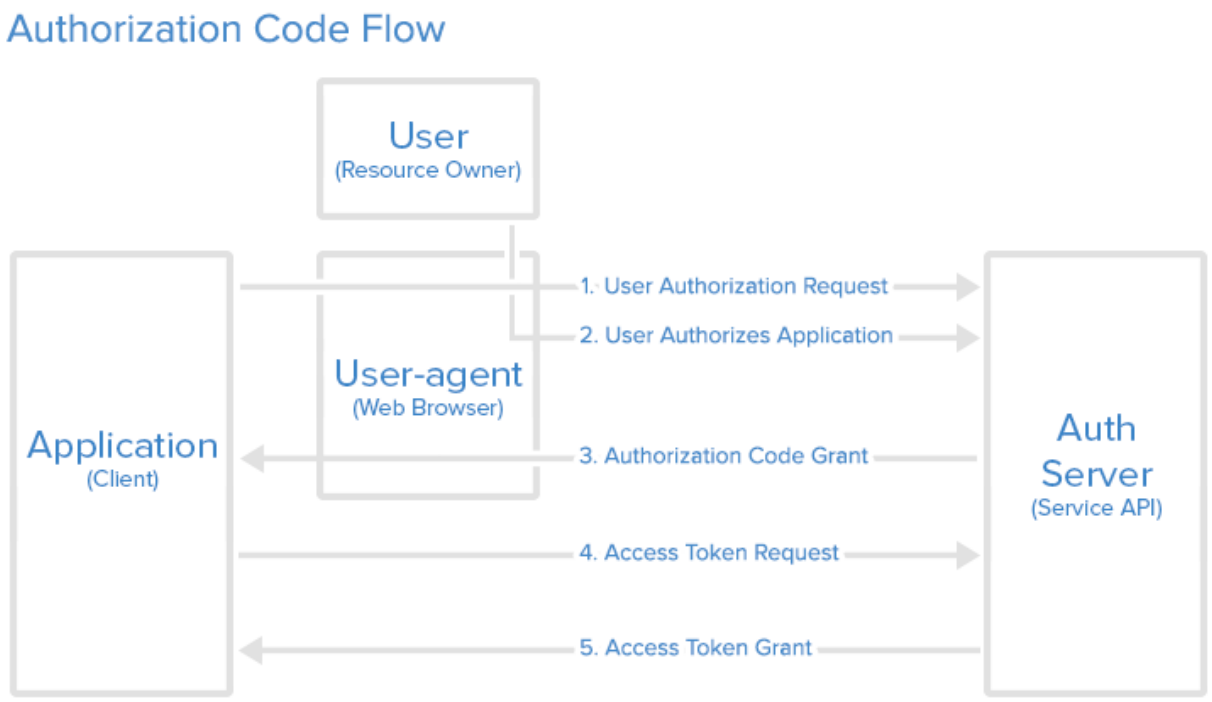
Visma.net Integrations-OAuth 2.0 Authorization Flow



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

1) 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.

- The first step in the process will be to get the user to login to Visma.net and give consent to access the required data and functions (scopes) on behalf of the user. Since this step requires the user to provide a username and password, it is very important that the user is redirected to the Visma.net login-page and provides the credentials here.

- Username and password should not under any circumstances be provided to any other party than this Login-page. The integration / application should never be aware of the username/password.

- The first step is therefore to generate a URL to the Visma.net login-page and redirect the user to this URL. The URL must contain the following parts/parameters:

GET request: https://integration.visma.net/API/resources/oauth/authorize

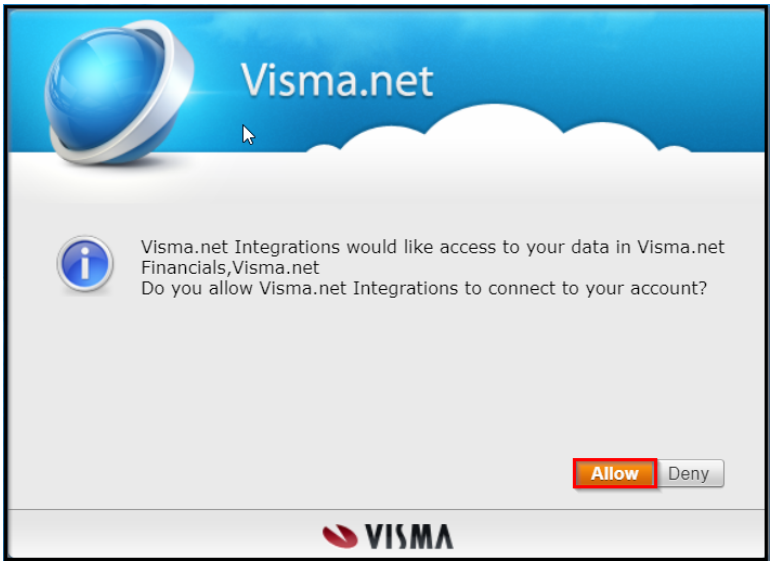
?**response_type**=code
&**client_id**={yourApiClientID}
&**redirect_uri**={yourRegisteredRedirectURI}
&**scope**=financialstasks
&**state**={randomStringGeneratedByApplication}

Parameter	Definition
URL	Visma.net login-page. https://integration.visma.net/API/resources/oauth/authorize
response_type	This tells the authorization server that the application is initiating the authorization code flow. - This parameter is **mandatory** and should be set as "**Code**".
client_id	The ClientId of the application that is requesting the authorization. This parameter is **mandatory.** *"Client_Id & client_secret" can be obtained during the onboarding process.* **Example:** client_id=visma_test_client_1234
Redirect_uri	Tells the authorization server where to send the user back to after they approve the request. - This parameter is **mandatory** and must **match** one of the Redirect_URIs that you've registered for Visma.Net Integrations API Client. Otherwise, the request will return an error.

