FACEIT Connect (OAuth2)

Provides secure access to FACEIT online services

Last review: August 4, 2020

Version 3.0

SDK verified version `faceit-oauth-sdk-1.3.0`

FACEIT Single Sign-On (SSO) supports the OAuth 2.0 protocol for authorizing access to private user data and all FACEIT online services.

The list below explains a basic OAuth 2.0 scenario:

1. Your application must initiate the OAuth2 authorization process (using the javascript SDK or navigating to the **FACEIT Connect**) when a user attempts to use functionality that requires them to be logged in as a FACEIT User.
2. Your application will direct the user to the FACEIT authorization server. The link you provide will specify the scope of access that your application is requesting for the user's account. The scope of access specifies the resources that your application can access when acting as the authenticated user.
3. If the user consents to authorize your application to access the requested resources, then FACEIT will return a token to your application. Depending on the type of your application, it will either validate the token or exchange it for a different type of token.

The actual implementation flows may vary according to the architecture of your application and whether you are accessing FACEIT resources on behalf of a specific logged-in user, or programmatically from another application.

OAuth 2.0 Flows

There are three ways to get a valid Access Token to access FACEIT online resources:

1. **Authorization Code Flow** is used for server side applications that access FACEIT on behalf of a user
2. **Implicit Grant Flow** is used for client side applications (such as a client-side JavaScript app running in a web browser) that access FACEIT on behalf of a user
3. **Client Credential Authentication** is used for server side applications that programmatically access FACEIT without acting on behalf of a specific logged-in user

Developer setup

Before being able to implement FACEIT SSO in your application, you must be registered to the *FACEIT Developer Portal* (https://developers.faceit.com).

The following information is associated with a developer application that is authorised on FACEIT:

1. **Logo** - an image representing your application’s logo.

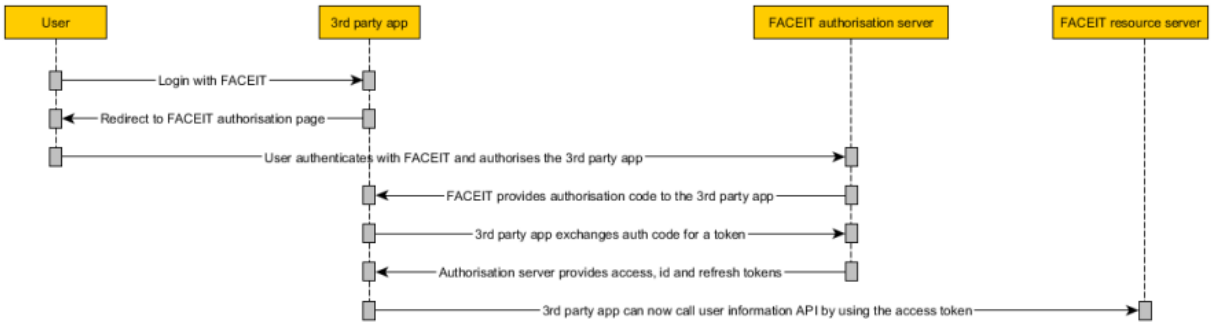
2. **Client ID** - a public string generated by FACEIT that represents the name of your app. You can give your app a label on FACEIT to make it easier to identify, but the Client ID provided by FACEIT will be always in a GUID format.

3. **Client Secret** – a secret token randomly generated by FACEIT for a specific Client ID. Your Client Secret is a password, so make sure to keep it somewhere safe. Additionally, generating a new client secret on FACEIT will immediately invalidate the current one, which will make your API requests fail until your app is updated.

4. **Redirect URL** - a URL pointing to a web page on your application or website to which FACEIT will redirect the user after successful authentication. FACEIT will append the Authorization code or OAuth token as a query parameter on this URL.

Authorization Code Flow

The following sequence diagram represents the FACEIT implementation of the OAuth2 Authorization Code Grant Type.



Remember, you must keep your Client Secret *confidential*. Make sure to never expose it to users, even in an obscured form.

All the authentication related endpoints can be found here:
https://api.faceit.com/auth/v1/openid_configuration

This grant type involves *server to server* communication to retrieve access and user tokens.

