Snowflake OAUTH Builtin and External OKTA and AzureAD

- Built-In Snowflake OAuth
- Snowflake Side Configuration
  - Step 1: Snowflake New Setup
    - Create Security Integration
    - Collect newly created Security Integration complete details
    - Collect newly created Security Integration Client ID and Secret
    - Delete the above Config (Optional)
  - Step 2: Alation Configuration for Builtin OAUTH
  - Step 3: After Authentication Test in Builtin Oauth
- OKTA External:
  - Reference https://docs.snowflake.com/en/user-guide/oauth-okta
    - Stage1.1 OKTA Auth Client:
      - Create application OIDC Sign in method
      - Choose Web application Type
      - Set Auth code and refresh token Grant type
      - Set redirect URI to Alation Callback
      - Set For internal testing -allow everyone
      - Change issuer from Dynamic to Static in Signon tab - openid
      - Click save
      - Full config sample
    - Stage1.2 OKTA Auth Server:
      - Step1: Create Auth Server in OKTA
      - In Input for Audience give our Snowflake Server Address and Save
      - Step2:Change Issuer from Dynamic to Static and save
      - Step3: On scopes tab> Add scope > SESSION:ROLE-ANY
      - Step4: Add policy, can create a policy to allow all users' use of this scope.
        - Map the Client Application created for snowflake under this policy
        - Now Add rule and create rule
        - Sample :
      - Collect Info
        - Issuer Okta URL https://dev-66731772.okta.com/oauth2/aus8jqjjrwUVSKfV85d7
        - Click on Metadata URI
        - link https://dev-66731772.okta.com/oauth2/aus8jqjjrwUVSKfV85d7/.well-known/oauth-authorization-server
          - Used in Snowflake
          - Used in Alation
  - Stage2:In Snowflake: set up External OAuth with Okta;
  - Stage3:In Alation: enable and configure the OAuth for your Snowflake data source.
  - TESTING OKTA :
    - Logs at OKTA Server Side for Success-full Login
- Azure AD_Snowflake_Alation:
  - Prerequisite Step: Determine the OAuth Flow in Azure AD
  - Stage1.1 Create App registration for AUTH SERVER
    - Set Application ID URI (We will use https://alation.com/ as Application ID URI)
    - Now click Add scope
      - Provide:
  - Stage1.2 Create App registration for AUTH Client
  - CREATE AUTH CLIENT
    - Generate Client Secret and Copy
    - Add API Permissions to Auth Server created
      - Add permission
      - Switch to My API s on popup
      - Double click Auth Server Created
      - Press Delegated Permissions ,select and add Permission
  - Collect Azure AD Information for Snowflake
    - Token endpoint v2https://login.microsoftonline.com/f8e32097-5c65-485f-9eb7-f40149462461/oauth2/v2.0/token
    - jwks_uri Locate Locate the "jwks_uri" here
    - Next Capture Entity ID:
  - Stage2:Snowflake Create a Security Integration Azure type
    - Just for understand what to put in Alation for User Field Name (Like EMail or Username or UPN)
    - Sample Users Mapping
      - In Azure AD:
      - In Snowflake:We use Login Name
  - Stage3 Configure Alation
  - Test Azure Ad Flow
    - Error 1: No reply Address
      - Solution: After Adding Client Redirect URI on Auth Client this got fixed
    - Error2 Callback/ vs callback
    - Tracing Success Routing Azure:
      - Step1 On Opening connection -on compose Redirect OAUTH : As soon we open Compose with Respective DataSource
      - Step2:On Add SSO User
      - Step 2.1 SSO page1
      - Here we can see the username

Built-In Snowflake OAuth

References:

- CREATE SECURITY INTEGRATION (Snowflake OAuth) | Snowflake Documentation
- Configure Snowflake OAuth for Custom Clients | Snowflake Documentation
- Alation Help Center: Snowflake SSO

Overview of the steps to perform:

The configuration workflow has 2 Stages:

Stage 1: Configure Snowflake with Alation Instance details

- Stage 1 Input: Run the **security integration** register query with the Alation endpoint details
- Stage 1 Output: Client ID , Client Secret , Authorization endpoint , Token Endpoint

Stage 2: Configure Alation Instance with Snowflake details

- Stage 2 Input: Use Stage 1 output in Alation's data source Compose connection setting

Snowflake Side Configuration

Step 1: Snowflake New Setup

Create Security Integration

You can run a query similar to the one shown below. Following adjustments are required:

- Change name from `SURAJ_SNOWFLAKE_BUILTIN_OAUTH` to yours. Avoid special chararacters.
- Change `OAUTH_REDIRECT_URI` to your own `https://ALATION_INSTANCE_FQDN/api/datasource_auth/oauth/callback`

```
CREATE SECURITY INTEGRATION
  "SURAJ_SNOWFLAKE_BUILTIN_OAUTH"
TYPE = OAUTH
ENABLED = TRUE
OAUTH_CLIENT = CUSTOM
OAUTH_CLIENT_TYPE = 'CONFIDENTIAL'
OAUTH_REDIRECT_URI = 'https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/callback'
OAUTH_REFRESH_TOKEN_VALIDITY = 7776000
OAUTH_ISSUE_REFRESH_TOKENS = TRUE ;
```

```
CREATE SECURITY INTEGRATION
  "SURAJ_SNOWFLAKE_BUILTIN_OAUTH"
  TYPE = OAUTH
  ENABLED = TRUE
  OAUTH_CLIENT = CUSTOM
  OAUTH_CLIENT_TYPE = 'CONFIDENTIAL'
  OAUTH_REDIRECT_URI = 'https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/callback'
  OAUTH_REFRESH_TOKEN_VALIDITY = 7776000
  OAUTH_ISSUE_REFRESH_TOKENS = TRUE ;


CREATE SECURITY INTEGRATION
```

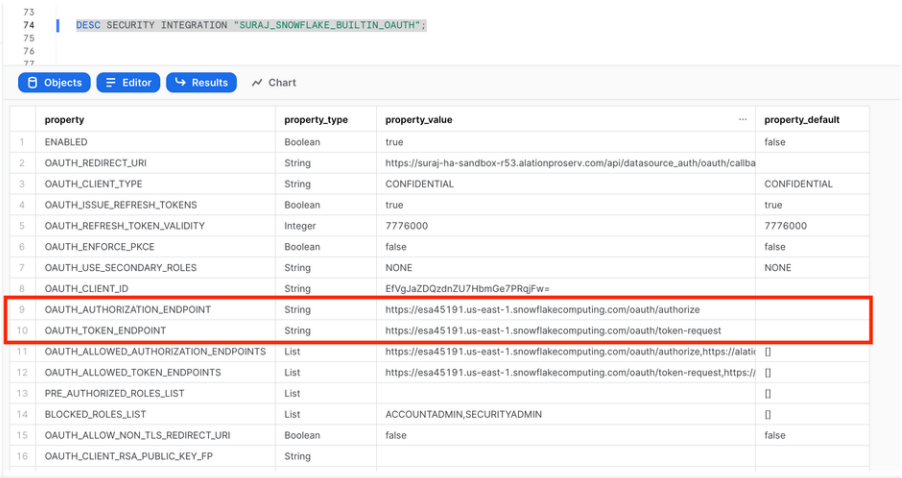
Objects | Editor | Results | Chart

	status
1	Integration SURAJ_SNOWFLAKE_BUILTIN_OAUTH successfully created.

Collect newly created Security Integration complete details

Execute the following query to get the details about **security integration** we created earlier on:

```
DESC SECURITY INTEGRATION "SURAJ_SNOWFLAKE_BUILTIN_OAUTH";
```



```
DESC SECURITY INTEGRATION "SURAJ_SNOWFLAKE_BUILTIN_OAUTH";
```

Objects | Editor | Results | Chart

	property	property_type	property_value	property_default
1	ENABLED	Boolean	true	false
2	OAUTH_REDIRECT_URI	String	https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/callba	
3	OAUTH_CLIENT_TYPE	String	CONFIDENTIAL	CONFIDENTIAL
4	OAUTH_ISSUE_REFRESH_TOKENS	Boolean	true	true
5	OAUTH_REFRESH_TOKEN_VALIDITY	Integer	7776000	7776000
6	OAUTH_ENFORCE_PKCE	Boolean	false	false
7	OAUTH_USE_SECONDARY_ROLES	String	NONE	NONE
8	OAUTH_CLIENT_ID	String	EfVgJaZDQzdnZU7HbmGe7PRqjFw=	
9	OAUTH_AUTHORIZATION_ENDPOINT	String	https://esa45191.us-east-1.snowflakecomputing.com/oauth/authorize	
10	OAUTH_TOKEN_ENDPOINT	String	https://esa45191.us-east-1.snowflakecomputing.com/oauth/token-request	
11	OAUTH_ALLOWED_AUTHORIZATION_ENDPOINTS	List	https://esa45191.us-east-1.snowflakecomputing.com/oauth/authorize,https://alati	[]
12	OAUTH_ALLOWED_TOKEN_ENDPOINTS	List	https://esa45191.us-east-1.snowflakecomputing.com/oauth/token-request,https://	[]
13	PRE_AUTHORIZED_ROLES_LIST	List		[]
14	BLOCKED_ROLES_LIST	List	ACCOUNTADMIN,SECURITYADMIN	[]
15	OAUTH_ALLOW_NON_TLS_REDIRECT_URI	Boolean	false	false
16	OAUTH_CLIENT_RSA_PUBLIC_KEY_FP	String		

In the result set you will find entries for:

- **Authorization Endpoint** as `OAUTH_AUTHORIZATION_ENDPOINT`
- **Token Endpoint** as `OAUTH_TOKEN_ENDPOINT`

Collect newly created Security Integration Client ID and Secret

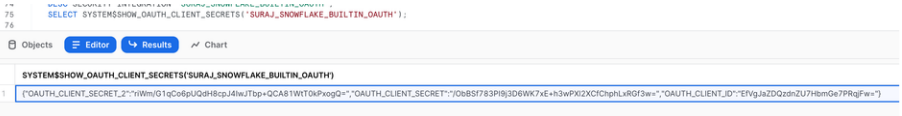
Retrieve the Client ID and Secret by running this query (replace the name of the security integration):

```
SELECT SYSTEM$SHOW_OAUTH_CLIENT_SECRETS('<SECURITY_INTEGRATION_NAME>');
-- in example:
SELECT SYSTEM$SHOW_OAUTH_CLIENT_SECRETS('SURAJ_SNOWFLAKE_BUILTIN_OAUTH');
```

Here `OAUTH_CLIENT_SECRET_2` can be ignored:

Security Integration Values (example output):

```
{"OAUTH_CLIENT_SECRET_2":"riWm/G1qCo6pUQdH8cpJ4IwJTbp+QCA81WtT0kPxogQ=",
"OAUTH_CLIENT_SECRET":"/ObBSf783PI9j3D6WK7xE+h3wPXl2XCfChphLxRGf3w=",
"OAUTH_CLIENT_ID":"EfVgJaZDQzdnZU7HbmGe7PRqjFw="}
```



```
SELECT SYSTEM$SHOW_OAUTH_CLIENT_SECRETS('SURAJ_SNOWFLAKE_BUILTIN_OAUTH');
```

Objects | Editor | Results | Chart

	SYSTEM$SHOW_OAUTH_CLIENT_SECRETS('SURAJ_SNOWFLAKE_BUILTIN_OAUTH')
1	{"OAUTH_CLIENT_SECRET_2":"riWm/G1qCo6pUQdH8cpJ4IwJTbp+QCA81WtT0kPxogQ=","OAUTH_CLIENT_SECRET":"/ObBSf783PI9j3D6WK7xE+h3wPXl2XCfChphLxRGf3w=","OAUTH_CLIENT_ID":"EfVgJaZDQzdnZU7HbmGe7PRqjFw="}

Delete the above Config (Optional)

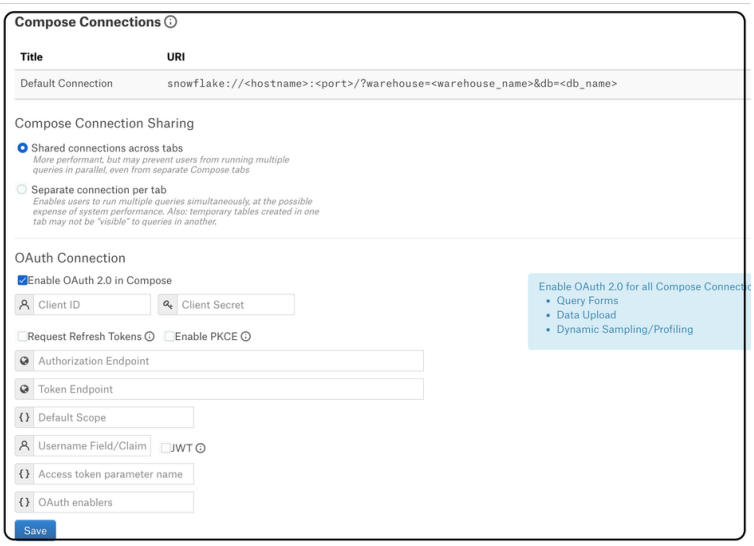
> ⚠ Only do this if you were testing the setup and want to clear down everything!

In case you want to drop the configuration (because you are just testing this setup), you can use this:

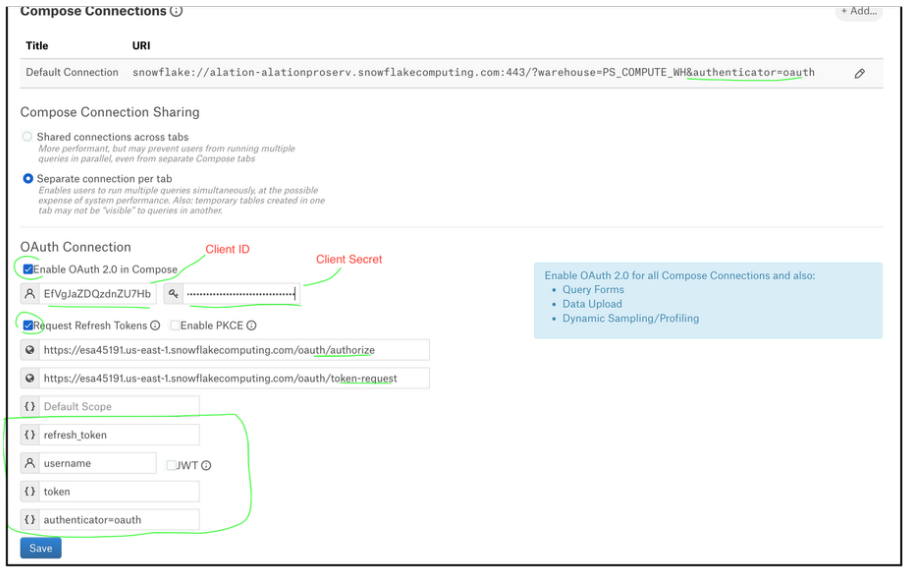
```
DROP SECURITY INTEGRATION IF EXISTS "<SECURITY_INTEGRATION_NAME>" CASCADE;
```

Step 2: Alation Configuration for Builtin OAUTH

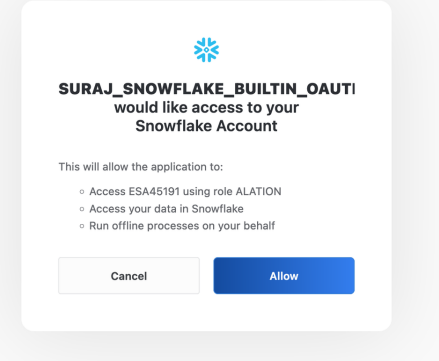
Under **Data Source > Settings > Compose**. Toggle **Enable in Compose** (if it is not already switched on) scroll down and edit it as follows:

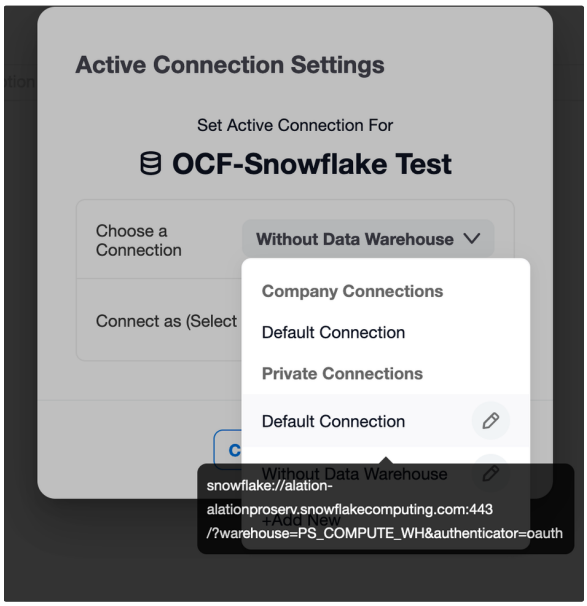


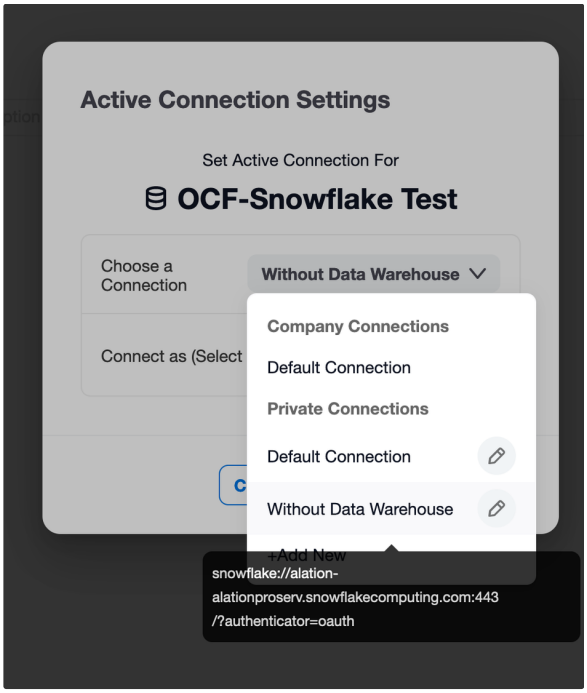
1. Add `authenticator=oauth` to the connection string.
2. Tick **Enable OAuth 2.0 in Compose** checkbox
3. Provide **Client ID** and **Client Secret**.
4. Tick **Request Refresh Tokens** checkbox.
5. Provide **Authorization Endpoint**.
6. Provide **Token endpoint**.
7. **Default Scope**: Leave Blank
8. **Refresh Scope**: Insert `refresh_token`
9. **Username Field**: Insert `username` . (here we specify the username since the same is configured at Snowflake side for user identification)
10. **Access token parameter**: Insert `token`
11. **OAuth Enablers**: Insert `authenticator=oauth`



