Logged-in Users (client-side)

> ⚠ This scenario requires a unique client ID and redirect URI to be set up with movingimage in advance. Contact movingimage Professional Services for further assistance.

Use this scenario if you are making a client-side application that requires the user to login to their VideoManager Pro account. This scenario will allow you to redirect the user to the authorization server's login page. The workflow is shown in the diagram to the right and corresponds to the OAuth 2.0 "authorization code" grant type.

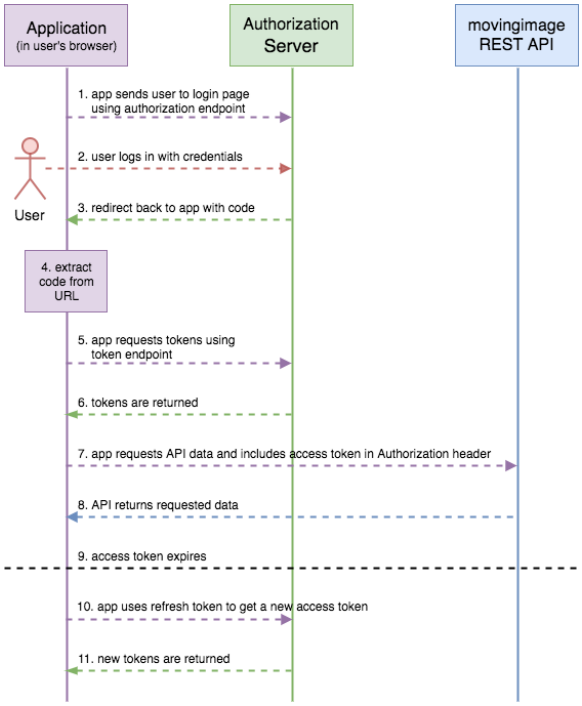
There are several authentication APIs available that can be used instead of writing the code for the requests below from scratch. The framework used to develop the integration software may even already provide this functionality. A list of authentication APIs can also be found on the OAuth website. Make sure the one you choose supports the "authorization code" grant type. Continue reading to learn how to build these requests from scratch.

In order to acquire the access and refresh tokens, the application must perform two requests to the authorization server:

1. Get an authorization code
    - (this covers steps 1 - 3 in the diagram)
2. Exchange the authorization code for access and refresh tokens
    - (this covers steps 4 - 6 in the diagram)

Once the tokens are acquired, they are used in the following ways:

1. Use the access token - include it in the authorization header for every request to the movingimage REST API
    - (this covers steps 7 & 8 in the diagram)
2. Use the refresh token - after the access token expires, use the refresh token to get a new one
    - (this covers steps 9 - 11 in the diagram)



(this diagram assumes all requests are successful)

Get an Authorization Code

1. The application must send the user to a login page using the authorization endpoint (see the Endpoint Discovery chapter).
    - Use the following data for your request:
        - client_id - you must get this from movingimage in advance (see yellow note at the top of the page)
        - redirect_uri - this is the location the login page will redirect the browser to after the user logs in - this must also be set up with movingimage in advance
        - response_mode - this should be set to "fragment", though it is not necessary to include this parameter because "fragment" is already the default value (it is included here for the sake of transparency)
        - response_type - set this to "code"
        - scope - set this to "openid"
        - state - (optional) - generate a random URL encoded string for this value

        > ⓘ Though the state value is optional, it is highly recommended to use it, as it protects against malicious software attacks. Click here for more information.

    - Example:

        ```
        https://login.movingimage.com/auth/realms/platform/protocol/openid-connect/auth?
                client_id=<---CLIENT_ID--->&
                redirect_uri=<---REDIRECT_URI--->&
                response_mode=fragment&
                response_type=code&
                scope=openid&
                state=<---RANDOM_STRING--->
        ```

        > ✓ Remember to URL encode your redirect URI (so "https://" should be "https%3A%2F%2F")

2. This link will send the user to a login page where they can enter their username and password to log in.

3. The user is then redirected to the location in the redirect_uri parameter. This location will contain an authorization code similar to the example below:

```
https://redirect.example.com/#state=7f4jKO98p0&code=beEGJ7l7OmpTI6DuCmOsAWmcPMz4EwajvLE-0BU5NdA.
3a21d1a0-23ad-40c0-8d79-332ac758dee2
```

> The state will also be given back to you if you used this parameter in the login URL. If the state value matches the string from the original login URL, it is safe to use the code.

Exchange the Authorization Code for Access and Refresh Tokens

4. Extract the authorization code from the URL.
5. Use the token endpoint to build a POST request that will return your access and refresh tokens (if the code is valid).
    - Use the following data for your request:
        - grant_type - this must be "authorization_code"
        - client_id - same client id used in the last step
        - code - the authorization code retrieved in the last step
        - redirect_uri - this must match the redirect uri used in the previous step
    - Example Request:
    - ```
      curl -X POST -H 'Content-Type: application/x-www-form-urlencoded' -d '
              grant_type=authorization_code&
              client_id=<---CLIENT_ID--->&
              code=<---AUTHORIZATION_CODE--->&
              redirect_uri=<---REDIRECT_URI--->'
      https://login.movingimage.com/auth/realms/platform/protocol/openid-connect/token
      ```
6. Example Response:

- ```
  {
          "access_token": "<----ACCESS_TOKEN---->",
          "expires_in":300,
          "refresh_expires_in":3600,
          "refresh_token":"<----REFRESH_TOKEN---->",
          "token_type":"bearer",
          "id_token":"<----ID_TOKEN---->",
          "not-before-policy":0,
          "session_state":"b2e84e67-f61e-4193-88d8-5846a0b76178"
      "scope": "openid profile email"
  }
  ```

> Note the "expires_in" and "refresh_expires_in" values. These give the timeframe (in seconds) during which the access and refresh tokens are valid, respectively. The access token's timeframe will be short, making the refresh token necessary (as demonstrated below under the "Use the Refresh Token" heading).

With the access and refresh tokens, it is now possible to use them to authorize the user to access resources from the API.

Use the Access Token

Once you have a valid access token, you must include it in the header of each request to the movingimage REST API. The following uses the "Get VideoManagers" method to demonstrate this:

```
curl -X GET -H "Authorization: Bearer <ACCESS_TOKEN>" https://api.video-cdn.net/v1/vms
```

Use the Refresh Token

The access token will expire after a short time, but it is possible to maintain uninterrupted access. After the access token expires, you can use the refresh token to get a new one. You will also get a new refresh token and the expiration time for the tokens will restart. Refreshing the tokens is possible until the latest refresh token expires. If both tokens expire without refreshing, the session will end and you will need to authenticate again.

Perform the following POST request (again using the discovered token endpoint) to refresh the tokens:

- Use the following data for your request:
  - grant_type - this must be "refresh_token"
  - client_id - same client id used in the previous steps
  - refresh_token - the refresh token
- Example Request:

```
curl -X POST -H 'Content-Type: application/x-www-form-urlencoded' -d '
        grant_type=refresh_token&
        client_id=<---CLIENT_ID--->&
        refresh_token=<---REFRESH_TOKEN--->'
https://login.movingimage.com/auth/realms/platform/protocol/openid-connect/token
```

The response data will be formatted just as it was the first time the token endpoint was used. It will contain new access and refresh tokens to use until the next time the access token expires.