# mydlink Open API Specification

Version 1.2.2

## Revision History

| Version | Publish Date | Author | Descriptions |
|---|---|---|---|
| 0.1 | Jan. 28, 2011 | Edward Wang | 1. initial Draft. |
| 0.2 | Feb. 16, 2011 | Edward Wang | 1. add the information access API. |
| 0.3 | Mar. 01, 2011 | Edward Wang | 1. fix the typo of the example in 2.b.2.2. |
| 0.4 | Feb. 19, 2013 | Jason Syu | 1. add 'online' & 'device_name' fields to 'device listing (4.b.1)' API.<br>2. add 'device_name', 'online', 'auth_key', 'signal_addr', 'fw_ver', and 'agent_ver' fields to 'more device information (4.b.4)" API.<br>3. add 'manage app token (3.a.4)' API.<br>4. add 'set device ordering (4.b.6)' API.<br>5. add 'f/w upgrade (4.b.7)' API. |
| 0.4.1 | Feb.26, 2013 | Jason Syu | 1. correct json format error of 'f/w upgrade (4.b.7)' API |
| 0.5 | Apr. 01, 2013 | Jason Syu | 1. revise the value definition of 'app_type' (3.a.4) |
| 0.6 | May 09, 2013 | Jason Syu | 1. add 3.a.5 for mydlink service control<br>2. add Appendix-I HTTP error code handling<br>3. add Appendix-II the device capability table |
| 0.7 | May 24, 2013 | Jason Syu | 1. add 'sig' & 'timestamp' fields to requests (2.a.1, 2.b.1.2, 2.b.2.1, 2.b.2.2)<br>2. hash the user password in exchange (2.b.2.2)<br>3. put more info for redirect behavior (2.b.2.2)<br>4. add req_region field (2.b.2.2)<br>5. add field for changing password (3.a.3)<br>6. add h/w version of device (4.b.1)<br>7. remove unnecessary mac field (4.b.4)<br>8. add Appendix III & Appendix IV |
| 0.7.1 | May 27, 2013 | Jason Syu | 1. revise info carried in the redirect action (2.b.2.1, 2.b.2.2) |
| 0.8 | May 28, 2013 | Jason Syu | 1. added 'code' field to error response.<br>2. add error code definition to Appendix-I.<br>3. set the hash algorithm from SHA1 to MD5.<br>4. remove [app_id] field of service setting API (3.a.5) |
| 0.8.1 | Jun. 04, 2013 | Jason Syu | 1. validate the request path (start with /me)<br>2. revise 3.a.1 for language setting<br>3. set the email field readonly (3.a.3) |
| 0.8.2 | Jun. 24, 2013 | Jason Syu | 1. add 'api_target' field (2.b.2.1, 2.b.2.2) |
| 0.8.3 | Jul. 04, 2003 | Mac Mac | 1. add app supporting capacity (Appendix II) |
| 0.8.4 | Aug. 26, 2013 | Jason Syu | 1. add field 'email_verified' and 'account_expired' (2.b.2.2)<br>2. add API 3.a.4 for sending confirmation mail. |
| 0.9 | Aug. 29, 2013 | Jason Syu | 1. add service linking api (3.a.5) |

| | | | |
|---|---|---|---|
| 1.0 | Aug 30, 2013 | Jason Syu | 1. revising authentication & authorization API (2.a, 2.b, 2.c)<br>2. add application registration form (Appendix VI, VII)<br>3. add API for 'connect' with external services |
| 1.0.1 | Oct. 14, 2013 | Jason Syu | 1. revise binding request for device with legacy f/w (4.b.2) |
| 1.0.2 | Feb. 10, 2014 | Jason Syu | 1. add query user info by access_token (3.a.6) |
| 1.0.3 | Feb. 17, 2014 | Jason Syu | 1. add soft restriction that the numbers of info to be queried should be not over 3 devices. (4.b.4). Client is suggested to query device info on daemon instead querying all at a time. |
| 1.1 | Feb. 26, 2014 | Jason Syu | 1. add chapter 5, logging support. (5.a.1)<br>2. add (4.b.8) for device statistics query. |
| 1.1.1 | Mar. 03, 2014 | Jason Syu | 1. add new capabilities (18,19,20,21,22) to Appendix II.<br>2. remove unsupported sections. (2.b.2.2, 4.c.1) |
| 1.1.2 | Mar. 10, 2014 | Jason Syu | 1. add Appendix-VIII to guide the access token usage.<br>2. add 'lang' field to API (2.b.1.1)<br>3. add 'target_site' and 'api_site' fields to the response of (2.b.1.1) |
| 1.1.3 | Apr. 08, 2014 | Jason Syu | 1. tune fields in (3.a.3) & (3.a.6) to match (3.a.1)<br>2. update error code (Appendix I)<br>3. add account deletion definition (3.a.2)<br>4. validate the content-type of the openapi message to 'application/json'<br>5. add preferences setup api (3.a.7), (3.a.8), (4.b.9), and (4.b.10) |
| 1.1.4 | Aug. 05, 2014 | Jason Syu | 1. correct error code type to int.<br>2. add 'receive_news' field (3.a.1)<br>3. add 'add_dialog' API (3.a.9)<br>4. tune the description of 302 redirection handling in (2.b.1.1, 2.b.2.1, 2.d.1)<br>5. tune oauth flow dialog (2.b.1), (2.b.2)<br>6. add 'module_info' field in device info response (4.b.4) |
| 1.1.5 | Sep. 29, 2014 | Jason Syu | 1. add API for set device name |
| 1.1.6 | Mar. 03, 2015 | Jason Syu | 1. add utc_offset field to (4.b.4)<br>2. add lvr_info field to (4.b.4) |
| 1.2.0 | Mar. 16, 2015 | Jason Syu | 1. add cnvr recording api, section 6.<br>2. update (4.c.2) for cnvr service subscription. |
| 1.2.1 | Aug. 10, 2015 | Jason Syu | 1. add more fields on account creation (3.a.1)<br>2. add password field for delete account (3.a.2)<br>3. more fields to set of account info (3.a.3)<br>4. carry more fields in account info (3.a.6)<br>5. add forgot password flow (3.a.10)<br>6. add change email API (3.a.11)<br>7. add device password change flag 'verified' (4.b.4)<br>8. add force and manual f/w upgrade flag (4.b.4)<br>9. add 'target' field in streamer block of lvr info (4.b.4)<br>10. add more error codes (Appendix I) |

| | | | |
|---|---|---|---|
| 1.2.2 | Nov. 30, 2015 | Jason Syu | 1. revise target field definition in streamer of lvr_info (4.b.4)<br>2. add event filter field in event subscription (4.c.2)<br>3. remove cnvr settings from (4.c.2)<br>4. remove cnvr API to separate doc (6.) |

# 1. Overview

This specification describes the details of mydlink Open Access API to enable the 3rd party development to integrate mydlink features into their applications. It presents a simple, consistent view of mydlink device-ownership graph, uniformly representing objects with the real-time status in the graph and the corresponding access mechanism. All communication between the application and mydlink shall be through the secure http connection (https).

# 2. Authentication

mydlink Open Access API follows the OAuth 2.0 recommendation to implement the application and user authentication/authorization in order to provide the flexible and secure access control to the end user.
Before the application development, the development party has to register the application with mydlink management team to get the application identifier and application privilege code. (refer to Appendix VII)

## 2.a Application Authentication

Each application needs to pass the authentication process and get the application access token before it can access the other administrative features, such as creating a user account, deleting a user account, or retrieve analysis of the application etc., of mydlink Open Access API.

