How to configure published web services with Oauth and Azure AD

Revision 20220520

Denodo Inc.: 525 University Avenue, Suite 31, Palo Alto, CA 94301. USA www.denodo.com

1 INTRODUCTION

Azure Active Directory (Azure AD) is an Identity Provider service offered by Microsoft as part of its Azure cloud offerings. In this article, Azure AD will be configured to function as the Authorization Server in the OAuth flow described in the OAuth 2.0 authentication in Virtual DataPort Web Applications (Northbound) section of the Knowledge Base. This will allow users to connect to the web services provided by Denodo using their credentials in Azure AD.

2 PREREQUISITES

An Azure account. Azure comes with Azure AD, which will function as the Identity Provider for the Denodo web services. More information about the OAuth protocol can be found in the OAuth 2.0 Protocol Overview article from the Denodo Knowledge Base. It is important to note that OAuth can be streamlined for machine to machine communication, while SAML (another popular web authentication protocol) generally requires user consent.

After performing the configuration in this article, Denodo web services will be able to use the OAuth authentication method, which will allow users with the appropriate credentials in Azure AD to connect automatically to the web service.

This article will review:

1. How Azure AD should be configured to function as the Identity Provider for the Virtual DataPort server.
2. The configuration necessary in the Virtual DataPort server to have OAuth tokens validated against Azure AD.
3. The steps necessary to publish a web service endpoint with OAuth authentication enabled.
4. How access tokens can be generated in Denodo, and how these can be used to create a data source in Denodo that consumes the OAuth web service endpoint.

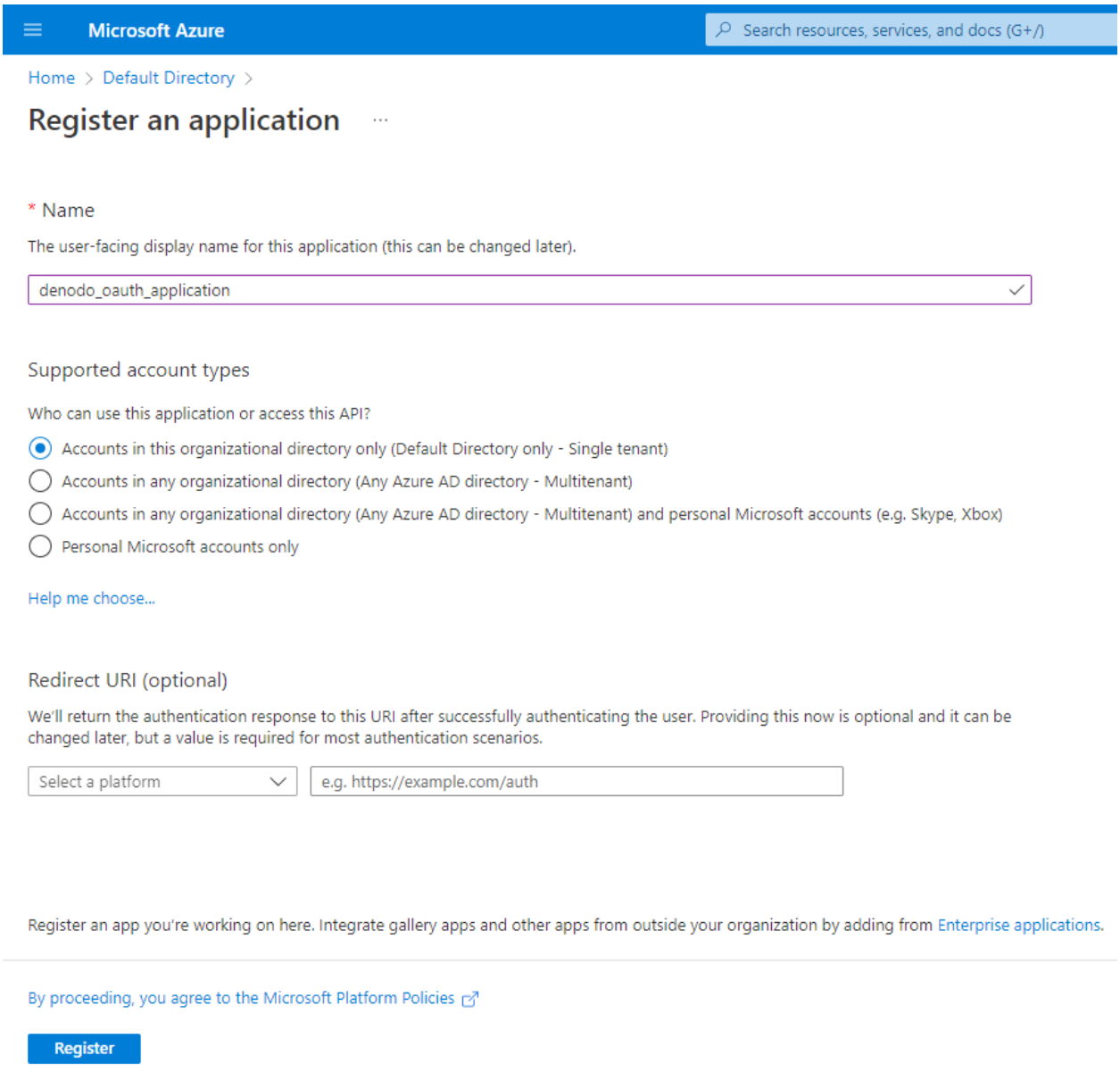
In the case that the user configuring the Denodo application in Azure AD is not an administrator, it will be necessary to have an administrator grant permissions to request the following scopes for the application:

- `offline_access`: this scope is required in the case that the user wants to request refresh tokens to continue accessing the Denodo web service.
- Any other scopes that are created to correspond to roles in Denodo.

We recommend checking in with your Azure AD administrator to make sure that permissions for your users are correctly configured, and that they will be able to successfully request the above scopes.