The high level flow is as follows:

1. The application renders a "Connect with FACEIT" link or button.
2. The user clicks the "Connect with FACEIT" link or button.
3. A pop-up is opened and directs the user to FACEIT Connect, asking them to authorize the application to access their details.
4. The pop-up closes and FACEIT redirects the user back to a URL under the application's control, passing a one-time authorization code for the specific user and application.
5. The application uses the authorization code in a server-side call to retrieve
    a. Access Token
    b. Refresh Token
    c. ID Token (containing all the info agreed by the user).

FACEIT keeps track of the authorizations granted, so users that are already signed in to faceit.com and who have authorized the application will simply be automatically redirected to your application.

The FACEIT Connect url is the following:

```
https://accounts.faceit.com/

    ?response_type=[can be either code or token]
```

```
&client_id=[your client ID]
&redirect_popup=[boolean indicating if a pop-up is needed]
```

You can directly open the above url in a popup window (or on top of your application) with the appropriate query string parameters or you can include the FACEIT JS SDK and invoke the `FACEIT.loginWithFaceit()` method (see examples below).

Please note that for security reasons, specifying multiple redirect URIs per OAuth2 client is not currently supported.

For more details about each grant type, please check the Examples section.

Implicit Grant Flow

This grant type can be used in client-side applications and is less secure than the authorization code grant type. The simpler flow is as follows:

1. The application renders a "Connect with FACEIT" link or button.
2. The user clicks the "Connect with FACEIT" link or button.
3. A pop-up is opened and directs the user to FACEIT, asking them to authorize the application to access their details.
4. FACEIT redirects the user back to a URL under the application's control, passing the Access Token and the ID Token directly

Please note that when using the implicit grant type, a refresh token is not issued, so access tokens cannot be refreshed.

Please refer to the Examples section for more details.

Client Credentials Flow

When authenticating through client credentials, you will obtain the Access Token by providing your credentials by making a POST request, the value of `<credentials>` is a Base64 encoded string of `<CLIENT_ID:CLIENT_SECRET>`.

```
Content-Type: application/x-www-form-urlencoded
Authorization: Basic <credentials>
POST https://api.faceit.com/auth/v1/oauth/token
payload:    “grant_type=client_credentials”
```

You will get an Access Token response in the same format as the other grant types.

Examples

Authorization Code Flow with the FACEIT JS SDK

1. Initializing the FACEIT JS SDK

The very first step is to insert and activate the "FACEIT Login for Apps" Javascript SDK inside your website.

You can insert the following snippet just before the closure of the </body> tag of your web pages. Please note the second parameter passed to the init function is "**code**", which means we are using the authorization code grant type.

```
<script src="https://cdn.faceit.com/oauth/faceit-oauth-sdk-1.3.0.min.js"
type="text/javascript"></script>
<script type="text/javascript">
   var initParams = {
       client_id: 'client1',
       response_type: 'code',
       state: 'informationYouWantPassedToTheRedirectUri',
       redirect_popup: true,
       debug: true
   };
   FACEIT.init(initParams);
</script>
```

The optional "**state**" parameter will be returned to your callback URI, so it is useful for storing any information that you want to maintain throughout the request.

The "**redirect_popup**" parameter allows you to decide whether you would like the FACEIT login pop-up window or the parent page to be redirected back to your Redirect URI once FACEIT has completed the user authentication process. By default, this parameter is false, which means the parent page will be redirected.

2. First click

The user clicks the "Connect with FACEIT" link or button inside your app.

The button or link can be either:

- automatically injected into the page by the FACEIT SDK. To have the SDK inject a button, just add a div with id “**faceitLogin**” where you want the button to be, like below:

  ```
  <div id="faceitLogin"></div>
  ```

- programmatically executed by the calling the following Javascript function

```
<a href="#" class="button Faceit" onclick="FACEIT.loginWithFaceit()">Connect with
FACEIT</a>
```

The user is redirected to FACEIT (a new pop-up is opened and directed to FACEIT) and asked to authorize the 3rd party application to use their details.

The loginWithFACEIT function returns the reference to the opened pop-up or null if something went wrong.

3. Obtaining the Authorization Code

Once the user is authenticated by FACEIT, a one-time **Authorization Code** is generated by FACEIT and sent to the **Redirect URI** of the 3rd party app as a query parameter, similar to this:

http://example.com/faceit-callback?code=jYWioI

4. Exchanging the Authorization Code for an Access Token

The 3rd party app makes a server to server call, posting the **Authorization Code**, **Client ID** and **Client Secret**, the response will contain:

- an **Access Token** (used for calling FACEIT API on behalf of the User)
- an **ID Token** (a JWT that contains user information digitally signed by FACEIT)
- a **Refresh Token** (used for obtaining a new Access Token when the existing one expires).