Step 3: After Authentication Test in Builtin Oauth




Active Connection Settings
Set Active Connection For
OCF-Snowflake Test
Choose a Connection
Without Data Warehouse
Connect as (Select
Company Connections
Default Connection
Private Connections
Default Connection
Without Data Warehouse
+Add New
snowflake://alation-alationproserv.snowflakecomputing.com:443/?warehouse=PS_COMPUTE_WH&authenticator=oauth


Active Connection Settings
Set Active Connection For
OCF-Snowflake Test
Choose a Connection
Without Data Warehouse
Connect as (Select
Company Connections
Default Connection
Private Connections
Default Connection
Without Data Warehouse
+Add New
snowflake://alation-alationproserv.snowflakecomputing.com:443/?authenticator=oauth

OKTA External:

Reference https://docs.snowflake.com/en/user-guide/oauth-okta

Configuration Work Flow in 3 Stages:

Stage1.1 Configure OKTA Client with Alation Instance callback details

Stage1.1 Input- Alation callback url

Stage1.1 Output- client id and secret

Stage1.2 Configure OKTA Auth Server with Snowflake required details

Stage1 .2 Input- Give Snowflake URL as Audience and Snowflake role in scope

Stage1 .2 Output-Endpoints

Stage2: Configure Snowflake with OKTA Auth Server details

Stage2 Input- run integration register query with OKTA endpoint details

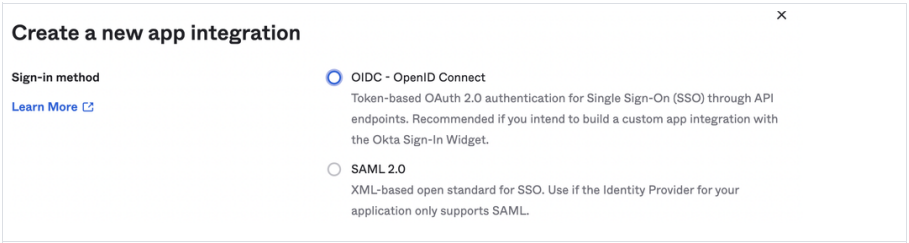
Stage3: Configure Alation Instance with OKTA Client ID and OKTA Auth server endpoints Details details

Stage3Input: Use Stage1 input in Alation's data source compose connection setting

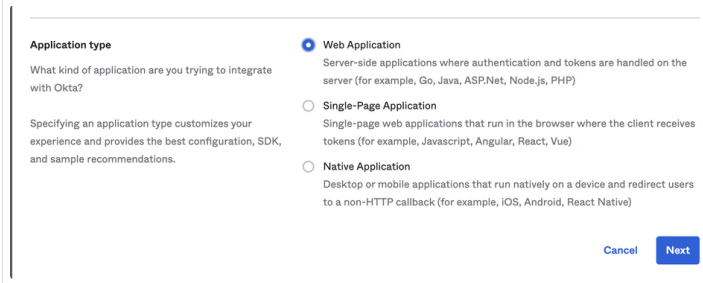
- In Snowflake: if this has not been done yet, set up External **OAuth** with Okta;
- In Alation: enable and configure the **OAuth** for your Snowflake data source.

Stage1.1 OKTA Auth Client:

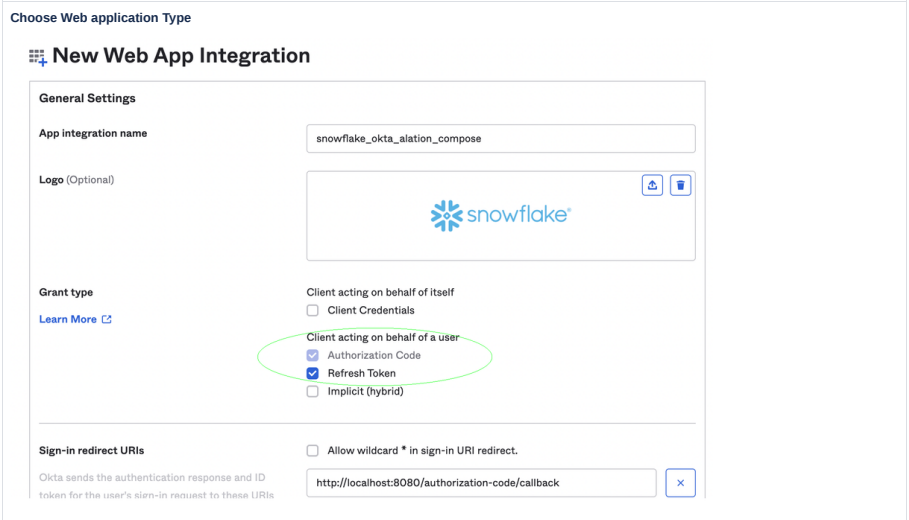
1. When creating the client application in Okta, use Web as the Application type.
2. In the client application settings:
   Under General > General Settings > Application > Allowed grant types, ensure that the Authorization Code and Refresh Token checkboxes are selected;
   (Optional) Under General > General Settings > User Consent > User consent, select Require consent;
   Under General > General Settings > LOGIN > Login redirect URIs, add https://<alation.company.com>/api/datasource_auth/oauth/callback/
   Ensure the URI ends with "/"
   HTTP can be specified without having to change any TLS settings



Create application OIDC Sign in method



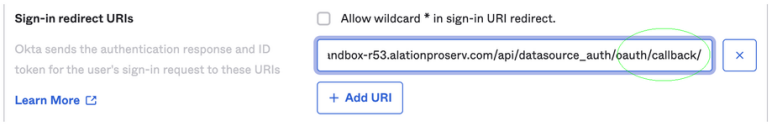
Choose Web application Type



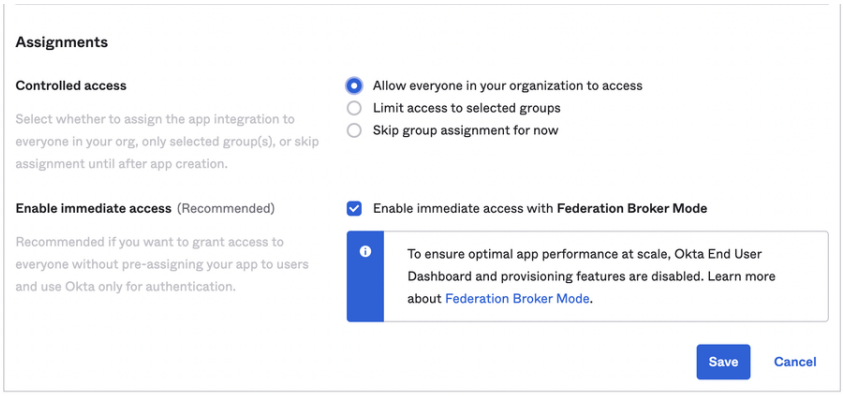
Set Auth code and refresh token Grant type

Set redirect URI to Alation Callback

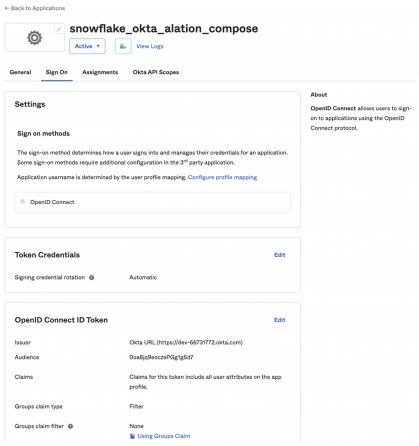
https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/callback/



Set For internal testing -allow everyone



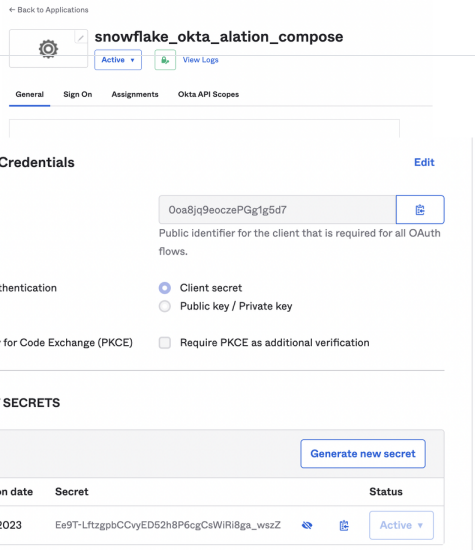
Change issuer from Dynamic to Static in Signon tab - openid



Click save

Full config sample

Collect Client Info



- **Client ID** 0oa8jq9eoczePGg1g5d7
- **Client Secret** Ee9T-LftzgpbCCvyED52h8P6cgCsWiRi8ga_wszZ

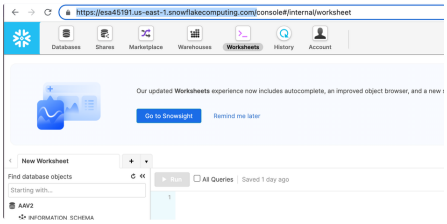
Stage1.2 OKTA Auth Server:

In the Okta Authorization Server for Snowflake settings:

1. Ensure that the audience URL does not have a trailing slash,

   working example: https://alation_test.us-east-1.snowflakecomputing.com
2. Consider adding SESSION:ROLE-ANY as a Scope to avoid adding a Scope entry for each role. You can create a policy to allow all users' use of this scope. This will allow users to connect to Snowflake as any role to which they have access in Snowflake and will also enable role switching in Compose with the USE ROLE statement. Otherwise, create Scopes linked to specific Snowflake roles and add policies limiting use of these roles to specific users.

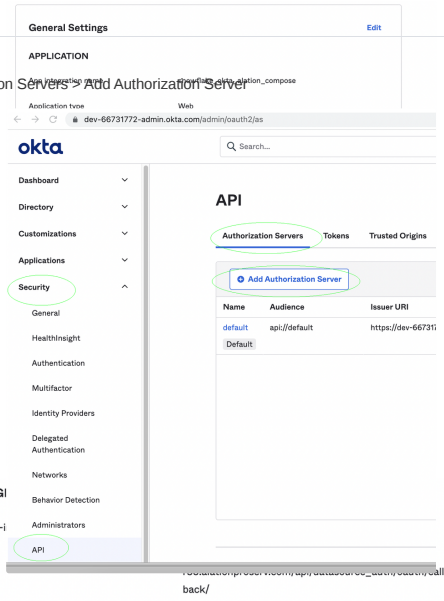
input:**Our Snowflake Server**:https://esa45191.us-east-1.snowflakecomputing.com



Snowflake Server
General Tab

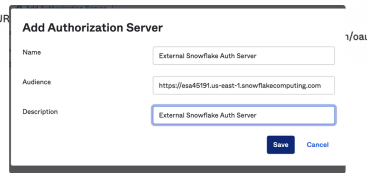
Step1: Create Auth Server in OKTA

Under Security > API> Authorization Servers > Add Authorization Server



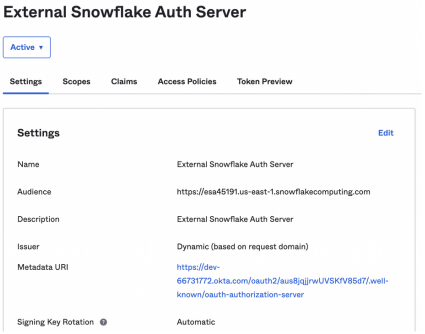
In Input for Audience give our Snowflake Server Address and Save

Our Snowflake Server:https://esa45191.us-east-1.snowflakecomputing.com

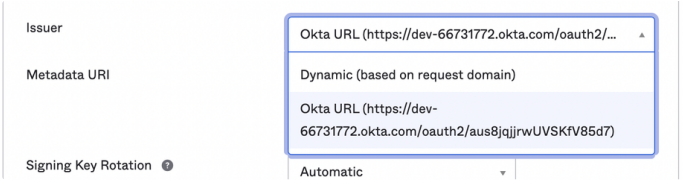


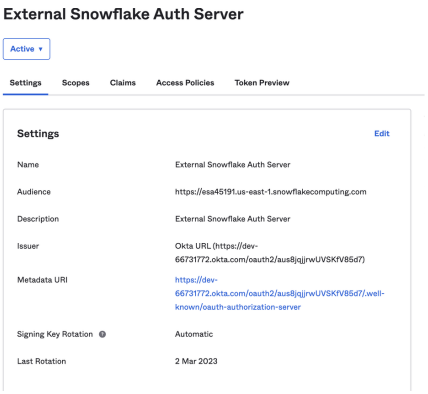
Step2:Change Issuer from Dynamic to Static and save

From Dynamic issuer

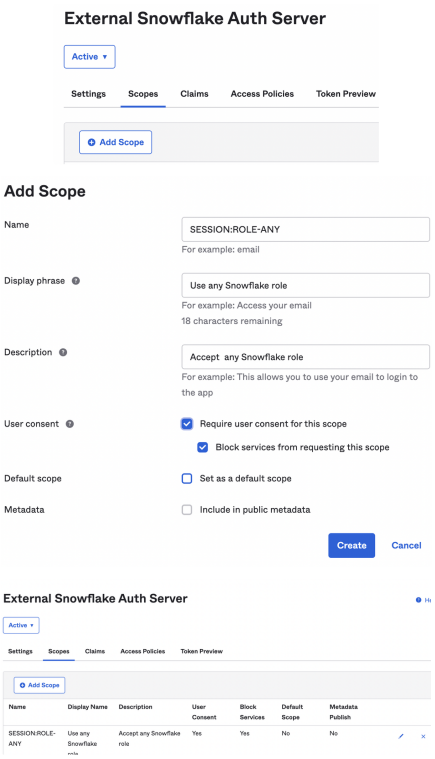


To Static OKTA Issuer URL



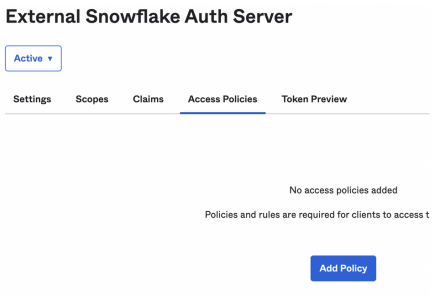


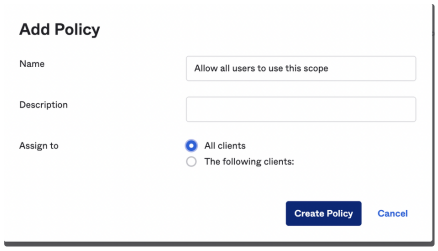
Step3: On scopes tab> Add scope > SESSION:ROLE-ANY



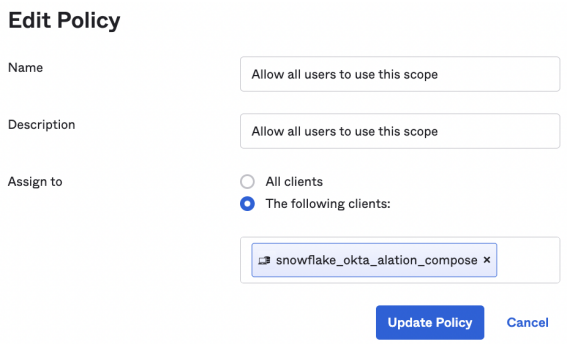
Step4: Add policy, can create a policy to allow all users' use of this scope.