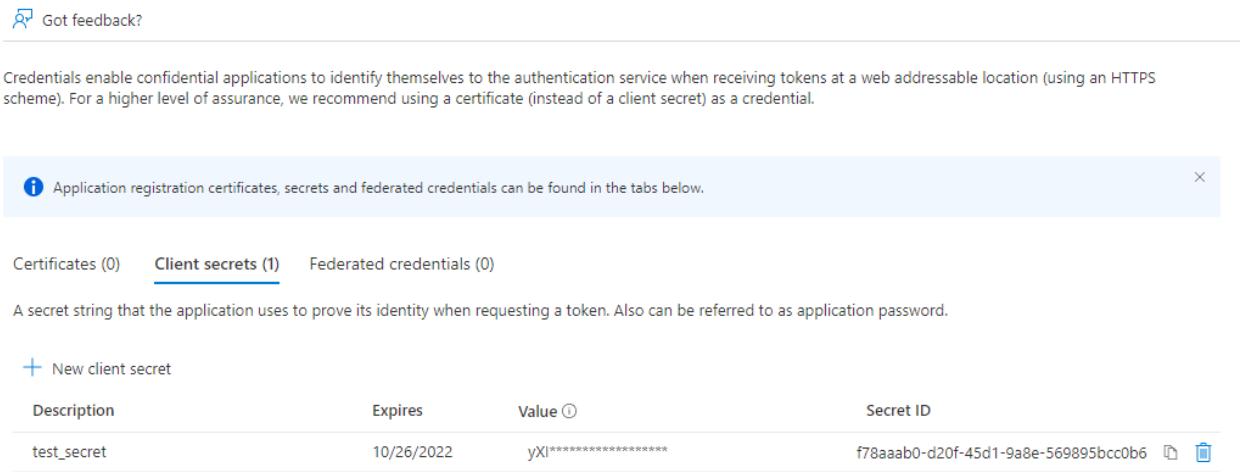
3 CONFIGURING AZURE AD

Navigate to "Azure Active Directory" in the Azure portal. Go to "App registrations", and click "New registration". This will bring up the following page:



Input a name for the application. The Supported account types will be determined by your internal security policy, so it is recommended to check this with your administrators. The Redirect URI will be configured later. Hit "Register".

In the "Certificates & Secrets" section, add a new secret. Then, make sure to immediately copy the secret value and save it for later.



Next, in "Expose an API", select "Add a scope". Use the default API URI, then add a name of a scope (this is necessary for Azure to provide a token). In this case, we will name the scope "oauth_read".

Add a scope

Scope name *

oauth_read

api://2e2bc87d-225d-45c3-a949-a0c870b60254/oauth_read

Who can consent?

Admins and users | Admins only

Admin consent display name *

denodo_oauth_read

Admin consent description *

Grants the oauth_read role in the Virtual DataPort server.

User consent display name

e.g. Read your files

User consent description

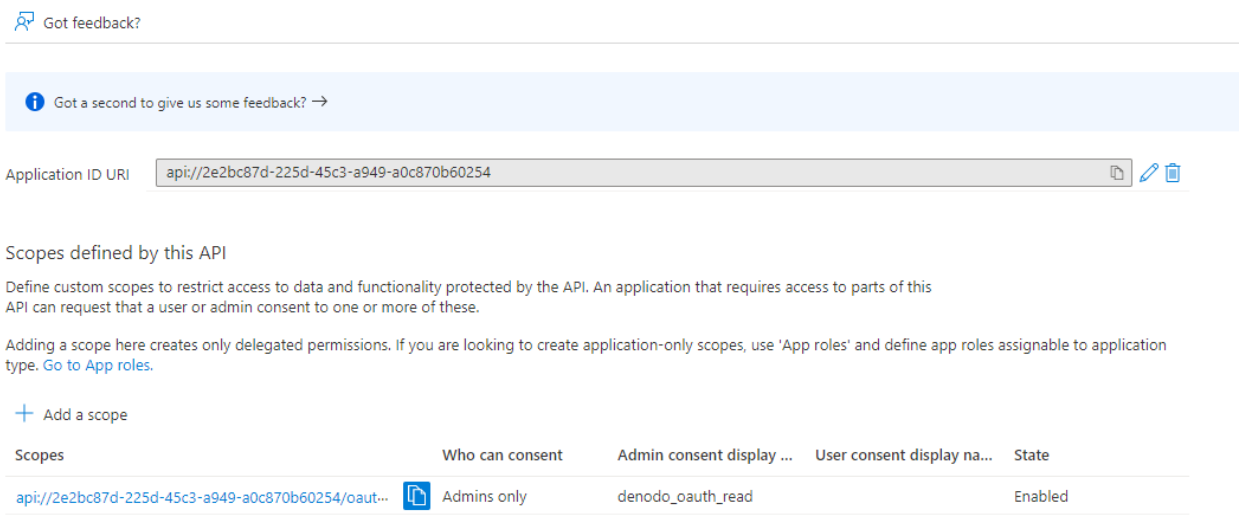
e.g. Allows the app to read your files.

