

Query params:
subId - Integer - company subscription id

triggerType - String - enum representing possible triggers. Available values : DOCK_TASK_COMPLETED_COPY_TO_S3, DOCK_TASK_COMPLETED, FLY_TO_COMPLETED, TAKEOFF_COMPLETED

**Description:** Calls client server with some test request aimed to ensure communication. For DOCK_TASK_COMPLETED_COPY_TO_S3 trigger uploads test_file.txt to the provided S3 path root. If POST is successful then returns 200, otherwise request error message and code. For failed upload to S3 webhook request's payload will contain S3UploadErrorMessage explaining why upload has failed.

**Example of webhook request client's server will receive:**

1) Success

```
{
  "trigger": "FLY_TO_COMPLETED",
  "triggerTimestamp": 1710167653077,
  "clientUserId": "clientUserId",
  "dockSn": "dockSn",
  "status": "SUCCESS",
  "payload": {}
}
```

2) S3 upload fail

```
{
  "trigger": "DOCK_TASK_COMPLETED_COPY_TO_S3",
  "triggerTimestamp": 1710167653077,
  "clientUserId": "clientUserId",
  "dockSn": "dockSn",
  "status": "FAIL",
  "payload": {
    "S3UploadErrorMessage": "S3 Error details. E.g. 403..."
  }
}
```

# Appendix

## A1

Examples of webhooks client will receive per trigger:

1. DOCK_TASK_COMPLETED_COPY_TO_S3

   **Note:** siteId will only be present in the webhook body if the site with the mission is saved (at the moment of task creation) (*Save site* from the main menu).

```
//SUCCESS
{
  "trigger": "DOCK_TASK_COMPLETED_COPY_TO_S3",
  "triggerTimestamp": 1640995200000,
  "clientUserId": "dh_test",
  "dockSn": "HFBCJ837",
  "status": "SUCCESS",
  "payload": {
    "taskId": "783874KJBFBB",
    "siteId": "guid",
    "missionId": "guid"
  }
}

//FAIL
{
  "trigger": "DOCK_TASK_COMPLETED_COPY_TO_S3",
  "triggerTimestamp": 1640995200000,
  "clientUserId": "dh_test",
  "status": "FAIL",
  "payload": {
    "taskId": "783874KJBFBB",
    "S3UploadErrorMessage":
"software.amazon.awssdk.services.s3.model.S3Exception: The request
signature we calculated does not match the signature you provided. Check
your key and signing method. (Service: S3, Status Code: 403, Request ID:
F08FQWE44YGNC, Extended Request ID:
biwlhR+SPcdIALBjW1wChuJr2lS8bbSFTNBnwfzwS87PeXCP0Kr6Yb0GIvIj9CCshzvK9kOyIQt
a035WAVYz1KZoOPMik3T0uCpxaOb4\u003d)", //check provided S3 credentials
    "siteId": "guid",
    "missionId": "guid"
  }
}
```

2. FLY_TO_COMPLETED

```
{
  "trigger": "FLY_TO_COMPLETED",
  "triggerTimestamp": 1640995200000,
```

```
  "clientUserId": "dh_test",
  "dockSn": "JE56ND78",
  "status": "SUCCESS", //FAIL
  "payload": {
    "flightId": "4f67d7a3-a7bf-4fec-92c3-367dd5b8d6a3"
  }
}
```

3. TAKEOFF_COMPLETED

```
{
  "trigger": "TAKEOFF_COMPLETED",
  "triggerTimestamp": 1640995200000,
  "clientUserId": "dh_test",
  "dockSn": "JE56ND78",
  "status": "SUCCESS", //FAIL
  "payload": {
    "flightId": "4f67d7a3-a7bf-4fec-92c3-367dd5b8d6a3"
  }
}
```

4. DOCK_TASK_COMPLETED

```
{
  "trigger": "DOCK_TASK_COMPLETED",
  "triggerTimestamp": 1640995200000,
  "clientUserId": "dh_test",
  "dockSn": "HFBCJ837",
  "status": "SUCCESS", //FAIL
  "payload": {
    "taskId": "783874KJBFBB",
    "siteId": "guid",
    "detailedStatus": "OK" //REJECTED, FAILED, TIMEOUT, PARTIALLY_DONE,
CANCELLED
  }
}
```

5. DOCK_TASK_COMPLETED_MEDIA_UPLOADED

```
{
  "trigger": "DOCK_TASK_COMPLETED_MEDIA_UPLOADED",
```

```
  "triggerTimestamp": 1640995200000,
  "clientUserId": "dh_test",
  "dockSn": "HFBCJ837",
  "status": "SUCCESS",
  "payload": {
    "taskId": "783874KJBFBB",
    "siteId": "guid"
  }
}
```