



Application Programming Interface (API) Introduction

LAST UPDATE: 17 May 2024

API Overview

Terms Used in this Document

- **DH** - Drone Harmony.
- **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.

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

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:

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).
- In response, the DH API furnishes an Auth Token, possessing a validity of 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: The Client User ID must be unique to each Client User account. Given that these IDs are scoped per Client, there's no overlapping with other Client User IDs.
- Retrievability: The Client Backend should have the capability to retrieve the user with this ID. For instance, utilizing Hash(database_id) won't suffice unless the hash result is retained 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 disseminated to the DH web and mobile clients, nor will they be visible to DH users. The DH Web App provides an option for the user to obliterate 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.

Getting SSO Link status

Request: GET /auth/sso/{client_user_id}

```
{  
  email: String, // if SSO link is present, the email of DH user  
}
```

Deleting 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. Returns general media features.

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 by 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 guid, image source.

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://app.droneharmony.com/v1/download?key=5c0ddeef21d98e...\", //with original content
type
        "name": "DJI_20231110164944_0001_W.jpeg",
        "missionName": "Manual",
        "missionGuid": "cb09a6e1-606e-4c91-ba18-0606aa91e35b",
        "logId": null,
        "viewMediaLink":
"\"https://app.droneharmony.com/v1/download?key=5c0ddeef21d98e4ec5f2ab7b8da89b34a6552d9dae888
930dc75b...\", //with original content type
        "id": 6041,
        "downloadMediaLink":
"\"https://app.droneharmony.com/v1/download?key=5c0ddeef21d98e4ec5f2ab7b8da89b34a6552d9dae888
930dc7...\", //always OCTET_STREAM
        "timestampStartMission": 1699606146282,
        "taskId": "cb09a6e1-606e-4c91-ba18-0606aa91e35b"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          0.000030555555555556,
          -0.000017527777777778
        ]
      }
    }
  ]
}
```


Get media metadata

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

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

Description: Gets all extracted 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": 47.14352633333333,
  "sharpness": 0,
  "imageWidth": 8000,
  "meteringMode": "1",
  "megapixels": 48,
  "relativeAltitude": 10.01,
  "shutterSpeed": "0.066666666667",
  "fileSize": 8291434,
  "fNumber": 2.8,
  "flightPitchDegree": -7.3,
  "contrast": 0,
  "gimbalYawDegree": -73.9,
  "exposureTime": "0.066666666667",
  "gpsLongitude": 8.50589677777778,
  "focalLengthIn35mmFormat": "24",
  "fov": "73.7398575770812",
  "focalLength35efl": 24,
  "scaleFactor35efl": 5.333333333333333,
  "createDate": 1664471582000,
  "exposureCompensation": "1",
  "imageHeight": 6000,
  "lightValue": 5.87774424987687,
  "aperture": 2.8,
  "gimbalPitchDegree": 0,
  "fileType": "JPEG",
  "flightRollDegree": -4.5,
  "focalLength": "4.5"
```

```
}
```

List medias per day count

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

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

Description: Gets medias 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. Maximum 100 medias 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://app.droneharmony.com/v1/download/?key=5c0dde..",
        //always OCTET_STREAM
        "downloadMediaThumbnailLink":
        "https://app.droneharmony.com/v1/download/?key=5c0dde..", //with original content type
        "viewMediaLink": "https://app.droneharmony.com/v1/download/?key=5c0ddeea.." //with
        original content type
      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          8.133535111111111,
          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 use 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 medias 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.

Webhook

Subscribe

Request: POST `v1/webhook`

Headers: X-API-KEY

Description: Upon registration, the client's server endpoint will be notified whenever a relevant event is triggered within the company's user base. The triggers can be activated by: (a) an 'External Company User': This user is registered with a linked 'client_user_id' and is considered the owner of their own team. (b) 'Team Members': These are users who are part of a team owned by an 'External Company User' (they selected the team on login). The notification will contain details of the event that triggered the webhook.

Subscription is unique per company and destination URL. That is, all company's triggers for the same destination URL will have the same subscriptionId even if registered separately. If some trigger has already existed, it will be updated (this especially applies for triggers that come with configs).

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 and all images 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"
      }
    },
    {
      "triggerType": "FLY_TO_COMPLETED"
    },
    {
      "triggerType": "TAKEOFF_COMPLETED"
    }
  ]
}
```

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":
  "{\n\"trigger\":\n\"DOCK_TASK_COMPLETED_COPY_TO_S3\",
  \"triggerTimestamp\":1708419244911,\n\"client
  UserId\":\n\"41\",
  \"status\":\n\"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":
  "{\n\"trigger\":\n\"DOCK_TASK_COMPLETED_COPY_TO_S3\",
  \"triggerTimestamp\":1708419880159,\n\"client
  UserId\":\n\"41\",
  \"status\":\n\"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,
  "clientId": "clientId",
  "dockSn": "dockSn",
  "status": "SUCCESS",
  "payload": {}
}
```

2) S3 upload fail

```
{
  "trigger": "DOCK_TASK_COMPLETED_COPY_TO_S3",
  "triggerTimestamp": 1710167653077,
  "clientId": "clientId",
  "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,
  "clientId": "dh_test",
  "dockSn": "HFBCJ837",
  "status": "SUCCESS",
  "payload": {
    "taskId": "783874KJBFBB",
    "siteId": "guid",
    "missionId": "guid"
  }
}

//FAIL
{
  "trigger": "DOCK_TASK_COMPLETED_COPY_TO_S3",
  "triggerTimestamp": 1640995200000,
  "clientId": "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:
biw1hR+SPcdIALBjW1wChuJr2lS8bbSFTNBnwfzwS87PeXCP0Kr6Yb0GivIj9CCshzvK9k0yIQta035WAVYz1KZoOPMik3T0uCpxa0b4\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"  
  }  
}
```

}