OMPASS

REST API - U2F

Content

What is U2F?

Universal 2nd Factor (U2F), which is also known as Two-Factor Authentication (2FA), is an authentication method that allows final login through OMPASS secondary authentication after password authentication is completed.

Preparation

Please check the secret key frist from "Edit" that is automatically assigned when registering the application from "App Management".

Please note that security problems may occur if the secret key is exposed to others, so make sure to be private.

Process for Applying OMPASS

1. Verify User ID and password
2. OMPASS registration & authentication API
3. OMPASS registration & U2F authentication
4. Redirect access token
5. Validate access token
6. Login

OMPASS-U2F

From the server-side

Once verifying user ID and password is completed from the server-side, call OMPASS API including the secret key in "HTTP HEADER" and the user ID in "Request Body".

OMPASS Registration & Authentication API

```
POST
URL /v1/ompass/u2f
URL EXAMPLE – https://interface-api.ompasscloud.com/v1/ompass/u2f
```

Header

Key	Type	Description
Authorization	Bearer	· Secret Key assigned to the application · "Bearer" is required to be specified as the authorization type, and a space is required between "Bearer" and "Secret Key" · Example : Bearer djfk39dkfdl39dldjmgjd4idls83jflghidhs83jfk

Request Body (JSON)

Key	Type	Description
user_id	String	User ID
lang_init	String	English (EN) as an Initial language setting value of OMPASS URI to recieve a response

Example of Request Body

```
{
  "user_id" : "omsecurity",
  "lang_init" : "KR"
 }
```

Response (JSON)

· In case of authentication success

Key	Type	Description
user_id	String	User ID
register	Boolean	True or false about the registration status in OMPASS
ompass_uri	String	A URI of the registration page in case the user is not registered yet in OMPASS A URI of the authentication page in case the user is already registered in OMPASS

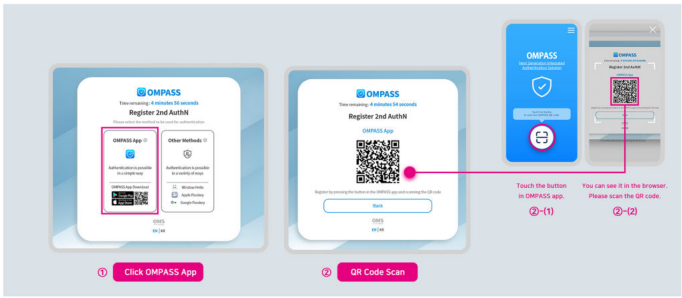
```
{
   "code" : 200,
   "message" : "ok",
   "data" : {
    "user_id" : "omsecurity",
    "is_register" : false,
    "ompass_uri" : "https://interface-api.ompasscloud.com/register/did/14?do..."
   }
 }
```

OMPASS Registration & U2F Authentication

From the client-side (user)

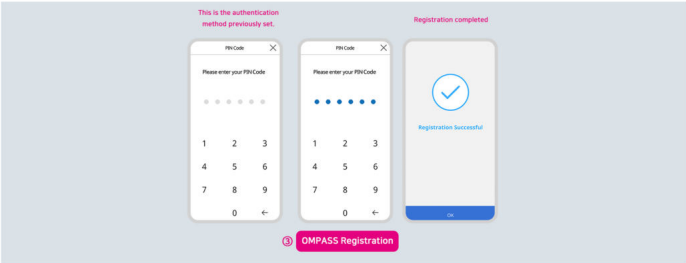
The browser from client-side calls OMPASS URI that is received a response.

Example of registration interface call in pop-up window
- In case the user is not registered in OMPASS, the following pop-up window will be displayed.



❶ Click the [OMPASS App] button in order to register second authentication methods for login.