### *2.a.1 Get the access token*

The application can obtain the `access_token` from mydlink Open Access API token endpoint at `https://api.mydlink.com/oauth/access_token` by specifying the application identifier and `grant_type` parameter with a value of `app_credential`. This request must carry signature for authentication. Refer to Appendix-III for more details.

```
GET /oauth/access_token?client_id=[CLIENT_ID]&grant_type=app_credential&
             timestamp=[CURRENT_TIME_SEC]&sig=[SIGNATURE] HTTP/1.1
Host: api.mydlink.com
Accept: */*
```

Sending the GET request to the above URL will return an `access_token` in the body of the response. The application then can use this returned `access_token` to call the restricted APIs of mydlink OpenAPI.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "access_token":"SlAV32hkKG",            # access_token for access mydlink resource
  "expires_in":3600,                       # valid seconds of the access_token
}
```

If the authentication fails or invalid parameters are carried, mydlink platform will reply HTTP 400 with the error content in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
```

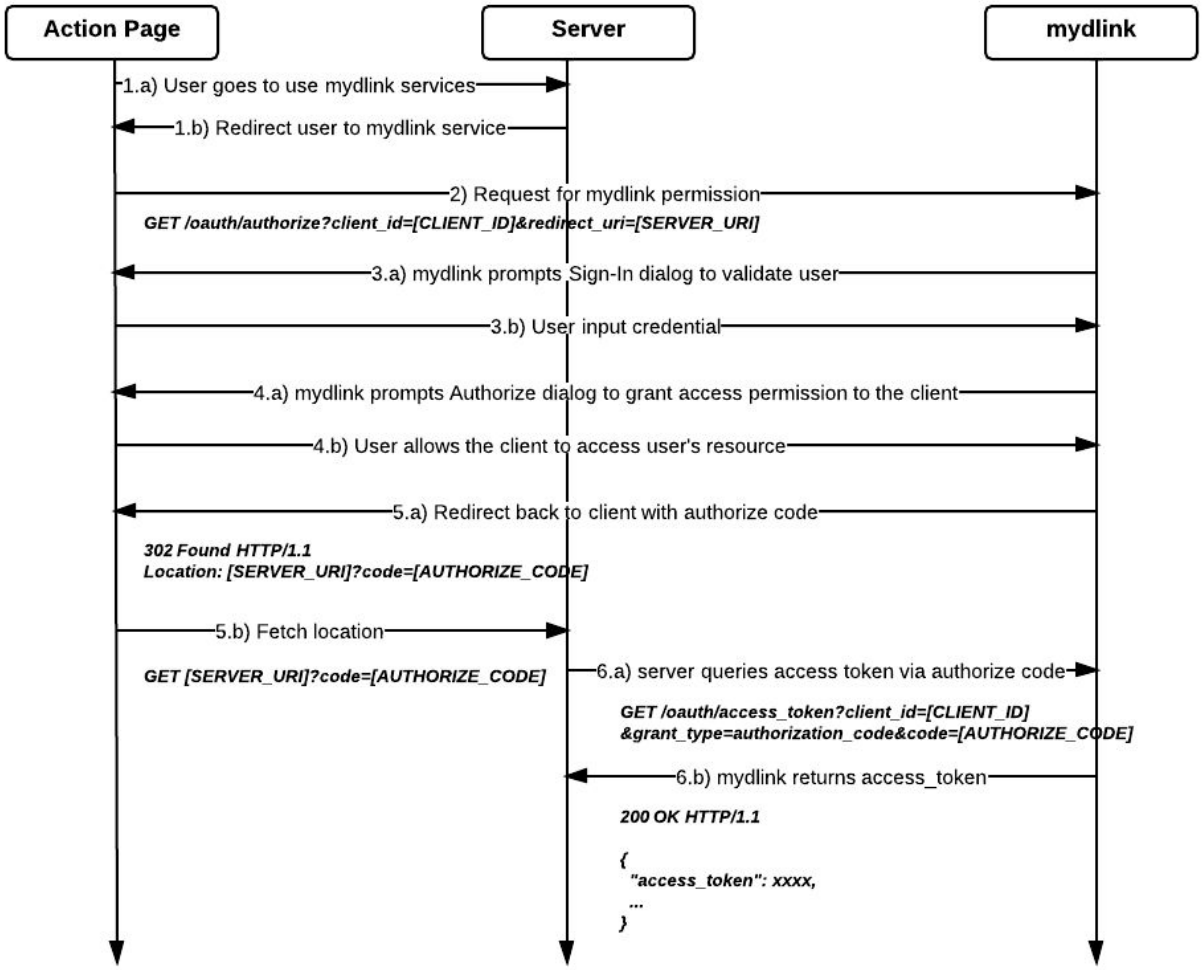
```
  "error": {
    "type": "OAuthException",
    "code": 10,
    "message":"Error validating this request."
  }
}
```

## 2.b Authentication for User Permission

mydlink Open Access API supports two different OAuth 2.0 flows for user login: server-side and client-side. The server-side flow is used whenever you need to call the Open Access API from your web service. The client-side flow is used when you need to make calls to the Open Access API from a client, such as JavaScript running in a Web browser or from a native mobile or desktop application.

### *2.b.1 Server-side Authentication*

The diagram below shows the message flow, between end-user, 3rd party web service, and mydlink platform.




### 2.b.1.1 Application Authorization

Before the application can access the user resources, it needs to be authorized by the end user. User authentication and app authorization are handled at the same time by redirecting the user to mydlink Sign-In Dialog. When invoking this dialog, the application must pass in the `client_id` that is assigned by mydlink and the URL that the user's browser will be redirected back to once app authorization is completed (the `redirect_uri` parameter). The `redirect_uri` must be within the same domain as the application site URL and in the uri list of the application registration table. `state` field is an optional field for client to carry its session information. The `state` field will be carried back in the response of the authorize request. The application will send GET request below to mydlink platform to get the authorization code. The `lang` field can be set for prompt sign-in dialog. (refer Appendix-V for language code)

```
GET
       /oauth/authorize?client_id=[CLIENT_ID]&redirect_uri=[APP_URI]&response_type=c
       ode&scope=basic&state=[CLIENT STATE]&hint=[USER_ACCOUNT]&lang=[LANGUAGE]
```

```
      HTTP/1.1
Host: api.mydlink.com
Accept: */*
```

Once the end user signs in successfully, the Sign-In Dialog will prompt the user to authorize the app. If the user grants the application to access resources, the user will be redirected to the URL specified in the `redirect_uri` parameter with the authorization code. The 3rdparty server shall handle below message at the specified `redirect_uri`.

```
GET [APP_URI]?code=[AUTHORIZATION_CODE]&expires_in=[EXPIRE_SEC]&state=[CLIENT
STATE]&target_site=[PORTAL_SITE]&api_site=[API_SITE]
Host: [host portion of APP_URI]
```

Note that the app shall send the following requests to the location specified in `api_site` field.

If the user doesn't grant the access, mydlink OpenAPI service will do the same way with error message.

```
GET [APP_URI]?state=[CLIENT
STATE]&error=%7B%22type%22%3A%22OAuthException%22%2C%22code%22%3A%2212%22%2C%22messa
ge%22%3A%22User%20denied.%22%7D%0A
Host: [host portion of APP_URI]
```

#### 2.b.1.2 Grant Access Token from Authorized Code

After getting the authorization code, the application shall conduct the application authentication to gain the privilege to access the restricted resources. It can obtain the `access_token` from mydlink Open Access API token endpoint at `https://api.mydlink.com/oauth/access_token` by specifying the `client_id`, `client_secret`, and authorization code. *(refer Appendix III for sig value generation)*

```
GET
/oauth/access_token?client_id=[CLIENT_ID]&grant_type=authorization_code&code=[AUTHOR
IZATION_CODE]&timestamp=[current time]&sig=[signature] HTTP/1.1
Host: api.mydlink.com
Accept: */*
```

If the carried authorize code is valid, mydlink platform will carry `access_token` and `exipires_in` fields in the response.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
 "access_token":"SlAV32hkKG",            # access_token for access user's resource
 "expires_in":3600,                      # valid seconds of the access_token
```

```
 "refresh_token": "hjs723h72h3a",        # refresh_token to get the new access_token
}
```

If the authentication fails, mydlink platform will reply HTTP 400 with the error message in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "OAuthException",
    "code": 10,
    "message":"Error validating this request."
  }
}
```

### 2.b.1.3 Grant Access Token from Refresh Token

Once the `access_token` expired, client can get the valid `access_token` from the '`refresh_token`' returned in 2.b.1.2. The refresh endpoint is as below:

```
GET
/oauth/access_token?client_id=[CLIENT_ID]&grant_type=refresh_token&code=[REFRESH_TOK
EN]&timestamp=[current time]&sig=[signature] HTTP/1.1
Host: api.mydlink.com
Accept: */*
```