State

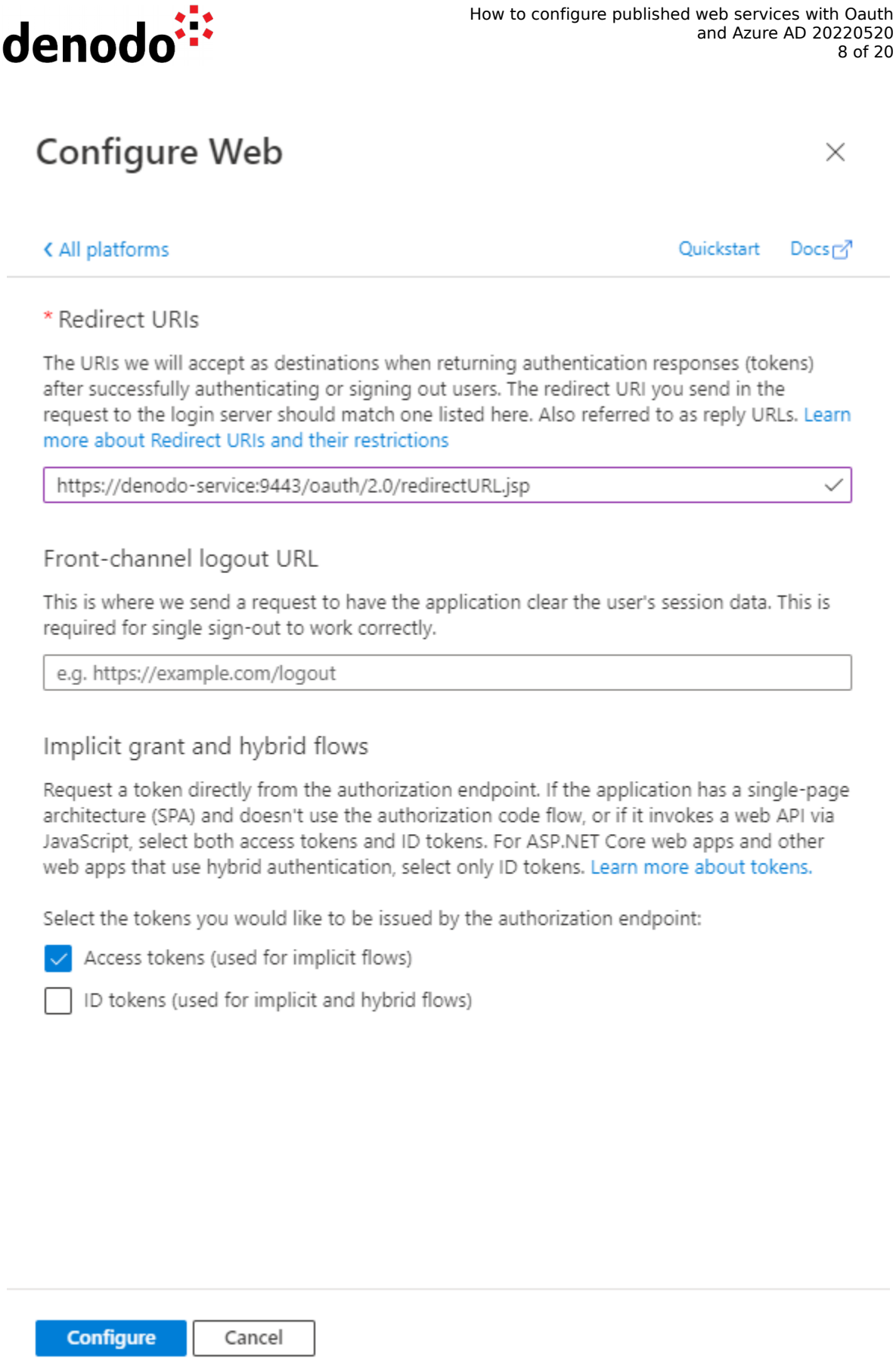
Enabled | Disabled

Add scope | Cancel

After adding the scope, it can be seen in the list of scopes for the application:



Finally, in the "`Overview`" page of the application, click the "`Add a Redirect URI`" link. Click on "`Add a platform`", select "`Web`", and configure the "`Redirect URI`" and "`Implicit grant and hybrid flows`" fields:


Configure Web
All platforms
Quickstart
Docs
* Redirect URIs
The URIs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating or signing out users. The redirect URI you send in the request to the login server should match one listed here. Also referred to as reply URLs. Learn more about Redirect URIs and their restrictions
https://denodo-service:9443/oauth/2.0/redirectURL.jsp
Front-channel logout URL
This is where we send a request to have the application clear the user's session data. This is required for single sign-out to work correctly.
e.g. https://example.com/logout
Implicit grant and hybrid flows
Request a token directly from the authorization endpoint. If the application has a single-page architecture (SPA) and doesn't use the authorization code flow, or if it invokes a web API via JavaScript, select both access tokens and ID tokens. For ASP.NET Core web apps and other web apps that use hybrid authentication, select only ID tokens. Learn more about tokens.
Select the tokens you would like to be issued by the authorization endpoint:
Access tokens (used for implicit flows)
ID tokens (used for implicit and hybrid flows)
Configure
Cancel

The Denodo redirect endpoint is the following by default:
http<s>://<server_hostname>:<server_port>/oauth/2.0/redirectURL.jsp

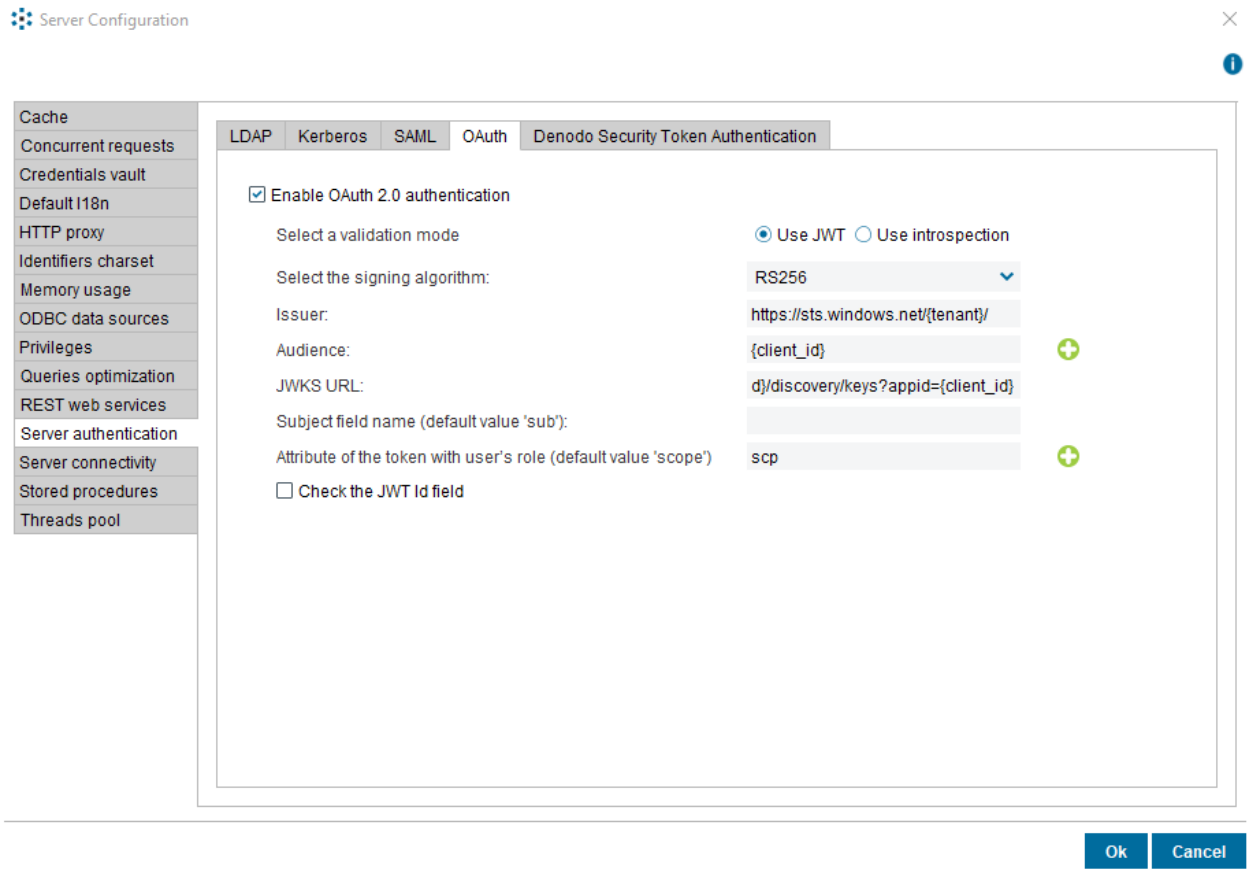
For “`Implicit grant and hybrid flows`”, “`Access tokens`” should be selected. ID tokens are used for the OpenID protocol.

