OpenID

Back

OpenID user authentication (TARGIT 2019 Update 3 feature)

TARGIT can now delegate user authentication to *external identity providers* such as Azure, ADFS, Google etc., opening up for many new features supplied by these providers - including two-factor authentication.

OpenID is increasingly becoming a standard for user authentication. Organizations that already embrace this technology will be delighted to learn that TARGIT now also supports this.

Disclaimer: Administrators working with setting TARGIT up for OpenID user authentication will need to know in advance how to work with the interface of the external identity provider of choice.

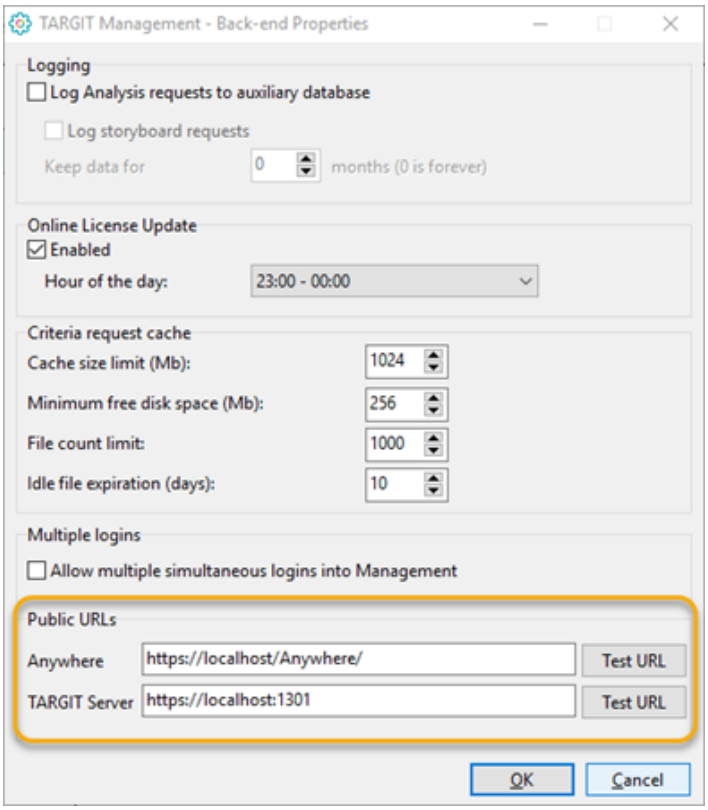
Overview

When you add a new external identity provider to your TARGIT solution, these are the general steps to take:

a. TARGIT needs to be registered as an *Application* with the external identity provider.
b. As part of the application registration process, you will get the *Client ID* and the *Client Secret* that are necessary for the TARGIT server to communicate with the external identity provider.
c. When the application registration is completed, the TARGIT server will also need to know about the *Authorization Endpoint,* the *Toke n Endpoint* and the *Scope*. TARGIT offers *OpenID Discovery*, which, with a single key, will automatically insert all of these properties. Alternatively, copy these properties – one by one – from the external Identity provider.
d. The external identity provider will in return need to know the URIs of the Anywhere component and the TARGIT Server. These URIs are available as soon as you have set up the external identity provider in TARGIT Management.
e. The AD groups from the external identity provider must be mapped with the Windows AD groups in the domain where the TARGIT server is installed. The mapping is done via a script that is connected to the added identity provider in the TARGIT Management client.