If the carried refresh_token is valid, mydlink platform will carry `access_token` and `exipires_in` fields in the response.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
 "access_token":"SlAV32hkKG",            # access_token for access user's resource
 "expires_in":3600,                      # valid seconds of the access_token
}
```

If the refresh_token is invalid, mydlink platform will reply HTTP 400 with the error message in the body of the response.

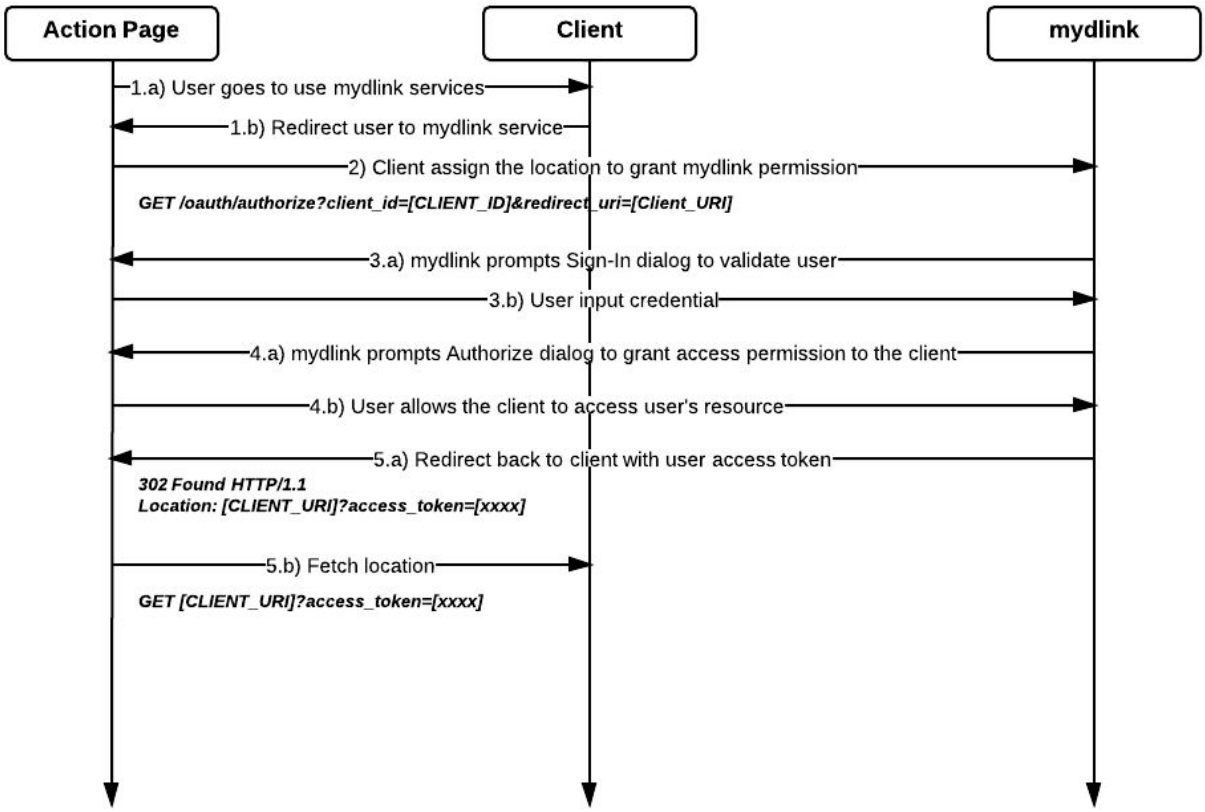
```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "OAuthException",
    "code": 10,
```

```
    "message":"Error validating this request."
  }
}
```

### *2.b.2 Client-side User Authentication*

The application running on the client side has to possess the simple browsing capability to deal with the Sign-In Dialog to get the `access_token` in one request.

#### 2.b.2.1 Authorization for Application with browser capability




This type of application also uses the OAuth Dialog for user authentication and app authorization. The only difference is that you must specify the `response_type` parameter with a value of **token**.

```
GET
/oauth/authorize?client_id=[CLIENT_ID]&redirect_uri=[APP_URI]&response_type=token&st
ate=[CLIENT STATE]&scope=basic&hint=[USER_ACCOUNT] HTTP/1.1
Host: api.mydlink.com
Accept: */*
```

Once the end user signs in successfully, the Sign-In Dialog prompts for user to authorize the app. If the user allows the application to access resources, the Sign-In Dialog will redirect the client to the URL in the `redirect_uri` parameter with the `acces_token` directly. The value of `redirect_uri` must be in the uri list of the application registration table. The application shall wait to handle the message below:

```
GET [APP_URI]?access_token=[ACCESS_TOKEN]&expires_in=[EXPIRE_TIME]
&target_site=[TARGET_LOCATION_OF_PORTAL_SERVICE]&api_site=[TARGET_LOCATION_OF_OpenAP
I_SERVICE]
Host: [host portion of APP_URI]
```

It's important that the client SHALL send the further requests to the location as `target_site` field specified.

If the authentication fails, the same behavior will be applied with error message.

```
GET
[APP_URI]?error=%7B%22type%22%3A%22OAuthException%22%2C%22code%22%3A%2212%22%2C%22me
ssage%22%3A%22User%20denied.%22%7D%0A
Host: [host portion of APP_URI]
```

#### 2.b.2.2 Authentication for Standalone Application

Not supported. Check **2.b.2.3** for the authorization process of standalone type application.

#### 2.b.2.3 Authentication for Standalone Application

The authentication of this type application is similar as 2.b.1. The difference is the redirect_uri field MUST be in the form of "`http://localhost:`**`PORT`**".

## 2.c Revocation of Authorized and Authenticated Client

### 2.c.1 Access token revocation

Client can revoke the given `access_token` with the following command, to act as 'Sign-Out' user from the client application.

```
GET
/oauth/revoke?client_id=[CLIENT_ID]&access_token=[ACCESS_TOKEN]&revoke_type=token
HTTP/1.1
Host: api.mydlink.com
Accept: */*
```

If the request is valid, the mydlink platform replies the message as:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": {
    "result": "success"
  }
}
```

If the authentication fails, mydlink platform will reply HTTP 400 with the error message in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "OAuthException",
    "code": 14,
    "message":"Access token invalid."
  }
}
```

### 2.c.2 Authorization revocation

Client can revoke the grant permission with the following command. The authorization step will be proceeded after the revocation process.

```
GET /oauth/revoke?client_id=[CLIENT_ID]&access_token=[ACCESS_TOKEN]&revoke_type=code
HTTP/1.1
Host: api.mydlink.com
Accept: */*
```

If the request is valid, the mydlink platform replies the message as:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": {
    "result": "success"
  }
}
```

If the authentication fails, mydlink platform will reply HTTP 400 with the error message in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "OAuthException",
    "code": 14,
    "message":"Access token invalid."
  }
}
```

## 2.d Integration of 3rd Party Services

In this section, APIs are provided for integration with 3rd party services via oAuth interaction. mydlink service acts as the gateway between client application and target 3rd party services. To invoke APIs in this section, the client has to grant app level access token first. (refer to section 2.a)

### 2.d.1 Sign-In with 3rd party service

After granted the `access_token`, the client can issue the request to start the sign-in process via 3rd party services. Note that the `access_token` here can be the application level token granted in 2.a.1 or from mydlink sign-in process. The value of `service` and `redirect_uri` must be obeyed what Appendix-VI and Appendix-VII defines. The `policy` field defines the action when the 3rd party account doesn't exist in mydlink service. The value of the policy can be "`default`" that mydlink service will respond fail if the account doesn't exists in mydlink service, or "`create`" that mydlink service will create and associate the 3rd party account with mydlink implicitly if the account doesn't exist in mydlink service.

```
GET
/oauth/connect?client_id=[CLIENT_ID]&access_token=[ACCESS_TOKEN]&redirect_uri=[APP_U
RI]&service=[SERVICE]&policy=[ACCOUNT_POLICY]&bind_info="%7B%20%22mac%22%3A%22F07D68
022D93%22%2C%20%22mydlink_id%22%3A%2230038291%22%2C%20%22session_id%22%3A%2239847184
729%22%20%7D%0A" HTTP/1.1
Host: api.mydlink.com
Accept: */*
```

If the request is valid and the specified 3rd party service is supported, the mydlink platform will proceed the oAuth procedure with 3rd party service. Several page redirections may occur between 3rd party service and mydlink service.
The possible cases may happen:

1. **User doesn't have account of the specified service:**
   1.1. User have to go to that 3rd party service and create an account manually.
2. **User doesn't have account in mydlink service:**
   2.1. A sign-in page of the specified service will be provided for user to input the

credential.

2.2. An authorize page may show to ask user to grant access permission of that platform to mydlink service.

2.3. If failed to sign-in or authorize denied by user, mydlink service will respond fail.

2.4. If the `policy` field is "`create`", a mydlink account will be created automatically and associated with the 3rd party account. The 3rd party user email or unique-id will be used for mydlink account creation.

2.5. If the `policy` field is "`default`", the fail is responded.

**3. User already has account of mydlink service:**

3.1. A sign-in page of the specified service will be provided for user to input the credential.

3.2. An authorize page may show to ask user to grant access permission of that platform to mydlink service.

3.3. A sign-in page of mydlink will show up. It's to validate that user of the specified service is the same one in mydlink service, to complete the account association.

3.4. If the carried `access_token` is the user-level token, then mydlink will automatically associate the 3rd party account with the `access_token` owner without going 3.3.

Once the process completes, mydlink service will respond the success (or fail if user disallow/cancel the operation) to the specified `redirect_uri` with error status. The client shall be able handle the message at specified `redirect_uri`.

