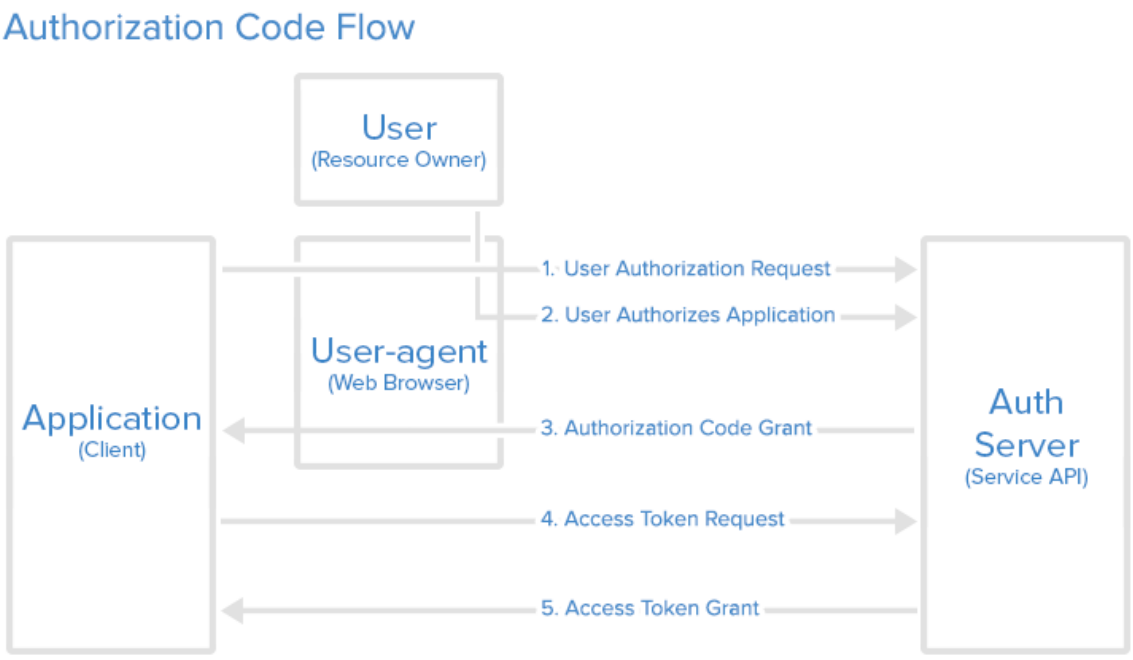
Visma.net Integrations-OAuth 2.0 Authorization Flow



What is OAuth 2.0 Grant Type?

In OAuth 2.0, the term “grant type” refers to the way an application gets an access token. OAuth 2.0 defines several grant types, in Visma.Net Integrations the grant type used as of today is “Authorization Code”.

Each grant type is optimized for a particular use case, whether that’s a web app, a native app, a device without the ability to launch a web browser, or server-to-server applications.

The Authorization Code Flow

The “Authorization Code” grant type is generally used by web and mobile apps. It’s different from other grant types in that it requires the application to open a browser to initiate the flow. Therefore the client application has to be able to interact with the users browser and receive incoming requests from the authorization server.

The flow is split up in two parts:

1. The application starts an authorization code request.
    a. The application opens a browser to send the user to the OAuth server.
    b. The user sees the authorization prompt and approves the app’s request.
    c. The user is redirected back to the application with an authorization code in the query string.
2. The application starts an access token request.

a. The application requests an access token with the authorization code provided in the previous step.
b. The OAuth server exchanges this authorization code for an access token

Getting the User's Permission

The point of using OAuth is to enable users to get access to the parts of the Application that they need. To be able to do this the application first has to decide what permissions it is going to ask for, then send the User to a browser to get their permission to do this. To start the flow the application constructs a URL like the following and opens a browser to that URL.

```
GET https://integration.visma.net/API/resources/oauth/authorize
?response_type=code
&client_id={yourApiClientID}
&redirect_uri={yourRegisteredRedirectURI}
&scope=financialstasks
&state={randomStringGeneratedByApplication}
```

Here's each of the query parameters explained:

- **response_type=code** - This tells the authorization server that the application is initiating the authorization code flow. - *This field is **mandatory** and has to be "Code".*
- **client_id** - The public identifier for the application. - *This field is **mandatory** and is obtained when registering for Visma.Net Integrations*
- **redirect_uri** - Tells the authorization server where to send the user back to after they approve the request. - *This field is **mandatory** and must match the URI you registered for Visma.Net Integrations*
- **scope** - One or more space-separated strings indicating which permissions the application is requesting. - *This field is **mandatory** and case sensitive, it has to be **financialstasks***
- **state** - The application generates a random string and includes it in the request. It should then check that the same value is returned after the user authorizes the app. This is used to prevent CSRF attacks. - *This field is **recommended** for the reasons stated previously*

When the user is redirected to the authorization server they will be prompted to enter their credentials to allow the applications request.