Requirements

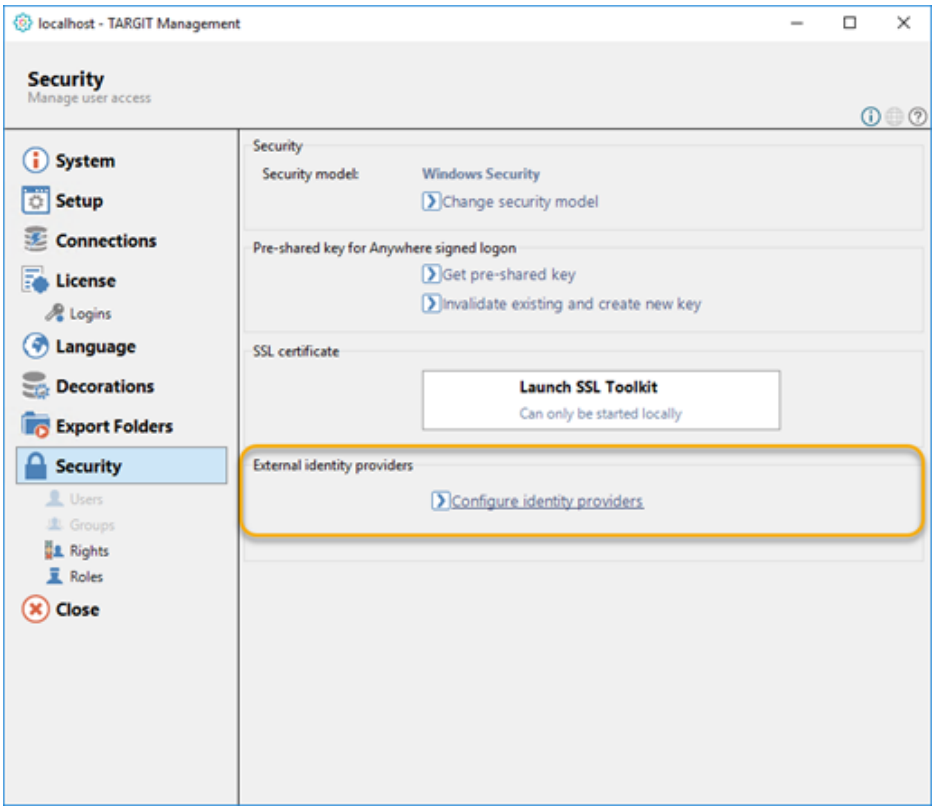
Before adding an external identity provider, make sure that Public URLs for the TARGIT Server and the Anywhere component are set up correctly. This is done in the TARGIT Management client, in *Setup / Back-end*:



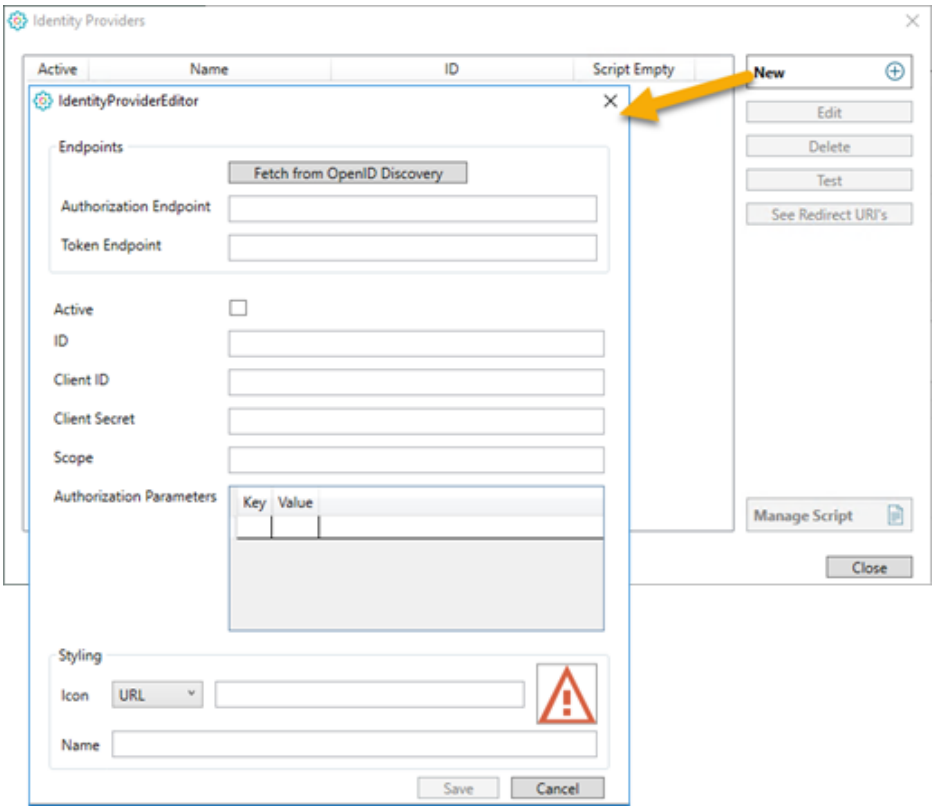
Note: Replace “localhost” with the correct server name or IP address within your organization. Also note that the Anywhere component (on the IIS) and the TARGIT server are not necessarily installed on the same server.