Scope	The different scopes you want the user to authorize access to. In Visma.net Financials, only one scope exists, "**financialstasks**". This field is **mandatory** and **case-sensitive**, this should be set as "**financialstasks**"
state	State is an **optional** parameter you could provide to differentiate different sessions or calls to the login-page. The value of the parameter is sent back to your application in the redirect_uri call. For instance, If you are hosting a web-application, you can have multiple requests going at the same time for multiple users. The state will then be used to differentiate the calls when the redirect-call is done. Additionally, 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.

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

- When signed in, the user will be presented with the consent-screen, where the user will be prompted to allow the client to access the users' data (Scope).



2) 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}

Parameter	Definition
code	The authorization code generated by the authorization server that will be used to exchange for a Token. "Code" **expires in 10 minutes**
state	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.

3) Exchange the Authorization Code for an Access Token

The last 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.
(https://integration.visma.net/API/security/api/v2/token)

There are two ways of doing this, **HTTP Basic authentication** or sending the *client_ID* and *client_Secret* in the **Request Body**. Both ways require a couple of parameters. (All the following keys are **mandatory**.)

Field	Definition
grant_type	This tells to the token endpoint that the application is using the "Authorization Code" grant type. - This should be set as **"authorization_code"**
Code	The code obtained from the callback to the Redirect URI
redirect_uri	The same redirect URI used during the "authorization" process.
client_id	The Client ID of the Client-application. *"Client_Id & client_secret" can be obtained during the onboarding process.* **Example:** client_id=visma_test_client_1234
client_secret	The Client Secret of the Client-application. 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. **Example:** client_secret=98ana4-7a8d-4e14-2247-957812ac9ec

3.1) Access Token Generation methods

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

An example request can be seen below:

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

Request Header:

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 its **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.

An example of this can be seen below:

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

Request Header:
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 be advised:

All the request parameters sent to https://integration.visma.net/API/security/api/v2/token ***must*** *be sent on request body. Even though the entire transmission is encrypted when using HTTPS, it is* ***not recommended sending*** *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 is 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

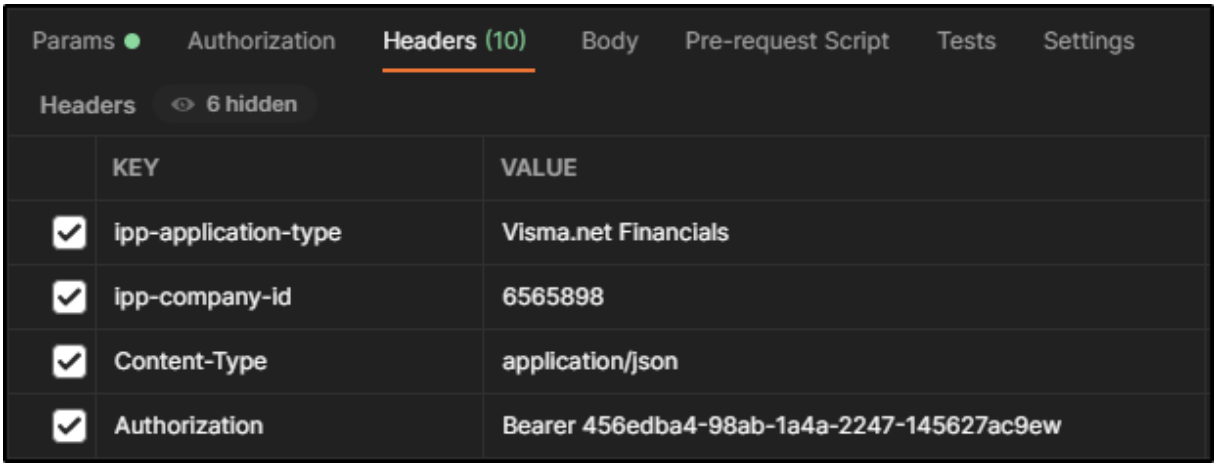
Response body:

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

Now the client can start using the token and token_type to make requests to the Visma.net Financials resources.

Please note: *Currently, the "token" does not expire, and it 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. Each consecutively generated token automatically invalidates the recently created one.*
(Request made from the Same CLIENT ID & the Same Visma.Net Financials user for all integrations/company)

Example API request: Request Headers



Key	Value
ipp-application-type	Static Value. Should always be set as "Visma.net Financials"
ipp-company-id	Financials ERP Company ID (Can be checked by making a GET Call to "Context" Endpoint)
Content-Type	original media type of the resource
Authorization	HTTP authentication scheme that involves security tokens called bearer tokens