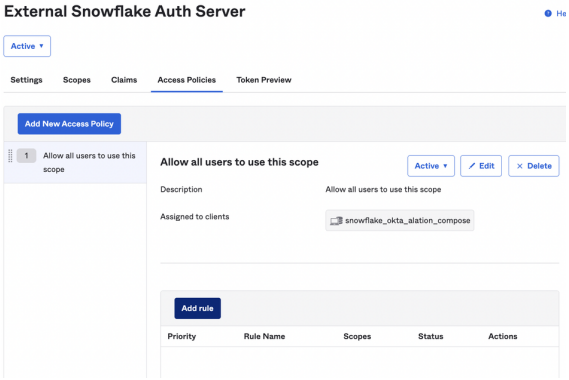
Add policy



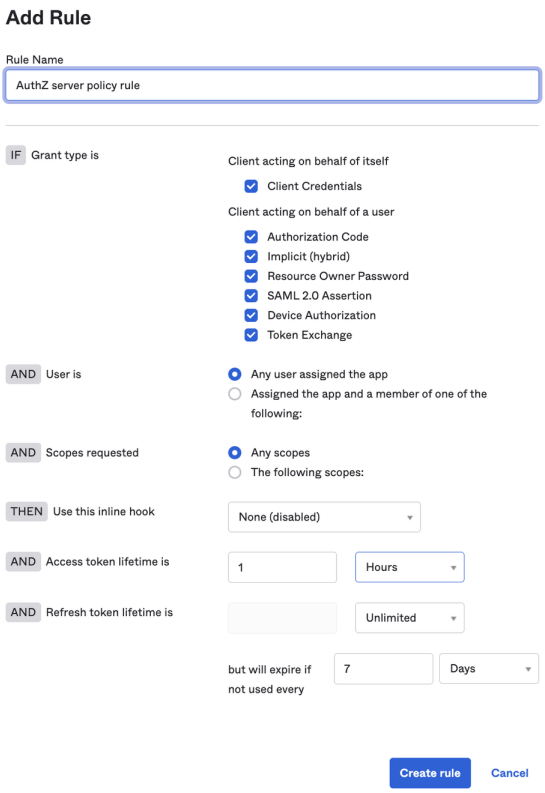


Map the Client Application created for snowflake under this policy

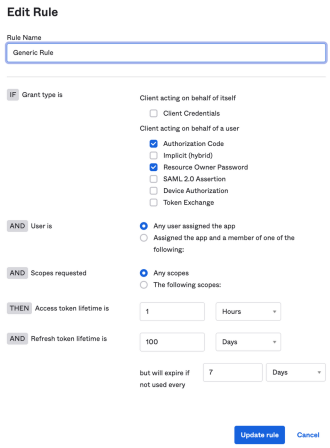




Now Add rule and create rule

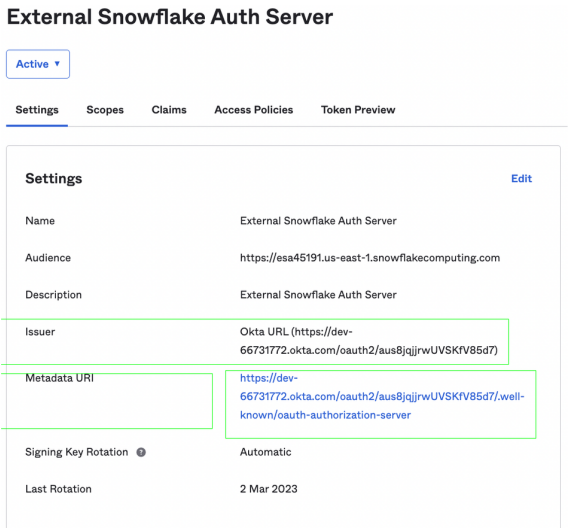


Sample :



Collect Info

Issuer Okta URL https://dev-66731772.okta.com/oauth2/aus8jqjjrwUVSKfV85d7



Click on Metadata URI

link https://dev-66731772.okta.com/oauth2/aus8jqjjrwUVSKfV85d7/.well-known/oauth-authorization-server

In the **Metadata** document:

1. Copy the **Metadata URI** value, open a browser tab, and paste the URL in the address bar.
2. You should see JSON text in the browser. You can work with this text in a text editor or in the browser itself.
3. Locate the `"jwks_uri"` parameter and copy its value. Its format should resemble `https://dev-111111.oktapreview.com/oauth2/auslh9j9vf9ej7NfT0h7/v1/keys`. This endpoint will be known as the `<OKTA_JWS_KEY_ENDPOINT>` in the following steps.
4. Locate the `"token_endpoint"` parameter and copy its value. Its format should resemble `https://dev-111111.oktapreview.com/oauth2/auslh9j9vf9ej7NfT0h7/v1/token`. This endpoint will be known as the `<OKTA_OAUTH_TOKEN_ENDPOINT>` in the following steps.

Used in Snowflake

"**issuer**":"https://dev-66731772.okta.com/oauth2/aus8jqjjrwUVSKfV85d7",

"**jwks_uri**":"https://dev-66731772.okta.com/oauth2/aus8jqjjrwUVSKfV85d7/v1/keys",

Audience from Auth Server https://esa45191.us-east-1.snowflakecomputing.com

Used in Alation

"**authorization_endpoint**":"https://dev-66731772.okta.com/oauth2/aus8jqjjrwUVSKfV85d7/v1/authorize",

"**token_endpoint**":"https://dev-66731772.okta.com/oauth2/aus8jqjjrwUVSKfV85d7/v1/token"

Stage2:In Snowflake: set up External OAuth with Okta;

In the security integration created for this Okta authorization server in Snowflake,

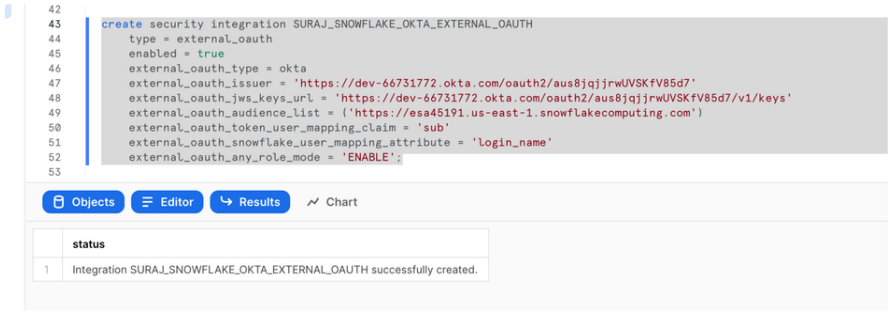
Run below query with inputs from OKTA

create security integration NAME_OF_CLIENT_REGISTERING
type = external_oauth
enabled = true
external_oauth_type = okta
external_oauth_issuer = 'ISSUER FROM OKTA AUTH SERVER'

external_oauth_jws_keys_url = 'KEYS FROM OKTA AUTH SERVER'
external_oauth_audience_list = ('AUDIENCE LIST FROM OKTA AUTH SERVER')
external_oauth_token_user_mapping_claim = 'sub'
external_oauth_snowflake_user_mapping_attribute = 'login_name'
external_oauth_any_role_mode = 'ENABLE';

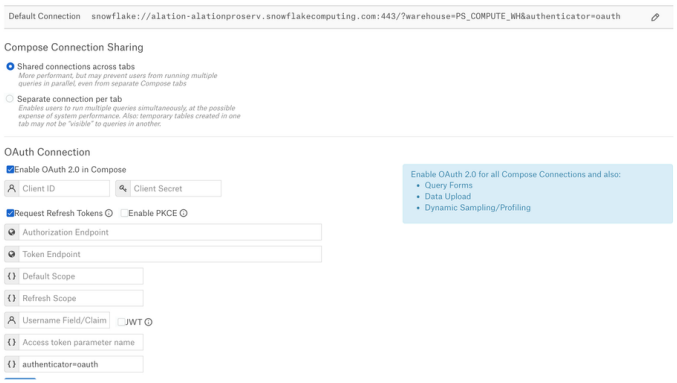
Note: to enable role switching, set the EXTERNAL_OAUTH_ANY_ROLE_MODE field to ENABLE.

```
create security integration SURAJ_SNOWFLAKE_OKTA_EXTERNAL_OAUTH
    type = external_oauth
    enabled = true
    external_oauth_type = okta
    external_oauth_issuer = 'https://dev-66731772.okta.com/oauth2/aus8jqjjrwUVSKfV85d7'
    external_oauth_jws_keys_url = 'https://dev-66731772.okta.com/oauth2/aus8jqjjrwUVSKfV85d7/v1/keys'
    external_oauth_audience_list = ('https://esa45191.us-east-1.snowflakecomputing.com')
    external_oauth_token_user_mapping_claim = 'sub'
    external_oauth_snowflake_user_mapping_attribute = 'login_name'
    external_oauth_any_role_mode = 'ENABLE';
```



Stage3:In Alation: enable and configure the OAuth for your Snowflake data source.

Empty Config At Alation Side



Data at Snowflake Side:

```
DESC SECURITY INTEGRATION "SURAJ_SNOWFLAKE_OKTA_EXTERNAL_OAUTH";
```

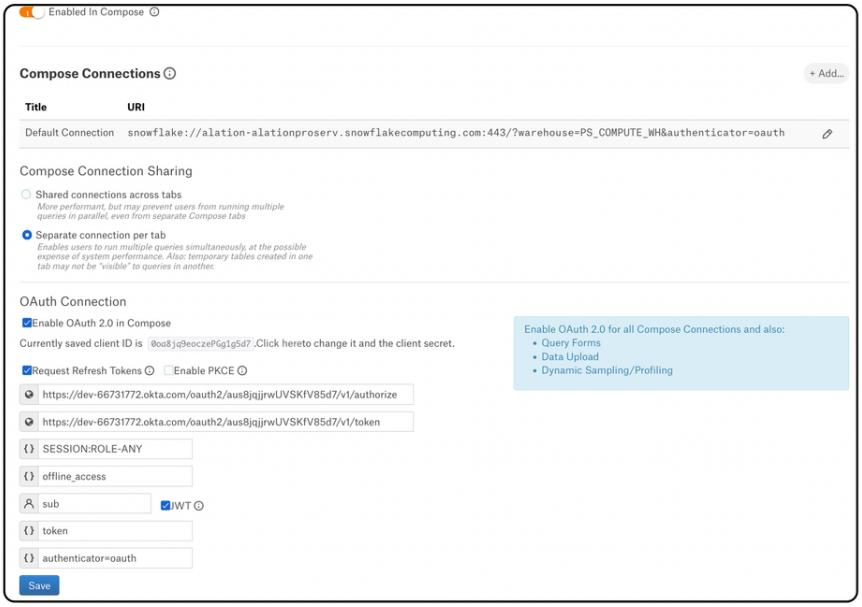
	property	property_type	property_value
1	ENABLED	Boolean	true
2	EXTERNAL_OAUTH_ISSUER	String	https://dev-66731772.okta.com/oauth2/aus8jqjjrwUVSKfV85d7
3	EXTERNAL_OAUTH_JWS_KEYS_URL	Object	https://dev-66731772.okta.com/oauth2/aus8jqjjrwUVSKfV85d7/v1/keys
4	EXTERNAL_OAUTH_ANY_ROLE_MODE	String	ENABLE
5	EXTERNAL_OAUTH_RSA_PUBLIC_KEY	String	
6	EXTERNAL_OAUTH_RSA_PUBLIC_KEY_2	String	
7	EXTERNAL_OAUTH_BLOCKED_ROLES_LIST	List	ACCOUNTADMIN,SECURITYADMIN
8	EXTERNAL_OAUTH_ALLOWED_ROLES_LIST	List	
9	EXTERNAL_OAUTH_AUDIENCE_LIST	List	https://esa45191.us-east-1.snowflakecomputing.com
10	EXTERNAL_OAUTH_TOKEN_USER_MAPPING_CLAIM	Object	['sub']
11	EXTERNAL_OAUTH_SNOWFLAKE_USER_MAPPING_ATTRIBUTE	String	login_name
12	COMMENT	String	

Data from OKTA Client Side

- Client ID, 0oa8jq9eoczePGg1g5d7
- Client Secret, Ee9T-LftzgpbCCvyED52h8P6cgCsWiRi8ga_wszZ

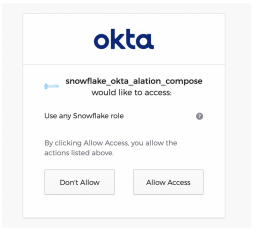
Data from OKTA Server Side

- Authorization endpoint, https://dev-66731772.okta.com/oauth2/aus8jqjjrwUVSKfV85d7/v1/authorize
- Token endpoint, https://dev-66731772.okta.com/oauth2/aus8jqjjrwUVSKfV85d7/v1/token\

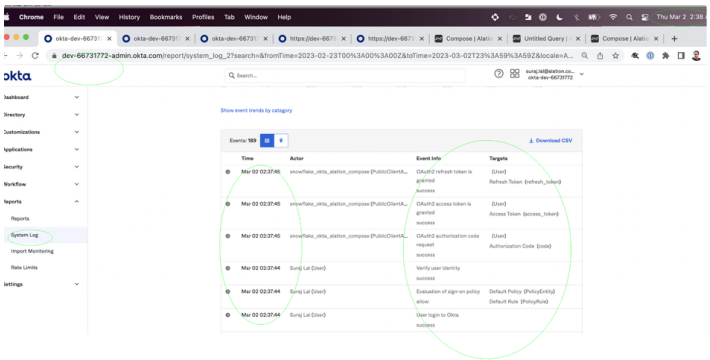


TESTING OKTA :

- After OkTA Redirect from Alation
- User Authenticates and then
- Consent from client application



Logs at OKTA Server Side for Success-full Login



severity	event_type	display_message	uuid	version	timestamp
INFO	user.session.start	User login to Okta	bd138fdb-b8c4-11ed-b66d-41d1699663fc	0	2023-03-02T06:37:44
INFO	user.authentication.verify	Verify user identity	bd1abbce-b8c4-11ed-b66d-41d1699663fc	0	2023-03-02T06:37:44
INFO	app.oauth2.as.token.grant.refresh_token	OAuth2 refresh token is granted	be00afba-b8c4-11ed-8905-81624f8eead9	0	2023-03-02T06:37:45
INFO	policy.evaluate_sign_on	Evaluation of sign-on policy	bd1871dc-b8c4-11ed-b66d-41d1699663fc	0	2023-03-02T06:37:44
INFO	app.oauth2.as.authorize.code	OAuth2 authorization code request	bd95b80b-b8c4-11ed-a2b6-f5f5729c420f	0	2023-03-02T06:37:45
INFO	app.oauth2.as.token.grant.access_token	OAuth2 access token is granted	be0088a9-b8c4-11ed-8905-81624f8eead9	0	2023-03-02T06:37:45

on	timestamp	outcome.result	outcome.reason	actor.id
0	2023-03-02T06:37:44.363Z	SUCCESS		00u7qelljmzf70Y7F5d7
0	2023-03-02T06:37:44.410Z	SUCCESS		00u7qelljmzf70Y7F5d7
0	2023-03-02T06:37:45.917Z	SUCCESS		0oa8jq9eoczePGg1g5d7
0	2023-03-02T06:37:44.395Z	ALLOW	Sign-on policy evaluation resulted in ALLOW	00u7qelljmzf70Y7F5d7
0	2023-03-02T06:37:45.216Z	SUCCESS		0oa8jq9eoczePGg1g5d7
0	2023-03-02T06:37:45.916Z	SUCCESS		0oa8jq9eoczePGg1g5d7

actor.id	actor.type	actor.display_name	actor.alternate_id	authentication_context.authentication_step	au
00u7qelljmzf70Y7F5d7	User	Suraj Lal	suraj.lal@alation.com	0	
00u7qelljmzf70Y7F5d7	User	Suraj Lal	suraj.lal@alation.com	0	
0oa8jq9eoczePGg1g5d7	PublicClientApp	snowflake_okta_alation_compose	0oa8jq9eoczePGg1g5d7	0	
00u7qelljmzf70Y7F5d7	User	Suraj Lal	suraj.lal@alation.com	0	
0oa8jq9eoczePGg1g5d7	PublicClientApp	snowflake_okta_alation_compose	0oa8jq9eoczePGg1g5d7	0	
0oa8jq9eoczePGg1g5d7	PublicClientApp	snowflake_okta_alation_compose	0oa8jq9eoczePGg1g5d7	0	

Azure AD_Snowflake_Alation:

https://docs.snowflake.com/en/user-guide/oauth-ext-overview

https://docs.snowflake.com/en/user-guide/oauth-azure

Prerequisite Step: Determine the OAuth Flow in Azure AD

Azure AD supports two different OAuth flows in which an OAuth Client can get an access token.

1. The authorization server can grant the OAuth client an access token on behalf of the user.(We will User based)
2. The authorization server can grant the OAuth client an access token for the OAuth client itself.

Configuration Work Flow in 3 Stages:

Stage1.1 Create App registration for **AUTH SERVER**

Stage1 .1 Input- Give Snowflake URL as Audience , Snowflake role in scope, and Application UID(can be anything that is there in trusted domain of AzureAD)

Stage1 .2 Output-Endpoints

Stage1.2 Create App registration for **AUTH Client**

Stage1.1 Input- Alation callback url, Genrate Client ID

Stage1.1 Output- client id and secret

Stage2: Configure Snowflake with Azure Auth Server details

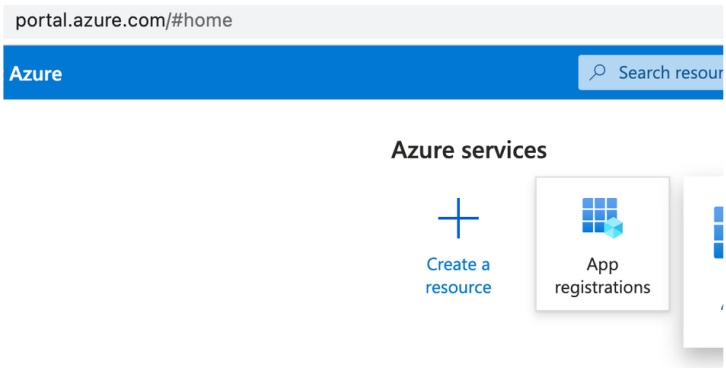
Stage2 Input- run integration register query with Azure Auth Server endpoint details

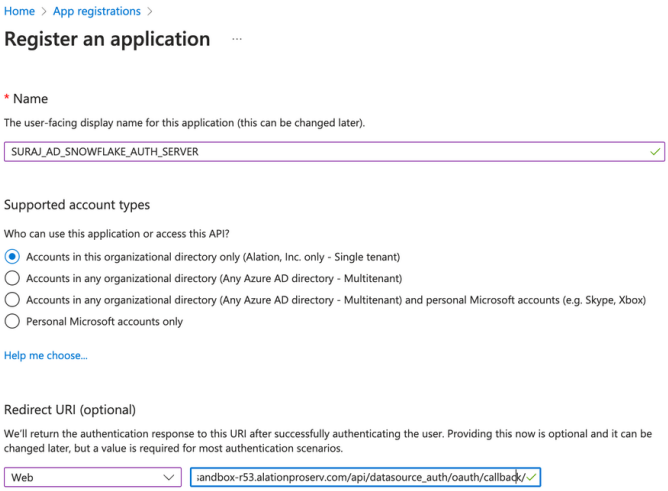
Stage3: Configure Alation Instance with Azure Client ID and Azure Auth server endpoints Details details

Stage3Input: Use Stage1 input in Alation's data source compose connection setting

- Complete either sub-step 10 or 11 in Step 1: Configure the OAuth Resource in Azure AD.
- Complete either sub-step 13 or 14 in Step 2: Create an OAuth Client in Azure AD.