Adding a new Identity Provider

The OpenID authentication method has become a new option in the TARGIT Management client’s Security settings:



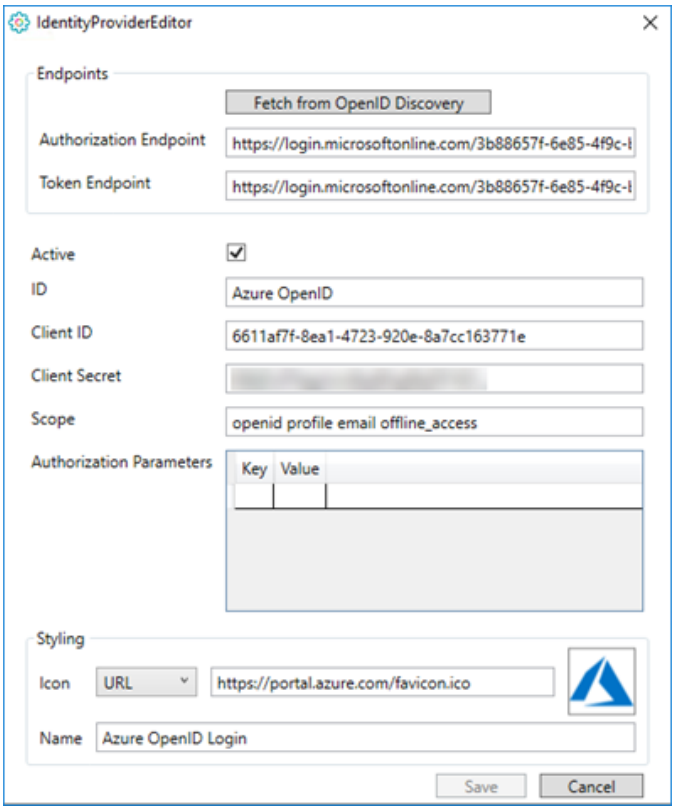
The *Identity Providers* dialog lets you add one or more identity providers to the list.



When you add a new Identity Provider, you must fill in its' settings in the IdentityProviderEditor dialog:

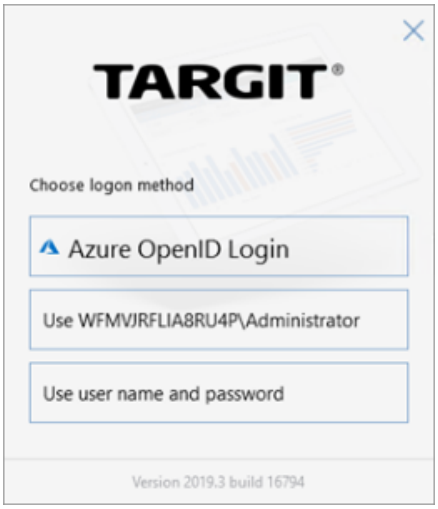
- **Fetch from OpenID Discovery:** This is an option to fill some of the other fields automatically. E.g. in an Azure Active Directory, App Registrations you will have an Endpoint called *OpenID connect metadata document*. This Endpoint can be copied and pasted and used to fetch some of the other settings (or they can be copied/pasted individually):
  - Authorization Endpoint
  - Token Endpoint
  - Scope
  - Authorization Parameters
  - **Active:** You have an option to disable (= not active) an identity provider. This will prevent end-users from using that particular login method.
  - **ID:** The ID is a name/ID you give to this Identity Provider setting. The ID will become part of the URI strings.
  - **Client ID:** On the external identity provider, TARGIT is registered as an application with a Client ID (sometimes called an Application ID.
  - **Client Secret:** The *Secret* (i.e. password) for the registered client is revealed during the application registration. **Important:** On Azure (and potentially other external identity providers) it is not possible to see the Secret after registration is completed.