Your server makes the exchange by sending an HTTPS POST request. The POST request is sent to the token endpoint, which you should retrieve from the OpenID configuration endpoint using the key “**token_endpoint**”.

The request must use HTTP Basic Authentication "**Authorization**" header (using your **Client ID** and **Client Secret**) and the `application/x-www-form-urlencoded` “**Content-type**” header, including the following parameters in the POST body:

Field	Description
code	The one-time authorization code that is received on the provided callback page
grant_type	This field must contain the value "authorization_code", as defined in the OAuth 2.0 specification

A successful response to such a request will contain the following fields in a JSON object:

Field	Description
access_token	The access token you can use to call the FACEIT APIs

refresh_token	The refresh token you need to use to get a new access token when your current one expires
expires_in	The remaining lifetime of your access token in seconds
id_token	The requested user data in JWT format, digitally signed by FACEIT, as described by OpenID Connect
scope	The scopes that this access token is authorized to use

Notes:

- By default, FACEIT Access Tokens issued with the authorization code grant type are valid for 24 hours.
- Make sure you store the Refresh Token in a safe place as you will need it to obtain a new Access Token when your current one expires. Your Refresh Token never expires. However, you must expect it to change at some point (make sure you update it in your storage every time you request a new Access Token).
- Normally, it is critical that you validate an ID Token before you use it, but since you are communicating directly with FACEIT over an intermediary-free HTTPS channel and using your client secret to authenticate yourself to FACEIT, you can be confident that the token you receive really comes from FACEIT and is valid. If your server passes the ID Token to other components of your app, it is **extremely important** that the other components validate the token before using it. Check the next section to find out how to do this.

By using information found in the **ID Token**, the 3rd party app can now query its own database to find out if the user is already registered and add the FACEIT specific information to that record, or otherwise to automatically create a new account for that specific user.

Please note that you need to validate all ID tokens on your server unless you know that they came directly from FACEIT. For example, your server must verify any ID tokens it receives from your client apps. However, validation is not necessary if you used the authorization_code grant type (in this case, the ID token was passed directly from FACEIT to your server over a secure connection).

ID tokens are sensitive and can be misused if intercepted. You must ensure that these tokens are handled securely by transmitting them only over HTTPS and only via POST data or within request headers. If you store them on your server, you must also store them securely.

One thing that makes ID tokens useful is the fact that you can pass them around different components of your app. These components can use an ID token as a lightweight authentication mechanism to authenticate the app and the user, but must ensure that it is validated before doing so.

ID Token Validation:

1. Verify that the ID token is a JWT that is properly signed with an appropriate FACEIT public key.
2. Verify that the value of aud in the ID token is equal to your app's client ID.

3. Verify that the value of iss in the ID token is equal to the value of the "issuer" field in the OpenID configuration.

To accomplish step 1 above, you need to retrieve FACEIT's public key from the "jwks_uri" endpoint defined in the OpenID configuration. You then need to use this key to verify that the ID Token is signed by FACEIT. Here is an example written in Java (http://stackoverflow.com/questions/24875253/openid-connect-how-to-verify-id-token-in-java).

5. Refreshing an Access Token

You should always refresh your access token before it expires, to avoid involving the user in authorizing your application again. To refresh your access token you need to make an authenticated (HTTP Basic Authentication) HTTPS POST request to the token endpoint defined in the OpenID configuration, using the `application/x-www-form-urlencoded` Content-Type header, sending the following parameters:

Field	Value	Description
grant_type	refresh_token	This tells the API that you want to refresh your Access Token
refresh_token	Your refresh token	The Refresh Token you received when your Access Token was issued

Implicit Grant Flow with the FACEIT JS SDK

1. Initialising the SDK on the login page(s)

The very first step is to insert and activate the "FACEIT Login for Apps" Javascript SDK on the page(s) where you want a "Connect with FACEIT" button to be present.