```
GET
[redirect_uri]?access_token=[ACCESS_TOKEN]&expires_in=[EXPIRE_SEC]&refresh_token=[RE
FRESH_TOKEN]&service=[SERVICE]&info=[CONNECTED_SERVICE_INFO]
Host: [host portion of redirect_uri]
```

The `info` field carries the info of the 3rdparty service, in json format that urlencoded, such as:

"%7B%22uid%22%3A%221234567890%22%2C%22access_token%22%3A%221234567890%22%7D"

Instead, if user cancel the sign-in action, mydlink service responds as:

```
GET
[redirect_uri]?error=%7B%22type%22%3A%22OAuthException%22%2C%22code%22%3A%2224%22%2C
%22message%22%3A%22User%20cancels.%22%7D%0A&service=[SERVICE]
Host: [host portion of redirect_uri]
```

The value of the error field is an url-encoded json message.

If user denis in the authorization page, the result will be:

```
GET
```

```
[redirect_uri]?error=%7B%22type%22%3A%22OAuthException%22%2C%22code%22%3A%2212%22%2C
%22message%22%3A%22User%20denied.%22%7D%0A&service=[SERVICE]
Host: [host portion of redirect_uri]
```

The caller client shall hook handler in provided `redirect_uri`, and reacts based on what kind of request is received.

For role, such as setup wizard, can use the API to finish account sign-in & device binding in a request, via carry binding info in `bind_info` field. The value of this field shall include information as below and be url-encoded.

```
{
  "mac":"F07D68022D93",         # MAC address of the device to be bound
  "mydlink_id":"30038291",      # mydlink number of the device
  "session_id":"39847184729"    # the 'footprint' value got from device agent
}
```

# 3. Application-related API

## 3.a Account Management

### *3.a.1 Create an user account*

The application can create a user account via POST the user information to the following interface after it got the `access_token`. Please refer **2.a** for the detail.

```
POST /me/user/add?access_token=[APP_ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    "email": "johnjones@dlink.com",      # mandatory
    "password": "asd923hc02"             # mandatory
    "first_name": "John",                # mandatory
    "last_name": "Jones",                # mandatory
    "language": "en",                    # refer to Appendix V. mandatory
    "nickname": "johnyboy",
    "receive_news": true,                # subscribe news from D-Link
    "country": "TW"                      # follow ISO 3166-1 alpha-2 definition
  }
}
```

If the information is validated, mydlink platform will create the user account and provide the `access_token` in the body of the response directly to simplify the following user-related operations.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data":{
    "access_token":"u39vljsf920447284",
    "expires_in":3600,
  }
}
```

If there is an issue creating the user account, mydlink platform will reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "User",
    "code": 14,
    "message": "Account duplicated."
  }
}
```

### *3.a.2 Delete an user account*

The authorized application can delete account via POST request below.

```
POST /me/user/delete?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    "email": "johnjones@dlink.com",
    "password": "c8837b23ff8aaa8a2dde915473ce0991" # user password to verify
                                                  # md5 hash of user password
  }
}
```

Note that the given email address must be same as the token associated, and all devices bound to the given account must be removed first.

If the information is validated, mydlink platform will delete the user account associated with `access_token` and responds:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data":{
    "result":"success"
  }
}
```

If there is issue deleting the user account, mydlink platform will reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "User",
    "code": 26,
    "message": "Need to unbind all devices first."
  }
}
```

### *3.a.3 Update account info*

The application has to get the `access_token` before it can update the user account information. It can POST the user information to the following interface with the `access_token`.

```
POST /me/user/update?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    "email": "johnjones@gmail.com", # read-only for current version
    "password": "umfruper1",
    "first_name": "John",
    "last_name": "Jones",
    "nickname": "johnyjon",
    "language": "en",
    "receive_news": false,
    "country": "TW"
  }
}
```

If the update is completed successfully, mydlink platform will reply HTTP 200 OK with the updated fields.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": {
    "email": "johnjones@gmail.com",
    "first_name": "John",
    "last_name": "Jones",
    "nickname": "johnyjon",
    "language": "en",
    "receive_news": false,
    "country": "TW"
  }
}
```

If there is an issue updating the user account, mydlink platform will reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "User",
    "code": 15,
    "message": "Field format invalid"
  }
}
```

### *3.a.4 Send activation email to user*

The API provides ability for client to send out the confirmation email to activate an account.

```
GET /me/user/activate_email?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json
```

If the token is validated, mydlink platform shall send confirmation email to user if the user isn't verified yet and respond success.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": {
    "result": "success"
  }
}
```

If there is an issue in sending the notification, mydlink platform shall reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Application",
    "code": 13,
    "message": "Access token invalid."
  }
}
```

### *3.a.5 Connect user account with external service*

The API provides ability for client to link the user account with given 3rd party account. After the linking process, user can use 3rd party account to sign-in the mydlink service and access the resource of the linked mydlink user.

```
POST /me/user/connect?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    "service": "3rdparty service",               # the supported 3rdparty service
name
    "redirect_uri": "http://192.168.0.1/result", # the result handling page
  }
}
```

If the token is validated and given parameters are correct, mydlink responds the success result as:

```
HTTP/1.1 302 Found
Location: [redirect_uri]?access_token=[ACCESS_TOKEN]
```

If there is an issue in account linking, mydlink platform shall reply the error code in the response. The possible errors of the request may be 'invalid access token', 'unsupported service', and 'already linked with different id'. Check appendix I for more details.

```
HTTP/1.1 302 Found
Location:
[redirect_uri]?error=%7B%0A%20%20%22type%22%3A%20%22Application%22%2C%0A%20%20%22cod
e%22%3A%20%2213%22%2C%0A%20%20%22message%22%3A%20%22Access%20token%20invalid.%22%0A%
7D
```

### *3.a.6 Get user information*

Client can get user information to the following interface with the `access_token`.

```
GET /me/user/info?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json
```

If the `access_token` is valid, mydlink platform will reply HTTP 200 OK with user's information.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": {
    "email": "johnjones@gmail.com",
    "first_name": "John",
    "last_name": "Jones",
    "nickname": "johnyjon",
    "language": "en",
    "receive_news": false,
    "country": "TW",
    "email_verified": true,             # is the account email verified
    "account_expired": false,           # not verified and exceed available window
    "expired_at": 1436151624            # seconds from 1970-01-01
  }
}
```

If the given `access_token` is invalid, mydlink platform will reply an HTTP 400 with the error

in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Application",
    "code": 13,
    "message": "Access token invalid."
  }
}
```

### *3.a.7 Set user preferences*

Client can store its preferences for the user via this API. mydlink platform provides **N** user attributes for each client application, and max **M** bytes for each content.

For this version, N and M are set as **20** and **50K** respectively.

```
POST /me/user/set_prefs?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": [
    {
      "no": 1,                                  # attribute number (1 to N)
      "tag": "<attribute name>",                # tag name, application defined
      "value": "<attribute value>"              # tag value, application defined.
    },
    {
      "no": 5,
      "tag": "<attribute name>",
      "value": "<attribute value>"
    }
  ]
}
```

Note that the value must be in string. To store binary value, the data shall be **Base64** encoded first.

If the request is valid and proceeded successfully, `success` replies from mydlink platform.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": {
    "result": "success"
  }
}
```

If the given `no` is outside the acceptable range (1-N), mydlink platform will reply an HTTP 400

with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Application",
    "code": 10,
    "message": "Error validating this request."
  }
}
```

### *3.a.8 Get user preferences*

Client can query the stored user preferences via the API:

```
POST /me/user/get_prefs?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    "prefs": [
      "attr A",                              # attribute name app defined.
      "attr E"                               # attribute name app defined.
    ]
  }
}
```

If the request and the given access_token is valid, client can get the response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": [
    { "no": 1, "tag": "<attr A>", "value": "<val>" },
    { "no": 4, "tag": "<attr E>", "value": "<val>" }
  ]
}
```

If the given request is invalid, the message responded as:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Application",
    "code": 10,
    "message": "Error validating this request."
  }
```

```
}
```

### *3.a.9 Request create account dialog*

The application can request the sign-up dialog for account creation. When invoking this dialog, the application must pass `access_token` retrieved from 2.a.1 and the URL which user's browser to be redirected to once account creation is completed (the `redirect_uri` parameter). The `redirect_uri` must be within the same domain as the application site URL and also in uri list of the application registration table. `state` field is an optional field for client to carry its session information, and will be carried back in the response of the result. The `lang` field can be set to specify the language of sign-up dialog. (refer Appendix-V for language code). The `dtype` field is used to specify the style of the dialog page. Supported types are "`mobile`" and "`wizard`".

```
GET
       /me/user/add_dialog?access_token=[APP_ACCESS_TOKEN]&redirect_uri=[APP_URI]&st
       ate=[CLIENT_STATE]&lang=[LANGUAGE]&dtype=[DIALOG_TYPE] HTTP/1.1
Host: api.mydlink.com
Accept: */*
```

Once the sign-up process is success, the page will be redirected to the URL in the `redirect_uri` parameter with the `access_token` of the user. Client shall handle the below message in the given `redirect_uri`

```
GET
[APP_URI]?state=[CLIENT_STATE]&access_token=[ACCESS_TOKEN]&expires_in=[EXPIRE_SEC]
Host: [APP_HOST]
```

### *3.a.10 forgot password*

User can request the sign-up dialog to apply for password-reset request. The `dtype` field is used to specify the style of the dialog page.
Once user apply the password-reset request, system will send password-reset email to user.