Example on a filled in Identity Provider for Azure OpenID:



- **Styling, Icon and name:** You can upload or point to an image that you want to associate with this Identity Provider, and you can give it a name.

The icon and the name is what the end-user will see when logging on to a TARGIT client.

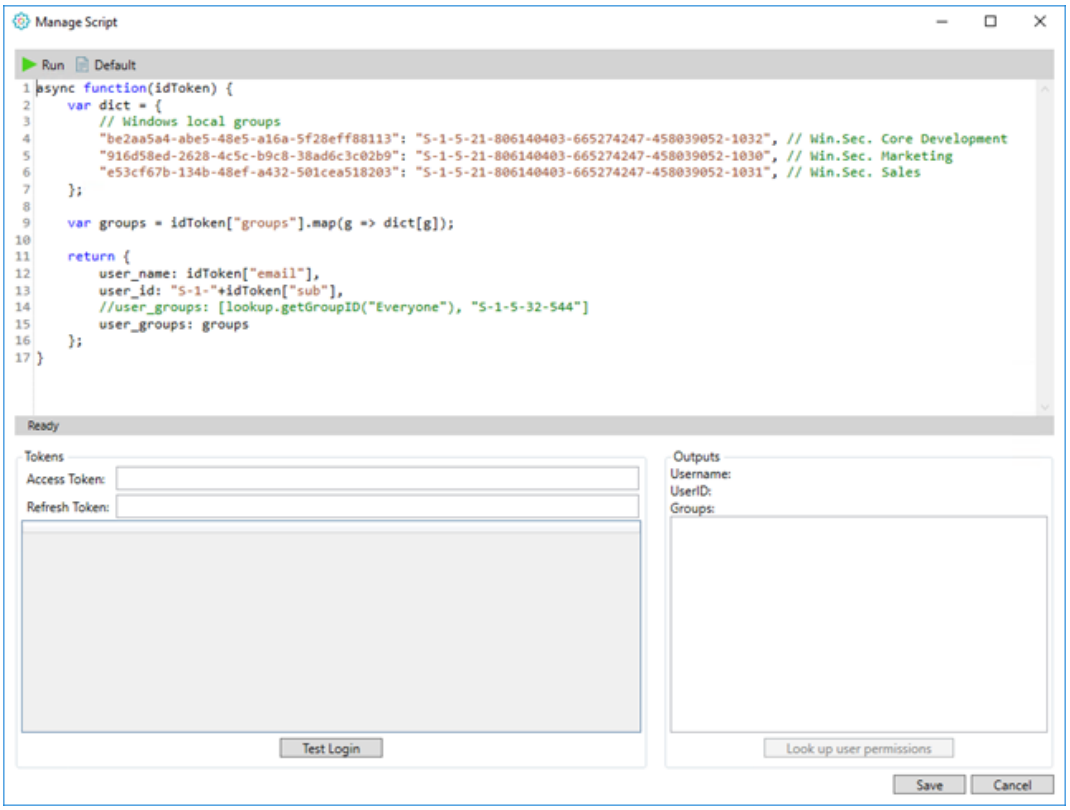


Scripts

An Identity Provider's associated script is essential for, at least, the mapping between the external identity provider's AD users and groups and the internal AD users and groups.

You will need to know the SIDs from both sides.