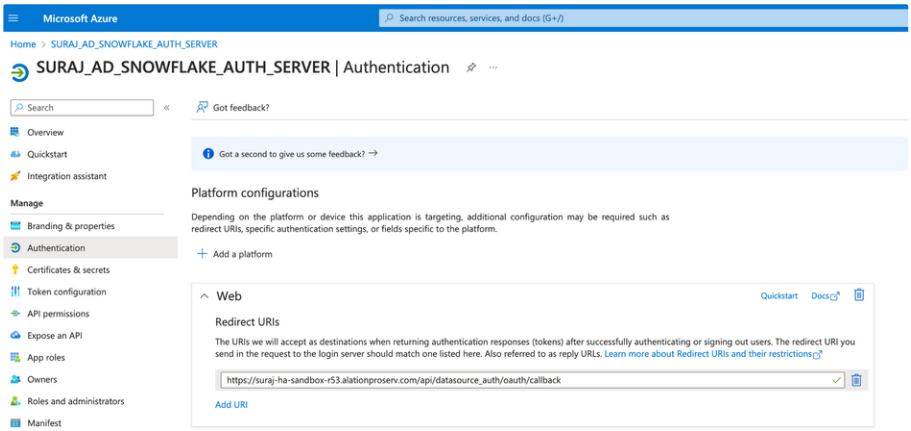
Stage1.1 Create App registration for AUTH SERVER





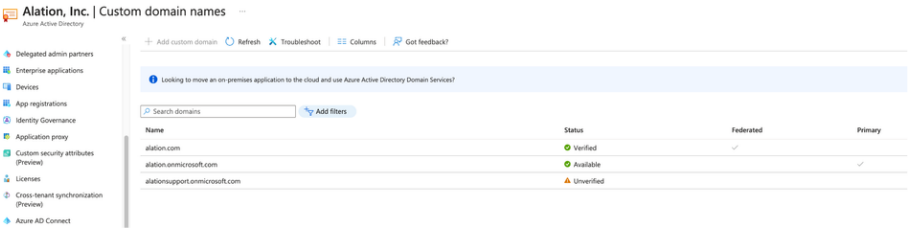
Provide details:

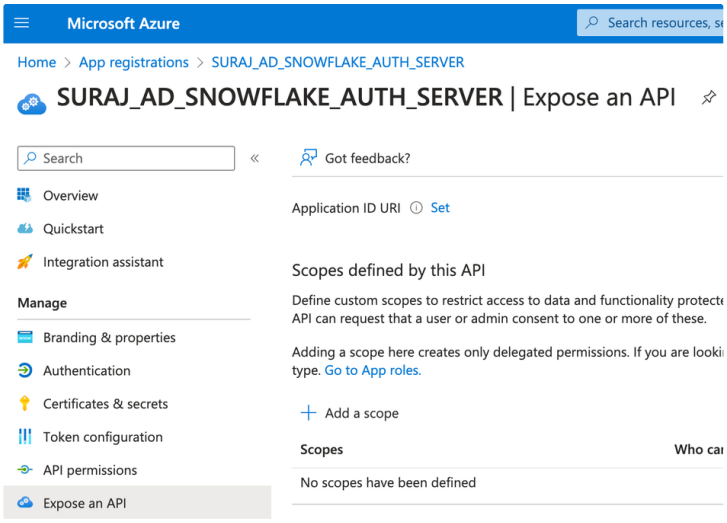
- Name SURAJ_AD_SNOWFLAKE_AUTH_SERVER
- Platform type web callbackurl (Note No forward)
- https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/callback



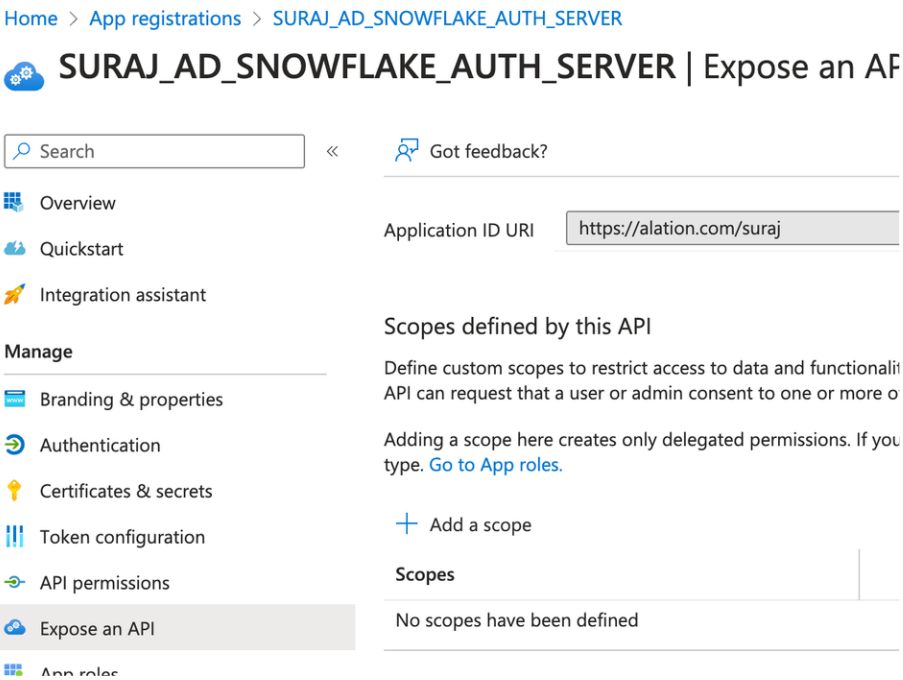
Set Application ID URI (We will use https://alation.com/ as Application ID URI)

- Note Since only allowed domains URL placeholders in our case are
- https://portal.azure.com/#view/Microsoft_AAD_IAM/ActiveDirectoryMenuBlade/~/Domains

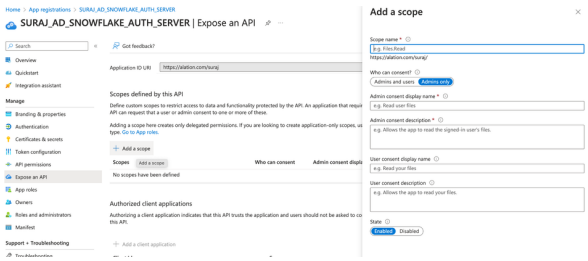




Application ID URI https://alation.com/suraj

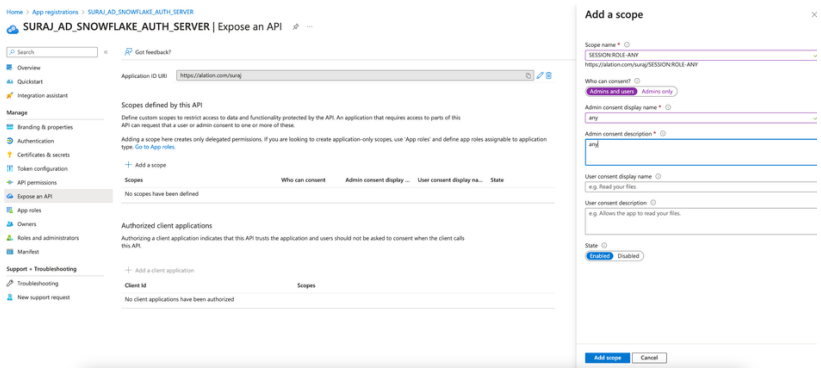


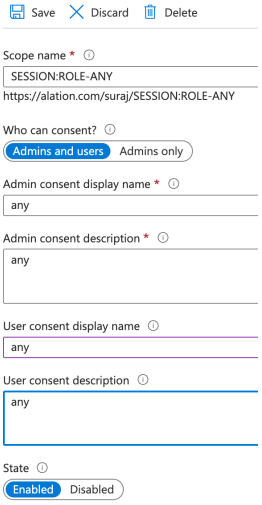
Now click Add scope



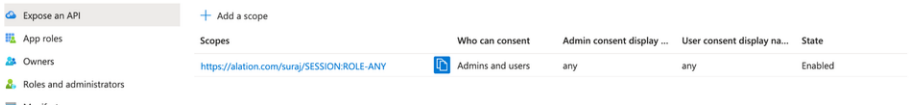
Provide:

- Scope Name: SESSION:ROLE-ANY
- Who can access: Admins and users
- Consent :any



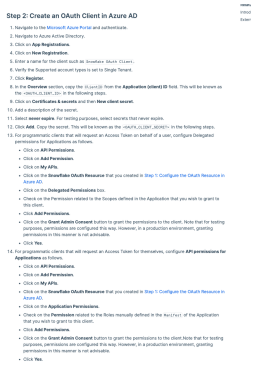


Scope

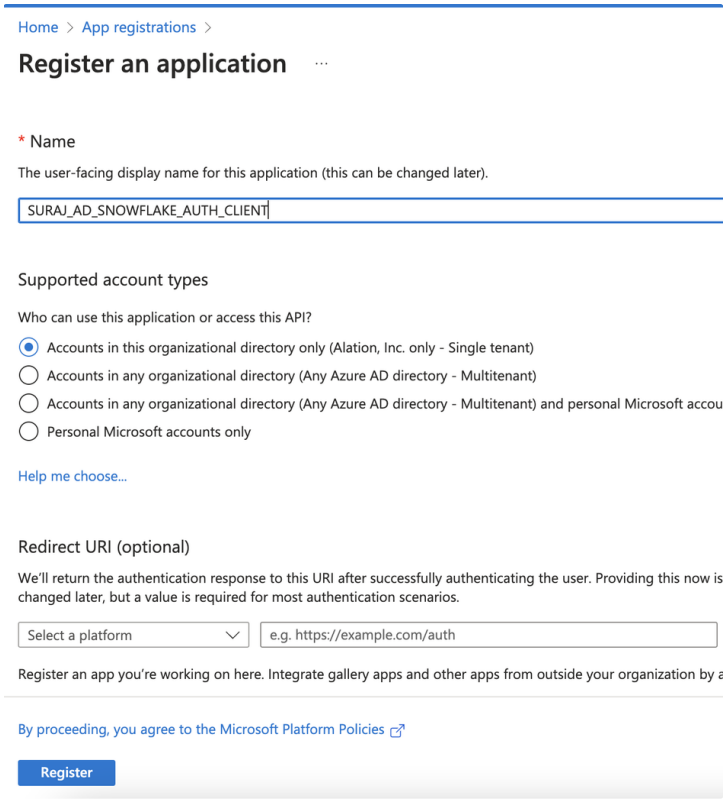


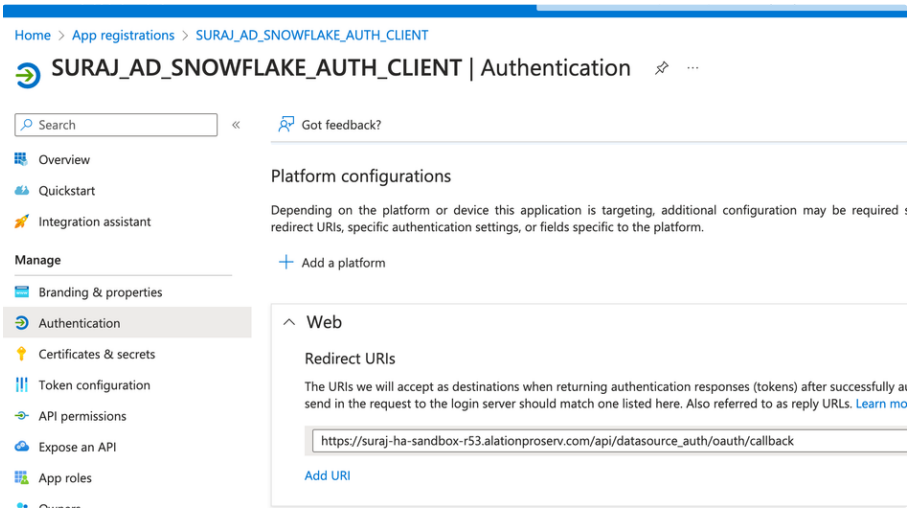
Stage1.2 Create App registration for AUTH Client

CREATE AUTH CLIENT



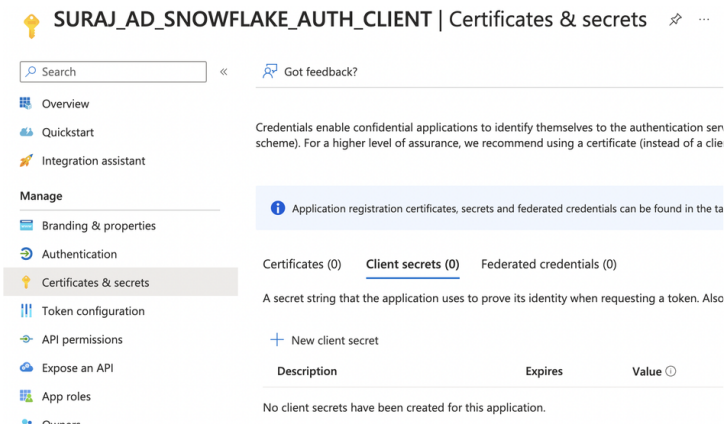
- SURAJ_AD_SNOWFLAKE_AUTH_CLIENT
- Add Web URL https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/callback

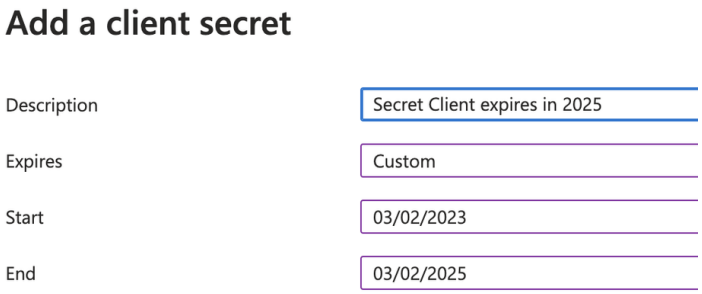


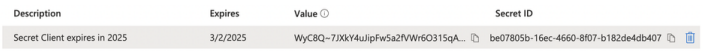


Generate Client Secret and Copy

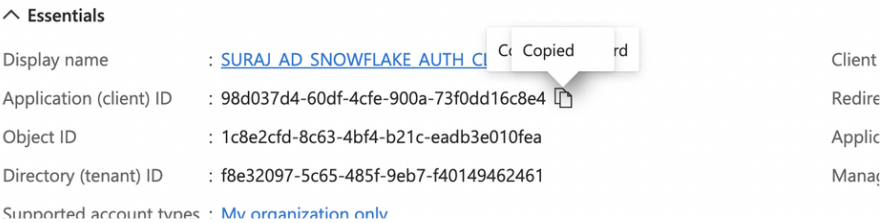
1. Click on **Certificates & secrets** and then **New client secret**.
2. Add a description of the secret.
3. Select **never expire**. For testing purposes, select secrets that never expire.
4. Click **Add**. Copy the secret. This will be known as the <OAUTH_CLIENT_SECRET> in the following steps.







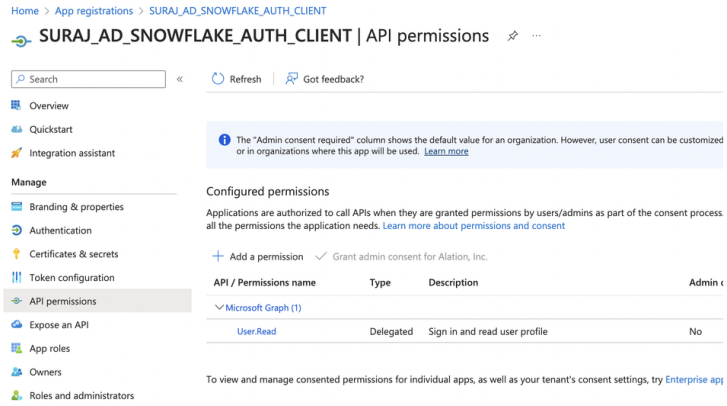
- CLIENT SECRET FOR CLIENT AUTH WyC8Q~7JXkY4uJipFw5a2fVWr6O315qAA4CEGc0D
- FOR CLIENT AUTH CLIENT ID 98d037d4-60df-4cfe-900a-73f0dd16c8e4



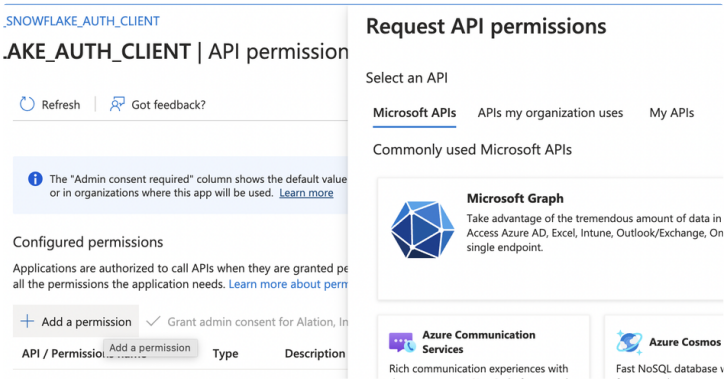
Add API Permissions to Auth Server created

For programmatic clients that will request an Access Token on behalf of a user, configure Delegated permissions for Applications as follows.