Note that Azure only allows the HTTP protocol to be used when redirecting to “localhost”.

4 CONFIGURING DENODO TO USE AZURE AD AS THE IDENTITY PROVIDER

Navigate to the "Administration > Server configuration > Server authentication > OAuth" window in the Virtual DataPort Administration Tool or Design Studio, and input the following:

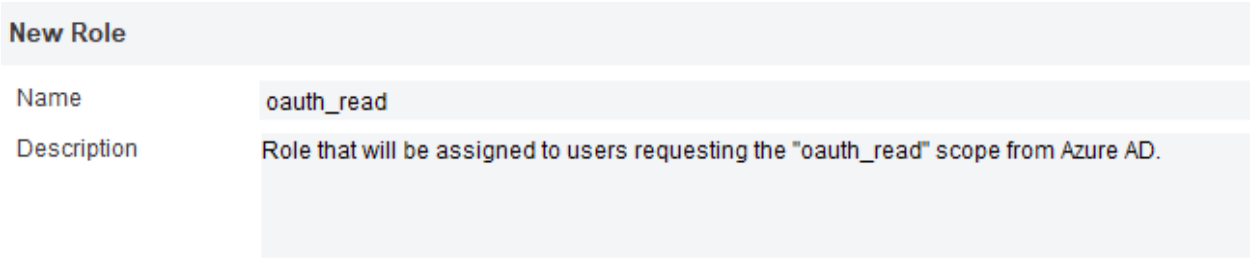
- Select "Use JWT". The Azure AD Identity Provider will supply JSON Web Tokens.
- Select the signing algorithm: RS256 (supported by Azure by default).
- Issuer: Found in the "issuer" property of the metadata page: https://login.microsoftonline.com/{tenant}/.well-known/openid-configuration?appid={client_id} (with {tenant} and {client_id} replaced with their corresponding values from the "Overview" page of the Azure ID application).
- Audience: Client ID of Azure AD client.
- JWKS URL: Found in the "jwks_uri" property of the metadata page: https://login.microsoftonline.com/{tenant}/.well-known/openid-configuration?appid={client_id} (same as before).
- Subject field name: leave empty, or provide "sub".
- Attribute of token with user's role: scp (to indicate the scope field)
- Check the JWT id field: Make sure that this is **disabled**. Azure AD uses a "uti" claim for this information instead of the standard "jti" claim.



Click "Ok".

Now, in order for authenticated users to be assigned permissions in the Virtual DataPort server, a role must be created corresponding to the scope we defined in Azure AD.

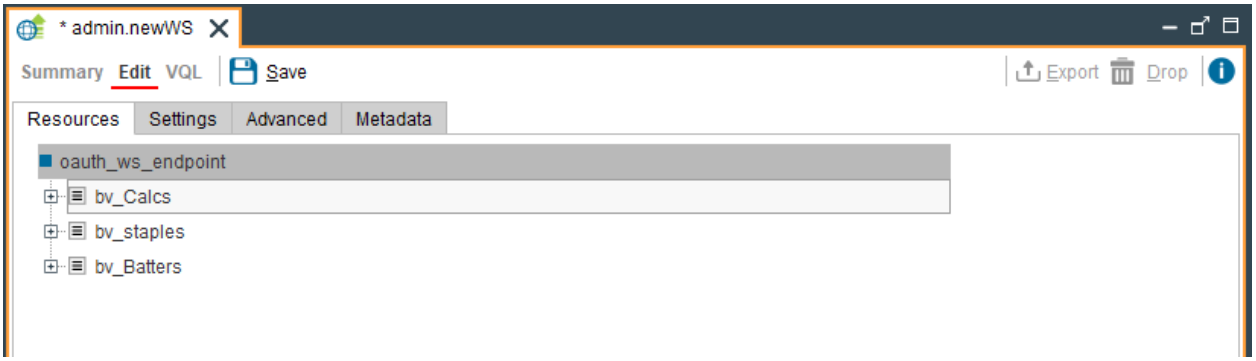
To do this, navigate to the "`Administration > Role management`" tab, and click on "`New`". Enter the name of the scope that you created in Azure AD, without the application URI. For example, in our case, we will create a role named "`oauth_read`".