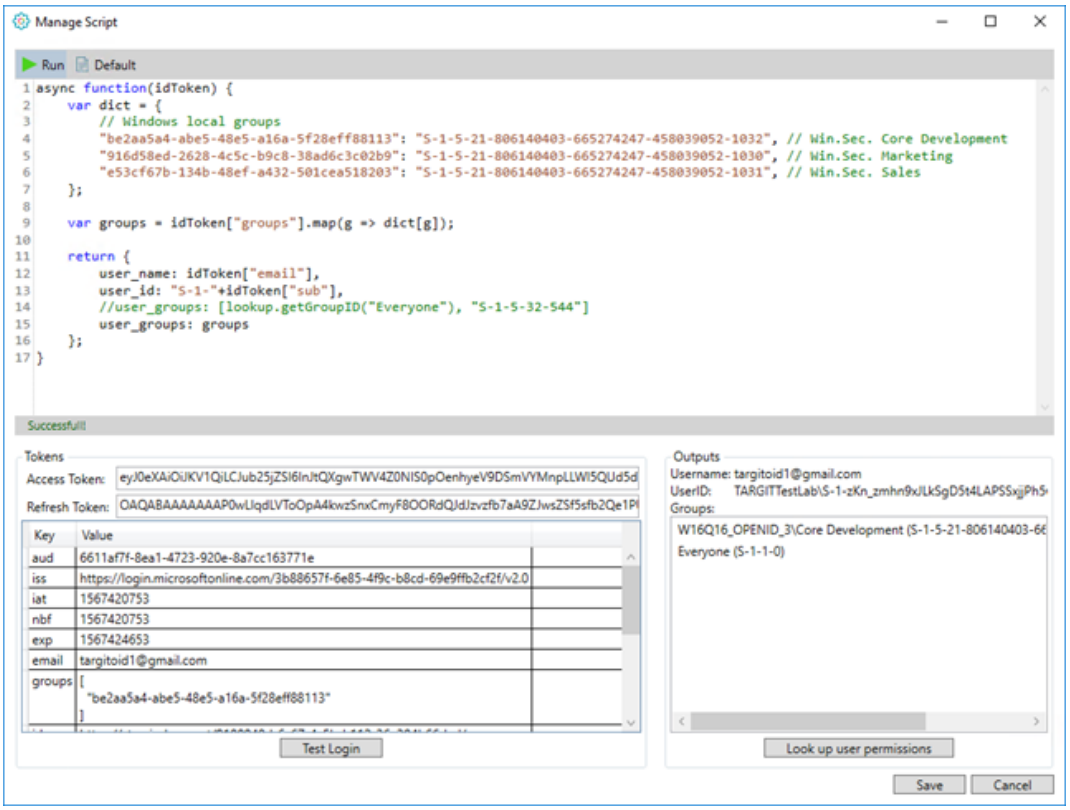
Example on a script related to an Azure OpenID identity provider:



Use the “Test Login” option to login as one of the authenticated users. The information you get in return can be used for filling parts of your script, e.g. a group SID.

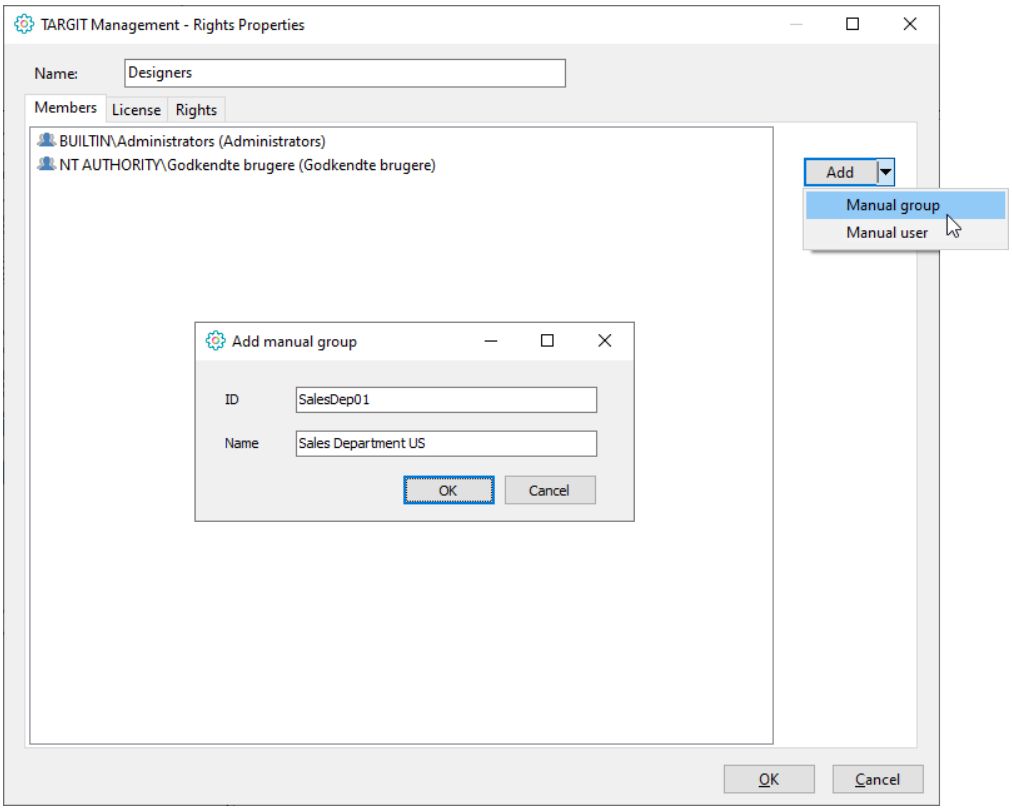
Furthermore, once you have done a Test Login and potentially modified your script, you can then “Run” the script. This will do the mapping and, in *Outputs,* give you additional information about the user, based on the internal AD.