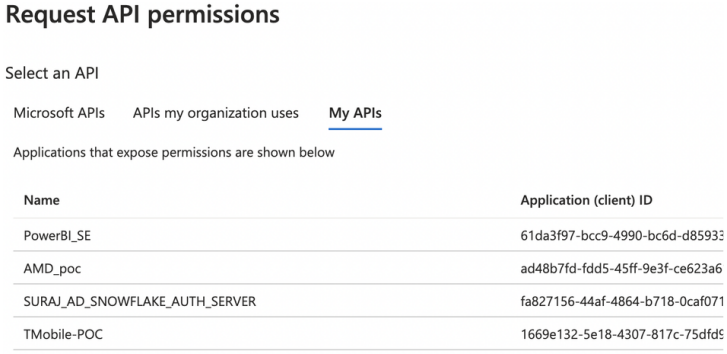
- Click on **API Permissions**.
- Click on **Add Permission**.
- Click on **My APIs**.
- Click on the **Snowflake OAuth Resource** that you created in Step 1: Configure the OAuth Resource in Azure AD.
- Click on the **Delegated Permissions** box.
- Check on the Permission related to the Scopes defined in the Application that you wish to grant to this client.
- Click **Add Permissions**.
- Click on the **Grant Admin Consent** button to grant the permissions to the client. Note that for testing purposes, permissions are configured this way. However, in a production environment, granting permissions in this manner is not advisable.
- Click **Yes**.



Add permission

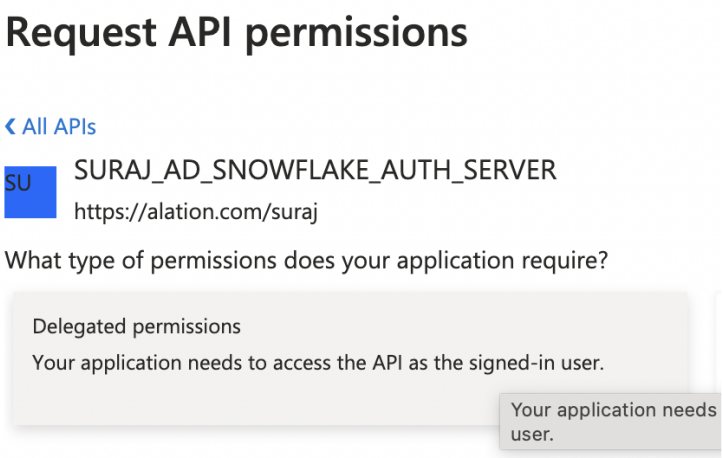


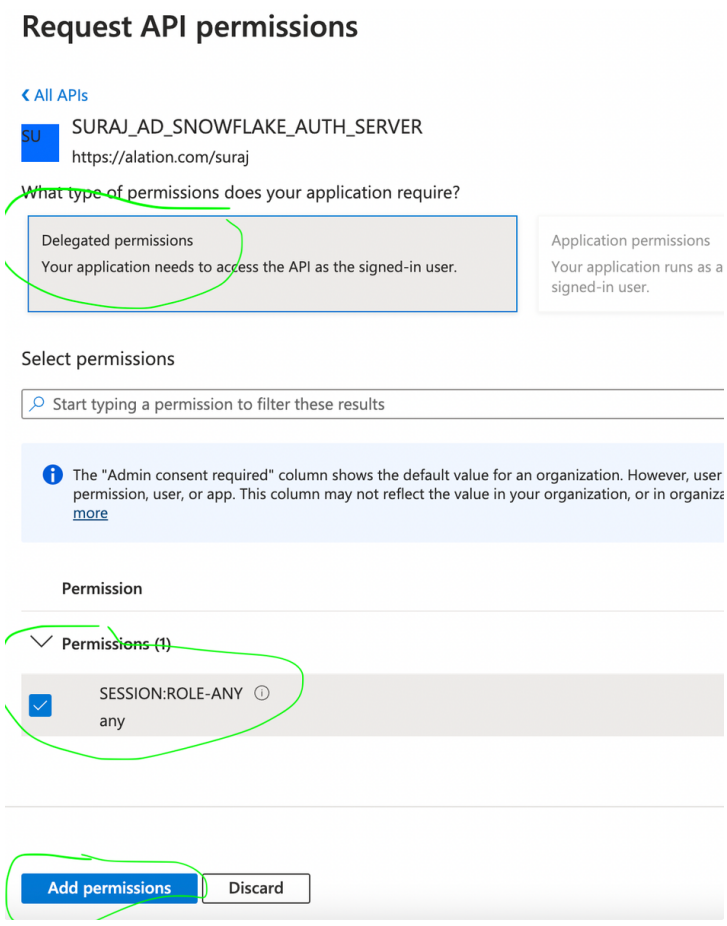
Switch to My API s on popup

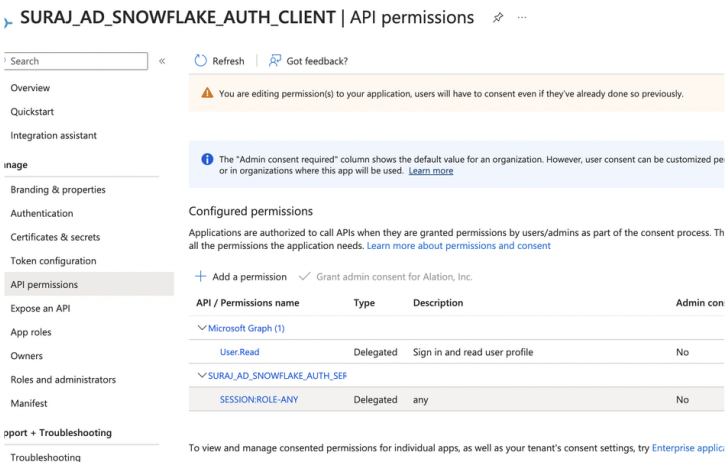


Double click Auth Server Created

Press Delegated Permissions ,select and add Permission



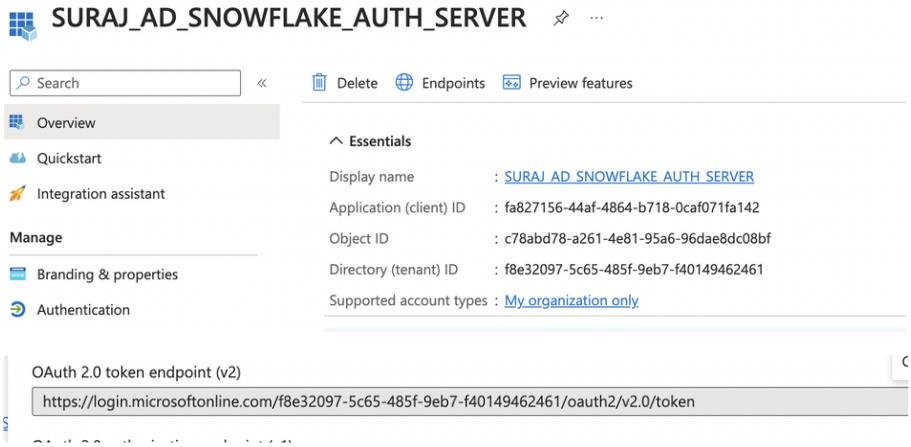

Request API permissions
All APIs
SU
SURAJ_AD_SNOWFLAKE_AUTH_SERVER
https://alation.com/suraj
What type of permissions does your application require?
Delegated permissions
Your application needs to access the API as the signed-in user.
Application permissions
Your application runs as a
signed-in user.
Select permissions
Start typing a permission to filter these results
The "Admin consent required" column shows the default value for an organization. However, user
permission, user, or app. This column may not reflect the value in your organization, or in organiz
more
Permission
Permissions (1)
SESSION:ROLE-ANY
any
Add permissions
Discard



Collect Azure AD Information for Snowflake

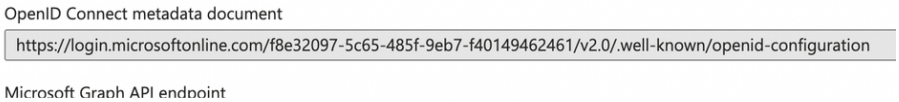
1. Navigate to the Microsoft Azure Portal and authenticate.
2. Navigate to Azure Active Directory.
3. Click on **App Registrations**.
4. Click on the **Snowflake OAuth Resource** that you created in Step 1: Configure the OAuth Resource in Azure AD.
5. Click on **Endpoints** in the **Overview** interface.
6. On the right-hand side, copy the **OAuth 2.0 token endpoint (v2)** and note the URLs for **OpenID Connect metadata** and **Federation Connect metadata**.

- The **OAuth 2.0 token endpoint (v2)** will be known as the <AZURE_AD_OAUTH_TOKEN_ENDPOINT> in the following configuration steps. The endpoint should be similar to https://login.microsoftonline.com/90288a9b-97df-4c6d-b025-95713f21cef9/oauth2/v2.0/token.
- For the **OpenID Connect metadata**, open in a new browser window.
  - Locate the "jwks_uri" parameter and copy its value.
  - This parameter value will be known as the <AZURE_AD_JWS_KEY_ENDPOINT> in the following configuration steps. The endpoint should be similar to https://login.microsoftonline.com/90288a9b-97df-4c6d-b025-95713f21cef9/discovery/v2.0/keys.
- For the **Federation metadata document**, open the URL in a new browser window.
  - Locate the "entityID" parameter in the XML Root Element and copy its value.
  - This parameter value will be known as the <AZURE_AD_ISSUER> in the following configuration steps. The entityID value should be similar to https://sts.windows.net/90288a9b-97df-4c6d-b025-95713f21cef9/.



Token endpoint v2https://login.microsoftonline.com/f8e32097-5c65-485f-9eb7-f40149462461/oauth2/v2.0/token

https://login.microsoftonline.com/f8e32097-5c65-485f-9eb7-f40149462461/v2.0/.well-known/openid-configuration



jwks_uri **Locate Locate the "jwks_uri" here**

"jwks_uri":"https://login.microsoftonline.com/f8e32097-5c65-485f-9eb7-f40149462461/discovery/v2.0/keys"

Next Capture Entity ID:

https://login.microsoftonline.com/f8e32097-5c65-485f-9eb7-f40149462461/federationmetadata/2007-06/federationmetadata.xml

Locate entityID here

entityID="https://sts.windows.net/f8e32097-5c65-485f-9eb7-f40149462461/"

Stage2:Snowflake Create a Security Integration Azure type

```
create security integration SURAJ_SNOWFLAKE_AZUREAD_EXTERNAL_OAUTH
    type = external_oauth
    enabled = true
    external_oauth_type = azure
    external_oauth_issuer = 'https://sts.windows.net/f8e32097-5c65-485f-9eb7-f40149462461/'
    external_oauth_jws_keys_url = 'https://login.microsoftonline.com/f8e32097-5c65-485f-9eb7-f40149462461/dis
    external_oauth_audience_list = ('https://alation.com/suraj')
    external_oauth_any_role_mode = 'ENABLE'
    external_oauth_token_user_mapping_claim = 'upn'
    external_oauth_snowflake_user_mapping_attribute = 'login_name';
```

```
create security integration SURAJ_SNOWFLAKE_AZUREAD_EXTERNAL_OAUTH
    type = external_oauth
    enabled = true
    external_oauth_type = azure
    external_oauth_issuer = 'https://sts.windows.net/f8e32097-5c65-485f-9eb7-f40149462461/'
    external_oauth_jws_keys_url = 'https://login.microsoftonline.com/f8e32097-5c65-485f-9eb7-f40149462461/discovery/v2.0/keys'
    external_oauth_audience_list = ('https://alation.com/suraj')
    external_oauth_any_role_mode = 'ENABLE'
    external_oauth_token_user_mapping_claim = 'upn'
    external_oauth_snowflake_user_mapping_attribute = 'login_name';
```

Objects | Editor | Results | Chart

status
Integration SURAJ_SNOWFLAKE_AZUREAD_EXTERNAL_OAUTH successfully created.

```
DESC SECURITY INTEGRATION "SURAJ_SNOWFLAKE_AZUREAD_EXTERNAL_OAUTH";
```

Objects | Editor | Results | Chart

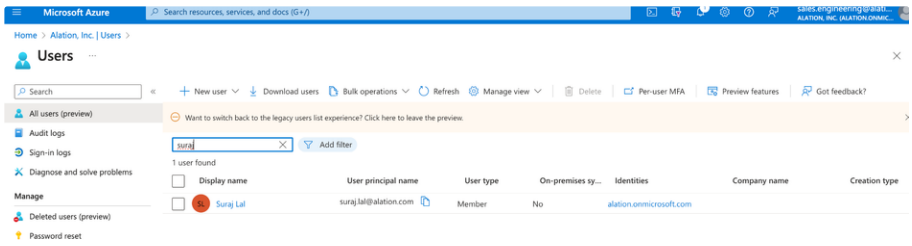
	property	property_type	property_value
1	ENABLED	Boolean	true
2	EXTERNAL_OAUTH_ISSUER	String	https://sts.windows.net/f8e32097-5c65-485f-9eb7-f40149462461/
3	EXTERNAL_OAUTH_JWS_KEYS_URL	Object	https://login.microsoftonline.com/f8e32097-5c65-485f-9eb7-f40149462461/disc
4	EXTERNAL_OAUTH_ANY_ROLE_MODE	String	ENABLE
5	EXTERNAL_OAUTH_RSA_PUBLIC_KEY	String	
6	EXTERNAL_OAUTH_RSA_PUBLIC_KEY_2	String	
7	EXTERNAL_OAUTH_BLOCKED_ROLES_LIST	List	ACCOUNTADMIN,SECURITYADMIN
8	EXTERNAL_OAUTH_ALLOWED_ROLES_LIST	List	
9	EXTERNAL_OAUTH_AUDIENCE_LIST	List	https://alation.com/suraj
10	EXTERNAL_OAUTH_TOKEN_USER_MAPPING_CLAIM	Object	['upn']
11	EXTERNAL_OAUTH_SNOWFLAKE_USER_MAPPING_ATTRIBUTE	String	login_name
12	COMMENT	String	

Just for understand what to put in Alation for User Field Name (Like EMail or Username or UPN)

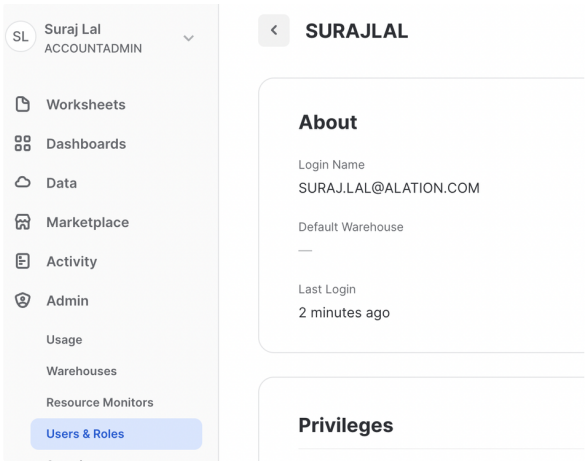
Sample Users Mapping

In Azure AD:

we used UPN (USER Principal name) external_oauth_token_user_mapping_claim = 'upn'

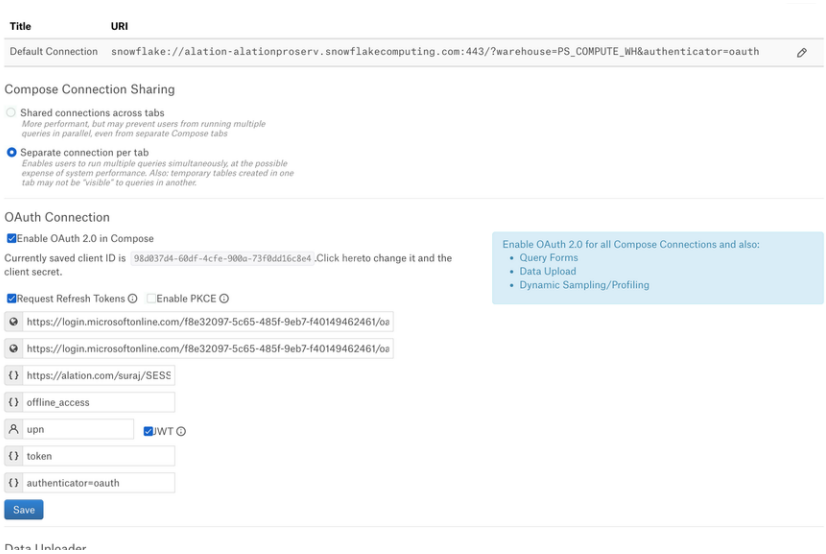


In Snowflake:We use Login Name



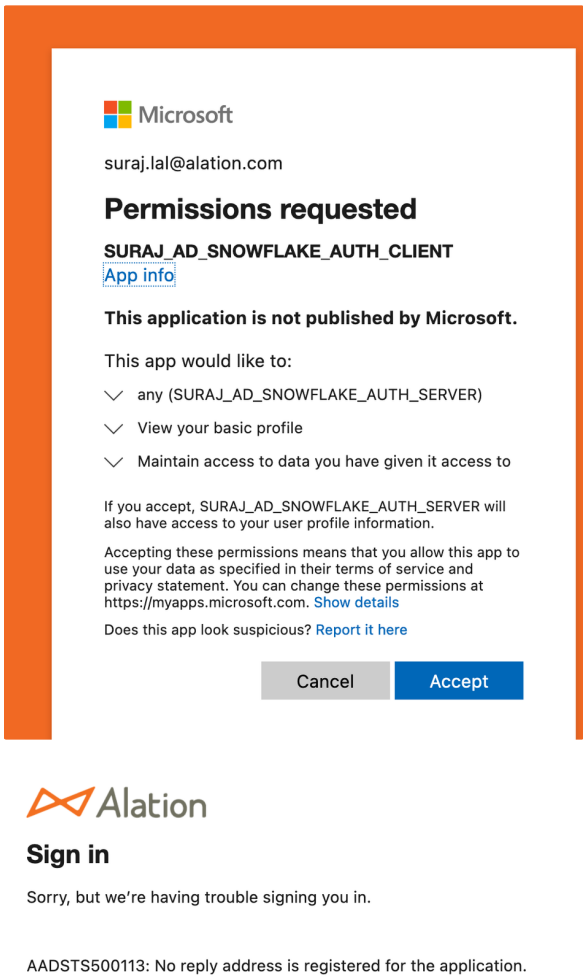
Stage3 Configure Alation

Configure Alation Instance with Azure Client ID and Azure Auth server endpoints Details details