```
GET /me/user/forgot_pwd?client_id=[CLIENT_ID]&lang=[LANGUAGE]&dtype=[DIALOG_TYPE]
       HTTP/1.1
Host: api.mydlink.com
Accept: */*
```

### *3.a.11 Change user email*

The application has to get the `access_token` before it can request change email. It can POST the user information to the following interface with the `access_token`.

```
POST /me/user/change_email?access_token=[ACCESS_TOKEN] HTTP/1.1
```

```
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    "email": "johnjonesnew@dlink.com",              # New email address
    "password": "c8837b23ff8aaa8a2dde915473ce0991" # user password to verify, it's
                                                    # md5 hash of user password
  }
}
```

Note that the given password must be same as the token associated account password.

If the information is validated, mydlink platform will send an email to the user and respond success:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data":{
    "result":"success"
  }
}
```

If there is issue processing the request, mydlink platform will reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "User",
    "code": 15,
    "message": "Account existed."
  }
}
```

## 4. Objects

To access the objects associated with the user account, the application first has to pass the user authentication, application authorization, and application authentication steps to get the `access_token`.

### 4.a Model

#### *4.a.1 Listing*

The application can retrieve the supported device models after it passes the application authentication. It can GET the device information to the following interface with the `app_access_token`.

```
GET /me/model/list?access_token=[APP_ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
```

If the `app_access_token` is valid, mydlink platform will reply HTTP 200 OK with the model list.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": [
    {"device_model": "DCS-1100L"},
    {"device_model": "DCS-5230L"},
    {"device_model": "DCS-930L"},
    {"device_model": "DCS-1130L"}
  ]
}
```

If there is an issue looking up the access_token, mydlink platform will reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Application",
    "code": 13,
    "message": "Access token invalid."
  }
}
```

#### *4.a.2 Model Information*

The application can retrieve the model information for the related resources. It can POST the device information to the following interface with the `app_access_token`.

```
POST /me/model/info?access_token=[APP_ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": [
    {"device_model": "DCS-1100L"},
    {"device_model": "DCS-930L"}
  ]
}
```

If the binding is completed successfully, mydlink platform will reply HTTP 200 OK with the associated mydlink id.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": [
    {
      "device_model": "DCS-1100L",
      "client_agent":"https://api.mydlink.com/resource/agent_dcs1100l.jar”,
      "dev_id_cgi":"/cgi/info”,
    },
    {
      "device_model": "DCS-930L",
      "client_agent":"https://api.mydlink.com/resource/agent_dcs930l.jar”,
      "dev_id_cgi":"/cgi/getinfo”
    }
  ]
}
```

If there is an issue looking up the model information, mydlink platform will reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Application",
    "code": 19,
    "message": "Unknown model."
}
```

## 4.b Device

### *4.b.1 Listing*

The application can retrieve the list of devices associated with the existing user account. It can GET the device information to the following interface with the `access_token`.

```
GET /me/device/list?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
```

If the `access_token` is valid, mydlink platform will reply HTTP 200 OK with the list of devices associated with the existing mydlink account. The ordering of devices in the list shall obey the result of **4.b.6**.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": [
    {
      "mac": "F07D68022D93",
      "mydlink_id": "30038291",
      "device_model": "DCS-930L",
      "device_name": "Living Room",
      "hw_ver": "A1",
      "online": true
    },
    {
      "mac": "F07D68024A81",
      "mydlink_id": "30039412",
      "device_model": "DCS-930L",
      "device_name": "Kitchen",
      "hw_ver": "B4",
      "online": true
    },
    {
      "mac": "F07D68012101",
      "mydlink_id": "30036291",
      "device_model": "DCS-1130L",
      "device_name": "Front Door",
      "hw_ver": "A1",
      "online": false
    }
  ]
}
```

If there is an issue looking up the devices associated with the user account, mydlink platform will reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
```

```
  "error": {
    "type": "User",
    "code": 13,
    "message": "Access token invalid."
  }
}
```

### *4.b.2 Binding*

The application can bind the device with the existing user account. It can POST the device information to the following interface with the `access_token`.

```
POST /me/device/add?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    "mac":"F07D68022D93",
    "mydlink_id":"30038291",
    "session_id":"39847184729" # for f/w with dcp1.0, it shall carry device
password.
  }
}
```

The above MAC address of the device, mydlink identifier and the session id (the DFP value) are retrieved from the device during the binding process. Be noted that the application should NOT cache the session id because it will be changed frequently. For legacy f/w (device agent version is less than 2.x), the device agent doesn't provide the DFP info. To bind the legacy device, the client shall carry device password in the '`session_id`' field, insteadly.

If the binding is completed successfully, mydlink platform will reply HTTP 200 OK with the associated mydlink id.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": {
    "mac": "F07D68022D93",
    "mydlink_id": "30038291",
    "device_model": "DCS-930L"
  }
}
```

If there is an issue binding the device to the user account, mydlink platform will reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store
```

```
{
  "error": {
    "type": "Device",
    "code": 16,
    "message": "Session id invalid."
  }
}
```

### *4.b.3 Unbinding*

The application can disassociate the device with the existing user account. It can POST the device information to the following interface with the `access_token`.

```
POST /me/device/del?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": [
    { "mydlink_id": "30038291" }
  ]
}
```

If the binding is completed successfully, mydlink platform will reply HTTP 200 OK with the associated mydlink id.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": [
    { "mydlink_id": "30038291" }
  ]
}
```

If there is an issue binding the device to the user account, mydlink platform will reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Device",
    "code": 17,
    "message": "Invalid ownership"
  }
}
```

### *4.b.4 Device Information*

The application can retrieve the access information of the devices with the existing user account.

It can POST the device information to the following interface with the `access_token`. Note that the number of queried devices shouldn't be over 3.

```
POST /me/device/info?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": [
    { "mydlink_id": "30038291" }
  ]
}
```

If the binding is completed successfully, mydlink platform will reply HTTP 200 OK with the associated mydlink id. Fields "`ctrl_stats`", and "`module_info`" are carried in the response. Content of fields "`ctrl_stats`" and "`module_info`" are device dependant.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": [
    {
      "mydlink_id": "30038291",
      "device_name": "Kitchen",
      "password": "123456",
      "verified": true,          # becomes false once device password changed
      "public_ip": "108.67.223.240",
      "public_port": "80",
      "private_ip": "192.168.0.132",
      "private_port": "8080",
      "activate_date": "2015-05-26 12:26:40", # when the device was bound
      "auth_key": "C4680EA26C7447FFA99C42F098240A19",
      "signal_addr": "signal2.us.mydlink.com",
      "online": true,
      "fw_uptodate": true,
      "fw_force": false          # default false. If fw_uptodate is false, client
                                 #  should enter f/w force upgrade flow.
      "fw_manual": false         # default false. If fw_uptodate is false, client
                                 #  should enter manual f/w upgrade flow.
      "fw_ver": "1.2",           # current f/w version of this device
      "agent_ver": "2.0.16",     # current agent version of this device
      "features": [ 1,2,3,6 ],   # capability of this device. check appendix II.
      "ctrl_stats": "<value>",   # device defined controllable stats
      "module_info": "<value>",  # device defined module related info
      "utc_offset": -6,          # time offset to UTC, in hour. it may be float
                                 #  point if the offset is half hour.

      "lvr_info": {              # info block of local recording feature
        "streamer": {
          "enabled": false       # true if it's enabled to stream video to target.
          "target": [
            {
              "mydlink_id": "20000001",
              "vol_id": "JetFlash_TS2GJF150_01"
```

```
            }
          ]

        },
        "storage_mgr": {         # binding info of storage manager
          "source": [
            {
              "mydlink_id": "44440000",         # receive stream from the mydlink_id

              "vol_id": "JetFlash_TS2GJF150_01" # and store to target volume id
            },
            {
              "mydlink_id": "44440001",         # receive stream from the mydlink_id
              "vol_id": "JetFlash_TS2GJF150_02" # and store to target volume id
            }
          ]
        }

      }
    }
  ]
}
```

If there is an issue binding the device to the user account, mydlink platform will reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Device",
    "17",
    "message": "Invalid ownership"
  }
}
```

### *4.b.5 Connect to the Device*

The application can ask for the access of the devices with the existing user account. It can POST the mac addr and the mydlink id of device to the following interface with the `access_token`.