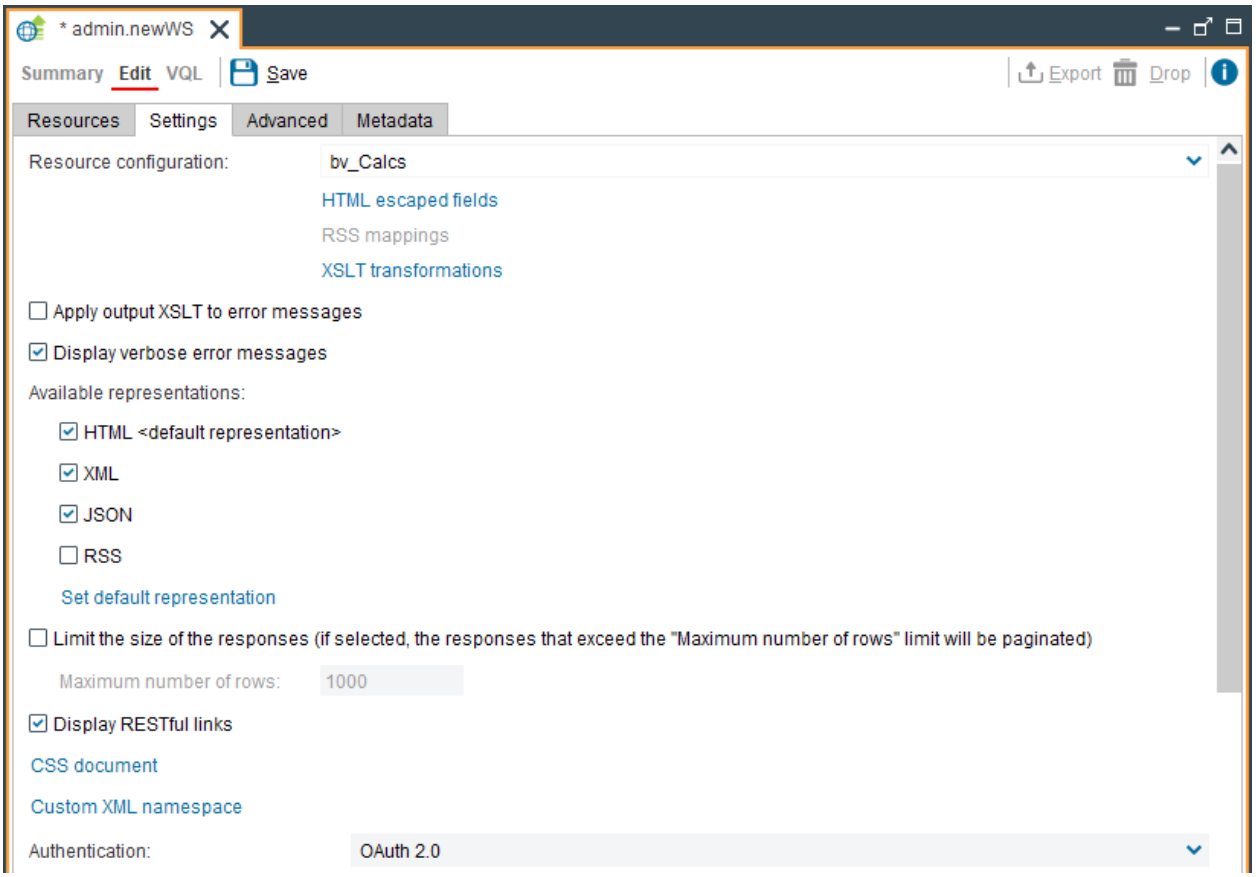
You will want to make sure that this role has permissions to access the views and web services that you will query when making requests to the OAuth secured endpoint in Denodo. For testing purposes, we assigned the role "`Admin`" privileges on a Virtual Database.

5 PUBLISHING A WEB SERVICE ENDPOINT USING OAUTH AUTHENTICATION

To start creating a web service, select "`File > New > Data Service > REST Web service`".
In the window that appears, drag a view (or multiple) into the "`Resources`" tab:



These views will be queryable from the endpoint.
In order to configure the web service to use OAuth authentication, all that will be required is that the "`Authentication`" property in the "`Settings`" tab is set to "`OAuth 2.0`".



Save the web service, and deploy it. It can now be queried using OAuth authentication, the semantics of which will be displayed in the next section.

6 MAKING REQUESTS TO AN OAUTH SECURED ENDPOINT

OAuth requests require access tokens to be included in the HTTP requests as an Authorization header. For example:

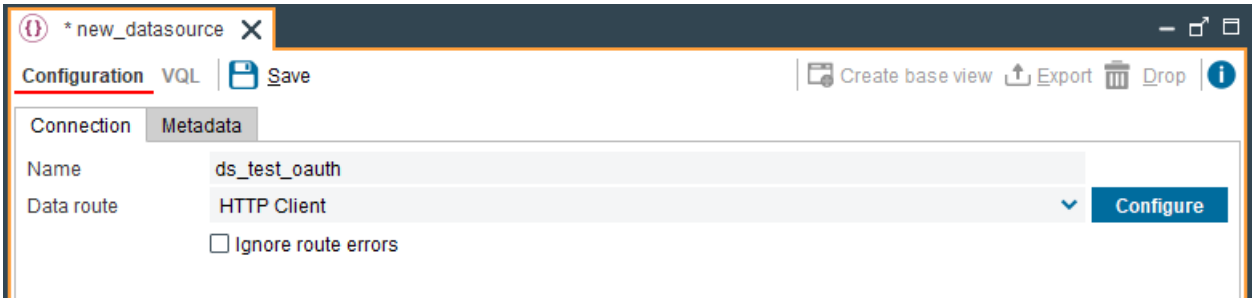
```
Authorization: Bearer <access_token>
```

There are other requests that must be sent to the Identity Provider in order to generate this access token, and other clients can usually perform this with some input from the user. However, in this case, the easiest way to obtain an access token is using the OAuth Credentials Wizard of the Virtual DataPort server. In the following sections, we will see connections to the Denodo OAuth endpoint using both Denodo itself, and using an external client (in this case, cURL).