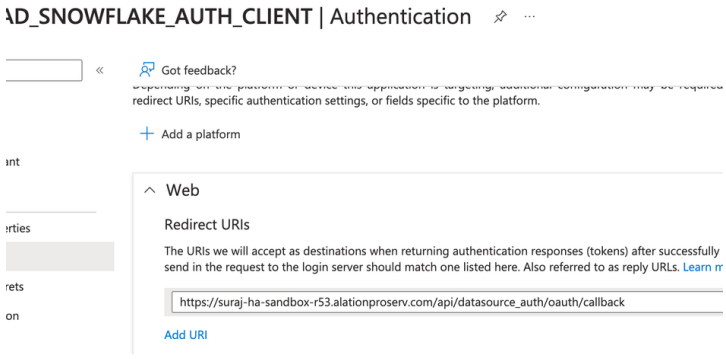


Test Azure Ad Flow

Error 1: No reply Address



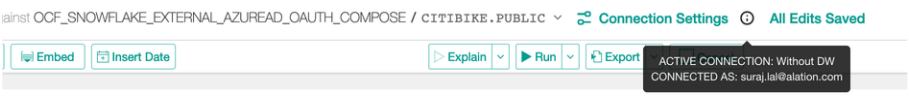
Solution: After Adding Client Redirect URI on Auth Client this got fixed



Error2 Callback/ vs callback

Here i tried with callback/ but it failed as the request was from callback only not callback/ with slash

Success Image



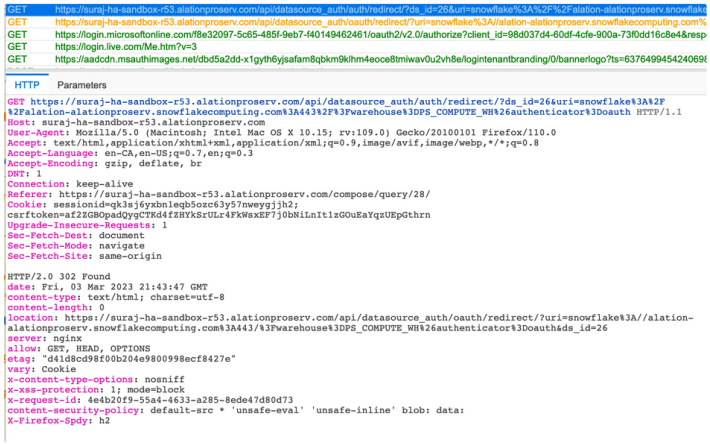
Tracing Success Routing Azure:

Step1 On Opening connection -on compose Redirect OAUTH : As soon we open Compose with Respective DataSource

1st call: GET is redirected and added with parameter ds_id=here(26)

GET **https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/auth/redirect/?ds_id=26&uri=snowflake%3A%2F%2Falation-alationproserv.snowflakecomputing.com%3A443%2F%3Fwarehouse%3DPS_COMPUTE_WH%26authenticator%3Doauth** HTTP/1.1 Host: suraj-ha-sandbox-r53.alationproserv.com User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/110.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 Accept-Language: en-CA,en-US;q=0.7,en;q=0.3 Accept-Encoding: gzip, deflate, br DNT: 1 Connection: keep-alive Referer: https://suraj-ha-sandbox-r53.alationproserv.com/compose/query/28/ Cookie: sessionid=qk3sj6yxbn1eqb5ozc63y57nweygjjh2; csrftoken=af2ZGBOpadQygCTKd4fZHYkSrULr4FkWsxEF7j0bNiLnIt1zGOuEaYqzUEpGthrn Upgrade-Insecure-Requests: 1 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: same-origin

HTTP/2.0 302 Found date: Fri, 03 Mar 2023 21:43:47 GMT content-type: text/html; charset=utf-8 content-length: 0 location: **https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/redirect/?uri=snowflake%3A//alation-alationproserv.snowflakecomputing.com%3A443/%3Fwarehouse%3DPS_COMPUTE_WH%26authenticator%3Doauth&ds_id=26** server: nginx allow: GET, HEAD, OPTIONS etag: "d41d8cd98f00b204e9800998ecf8427e" vary: Cookie x-content-type-options: nosniff x-xss-protection: 1; mode=block x-request-id: 4e4b20f9-55a4-4633-a285-8ede47d80d73 content-security-policy: default-src * 'unsafe-eval' 'unsafe-inline' blob: data: X-Firefox-Spdy: h2



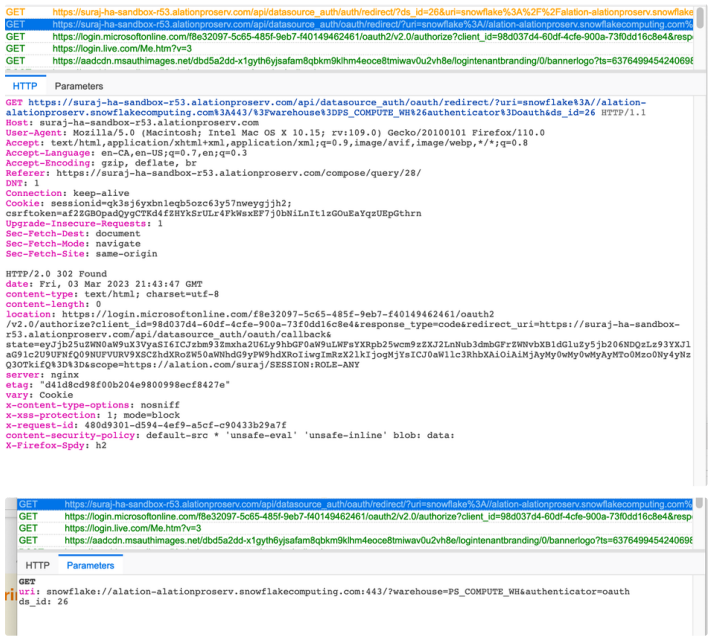
Step2:On Add SSO User

The movement we want to add SSO user , it is redirected to Authorize URL configured in Compose with client id, response_type=code and redirect URI = alation callback url

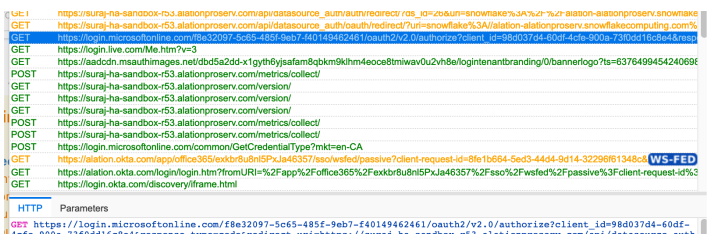
With state value to return back and final scope variable is passed for matching Audience names

GET https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/redirect/?uri=snowflake%3A//alation-alationproserv.snowflakecomputing.com%3A443/%3Fwarehouse%3DPS_COMPUTE_WH%26authenticator%3Doauth&ds_id=26 HTTP/1.1 Host: suraj-ha-sandbox-r53.alationproserv.com User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/110.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 Accept-Language: en-CA,en-US;q=0.7,en;q=0.3 Accept-Encoding: gzip, deflate, br Referer: https://suraj-ha-sandbox-r53.alationproserv.com/compose/query/28/ DNT: 1 Connection: keep-alive Cookie: sessionid=qk3sj6yxbn1eqb5ozc63y57nweygjjh2; csrftoken=af2ZGBOpadQygCTKd4fZHYkSrULr4FkWsxEF7j0bNiLnIt1zGOuEaYqzUEpGthrn Upgrade-Insecure-Requests: 1 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: same-origin

HTTP/2.0 302 Found date: Fri, 03 Mar 2023 21:43:47 GMT content-type: text/html; charset=utf-8 content-length: 0 location: **https://login.microsoftonline.com/f8e32097-5c65-485f-9eb7-f40149462461/oauth2/v2.0/authorize?client_id**=98d037d4-60df-4cfe-900a-73f0dd16c8e4&**response_type=code**&**redirect_uri**=https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/callback&state=eyJjb25uZWN0aW9uX3VyaSI6ICJzbm93Zmxha2U6Ly9hbGF0aW9uLWFsYXRpb25wcm9zZXJ2LnNub3dmbGFrZWNvbXB1dGluZy5jb206NDQzLz93YXJlaG91c2U9UFNfQ09NUFVURV9XSCZhdXRoZW50aWNhdG9yPW9hdXRoIiwgImRzX2lkIjogMjYsICJ0aW1lc3RhbXAiOiAiMjAyMy0wMy0wMyAyMTo0Mzo0Ny4yNzQ3OTkifQ%3D%3D&**scope**=https://alation.com/suraj/SESSION:ROLE-ANY server: nginx etag: "d41d8cd98f00b204e9800998ecf8427e" vary: Cookie x-content-type-options: nosniff x-xss-protection: 1; mode=block x-request-id: 480d9301-d594-4ef9-a5cf-c90433b29a7f content-security-policy: default-src * 'unsafe-eval' 'unsafe-inline' blob: data: X-Firefox-Spdy: h2


GET https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/auth/redirect/?ds_id=26&uri=snowflake%3A%2F%2Falation-alationproserv.snowflake
GET https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/redirect/?uri=snowflake%3A//alation-alationproserv.snowflakecomputing.com%
GET https://login.microsoftonline.com/f8e32097-5c65-485f-9eb7-f40149462461/oauth2/v2.0/authorize?client_id=98d037d4-60df-4cfe-900a-73f0dd16c8e4&resp
GET https://login.live.com/Me.htm?v=3
GET https://aadcdn.msauthimages.net/dbd5a2dd-x1gyth6yjsafam8qbkm9klhm4eoce8tmiwav0u2vh8e/logintenantbranding/0/bannerlogo?ts=637649945424069
HTTP
Parameters
GET https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/redirect/?uri=snowflake%3A//alation-alationproserv.snowflakecomputing.com%3A443/%3Fwarehouse%3DPS_COMPUTE_WH%26authenticator%3Doauth&ds_id=26 HTTP/1.1
Host: suraj-ha-sandbox-r53.alationproserv.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/110.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-CA,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://suraj-ha-sandbox-r53.alationproserv.com/compose/query/28/
DNT: 1
Connection: keep-alive
Cookie: sessionid=qk3sj6yxbnleqb5ozc63y57nweygjjh2;
csrftoken=aF22GBOpadGygcTKd4fZHYkSrULr4FkWexEF7j0bNiLnIt1zGOuEaYqzUEpGthrn
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
HTTP/2.0 302 Found
date: Fri, 03 Mar 2023 21:43:47 GMT
content-type: text/html; charset=utf-8
content-length: 0
location: https://login.microsoftonline.com/f8e32097-5c65-485f-9eb7-f40149462461/oauth2/v2.0/authorize?client_id=98d037d4-60df-4cfe-900a-73f0dd16c8e4&response_type=code&redirect_uri=https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/callback&state=eyJjb25uZWN0aW9uX3VyaSI6ICJzbm93flxha2U6Ly9hbGF0aW9uLWFsYXRpb25wcm9zZXJ2LnNub3dmbGFrZWNvbXB1dGluZy5jb206NDQzLz93YXJlaG91c2U9UFNfQ09NUFVURV9XSCZhdXRoZW50aWNhdG9yPW9hdXRoIiwgImRzX2lkIjogMjYsICJ0aWl1c3RhbXAiOiAiMjAyMy0wMy0wMyAyMTo0Mzo0Ny4yNzQ3OTkifQ%3D%3D&scope=https://alation.com/suraj/SESSION:ROLE-ANY
server: nginx
etag: "d41d8cd98f00b204e9800998ecf8427e"
vary: Cookie
x-content-type-options: nosniff
x-xss-protection: 1; mode=block
x-request-id: 480d9301-d594-4ef9-a5cf-c90433b29a7f
content-security-policy: default-src * 'unsafe-eval' 'unsafe-inline' blob: data:
X-Firefox-Spdy: h2

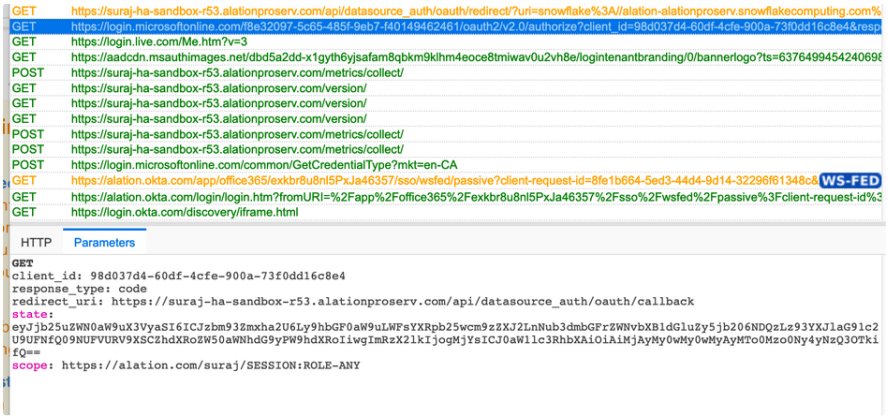
GET https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/redirect/?uri=snowflake%3A//alation-alationproserv.snowflakecomputing.com%
GET https://login.microsoftonline.com/f8e32097-5c65-485f-9eb7-f40149462461/oauth2/v2.0/authorize?client_id=98d037d4-60df-4cfe-900a-73f0dd16c8e4&resp
GET https://login.live.com/Me.htm?v=3
GET https://aadcdn.msauthimages.net/dbd5a2dd-x1gyth6yjsafam8qbkm9klhm4eoce8tmiwav0u2vh8e/logintenantbranding/0/bannerlogo?ts=637649945424069
HTTP
Parameters
GET
uri: snowflake://alation-alationproserv.snowflakecomputing.com:443/?warehouse=PS_COMPUTE_WH&authenticator=oauth
ds_id: 26


GET https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/redirect/?ds_id=26&uri=snowflake%3A%2F%2Falation-alationproserv.snowflake
GET https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/redirect/?uri=snowflake%3A//alation-alationproserv.snowflakecomputing.com%
GET https://login.microsoftonline.com/f8e32097-5c65-485f-9eb7-f40149462461/oauth2/v2.0/authorize?client_id=98d037d4-60df-4cfe-900a-73f0dd16c8e4&resp
GET https://login.live.com/Me.htm?v=3
GET https://aadcdn.msauthimages.net/dbd5a2dd-x1gyth6yjsafam8qbkm9klhm4eoce8tmiwav0u2vh8e/logintenantbranding/0/bannerlogo?ts=637649945424069
POST https://suraj-ha-sandbox-r53.alationproserv.com/metrics/collect/
GET https://suraj-ha-sandbox-r53.alationproserv.com/version/
GET https://suraj-ha-sandbox-r53.alationproserv.com/version/
GET https://suraj-ha-sandbox-r53.alationproserv.com/version/
POST https://suraj-ha-sandbox-r53.alationproserv.com/metrics/collect/
POST https://suraj-ha-sandbox-r53.alationproserv.com/metrics/collect/
POST https://login.microsoftonline.com/common/GetCredentialType?mkt=en-CA
GET https://alation.okta.com/app/office365/exkbr8u8nl5PxJa46357/sso/wsfed/passive?client-request-id=8fe1b664-5ed3-44d4-9d14-32296f61348c8 WS-FED
GET https://alation.okta.com/login/login.htm?fromURI=%2Fapp%2Foffice365%2Fexkbr8u8nl5PxJa46357%2Fsso%2Fwsfed%2Fpassive%3Fclient-request-id%3
GET https://login.okta.com/discovery/iframe.html
HTTP
Parameters
GET https://login.microsoftonline.com/f8e32097-5c65-485f-9eb7-f40149462461/oauth2/v2.0/authorize?client_id=98d037d4-60df-


/oauth/callback&
state=eyJjb25uZWN0aW9uX3VyaSI6ICJzbm93mxha2U6Ly9hbGF0aW9uLWFsYXRpb25wcm9zZXJ2LnNub3dmbGFrZWNvbXB1dGluZy5jb206NDQzLz93YXJl
aG91c2U9UFNfQ09NUFVURV9XSCZhdXRoZW50aWNhdG9yPW9hdXRoIiwgImRzX2lkIjogMjYsICJ0aW1lc3RhbXAiOiAiMjAyMy0wMy0wMyAyMTo0Mzo0Ny4yNz
Q3OTk1fQ%3D%3D&scope=https://alation.com/suraj/SESSION:ROLE-ANY HTTP/1.1
Host: login.microsoftonline.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/110.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-CA,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://suraj-ha-sandbox-r53.alationproserv.com/
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site

HTTP/1.1 200 OK
Cache-Control: no-store, no-cache
Pragma: no-cache
Content-Type: text/html; charset=utf-8
Content-Encoding: gzip
Expires: -1
Vary: Accept-Encoding
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Link: <https://aadcdn.msauth.net>; rel=preconnect; crossorigin, <https://aadcdn.msauth.net>; rel=dns-prefetch,
<https://aadcdn.msftauth.net>; rel=dns-prefetch
X-DNS-Prefetch-Control: on
P3P: CP="DSP CUR OTPi IND OTRi ONL FIN"
x-ms-request-id: 5d2030e3-4899-42e4-b987-375e191c0a01
x-ms-ests-server: 2.1.14649.20 - WUS2 ProdSlices
Referrer-Policy: strict-origin-when-cross-origin
X-XSS-Protection: 0
Set-Cookie: buid=0.ARcAlyDj-GVcX0iet_QBSUYkYdQ30JjfYP5MkApz8N0WyOQXAAA.AQABAAEAAAD--
DLA3VO7QrddgJg7Wevr2TYIrevrZAMz1NtdDDCT2vcWlwvTxuKlzJE5mlCv0q8l0CK3vJVNO4fsF5z5Ca3zOsZ0EdN9Urzlz4udXYJFa_e37BhvXF26sUwawq8
pVqogAA; expires=Sun, 02-Apr-2023 21:43:47 GMT; path=/; secure; HttpOnly; SameSite=None
fpc=AhBEzJW_r15Oooa-WiRofqKiCkvHAOAABNhlNsOAAAA; expires=Sun, 02-Apr-2023 21:43:47 GMT; path=/; secure; HttpOnly;

122 requests received (45 hidden)



Now User is prompted with default page at SSO Server (Azure AD)

On passing correct user details like username@domain.com .