Now that this user has been mapped from an external identity provider group to an internal AD group (and assuming that internal AD groups already have been added to TARGIT roles), you can then *Look up user permissions* to get an overview of the user’s effective permissions with regard to access to documents folders, databases, forced criteria etc.



Alternatively, if your internal AD does not hold the corresponding groups required for mapping to the groups of the identity provider's AD, you can manually add group information in the TARGIT Management client.

E.g., if you have some identity provider users that are members of an identity provider group, and you want that group's members to log on to TARGIT with specific rights and specific roles, you can simply add a new manual group as a member to the Rights and Roles definitions.

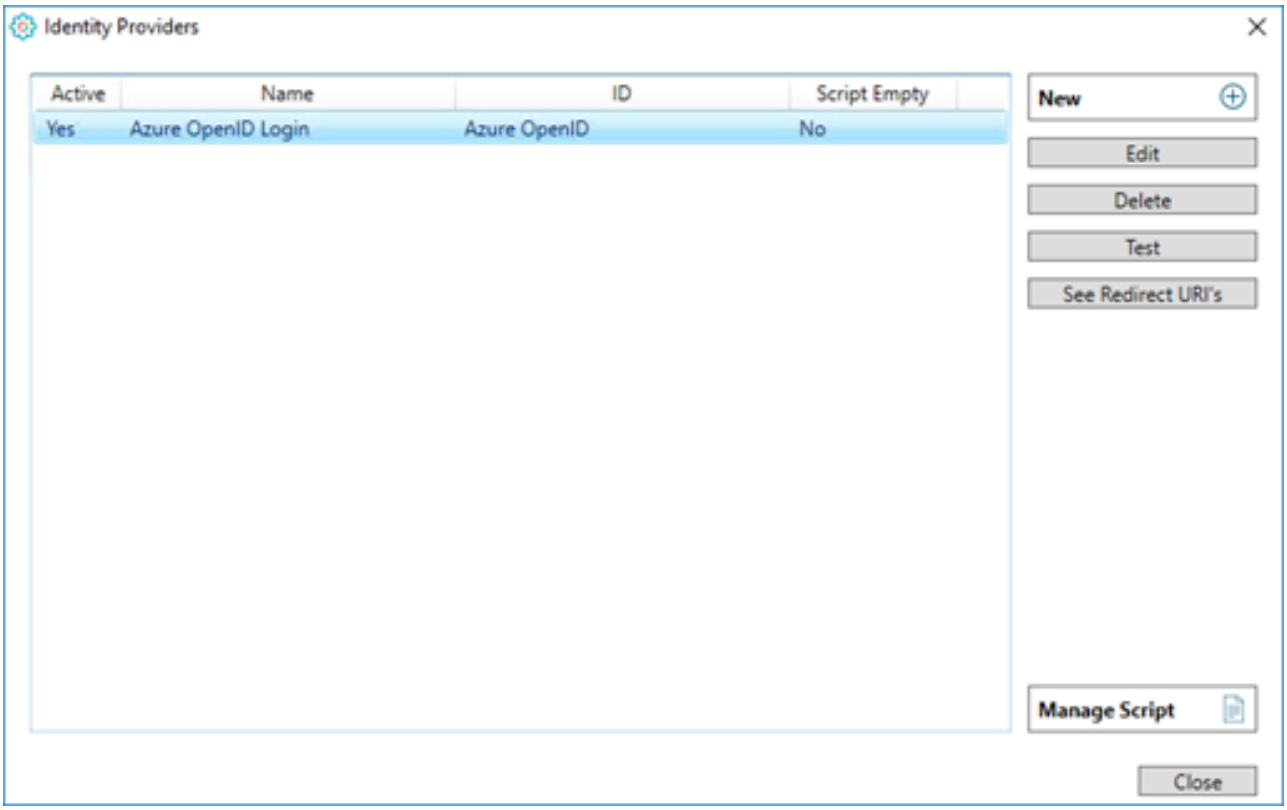


Even if you do have the corresponding groups in your internal AD, it may be easier to use the IDs from the manually created groups, rather than trying to retrieve IDs from your internal AD.

Identity Provider Management

For an existing identity provider, you may:

- Edit the identity provider.
- Delete the identity provider.