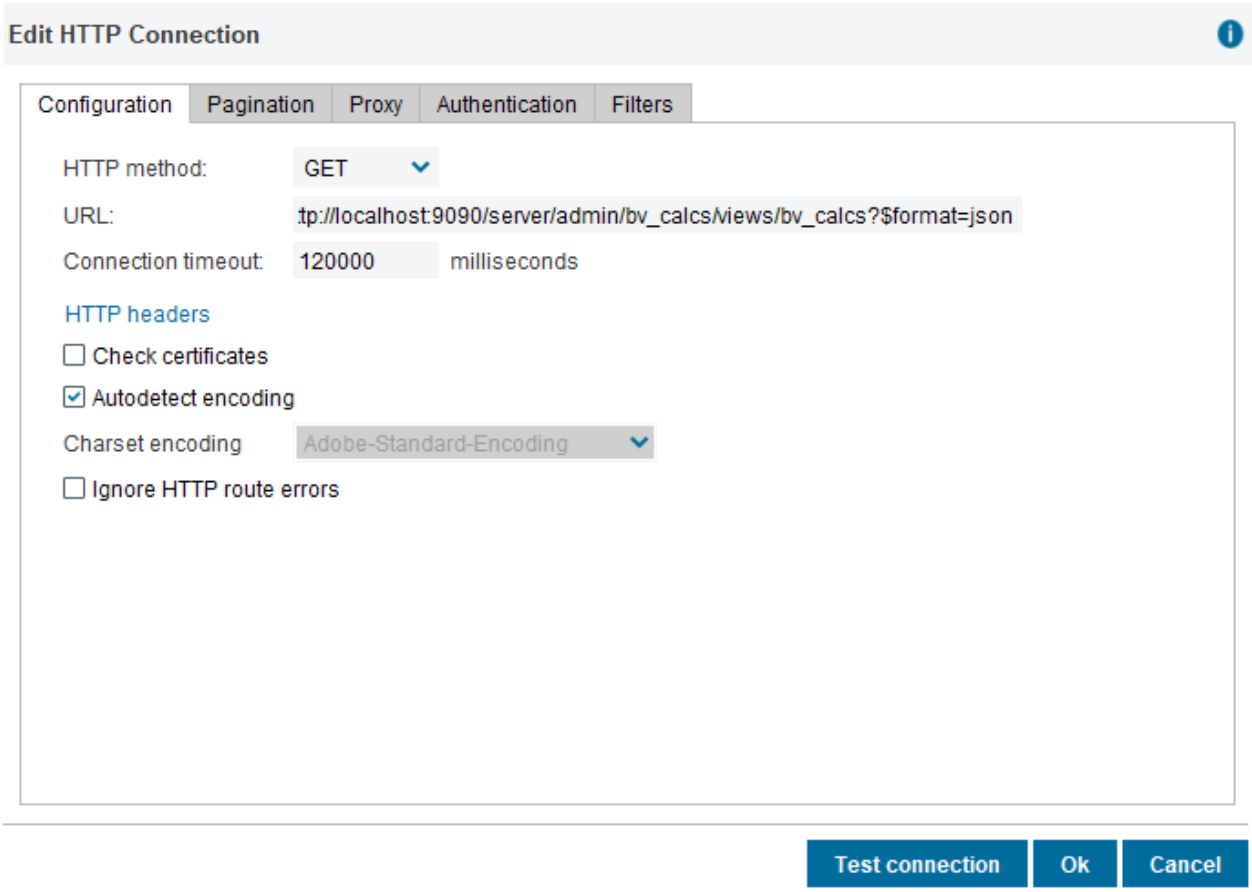
6.1 USING DENODO

When configuring data sources that authenticate using OAuth, the Virtual DataPort server includes the OAuth credentials wizard in the configuration of the data source itself.

In this example, a JSON data source will be created. Since the web service is an HTTP endpoint, we will specify “HTTP Client” as the Data Route:

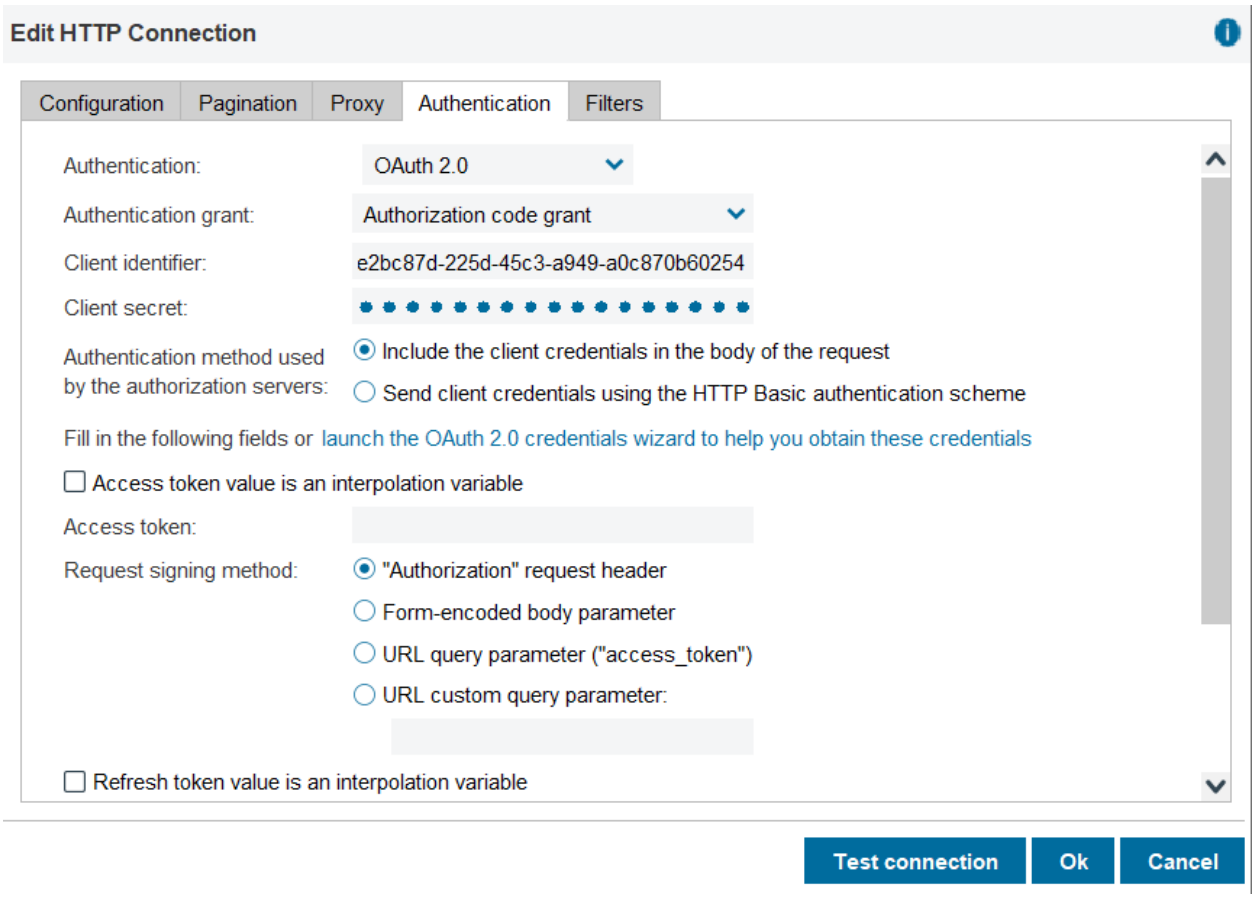


In the “Configuration” tab of the “Edit HTTP Connection” window, we will specify our OAuth endpoint. Since we are using a JSON data source, we will force the endpoint to return JSON formatted data by adding “?$format=json” to the end of the URI:



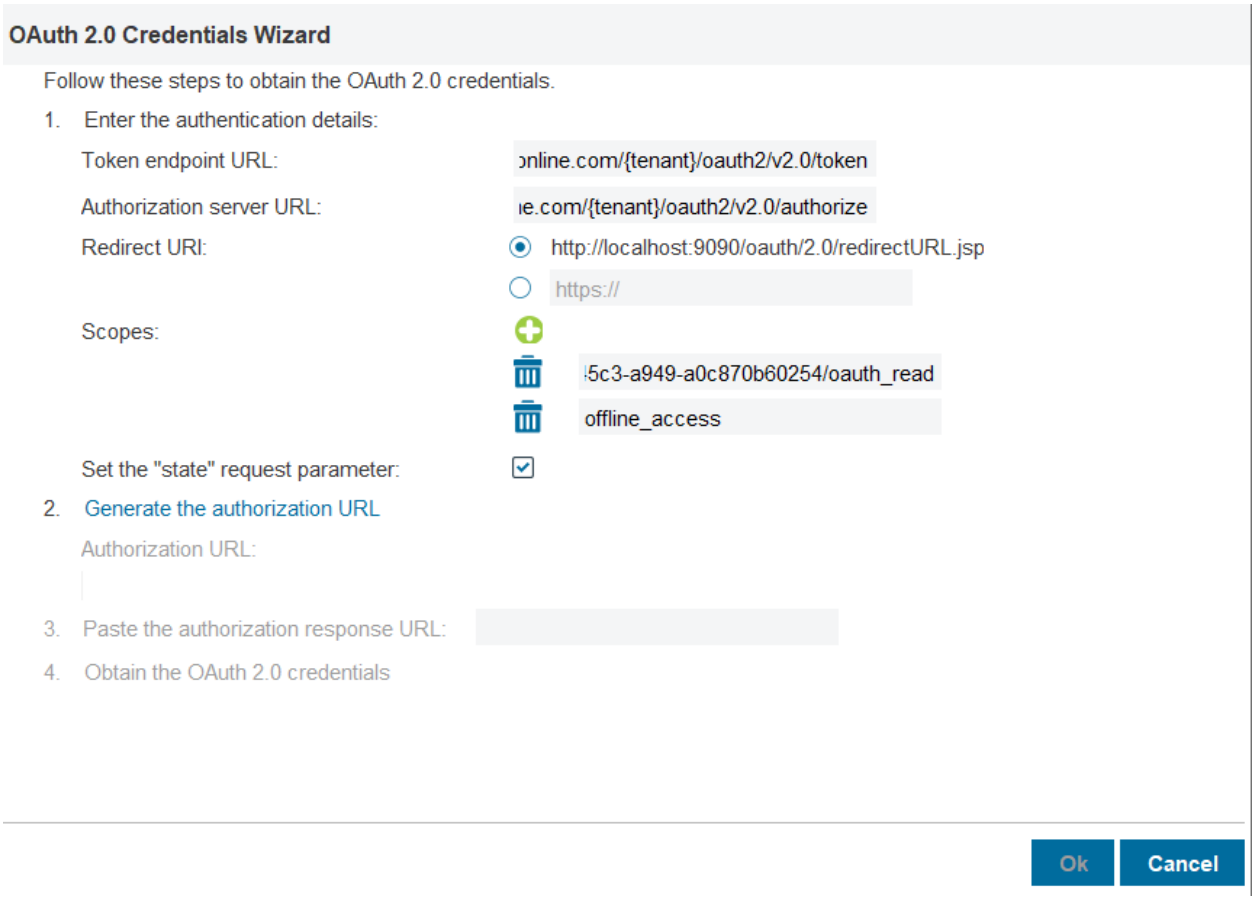
In the "Authentication" page, we will select "OAuth 2.0". Filling in the following fields:

- Authentication grant: Authorization code grant.
- Client identifier: the "Application (client) ID" property from the "Overview" page of the Azure AD application.
- Client secret: the secret value that was saved.
- Authentication method used by the authorization servers: "Include the client credentials in the body of the request".



We will then select "`launch the OAuth 2.0 credentials wizard …`" to complete the OAuth configuration. In the new window that appears, we will enter the following:

- `Token endpoint URL`: The value of the "`OAuth 2.0 token endpoint (v2)`" property in the "`Endpoints`" button on the "`Overview`" page.
- `Authorization endpoint URL`: The value of the "`OAuth 2.0 authorization endpoint (v2)`" property in the same page as above.
- `Redirect URI`: the default will usually work. If this does match the endpoint you listed in Azure AD, you can change it to match here.
- `Scopes`: we will add two:
  - The first should be the scope created in the "`Expose an API`" page. It should look like "`api://{client_id}/{scope_name}`".
  - The second scope is "`offline_access`", which will allow a refresh token to be retrieved. This token is necessary to continue querying the endpoint without having to generate the access token again.



Next, select "`Generate the authorization URL`". This will open the browser on the Identity Provider's login page, unless the browser was already authenticated with the Identity Provider.
After authenticating, an authorization response URL will be shown. Copy this URL back into the `OAuth 2.0 Credentials Wizard` and click on "`Obtain the OAuth 2.0 credentials`". Clicking on the button will automatically fill the "`Access token`" and "`Refresh token`" fields in the HTTP connection configuration, so it will only be necessary to click "`Ok`" at this point.