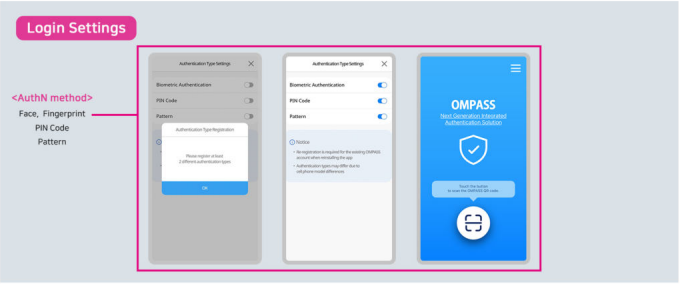
❷ Scan the QR code with the OMPASS app.



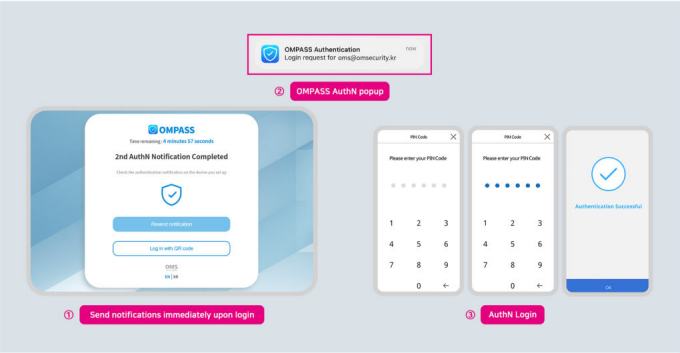
❸ Once the user has registered the authentication methods, complete the registration with the selected authentication method.

---

❸Example of how to register the authentication method)



Example of authentication interface call in pop-up window
- In case the user is not registered in OMPASS, the following pop-up window will be displayed.



❶ Notifications are sent immediately upon login.

❷ Tap on the notification of OMPASS to launch the app from your phone.

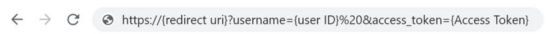
❸ Complete the authentication using previously selected authentication methods.

Redirect Access Token

From the client-side (user)

Whenever OMPASS registration or authentication is successfully completed, HTTP redirects to the redirect URI designated in the application at the pop-up window of the OMPASS page, and includes a Query String containing the access token.
Parsing the redirected authentication token from client-side will be passing to server-side.

Example



https://{redirect uri}?username={user ID}%20&access_token={Access Token}

Validate Access Token

From the server-side

The server will validate the token by calling the token validation API for OMPASS authentication including the access token passed from the client-side.
If the response for HTTP STATUS CODE of the API request is 200, login will proceed after verifying ID.

OMPASS Success Token Validation API

```
POST
URL /v1/ompass/token-verification
URL EXAMPLE – https://interface-api.ompasscloud.com/v1/ompass/token-verification
```

Header

Key	Type	Description
Authorization	Bearer	· Secret Key assigned to the application · "Bearer" is required to be specified as the authorization type, and a space is required between "Bearer" and "Secret Key" · Example : Bearer djfk39dkfdl39dldjmgjd4idls83jflghidhs83jfk

Request Body (JSON)

Key	Type	Description
user_id	String	User ID
access_token	String	The access token received as a redirect URI

Example of Request Body

```
{
  "user_id" : "omsecurity",
  "access_token" : "dfj2ld92lldj29cldl29llduuufnbsd229312000dfl2ldio2o019029dj10wj"
 }
```

Response (JSON)

· In case of authentication success

Key	Type	Description
user_id	String	User ID

```
{
  "code" : 200,
  "message" : "ok",
  "data" : {
   "user_id" : "omsecurity"
  }
 }
```

API Error Messages

code	message
000	Required Request Body is missing.
001	Please make a request including the secret key.
002	Please make a request including the user ID.
003	Please make a request including the access token.
004	Invalid secret key.
005	The secret key format does not match. example) Bearer dl239d29dl292kmdjf139f2ds
006	User ID cannot exceed 30 digits.
011	The token has expired.
012	It is a token of an unsupported format.
013	The token is not configured correctly.
014	Failed to verify the existing signature.