You can insert the following snippet just before the closure of the </body> tag of your web pages. Please note the second parameter passed to the init function is "token", which means we are using the implicit grant type.

```
<script src="https://cdn.faceit.com/oauth/faceit-oauth-sdk-1.3.0.min.js"
type="text/javascript"></script>
<script type="text/javascript">
   function callback(response){
       if(response.isIdTokenValid === true){
           return;
       }
       alert('The id token is not valid, something went wrong');
   }

   var initParams = {
       client_id: 'client1',
       response_type: 'token',
       state: 'informationYouWantPassedToTheRedirectUri'
   };
   FACEIT.init(initParams, callback);
```

```
</script>
```

Your **callback function** is called when authentication is completed by FACEIT. The response parameter sent to it will contain a boolean flag that represents the result of the OpenID Token validation on the FACEIT server. If the token has not been tampered with, the flag isIdTokenValid should be true. If this flag is false, the token cannot be trusted and should not be used.

2. Initialising the SDK on the redirect URI page

For the implicit grant type, the OpenID Connect JWT tokens need to be verified by the client to ensure that they haven't been tampered with over the wire.

```
<script src="https://cdn.faceit.com/oauth/faceit-oauth-sdk-1.3.0.min.js"
type="text/javascript"></script>
<script type="text/javascript">
   var initParams = {
       client_id: 'client1',
       response_type: 'token'
   };
   FACEIT.init(initParams);
</script>
```

3. First click

The user clicks the "Connect with FACEIT" link or button inside your app.

The button or link can be either:

- automatically injected into the page by the FACEIT SDK. To have the SDK inject a button, just add a div with id "**faceitLogin**" where you want the button to be, like below:

  ```
  <div id="faceitLogin"></div>
  ```

- programmatically executed by the calling the following Javascript function

```
<a href="#" class="button Faceit" onclick="FACEIT.loginWithFaceit()">Connect with
FACEIT</a>
```

The user is redirected to FACEIT (a new pop-up is opened and directed to FACEIT) and asked to authorize the 3rd party application to use their details.

The `loginWithFACEIT` function returns the reference to the opened pop-up or null if something went wrong.

4. User Authentication

Once the user is authenticated by FACEIT, the Access Token and ID Token are sent to the **Redirect URI** of the 3rd party app. As described above, it is important that the FACEIT JS SDK is included on the page where the request is redirected. The query string will contain a hash with all of the tokens and other parameters, similar to:

https://example.com/Faceit-callback#access_token=1359da65-c0a3-46fc-81d9-c50bbd9a6cbd&token_type=bearer&state=998412345&expires_in=5023793&scope=openid%20profile%20email&id_token=eyJhbGciOiJSUzI1NiJ9.eyJwaWN0dXJlIjoiaHcu79XloSMdxcQPA1TARxCOWK8-n4mqEn3_6ISU_KgiLtFT5s+_

If the FACEIT SDK is included on this page then you don't have to worry about processing the information in the URL, as the SDK will make this information available to you through functions that are described in the following section.

5. Using the JS SDK

Once the token has been passed to the Redirect URI, you can start using the FACEIT JS SDK to retrieve information about the user from FACEIT. You can do so in your callback function, as described in section 1.

The functions provided by the SDK are:

Function	Description

`FACEIT.init(params, [callback])`	Initializes the SDK. You must provide a `params` object containing at least the `client_id` and `response_type` properties:
	`{`
	`client_id: '...',`
	`response_type: 'token' | 'code',`
	`redirect_popup: true/false,`
	`state: '...'`
	`}`
	Additionally, you may provide a callback function which will be invoked asynchronously when the token is received from the FACEIT Api

`FACEIT.loginWithFaceit(options)`	Triggers the FACEIT pop-up that authenticates the user. You may provide an `options` object to specify the popup's `width` and/or `height`. `options` defaults to: `{width: 750, height: 825 }`
`FACEIT.getAuthenticationStatus()` (only works for the Implicit grant type)	Returns a boolean that represents the user's authentication status with FACEIT. If `false`, you cannot retrieve user information anymore and will need to display the "Connect with FACEIT" button again.

Final considerations

- If you are using the authorization code grant type, make sure to always refresh your Access Token before it expires to avoid involving the user in authorizing your application again
- Make sure you keep your tokens and client secret safe as they are sensitive information that should not be exposed
- When integrating the FACEIT Login into your application, always make sure that you use the latest endpoints defined by the FACEIT OpenID configuration service, available online at https://api.faceit.com/auth/v1/openid_configuration