Redirect back to the Application

If the user approves the request, the authorization server will redirect the browser back to the **redirect_uri** specified in the request, adding a **code** and **state** to the query string.

For example, the user will be redirected back to a URL such as:

```
https://yourApplicationLivesHere.com/redirect
?code={authorizationCodeGeneratedByAuthServer}
&state={randomStringGeneratedByApplication}
```

The **state** value will be the same value that the application initially set in the request. The application is expected to check that the state in the redirect matches the state it originally set. This protects against CSRF and other related attacks.

The **code** is the authorization code generated by the authorization server. The lifespan of this code is 10 minutes.

Exchange the Authorization Code for an Access Token

The second part of the Authorization Code flow is to exchange the Authorization Code the user just received for an Access token.

The application makes a POST request to the token endpoint. There are two ways of doing this, HTTP Basic authentication and sending the client_ID and client_Secret in the request body. Both ways require a couple of parameters.

Here's each of the query parameters explained:

- **grant_type=authorization_code** - This tells the token endpoint that the application is using the Authorization Code grant type. - *This has to be "authorization_code".*
- **Code** - The application includes the authorization code it was given in the redirect.
- **redirect_uri** - The same redirect URI that was used when requesting the code.
- **client_id** - The application's client ID. - *This is the ID obtained when registering for Visma.Net Integrations.*
- **client_secret** - The application's client secret. This ensures that the request to get the access token is made only from the application, and not from a potential attacker that may have intercepted the authorization code. - *This is the secret obtained when registering for Visma.Net Integrations.*

  Please note: All these fields are mandatory.

1. HTTP Basic authentication

The HTTP Basic authentication scheme is the preferred way and we encourage all clients that can utilize this authentication scheme to use it. It is done by providing an Authorization header on the request:

- **Authorization: Basic XXXXXXXXXX=** - The value of the Authorization header is a string composed from the authorization method a space("Basic ") followed by a Base64 encoded string obtained from combining **client_ID** and **client_Secret** separated by a colon(client_id:client_secret).

Example of this can be seen below:

Request Header:
POST https://integration.visma.net/API/security/api/v2/token
Content-Type: application/x-www-form-urlencoded
Authorization: Basic Y2xpZW50X2lkOmNsaWVudF9zZWNyZXQ=

Request Body:
grant_type=authorization_code&
code={authorizationCodeGeneratedByAuthServer}&
redirect_uri={yourRegisteredRedirectURI}&

2. Request body

The second option for the client is to send it's client_id and client_secret to Visma.net Integrations in the request body.

This option should be used by clients that cannot utilize HTTP Basic authentication directly.

Example of this can be seen below:

Request Header:
POST https://integration.visma.net/API/security/api/v2/token

Content-Type: application/x-www-form-urlencoded

Request Body:
grant_type=authorization_code&
code={authorizationCodeGeneratedByAuthServer}&
redirect_uri={yourRegisteredRedirectURI}&
client_id={yourApiClientID}&
client_secret={yourApiClientSecret}

Please note:
*All the request parameters sent to https://integration.visma.net/API/security/api/v2/token **must** be sent on request body. Event though the entire transmission is encrypted when using HTTPS, it is **not recommended** to send sensitive information (password and client_secret) in the URL as query parameters. The URLs are stored in web server logs which means that your sensitive data are saved in clear text on the server.*

The token endpoint will verify all the parameters in the request, ensuring the code hasn't expired and that the client ID and secret match. If everything checks out, it will generate an access token and return it in the response.

A successful response will look like this:

Response headers:
HTTP/1.1 200 OK
Content-Type: application/json

Response body:
{
"token": "1f729814-1a98-4c8e-860b-76ec004742f5",
"token_type": "bearer",
"scope": "financialstasks"
}

Now the client can start using the token and token_type to make request through VNA against Visma.net Visma.net Financials resources.
Please note: *Once generated, the token currently does not expire; the token can be used for making subsequent calls towards the exposed endpoints. A new token should not be generated before making a new call; the token is not connected to the session.*