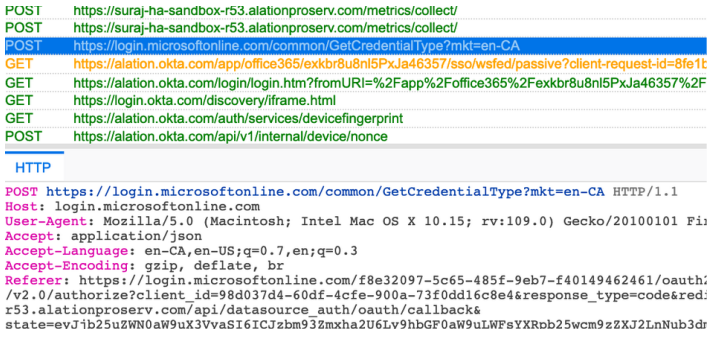
AD will redirect to respective Authenctication page

In our case Azure AD was configured with OKTA

Ignore Metrics and Version calls from Alation

Continuation : we see that SSO server (Azure AD calls) Internal method with reference data from Alation compose call to get proper Authentcion for this user

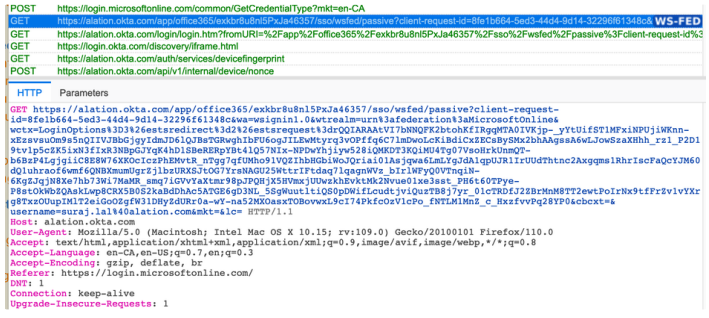
https://login.microsoftonline.com/common/**GetCredentialType**?mkt=en-CA



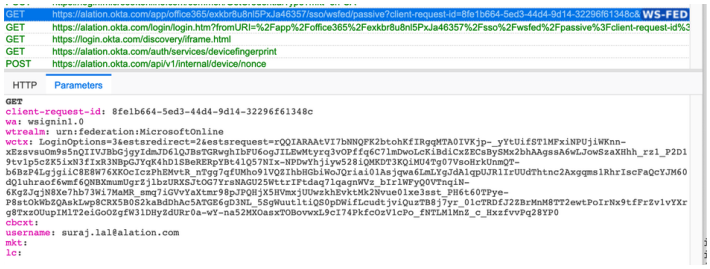
Step 2.1 SSO page1

Here for this specific user company , SSO Server (Azure AD) is configured with OKTA SSO server with WS federation method

So Azure passes all auth request to OKTA



Here we can see the username

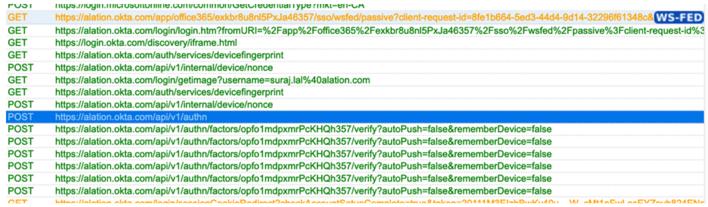


Step2.2 SSO page2

Now Actual User Authentication is Started (here OKTA does)

In our case MFA is configured at OKTA

So it ask for Verify Push notification too



```
GET   https://alation.okta.com/app/office365/exkbr8u8nl5PxJa46357/sso/wsfed/passive?client-request-id=8fe1b664-5ed3-44d4-9d14-32296f61348c& WS-FED
GET   https://alation.okta.com/login/login.htm?fromURI=%2Fapp%2Foffice365%2Fexkbr8u8nl5PxJa46357%2Fsso%2Fwsfed%2Fpassive%3Fclient-request-id%3
GET   https://login.okta.com/discovery/iframe.html
GET   https://alation.okta.com/auth/services/devicefingerprint
POST  https://alation.okta.com/api/v1/internal/device/nonce
GET   https://alation.okta.com/login/getimage?username=suraj.lal%40alation.com
GET   https://alation.okta.com/auth/services/devicefingerprint
POST  https://alation.okta.com/api/v1/internal/device/nonce
POST  https://alation.okta.com/api/v1/authn
POST  https://alation.okta.com/api/v1/authn/factors/opfo1mdpxmrPcKHQh357/verify?autoPush=false&rememberDevice=false
POST  https://alation.okta.com/api/v1/authn/factors/opfo1mdpxmrPcKHQh357/verify?autoPush=false&rememberDevice=false
POST  https://alation.okta.com/api/v1/authn/factors/opfo1mdpxmrPcKHQh357/verify?autoPush=false&rememberDevice=false
POST  https://alation.okta.com/api/v1/authn/factors/opfo1mdpxmrPcKHQh357/verify?autoPush=false&rememberDevice=false
POST  https://alation.okta.com/api/v1/authn/factors/opfo1mdpxmrPcKHQh357/verify?autoPush=false&rememberDevice=false
POST  https://alation.okta.com/api/v1/authn/factors/opfo1mdpxmrPcKHQh357/verify?autoPush=false&rememberDevice=false
```

Step2.3 Alation Gets Code for OAUTH call

Once User is Authenticated Successfully , OKTA sends info back to AZure AD in POST call with SAML response in WSFED Call

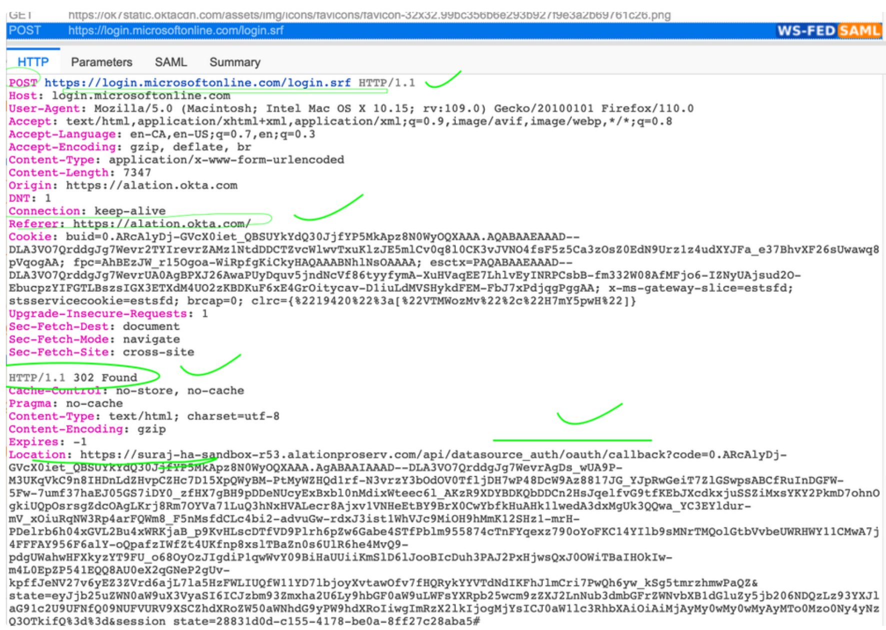
This info is redirected to Alation call back with Code value that Alation requested

We See OKTA make POST Call

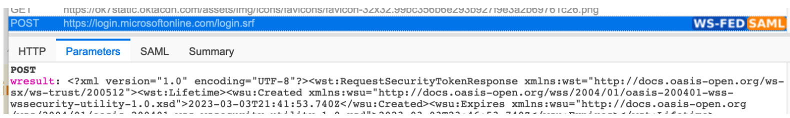
Referer: https://alation.okta.com/

POST https://login.microsoftonline.com/login.srf HTTP/1.1

Redirect Location is Location: https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/callback?code=0.A



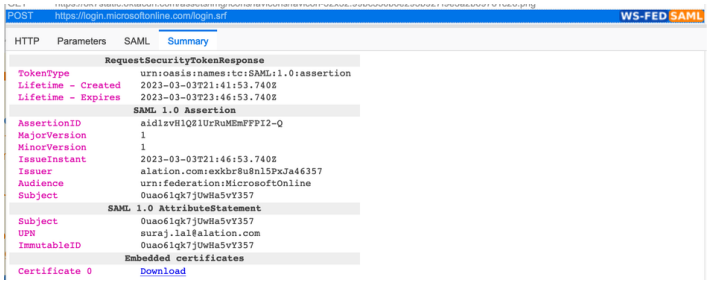
```
POST  https://login.microsoftonline.com/login.srf    WS-FED SAML
HTTP  Parameters  SAML  Summary
POST https://login.microsoftonline.com/login.srf HTTP/1.1
Host: login.microsoftonline.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/110.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-CA,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 7347
Origin: https://alation.okta.com
DNT: 1
Connection: keep-alive
Referer: https://alation.okta.com/
Cookie: buid=0.ARcAlyDj-GVcX0iet_QBSUYkYdQ30JjfYP5MkApz8N0WyOQXAAA.AQABAAEAAAD--
DLA3VO7QrddgJg7Wevr2TYIrevrZAMz1NtdDDCTZvcWlwvTxuKlzJE5mlCv0q810CK3vJVNO4fsF5z5Ca3zOsZ0EdN9Urz1z4udXYJFa_e37BhvXF26sUwawq8
pVqogAA; fpc=AhBEzJW_r15Ogoa-WiRpfgKiCkyHAQAAABNhlNsOAAAA; esctx=PAQABAAEAAAD--
DLA3VO7QrddgJg7WevrUA0AgBPXJ26AwaPUyDquv5jndNcVf86tyyfymA-XuHVaqEE7LhlvEyINRPCsbB-fm332W08AfMFjo6-IZNyUAjsud2O-
EbucpzYIFGTLBszsIGX3ETXdM4UO2zKBDKuF6xE4GrOitycav-D1iuLdMVSHykdFEM-FbJ7xPdjqgPggAA; x-ms-gateway-slice=estsfd;
stsservicecookie=estsfd; brcap=0; clrc={%2219420%22%3a[%22VTMWozMv%22%2c%22H7mY5pwH%22]}
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site

HTTP/1.1 302 Found
Cache-Control: no-store, no-cache
Pragma: no-cache
Content-Type: text/html; charset=utf-8
Content-Encoding: gzip
Expires: -1
Location: https://suraj-ha-sandbox-r53.alationproserv.com/api/datasource_auth/oauth/callback?code=0.ARcAlyDj-
GVcX0iet_QBSUYkYdQ30JjfYP5MkApz8N0WyOQXAAA.AgABAAIAAAD--DLA3VO7QrddgJg7WevrAgDs_wUA9P-
M3UKqVkC9n8IHDnLdZHvpCZHc7D15XpQWyBM-PtMyWZHQd1rf-N3vrzY3bOdOV0TfljDH7wP48DcW9Az8817JG_YJpRwGeiT7ZlGSwpsABCfRuInDGFW-
5Fw-7umf37haEJ05GS7iDY0_zfHX7gBH9pDDeNUcyExBxbl0nMdixWteec6l_AKzR9XDYBDKQbDDCn2HsJqelfvG9tfKEbJXcdkxjuSSZiMxsYKY2PkmD7ohnO
gkiUQpOsrsgZdcOAgLKrj8Rm7OYVa71LuQ3hNxHVALecr8Ajxv1VNHeEtBY9BrX0CwYbfkHuAHk1lwedA3dxMgUk3QQwa_YC3EYldur-
mV_xOiuRqNW3Rp4arFQWm8_F5nMsfdCLc4bi2-advuGw-rdxJ3ist1WhVJc9MiOH9hMmK12SHz1-mrH-
PDelrb6h04xGVL2Bu4xWRKjaB_p9KvHLscDTfVD9Plrh6pZw6Gabe4STfPblm955874cTnFYqexz790oYoFKC14YIlb9sMNrTMQolGtbVvbeUWRHWY11CMwA7j
4FFFAY956F6alY-oQpafzIWfZt4UKfnp8xslTBaZn0s6UlR6he4MvQ9-
pdgUWahwHFXkyzYT9FU_o68OyOzJIgdiP1qwWvY09BiHaUUiiKmSlD6lJooBIcDuh3PAJ2PxHjwsQxJ0OWiTBaIHOkIw-
m4L0EpZP541EQQ8AU0eX2qGNeP2gUv-
kpffJeNV27v6yEZ3ZVrd6ajL7la5HzFWLIUQfW11YD7lbjoyXvtawOfv7fHQRykYYVTdNdIKFhJlmCri7PwQh6yw_kSg5tmrzhmwPaQZ&
state=eyJjb25uZWN0aW9uX3VyaSI6ICJzbm93Zmxha2U6Ly9hbGF0aW9uLWFsYXRpb25wcm9zZXJ2LnNub3dmbGFrZWNvbXB1dGluZy5jb206NDQzLz93YXJl
aG91c2U9UFNfQ09NUFVURV9XSCZhdXRoZW50aWNhdG9yPW9hdXRoIiwgImRzX2lkIjogMjYsICJ0aW1lc3RhbXAiOiAiMjAyMy0wMy0wMyAyMTo0Mzo0Ny4yNz
Q3OTkifQ%3d%3d&session state=28831d0d-c155-4178-be0a-8ff27c28aba5#
```



```
POST  https://login.microsoftonline.com/login.srf    WS-FED SAML
HTTP  Parameters  SAML  Summary
POST
wresult: <?xml version="1.0" encoding="UTF-8"?><wst:RequestSecurityTokenResponse xmlns:wst="http://docs.oasis-open.org/ws-
sx/ws-trust/200512"><wst:Lifetime><wsu:Created xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">2023-03-03T21:41:53.740Z</wsu:Created><wsu:Expires xmlns:wsu="http://docs.oasis-open.org
```

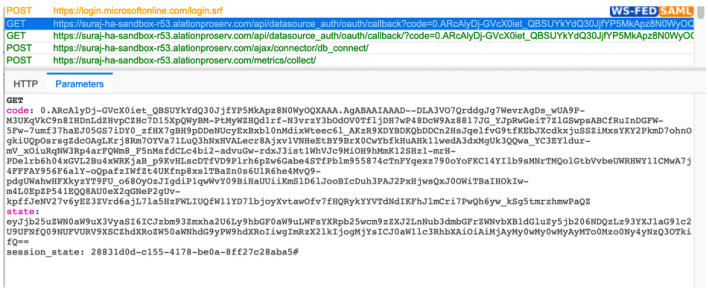
```
POST  https://login.microsoftonline.com/login.srf    WS-FED SAML
HTTP  Parameters  SAML  Summary
                    <saml1:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                          xsi:type="xs:string"
                                          >suraj.lal@alation.com</saml1:AttributeValue>
                </saml1:Attribute>
```

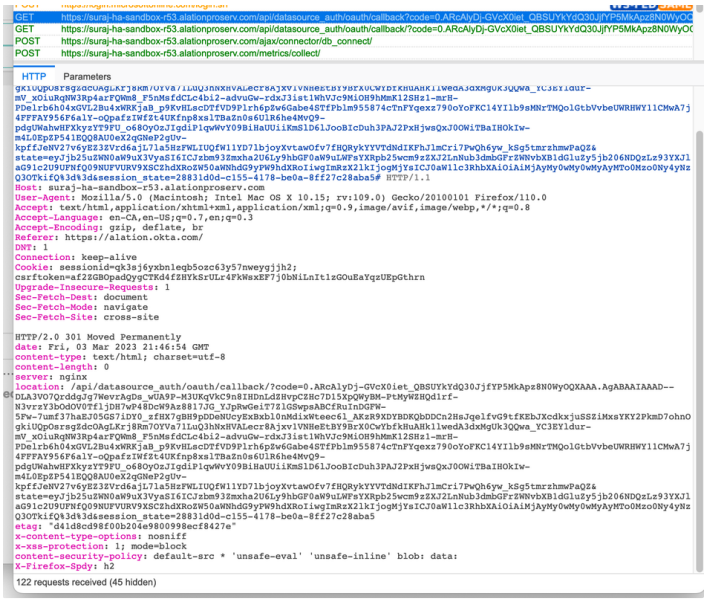
In summary we can see UPN from OKTA to Azure