```
POST /me/device/connect?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    { "mac": "F07D68022D93", "mydlink_id": "30038291" }
  }
}
```

If the device is online, mydlink platform will reply HTTP 200 OK with the associated access

information.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": {
    "mydlink_id":" 30038291",
    "relay_ip":"108.67.223.240",
    "session_number":"08583121-469e-11e0-8a6d-d49a20db4028",
    "private_ip":"192.168.0.132",
    "private_port":" 8080"
  }
}
```

If there is an issue retrieving the access information of the target device, mydlink platform will reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Device",
    "code": 18,
    "message": "Offline"
  }
}
```

### *4.b.6 Set device list orders*

Client app can set the device orders in the response list. It can POST the mydlink id of device to the following interface with the `access_token`.

```
POST /me/device/order_list?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    "orders": [ "30000005", "30000002", "30000001", "30000004", "30000003" ]
  }
}
```

If all passed device records are correct and belong to the operating account, mydlink platform will reply HTTP 200 OK with the given ordering.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
```

```
  "data": {
    "orders": [ "30000005", "30000002", "30000001", "30000004", "30000003" ]
  }
}
```

If there is an issue for the given device information, mydlink platform will reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Device",
    "code": 17,
    "message": "Invalid ownership"
  }
}
```

### *4.b.7 Upgrade device firmware*

Client app can trigger the firmware upgrade action on the bound devices. It can POST the MAC address and mydlink id of the device via the below request.

```
POST /me/device/fw_upgrade?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    "mac": "F07D68022D93",
    "mydlink_id": "30038291"
  }
}
```

If the passed device information is correct and belong to the operating account, mydlink service replies HTTP 200 OK with the status “progressing”. If the device already has the up-to-date f/w version, the status will be “done”. For fail of the f/w upgrade operation, the service responds 200 OK status code with value “fail” in the status field.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": {
    "mydlink_id": "30038291",
    "status": "progressing"    # value may be "done", "progressing", and "fail"
  }
}
```

If there is an issue for the given device information, mydlink service replies an HTTP 400 with the error in the response body, such as “Invalid ownership” or “Offline”.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Device",
    "code": 17,
    "message": "Invalid ownership"
  }
}
```

### *4.b.8 Device Statistics Query*

Client app can query the statistics info of the specified device if that device also supports the statistics reporting protocol of mydlink. To get this kind of info, client can invoke:

```
POST /me/device/statistics?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": [
    { "mydlink_id": "30038291" }
  ]
}
```

If the given device belongs to the user, mydlink service replies HTTP 200 OK with the information this device reported to mydlink service.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": [
    {
      "mydlink_id": "30038291",           # mydlink number of the device
      "timestamp": 1393495628,            # last update timestamp, since 1970/1/1
      "info": "DEVICE-DEFINED FIELDS"     # device info. client shall interpret
                                          # by the each device context.
    }
  ]
}
```

Note that device statistics will be kept in mydlink service for N seconds at most. The N is 300 sec for this version. Empty value will be returned if device doesn’t update the info and its expired.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
```

```
{
  "data": [
    {
      "mydlink_id": "30038291",
      "timestamp": 0,
      "info": ""
    }
  ]
}
```

If there is an issue for the given device information, mydlink service replies an HTTP 400 with the error in the response body, such as “Invalid ownership”.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Device",
    "code": 17,
    "message": "Invalid ownership"
  }
}
```

### *4.b.9 Set Device Preferences*

Client can store preferences of devices via this API. There are **N** attributes for each device, and max **M** bytes for each attribute content.
For this version, N and M are set as **20** and **256**.

```
POST /me/device/set_prefs?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    "mydlink_id": "30032891",               # mydlink number of the target device
    "info": [
      {
        "no": 1,                            # attribute no. (1-N)
        "tag": "<attribute name>",          # tag name the device defined
        "value": "<attribute value>"        # tag value the device defined
      },
      {
        "no": 3,
        "tag": "<attribute name>",
        "value": "<attribute value>"
      }
    ]
  }
}
```

If the given device belongs to the user and the request is validaed, mydlink service replies HTTP 200 OK with success result.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": {
    "result": "success"
  }
}
```

If there is an issue for the given device attributes, mydlink service replies an HTTP 400 with the error in the response body, such as “Invalid ownership”.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Device",
    "code": 17,
    "message": "Invalid ownership"
  }
}
```

### *4.b.10 Get Device Preferences*

Client can query the stored device preferences back from below API:

```
POST /me/device/get_prefs?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": [
    {
      "mydlink_id": "30032891",               # mydlink number of the target device
      "prefs": [
        "attr A"                              # attribute name to be queried.
      ]
    },
    {
      "mydlink_id": "30039412",
      "prefs": [
        "attr A",
        "attr B"
      ]
    }
  ]
}
```

If the request is correct, below message will be responded:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": [
    {
       "mydlink_id": "30032891",
       "info": [
         { "no": 1, "tag": "<attr A>", "value": "<val>" }
       ]
    },
    {
       "mydlink_id": "30039412",
       "info": [
         { "no": 1, "tag": "<attr A>", "value": "<val>" },
         { "no": 5, "tag": "<attr B>", "value": "<val>" }
       ]
    }
  ]
}
```

If the given request is invalid, the message responded as:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Application",
    "code": 10,
    "message": "Error validating this request."
  }
}
```

### *4.b.11 Verify Device Password*

If user changed device password from device console/webpage directly, the device will enter 'unverified' state ('verified' field is false in 4.b.4 response). Once device entered this state, client shall prompt user to enter the new device password before access to this device.

The application has to get the `access_token` before it can verify the device password. It can POST the device information to the following interface with the `access_token`.

```
POST /me/device/verify_password?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    "mydlink_id":"30038291", # mydlink number of the target device
    "password":"123456"      # device password
  }
```

```
}
```

If the request is correct, below message will be responded:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data":
    {
      "mydlink_id": "30038291",
      "device_name": "Kitchen",
      "password": "123456",
      "verified": true,
      "public_ip": "108.67.223.240",
      "public_port": "80",
      "private_ip": "192.168.0.132",
      "private_port": "8080",
      "activate_date": "2015-05-26 12:26:40",
      "auth_key": "C4680EA26C7447FFA99C42F098240A19",
      "signal_addr": "signal2.us.mydlink.com",
      "online": true,
      "fw_uptodate": true,
      "fw_force": false          # default is false. If fw_uptodate is false, read
the "force" value of the target version in fw config file.
      "fw_manual": false          # default is false. If fw_uptodate is false, read
the "manual" value of the target version in fw config file
      "fw_ver": "1.2",           # current f/w version of this device
      "agent_ver": "2.0.16",     # current agent version of this device
      "features": [ 1,2,3,6 ],   # capability of this device. check appendix II.
      "ctrl_stats": "<value>",   # device defined controllable stats
      "module_info": "<value>",  # device defined module related info
      "utc_offset": -6,          # time offset to UTC, in hour. it may be float
                                 # point if the offset is half hour.
      "lvr_info": {              # info block of nvr (local) feature
        "streamer": {
          "enabled": false       # true if it's enabled to stream video to target.
          "target": [
            {
              "mydlink_id": "20000001",
              "vol_id": "JetFlash_TS2GJF150_01"
            }
          ]

        },
        "storage_mgr": {         # binding info of storage manager
          "source": [
            {
              "mydlink_id": "44440000",         # receive stream from the mydlink_id

              "vol_id": "JetFlash_TS2GJF150_01" # and store to target volume id
            },
            {
              "mydlink_id": "44440001",         # receive stream from the mydlink_id
              "vol_id": "JetFlash_TS2GJF150_02" # and store to target volume id
```

```
            }
          ]
        }

      }
    }
}
```

If the given request is invalid, the message responded as:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Application",
    "code": 10,
    "message": "Error validating this request."
  }
}
```

### *4.b.12 Update Device Info*

The application has to get the `access_token` before it can update device information. It can POST the device information to the following interface with the `access_token`.

**Note :** Attributes provided by client will be updated (currently password and device_name). Only carry the attributes need to update.

```
POST /me/device/update?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    "mydlink_id": "30038291",     # mydlink number of the target device
    "force_unverify": true,       # force device to be unverified state
    "device_name": "中文Türkçe"    # UTF8 supported device name
  }
}
```

If the request is correct, below message will be responded:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  {
  "data": {
```

```
    "result": "success"
  }
}
```

If the given request is invalid, the message responded as:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Application",
    "code": 10,
    "message": "Error validating this request."
  }
}
```

## 4.c Services Configuration

### *4.c.1 Manage app token for push notification*

Not supported.

### *4.c.2 User services setting*

The API provides ability for client to setup the services state belong to the given account, such as push notification.