- Test the identity provider. (Login with a user already set up on the external identity provider.) A successful test indicates that the TARGIT server and the external identity provider were able to exchange authentication information as expected.
- See Redirect URIs. These are the two redirect URIs (for TARGIT server and Anywhere) that must be fed back to the external identity provider's list of authenticated URIs.



Embedding Provider ID with Anywhere URL

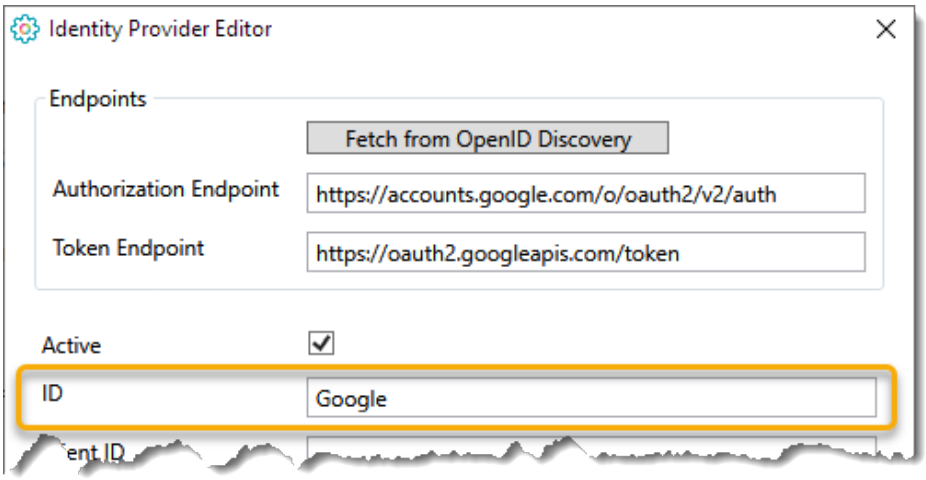
In an OpenID environment, you can prevent Anywhere users from choosing login method at first login. This is done by inserting the Provider ID in the Anywhere URL.

In this example, Google authentication has been used.

https://myTargitServer.com/anywhere/embed.html?**providerID=Google**##SalesforceHome

Please notice that "**providerID**" is case sensitive - "providerId" would not work.

The "**Google**" value is the ID of an Identity Provider that must be set up in the TARGIT Management client:



Appendix – solution description, technical

The solution is based on OpenID Connect, where the user's identity is encoded in a secure JSON Web Token (JWT), called an **ID token**, based on the standard OAuth 2.0 flow.