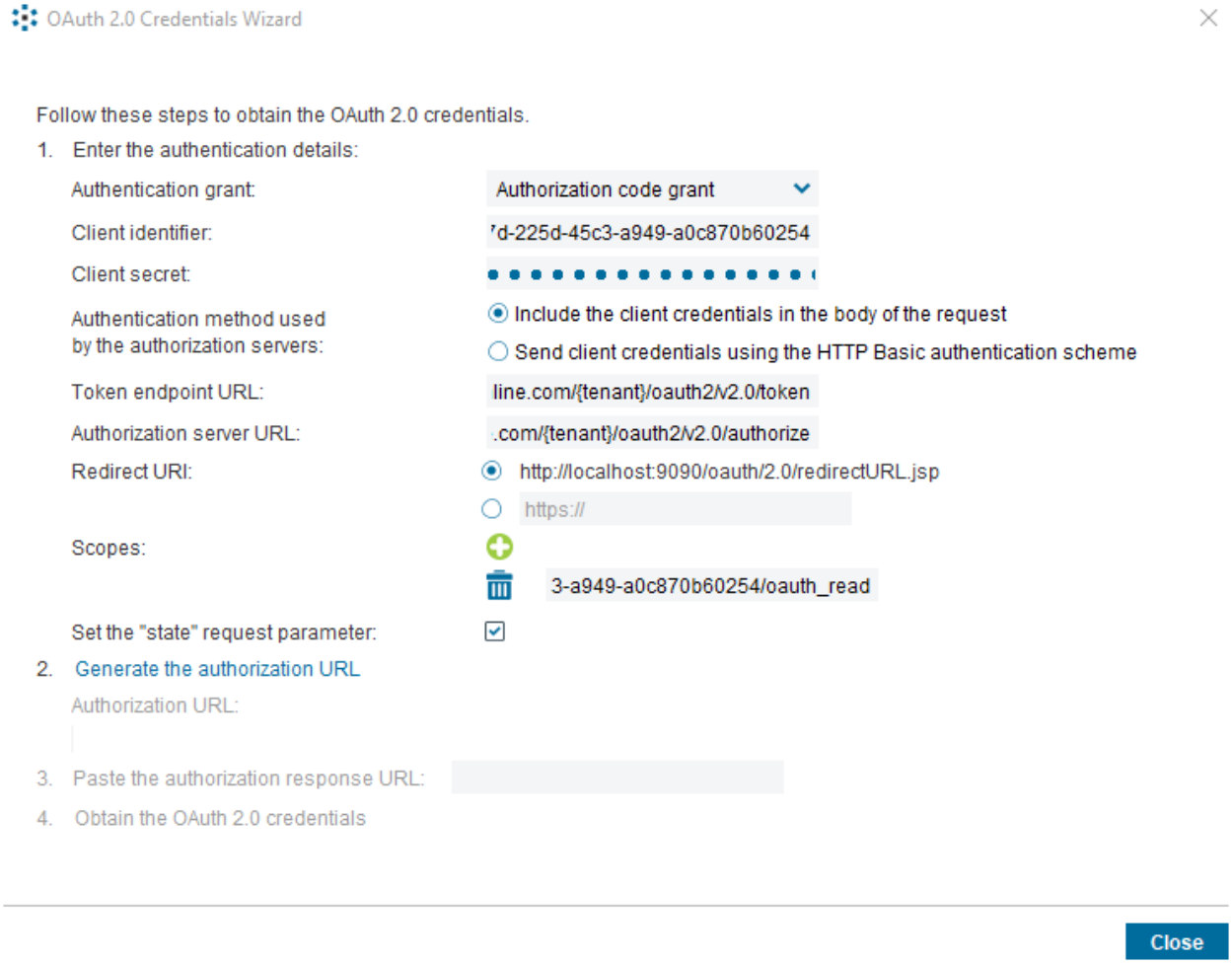
At this point, the data source will be created, and a base view can be created on top of it to retrieve data from the OAuth secured endpoint.

6.2 USING A CLIENT THAT CAN MAKE HTTP REQUESTS

To generate the access token, the "`Tools > OAuth credentials wizards > OAuth 2.0 wizard`" can be used. We will input the following into the wizard:

- `Authentication grant`: "`Authorization code grant`".
- `Client identifier`: the "`Application (client) ID`" property from the "`Overview`" page of the Azure AD application.
- `Client secret`: the secret value that was saved.
- `Authentication method used by the authorization servers`: "`Include the client credentials in the body of the request`".

- `Token endpoint URL`: The value of the "`OAuth 2.0 token endpoint (v2)`" property in the "`Endpoints`" button on the "`Overview`" page.
- `Authorization endpoint URL`: The value of the "`OAuth 2.0 authorization endpoint (v2)`" property in the same page as above.
- `Redirect URI`: the default will usually work. If this does not match the endpoint you listed in Azure AD, you can change it to match here.
- Scopes: this should be the full scope that was created in the "`Expose an API`" page. It should look like "`api://{client_id}/{scope_name}`".



After the above fields have been entered, the "`Generate the authorization URL`" button should be selected. This will redirect you to Azure AD, which will require you to login if you have not already authenticated in your browser.

When you have authenticated against Azure AD, you will be redirected to a page that provides an authorization response URL. The URL on this page should be copied and pasted back into the OAuth 2.0 Credentials Wizard. If this was done correctly, clicking the "`Obtain the OAuth 2.0 credentials`" button should generate an access token in the clipboard.

To use the access token in an external client, the value in the "access_token" attribute should be copied into the Authorization header of the request. In curl, this will look like the following:

```
curl -H "Authorization: Bearer <access_token>"
http<s>://<server_hostname>:<server_port>/server/<database>/<webservive>/vie
ws/<view_name>
```

Executing this, we can see that we are able to retrieve results from the Denodo endpoint when we provide a valid access token:

```
PS C:\Users> curl -v -H @{
>> "Authorization" = "Bearer [redacted]
"
>> "Accept" = "application/json"
>> } -Uri http://localhost:9090/server/admin/bv_calcs/views/bv_calcs | Select-Object -Expand Content
VERBOSE: GET http://localhost:9090/server/admin/bv_calcs/views/bv_calcs with 0-byte payload
VERBOSE: received -1-byte response of content type application/json; subtype=denodo-8.0;charset=UTF-8
{"name":"bv_calcs","elements":[{"key":"key00","num0":12.3,"num1":8.42,"num2":17.86,"num3":-11.52,"num4
e","int0":1,"int1":-3,"int2":5,"int3":8,"bool0":true,"bool1":true,"bool2":false,"bool3":true,"date0":"
20","time0":"1899-12-30T21:07:32","time1":"19:36:22","datetime0":"2004-07-09T10:17:35","datetime1":nul
```

Note that the syntax may be slightly different from the syntax of curl in other machines; this is because the curl command in the example was executed in Windows PowerShell.

7 TROUBLESHOOTING

In order to generate additional information about the HTTP requests being sent between the server and the Identity Provider, the following commands can be executed in the VQL Shell:

```
CALL LOGCONTROLLER('httpclient.wire', 'TRACE');
CALL LOGCONTROLLER('com.denodo.parser.connection.http', 'TRACE');
```

This logging will appear in the "<DENODO_HOME>/logs/vdp/vdp.log" file of the Virtual DataPort server. Additional HTTP related logging provides more information about the responses to Virtual DataPort server requests from the Identity Provider, and the "access_token" attribute can also be extracted from these requests. Parsing the "access_token" in a JWT parser like jwt.io allows for the verification that the token contains the expected parameters.

Note that raising the above logging levels will also output other HTTP related communication occurring in the Virtual DataPort server (it is not limited to OAuth related requests). In addition, note that **JWTs are credentials and can contain personal information**, so care should be taken when copying them into a web resource.

The other logger that can be useful when debugging the behavior of the Virtual DataPort server is the "com.denodo.vdb.security" logger, which can be set to "TRACE" level with the following command:

```
CALL LOGCONTROLLER('com.denodo.vdb.security','TRACE');
```

This logger records information about how the Virtual DataPort server processes the information it receives from the OAuth authentication process, and can be used to gather more information about unexpected behavior in the server itself. This usually becomes useful when all the requests from the HTTP loggers are showing up as successful, but OAuth authentication is still not working.

Finally, we recommend double checking that a role has been created corresponding to the name of your scope; otherwise, users authenticating with OAuth will not be assigned any permissions in the server.