Client can use GET method to query current services settings of the given account, and using POST method to update the service settings.

```
GET /me/user/services?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json
```

```
POST /me/user/services?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    "services": [
      {
        "name": "mpn",      # mpn for 'mydlink push notification' service
        "devices": [
          { "mydlink_id":"30038290",
                "type": "email", "enabled": 0, "token": "EMAIL" },
          { "mydlink_id":"30038290",
                "type": "ios",
                "enabled": 1,
                "token": "TOKEN",
                "filter": {
                  "enable": true,
                  "value": {
                    "sys": true,         # receive all system related events.
                    "srv": true,         # receive all service related events.
                    "ids": [ 256, 512 ]  # receive specified device events.
                  }
                }
          }
        ]
      }
    ]
  }
}
```

For service "mpn", the possible value of "**`type`**" field may be: "`email`", "`ios[;ARGS]`", "`android`", and "`win`". The `filter` field carries the interested event id(s). If it's specified, only events in the list will be emitted to the client. If the field is missing, no events will be sent to the client. If `filter` field is null or doesn't present, the filter function is disabled.

If the information is validated, mydlink platform shall respond the current service settings of the

associated user.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "data": {
    "services": [
      {
        "name": "mpn",      # mpn for 'mydlink push notification' service
        "devices": [
         { "mydlink_id": "30038290", "type": "email", "enabled": 0,
             "token": "EMAIL", "filter": { "enable": false } },
         { "mydlink_id": "30038290", "type": "ios", "enabled": 1,
             "token": "TOKEN",  "filter": { "enable": false } }
        ]
      }
    ]
  }
}
```

If there is an issue access the service settings, mydlink platform shall reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Application",
    "code": 13,
    "message": "Access token invalid."
  }
}
```

## 5. Logging

Clients are able to store client logs to mydlink platform for error reporting, issue tracking, and client behaviors mining.

### 5.a.1 Session logging

Client can post logs to mydlink service via this API. The `access_token` can be app-level token or user-level token. Extra info (ip, username) is stored automatically if user-level token is carried. Client can log single message or couples massages for same session id. To be consist, client shall carry info in “basic, info, and log sections”. It’s open for clients to carry “network” section or not. The given payload must be valid JSON format.

```
POST /me/log/session?access_token=[ACCESS_TOKEN] HTTP/1.1
Host: api.mydlink.com
Content-Type: application/json

{
  "data": {
    # [Basic] session information
    "session":  "xx-xx-xx-xx-xx",       # session id, should be in GUID form to
avoid
                                        # id confliction.
    "host":     "ios",                  # host type: web, ios, android, or win
    "version":  "1",                    # log format version , current is 1

    # [Info] client and host information
    "info": {
      "app_version":  "1.2",            # version of the app
      "build_number": "2.5",            # app build number
      "os_version":   "7.0",            # OS version of the host device
      "udid":         "xxxxxxxxx”,      # UDID of the host device
      "hardware_id”:  "iPad3.1"         # h/w version of the host device
    },

    # [Network] client network information (update on change)
    "network": {
      "ip":           "10.32.100.1",    # current IP of the device
      "country":      "united kingdom", # current country setting of the device
      "locale":       "english",        # current language setting of the device
      "mobile_net":   true,             # mobile net enabled or not: true or false
      "wifi":         false,            # device wifi is on or off: true or false
      "carrier":      "O2"              # name of mobile network operator
    },

    # [Log] it’s not limited to carry "log_data" field, depending on app context
    "log": {
      "log_data":     "app upgraded"    # log message
    }
  }
}
```

On success, the message below will be returned.

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json
Cache-Control: no-store

{
  "data": {
    "result": "success"
  }
}
```

If the given `access_token` is invalid, mydlink platform will reply an HTTP 400 with the error in the body of the response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": {
    "type": "Application",
    "code": 13,
    "message": "Access token invalid."
  }
}
```

## Appendix I - Handling of HTTP Error Codes

The client of Open API shall be able to handle the HTTP error code well. The listing error codes must be handled:

| Status Code | Client Actions |
|---|---|
| 301<br>Move Permanently | Client shall save the target location in local cache and always send request to the cached one. |
| 302<br>Found | Client shall save the target location in local cache till next login action. |
| 400<br>Bad Request | The request is malformed. May be caused by wrong AppID, signature, username/password combination, or missing/invalid access token. |
| 404<br>Not Found | The requested resource doesn't exist in OpenAPI. The client SHALL stop to request this resource. |
| 500<br>Internal Server Error | Client shall retry the request for 3 times with certain frequency in each specified scope. If fail, the client shall stop the auto retry, unless triggered by user. |
| 503<br>Service Unavailable | Client shall display the message carried in the response and stop to send any request to the service. The server may return the error code when it's overloading or under maintenance period. Client should send any request to the service unless user trigger it manually.<br><br>The response message of 503 status code shall be:<br>`HTTP/1.1 503 Service Temporarily Unavailable`<br>`Content-Type: application/json`<br>`Retry-After: Fri, 1 May 2013 23:59:59 GMT`<br><br>`{`<br>`  "data": {`<br>`    "type": "maintenance",`<br>`    "info": [`<br>`      { "lang": "en",`<br>`        "msg": "mydlink website will undergo scheduled maintenance on May 14 from 12:00AM to 4:00AM." },`<br>`      { "lang": "zh_TW",`<br>`        "msg": "mydlink網站將會於5/14 12:00AM至4:00AM進行維護" }`<br>`    ]`<br>`  }`<br>`}`<br><br>The client shall handle the '`Retry-After`' tag and be able to display the message carried in '`msg`' tag by user's language. If client receives 503 response, the client should display the carried message and stop to provide any function which related to the service to user. |

The table below lists the code id definitions for 400 Errors

| Code ID | Message | Description |
|---|---|---|
| 10 | Error validating this request. | Parameter(s) carried with this request is missing or wrong. |
| 11 | Request expired. | The timestamp carried with the request exceeded the acceptable intervals (300 sec) of server's time. The server shall carry the 'timestamp' field in the error response to indicate current timestamp of server's. |
| 12 | User denied. | User doesn't authorize the access from this client. |
| 13 | Wrong username/password combination. | The carried username/password combination in the request is incorrect. |
| 14 | Access token invalid. | The carried access token is invalid. |
| 15 | Account duplicated. | The account name to be created already exists. |
| 16 | Field format invalid. | The field value to be updated is invalid |
| 17 | Session id invalid. | The session id for binding the given device is invalid. |
| 18 | Invalid ownership. | The user doesn't have the permission to access the specified device. |
| 19 | Offline. | The device client going to manage is offline at this moment. |
| 20 | Unknown model. | The target model is unknown for the service. |
| 21 | Invalid Client ID | The carried client_id cannot be recognized. |
| 22 | Not supported 3rd party service. | The 3rd party service cannot be recognized or not supported yet. |
| 23 | Already linked. | The mydlink account to be linking is already linked with different user id of the 3rd party service. |
| 24 | User cancels. | User cancels the process. |
| 25 | Different account/device sites. | The account and the device to be bound are in different sites. |
| 26 | Device list is not empty | The error may occur in account deletion request. |
| 27 | Not activated | The request API isn't activated yet for the client. |
| 28 | Too many login fails | Too many login fails in the period. |

| | | |
|---|---|---|
| 29 | Account suspended | Account suspended. |
| 30 | No such record | No record found for the given information. |
| 31 | No privilege | The carried client_id has no privilege for this call |
| 32 | Value out of range | The specified value is out of defined range. |
| 51 | Fail to create account | Errors in account creation. |
| 52 | Internal error | DB access error. |
| 53 | Internal error | Failed to access external service (e.g., sqs, s3) |
| 60 | Subscription requires | Need to subscribe service on the selected target for the requesting services. |
| 61 | Subscription suspended currently | Cannot access the specified service currently for some reasons. properly causes by the subscription issue. |
| 62 | Invalid service session | The carried session is invalid to access the service. |
| 63 | Quota exceeded | Quota on access external service exceeded |
| 71 | canceled by forgot-password | current action is canceled by forgot-password action |
| 72 | canceled by signup | current action is canceled by signup action |
| 73 | canceled by signin | current action is canceled by signin action |
| 74 | Password mismatch | Device password mismatch |
| 75 | Bind limit (99) reach | Device bind limit per account (99) reached |

## Appendix II - Capability Definition of mydlink Enabled Devices

| Name | Value | Remarks |
|---|---|---|
| LAN management | 1 | Basic router management functions. |
| WiFi 2.4G | 2 | Support wifi 2.4G configuration. |
| WiFi 5G | 3 | Support wifi 5G configuration. |
| WAN management | 4 | Support WAN management functions. |
| Event message | 5 | Support mydlink event spec. |
| Live view | 6 | Support live view streaming functions. |
| Motion detection | 7 | Support motion detection configurations. |
| Night vision | 8 | Support night vision configurations. |
| SD card access | 9 | Support SD card access functions. |
| 2way audio | 10 | Support 2way audio functions. |
| Sound detection | 11 | Support sound detection functions. |
| PTZ | 12 | Support PTZ functions. |
| Extender mode | 13 | Support WiFi extender functions |
| Fisheye | 14 | Suuport Fisheye functions. |
| PIR | 15 | Support PIR detection. |
| ePTZ | 16 | Support ePTZ functions. |
| Time lapse | 17 | Support Time-lapse function. |
| Thermal detection | 18 | Support Thermal detection function for baby cam |
| Lullaby | 19 | Support Lullaby function for baby cam |
| Full duplex two-way audio | 20 | Support full duplex two-way audio (dgtalkie) function |
| White light LED | 21 | Support white light LED function |
| URLEncode | 22 | Support URL encode/decode function for device password, SSID and keyphrase of routers, DAPs and DSL modem |

| | | |
|---|---|---|
| Shareport | 40 | Support mydlink Shareport |
| View-NVR | 41 | Support mydlink View-NVR |
| View-Router | 42 | Support mydlink View-Router |
| Access-NAS | 43 | Support mydlink Access-NAS |
| Access-AP | 44 | Support mydlink Access-AP |
| App reserved 1 | 45 | Reserved for app supporting 1 |
| App reserved 2 | 46 | Reserved for app supporting 2 |
| App reserved 3 | 47 | Reserved for app supporting 3 |
| App reserved 4 | 48 | Reserved for app supporting 4 |
| App reserved 5 | 49 | Reserved for app supporting 5 |

## Appendix III - Generation of Request Signature

In OpenAPI, request without `access_token` should carry '`signature`' for the validation process of OpenAPI service. The signature is calculated from request path, timestamp, and the `client_secret` of the app.