The **ID token** resembles the concept of an identity card, in a standard JSON Web Token (JWT) format, signed by the **Identity Provider**. An **ID token** has a limited lifetime (e.g. 30 minutes), so a **Refresh token** is also provided that can be used to query for a new **ID token**. The **Refresh Token** will be necessary for e.g. running scheduled jobs, because we "simulate" a user login.

Authentication will take place at the **Identity Provider** in two steps.

First step is to request an **Authorization Code** from the **Identity Provider,** for that the TARGIT client will use a trusted agent (browser) separate from the TARGIT application. The browser (standard system browser) will handle the dialogue that send the End-User to the chosen **Id entity Provider**.

At the **Identity Provider**, the End-User will typically be authenticated by checking if they have a valid session (established by a browser cookie), and in the absence of that, by prompting the user to login. After that the user will typically be asked whether they agree to sign into TARGIT.

The TARGIT client will pass the **Authorization Code** to the TARGIT Server which in step two will do a “back-end” authorization against the **Ide ntity Provider,** and in exchange for the **Authorization Code** receive an **ID token** and the **Refresh token**. The **ID token** will be security validated by TARGIT

Functionality TARGIT Management:

In TARGIT Management there will be added an additional security model “OpenID”.

In the “OpenID” security model you will be able to specify/“add” which **Identity Provider** that you want to trust. For custom **Identity Provider** yo u will be able to define a URL with the address of the custom **Identity Provider** and request parameters in the URI query.

To integrate OpenID into our rights/role-based security model, it should be possible to define certain rules on each right/role that determine if the right/role will be active for a given user. These rules could be something like: if the value of claim x equals y, then this right/role should be active. The administrator can then create several rights/roles that defines what each user can do - based on the contents of their **ID token**.

Functionality TARGIT Client:

When an end-user log into TARGIT, the TARGIT Client should be responsible for launching the browser to initiate the OpenID connect login. When an **Authorization Code** is received, it should be handed to the TARGIT Server where the actual token handling takes place.

Functionality TARGIT Server:

The TARGIT Server will be the only one knowing the client secret, therefore the actual handling of the **ID token** and **Refresh Token** will take place here. Each right and role will be checked to see if any of the OpenID rules are met. After rights and roles have been determined, we will issue our own security token to the **TARGIT Client** to be used internally.

Appendix – example Azure App Registration

In this example, you already have an Azure portal account and your Azure AD is already set up with a number of users and groups.

Azure:

a. Log on to your Azure portal, e.g. http://portal.azure.com
b. Go to **Azure Active Directory**
c. Go to **App Registrations**
    i. Add a **New registration**

        i. Apply a proper name (can be changed later) and **Register**

        ii. Copy and store the **Application (client) ID**. You will need this later.

a. Go to **Certificates and Secrets**
    i. Add a **New client secret.** Copy and store the client secret – you will not be able to retrieve it later.
    ii. Go to **API permissions**
    iii. **Add a permission:**

        i. **Microsoft Graph**

        ii. **Delegated permission**

        iii. Checkmark **Group.Read.All**

        iv. Click the **Add permissions** button at bottom.

a. Go to **Manifest**
    i. Change **groupMembershipClaims** value to "**All**" (notice double quotation marks):

        i. "groupMembershipClaims": "All",

a. Go to **Overview, Endpoints**
    i. Copy to clipboard: **OpenID Connect metadata document**

TARGIT:

a. Create a **New Identity Provider**
b. Click "**Fetch from OpenID Discoverer**". This should automatically insert the copied URL.
    i. Click "Fetch". This should automatically fill in

        i. Authorization Endpoint

        ii. Token Endpoint

        iii. Scope

a. Enable "**Active"**
b. Provide an **ID** of your choice, e.g. "CompanyAzureID"
c. Paste in the previously stored **Client ID** and **Client Secret**.
d. Setup Styling **Icon** and **Name** of your choice.
e. **Save** the Identity Provider setup.
f. In the **Identity Provider list**, select the recently created provider.
    i. Click "**See Redirect URIs**"

Azure:

a. Go to **Authentication**
    i. Paste in the two **Redirect URIs** as two separate entries – **Type: Web**.