Step 3:Compose uses code to get token

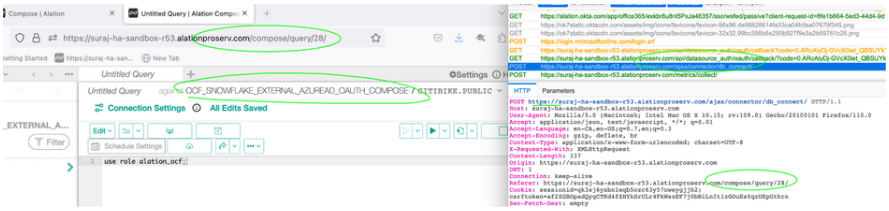
Using the Code value Alation compose make internal call for Token





Query id

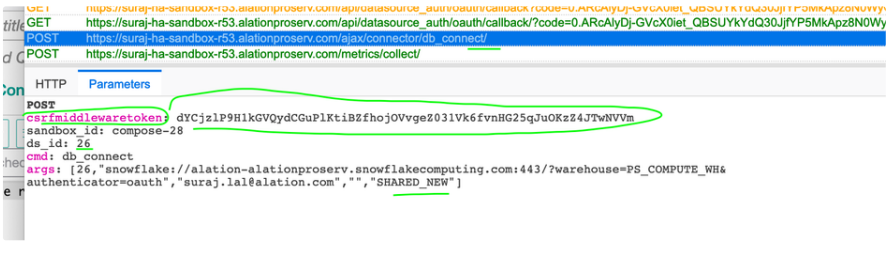
We see the query id we are running in the HTTP request https://suraj-ha-sandbox-r53.alationproserv.com/compose/query/28/



We See

Encrypted token received

csrfmiddlewaretoken: dYCjzlP9H1kGVQydCGuPlKtiBZfhojOVvgeZ031Vk6fvnHG25qJuOKzZ4JTwNVVm

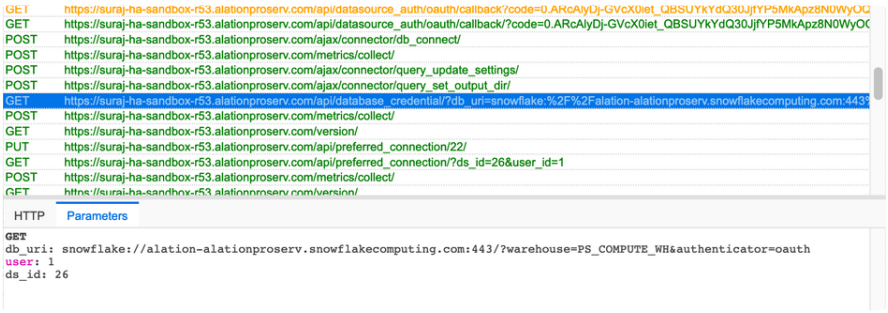


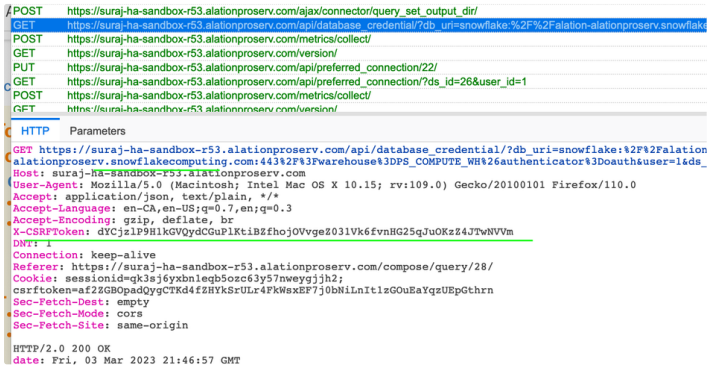
Step4 with token query snowflake

We see GET call to Snowflake via query with Token in request

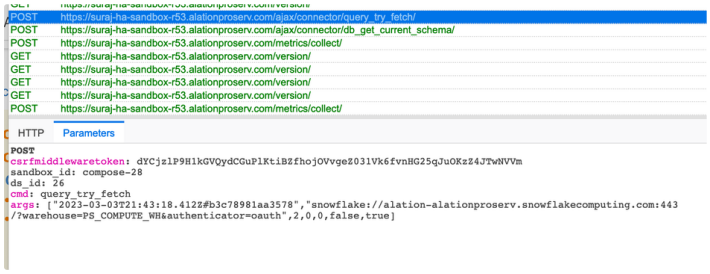
As

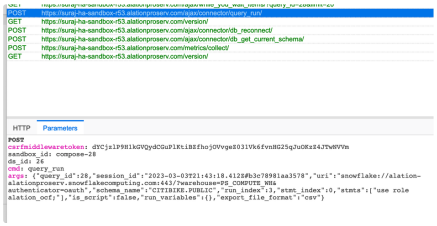
X-CSRFToken: dYCjzlP9H1kGVQydCGuPlKtiBZfhojOVvgeZ031Vk6fvnHG25qJuOKzZ4JTwNVVm





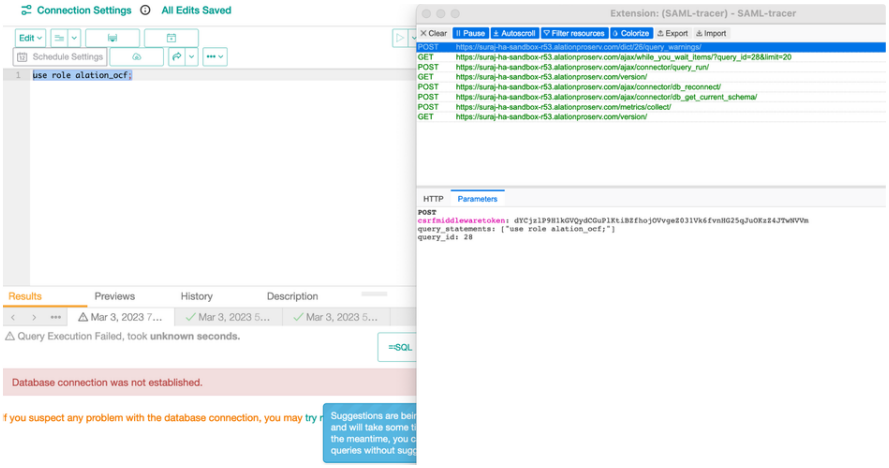
After Connection success if we make a run Query we see POST call recorded with Token

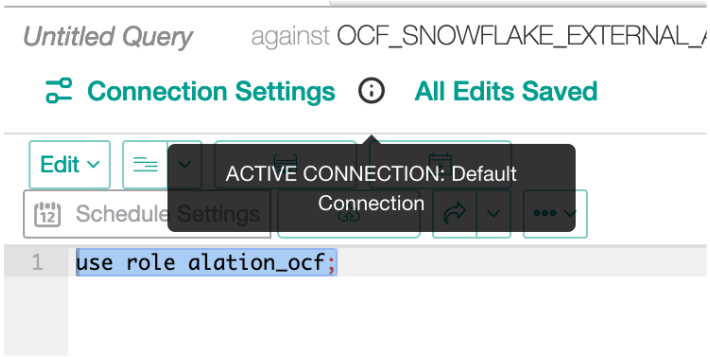




Step5: Re Connect if broken

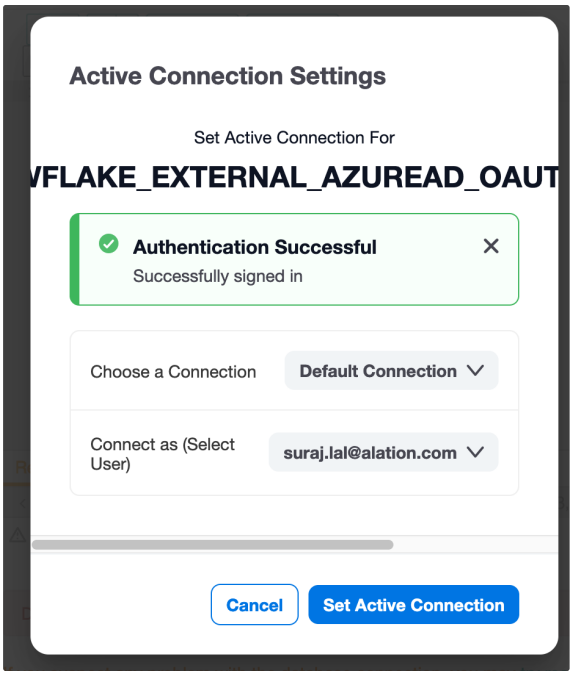
If we run query we get error

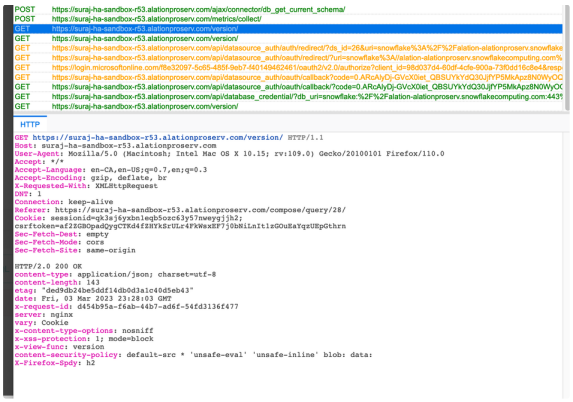




retry

Once we retry to connect, Since SSO token is available on browser it connected›



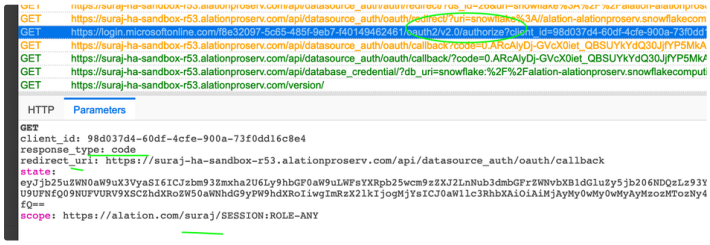


Flow Chart :

Code request, code, token(Backend call hidden from Browser) ,Snowflake query , result

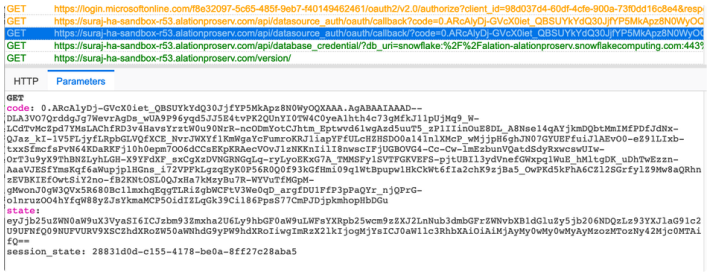
Code request

We see Authorize call initiated with code request

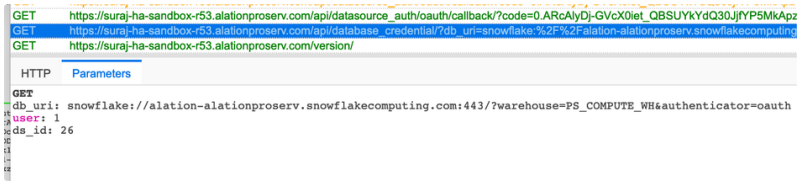


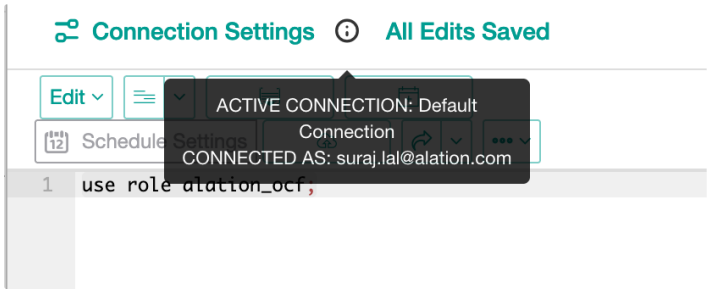
Code and token

In the next call we have the code so we initiated the token call

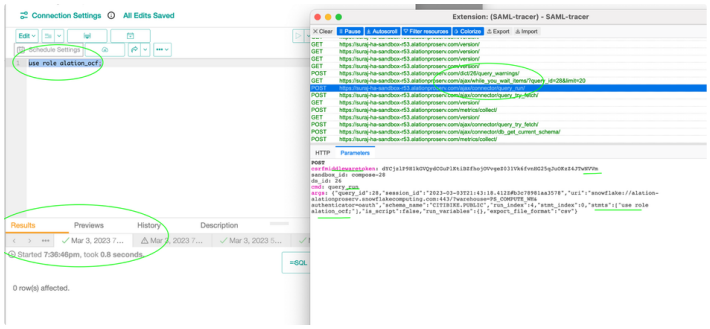


Snowflake call

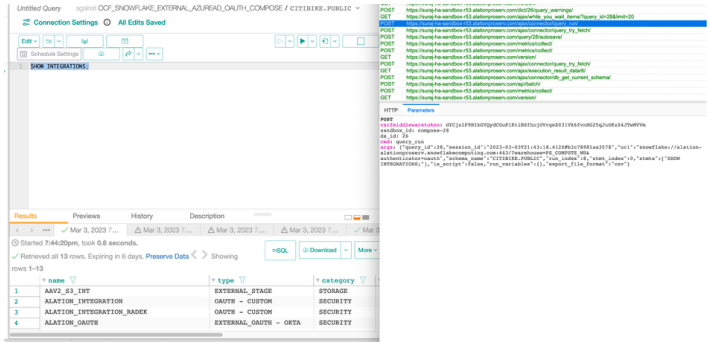




Now lets run the Query again



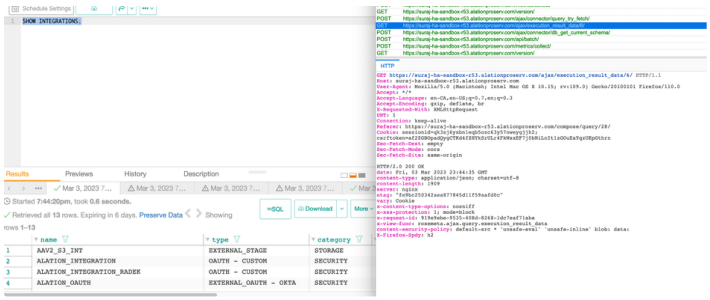
RAN NEW QUERY:query_run



Results:execution_result_data

Then we see execution_result_data call success

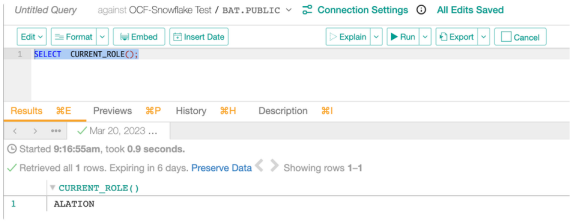
GET https://suraj-ha-sandbox-r53.alationproserv.com/ajax/execution_result_data/6/



Try Switching Roles with Builtin Oauth

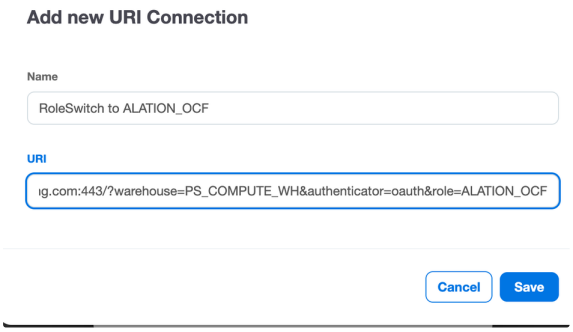
Switch using JDBC URL

Default Role here ALATION

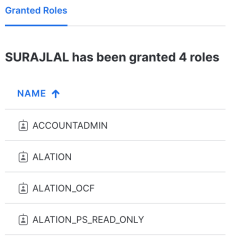


Adding new role

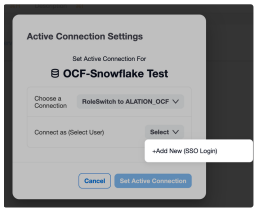
ADD role=role_name in JDBC url



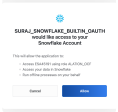
Validate User is mapped with the role

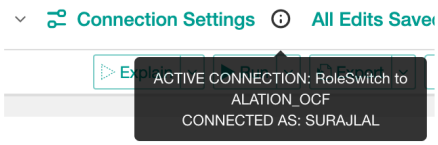


Test

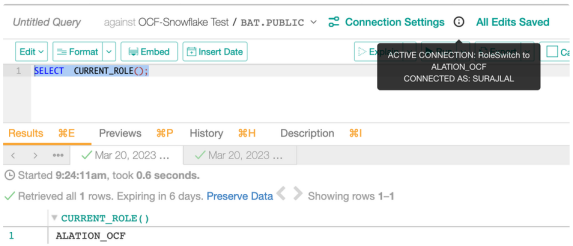


Allow OAUTH





Validate current role



Try Switching Roles with External Oauth

https://alationcorp.atlassian.net/wiki/spaces/ProServ/pages/12611060432 Can't find link

Before Connection

Tried creating a JDBC URL with role parameter

After successful authentication