Here is an example:

```
GET
/oauth/access_token?client_id=[CLIENT_ID]&grant_type=app_credential&timestamp=[curr
ent-time]&sig=[signature] HTTP/1.1
Host: api.mydlink.com
Accept: */*
```

- **`client_id`:** the app identifier of the client app issued by mydlink.
- **`timestamp`:** the UNIX time in seconds (UTC+0)
- **`sig`:** the request signature, and derived from:
  *`sig = HexString-of-MD5(request path + client_secret)`*

In this example:

1. Request path: (before urlencode)

   `/oauth/access_token?client_id=FakeAppID&grant_type=app_credenti al&timestamp=1369307910`

2. Assumed the privilege code of app 'FakeApp' is:

   `75a8ab07844640e99ea92d3330b625f2`

3. Then the sig value goes to:

   *`HexString-of-MD5`*`(/oauth/access_token?client_id=FakeAppID&grant_t ype=app_credential&timestamp=13693079107`**`5a8ab07844640e99ea92d333 0b625f2`**`)`
   `= b578153b792c2ca024fbc53188aa8dee`

4. So the request from app shall be:

```
GET
             /oauth/access_token?client_id=FakeAppID&grant_type=app_credential&times
             tamp=1369307
910&sig=b578153b792c2ca024fbc53188aa8dee HTTP/1.1
Host: api.mydlink.com
Accept: */*
```

Note that the OpenAPI service shall check both the validity of `sig` and the `timestamp` fields. If the timestamp of the request is exceed **`300`** seconds (before and after), compared with the current timestamp of OpenAPI service, the request will be rejected by the OpenAPI service with 400 error. For this case, the error code '10' and `'timestamp'` field will be responded. The

client shall use the returned `timestamp` value as timestamp, and send the request again to get the `access_token`.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error":{
    "type": "OAuthException",
    "code": 10,
    "message": "Error validating privilege code.",
    "timestamp": 1379054513
  }
}
```

## Appendix IV - Privileges of Access Token

In OpenAPI, there are different privileges of granted access token:

1. **Application level access token:**
   - 2.a.1

   has privileges to access APIs:
   - Create account (3.a.1)
   - Create account dialog (3.a.9)
   - Access token revocation (2.c.1)
   - Log message (5.a.1)

2. **User level access token:**
   - 2.b.1.2
   - 2.b.1.3
   - 2.b.2.1
   - 2.b.2.3
   - 2.d.1
   - 3.a.1

   (this section isn't up-to-date yet)

   has privileges to access APIs:
   - Access token revocation (2.c.1)
   - Authorization revocation (2.c.2)
   - Delete account (3.a.2)
   - Update account (3.a.3)
   - Send activation email to user (3.a.4)
   - Link user account with external service (3.a.5)
   - Model listing (4.a.1)
   - Device listing (4.b.1)
   - Bind device (4.b.2)
   - Unbind device (4.b.3)
   - Device info (4.b.4)
   - Device list ordering (4.b.6)
   - Firmware upgrade (4.b.7)
   - Manage app token (4.c.1)
   - Update service settings (4.c.2)
   - Log message (5.a.1)

## Appendix V - Language Code Definitions

| Code | Language |
|---|---|
| en | English |
| fr | French |
| ru | Russian |
| es | Spanish |
| pt | Portuguese(Portugal) |
| pt_BR | Portuguese(Brazil) |
| ja | Japanese |
| zh_TW | Traditional Chinese |
| zh_CN | Simplified Chinese |
| ko | Korean |
| cs | Czech |
| da | Danish |
| de | German |
| el | Greek |
| hr | Croatian |
| hu | Hungarian |
| it | Italian |
| nl | Dutch |
| no | Norwegian |
| pl | Polish |
| ro | Romanian |
| sl | Slovenian |
| sv | Swedish |
| fi | Finnish |

## Appendix VI - 3rd Party Services Information

| Service | Name | Supported | User Id Field | Reference |
|---|---|---|---|---|
| Facebook | facebook | N | **id** field of the response of: https://graph.facebook.com/me | https://developers.facebook.com/docs/reference/api/user/ |
| Google | google | N | **sub** field of the ID Token | https://developers.google.com/accounts/docs/OAuth2Login |
| Yahoo | yahoo | N | **guid** field of the response of social profile api. | http://developer.yahoo.com/social/rest_api_guide/extended-profile-resource.html |
| Baidu | baidu | N | **userid** field of the response of: https://openapi.baidu.com/rest/2.0/passport/users/getInfo | http://developer.baidu.com/wiki/index.php?title=docs/oauth/rest/file_data_apis_list#.E8.BF.94.E5.9B.9E.E7.94.A8.E6.88.B7.E8.B5.84.E6.96.99 |

## Appendix VII - Application Registration Table

| Field | Name | Description |
|---|---|---|
| **Input** | | |
| Name | name | Name of the application. It's the name show up in 'Authorize App' dialog. |
| Icon | icon | Icon of the application. It's the icon show up in 'Authorize App' dialog. |
| Description | description | Short description of the application. It's the description show up in 'Authorize App' dialog. |
| Client Type | client_type | Type of the application. The possible value are '`server`', '`web`', and '`app`'. |
| URI List | uri_list | The uri list that can be carried in '`redirect_uri`' file for this application. If the carried '`redirect_uri`' doesn't exist in the list, the oAuth process will fail. Only **hostname** of the `redirect_uri` will be checked. The protocols portion, such as 'http(s)://', and resource path will be skipped. |
| Type | type | To identify the app is 3rd party application or not. (0: false, 1: true). |
| **Generate** | | |
| Client ID | client_id | The application ID. |
| Client Secret | client_secret | The secret key of the application. |

## Appendix VIII - OAuth Procedure for Standalone Applications

For clients sign-in mydlink platform via (2.b.2.3), the token management procedure described below is recommended:

1. Listen on `localhost:PORT`.
   - All success/fail results will be forward to `localhost:PORT`.
   - The response are all in HTTP format.
   - Client shall have ability to parse HTTP messages.

2. Embed the WebView widget, and opening the page by request (2.b.1.1) to get the authorize code.
   - The `redirect_uri` must be : `http(s)://localhost:PORT/<client-defined-fields>`
   - mydlink Sign-in page prompts for user account and password.
   - Authorize code returns via HTTP '`302 Found`' if the given credential is correct.
   - App should switch to its handling UI and hide the WebView widget once the response received.

3. Retrieve the code from '`Location`' field and prepare for getting `access_token` & `refresh_token` via (2.b.1.2) request.
   - `access_token` is used for general API access. The token expires in day(s).
   - `refresh_token` is for renew `access_token` only. This kind of token has longer expiration time.

4. The client shall store the retrieved `access_token` and `refresh_token` in local storage.
   - On app starts, the `access_token` shall be used to access mydlink resources of the associated user.
   - If mydlink service responds 'Invalid Access Token (code: 14)', the app shall try the `refresh_token` to get a fresh `access_token`.
   - If mydlink service responds 'Invalid Access Token (code: 14)' in `access_token` refreshing request, the app shall go to step 1., to resend the request (2.b.1.1) to obtain the new token pairs.
   - Application shall well manage the retrieved tokens from mydlink service. Number of `refresh_token` one user can request is limited. Keep calling (2.b.1.1) will make `refresh_token(s)` with least expiration time be recycled by mydlink platform, and cause applications with recycled tokens fails to access mydlink service.

5. On user sign-out, app must issue (2.c.1) to revoke both stored `access_token` and `refresh_token`, to ensure these two tokens become invalid.
   - mydlink platform will invalid user tokens automatically on user password changed or account deleted.