



Polaris and OAuth 2.0 with OpenID Connect Integration Guide

7.3

© 2023

Legal Notices

© Innovative (Part of Clarivate) and/or its affiliates. All rights reserved. All trademarks shown are the property of their respective owners.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

The software and related documentation are provided under an agreement containing restrictions on use and disclosure and are protected by intellectual property laws.

Except as expressly permitted in your agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of the software, unless required by law for interoperability, is prohibited.

Contents

Introduction	1
Minimum Requirements	2
Polaris System Administration (Web-Based) with Active Directory and AD FS Authentication	2
Polaris System Administration (Web-Based) with Azure AD Authentication	2
Leap with Active Directory and AD FS Authentication	3
Leap with Azure AD Authentication	3
Configuring Active Directory with AD FS	4
Install Active Directory Federation Services	5
Configure Active Directory Federation Services	14
Verify Active Directory Federation Services Is Running	23
Verify that OAuth 2.0 is Enabled	26
Create an Application Group	28
Configure the AD FS Web Application: Claims and Permitted Scopes	34
Enable CORS on AD FS To Accept Requests from Polaris APIs	40
Set Up Web Services and Applications	40
Enable Session Storage for LeapWebApp	56
Customize the AD FS Pages	57
Change the Access Token Lifetime	58
Bind a New SSL Certificate	58
Troubleshoot	58
Configuring Azure AD	60
Register LeapWebApp with Azure AD	60
Create Client Credentials	64
Add Authentication Redirect URIs	66

Expose the Polaris.ApplicationServices API	69
Configure an ID Token	71
Set Up Users and Groups	73
Control Access to LeapWebApp Using Azure AD	75
Set Up Web Services and Applications	77
Add a URL Rewrite Rule for LeapWebApp	100
Sample Rewrite Rule Text	101
Additional URL Rewrite Resources	101

Introduction

Polaris System Administration (web-based) requires OAuth 2.0 with OpenID Connect and Proof Key for Code Exchange (PKCE). As of version 7.2, Leap supports using OAuth 2.0 with OpenID.

Staff authentication for Polaris System Administration (web-based) is handled by either Active Directory and Active Directory Federation Services (AD FS), or by Azure Active Directory (Azure AD).

Staff authentication for Leap is handled by either Active Directory and AD FS, or by Azure AD.

Review the [Minimum Requirements](#) for the configuration you plan to use:

- Polaris System Administration (web-based) with Active Directory and AD FS Authentication
- Polaris System Administration (web-based) with Azure AD authentication
- Leap with Active Directory and AD FS authentication
- Leap with Azure AD authentication

Then, continue to one of the following configuration procedures:

- [Configuring Active Directory with AD FS](#)
- [Configuring Azure AD](#)

Minimum Requirements

This section discusses minimum requirements for the following configurations:

- [Polaris System Administration \(Web-Based\) with Active Directory and AD FS Authentication](#)
- [Polaris System Administration \(Web-Based\) with Azure AD Authentication](#)
- [Leap with Active Directory and AD FS Authentication](#)
- [Leap with Azure AD Authentication](#)

Polaris System Administration (Web-Based) with Active Directory and AD FS Authentication

To use Polaris System Administration (web-based) with Active Directory and AD FS authentication, you must have the following installed:

- Windows Server 2019 Standard
 - Polaris requires OAuth 2.0 with PKCE support
 - AD FS on Windows Server 2019 supports PKCE
- Active Directory Domain Services
- SSL Certificate
 - Publicly trusted CA signed certificate
- Polaris 7.1 or later
- Polaris 7.1 or later PolarisAdmin

Polaris System Administration (Web-Based) with Azure AD Authentication

To use Polaris System Administration (web-based) with Azure AD authentication, you must have:

- Access to Microsoft's Azure AD services
- The Polaris 7.3 PolarisAdmin installed

Leap with Active Directory and AD FS Authentication

To use Leap with Active Directory and AD FS authentication, you must have the following installed:

- Windows Server 2019 Standard
 - Polaris requires OAuth 2.0 with PKCE support
 - AD FS on Windows Server 2019 supports PKCE
 - Active Directory Domain Services
 - SSL Certificate
 - Publicly trusted CA signed certificate
 - Polaris 7.2 or later
 - Polaris 7.2 or later LeapWebApp
-

Leap with Azure AD Authentication

To use Leap with Azure AD authentication, you must have:

- Access to Microsoft's Azure AD services
- The Polaris 7.3 LeapWebApp installed

Configuring Active Directory with AD FS

Important:

The mechanism used to connect an Active Directory user to a Polaris user is the user principal name (UPN) in the format of an email address. For example, user@mydomain.com. During the account verification process, we request the UPN claim from Active Directory. This must return a UPN in the name@domain format. The Polaris.AdminServices (API) can then use that information to map the AD user to a Polaris user.

To configure Polaris OAuth support with Active Directory and AD FS, perform the following tasks:

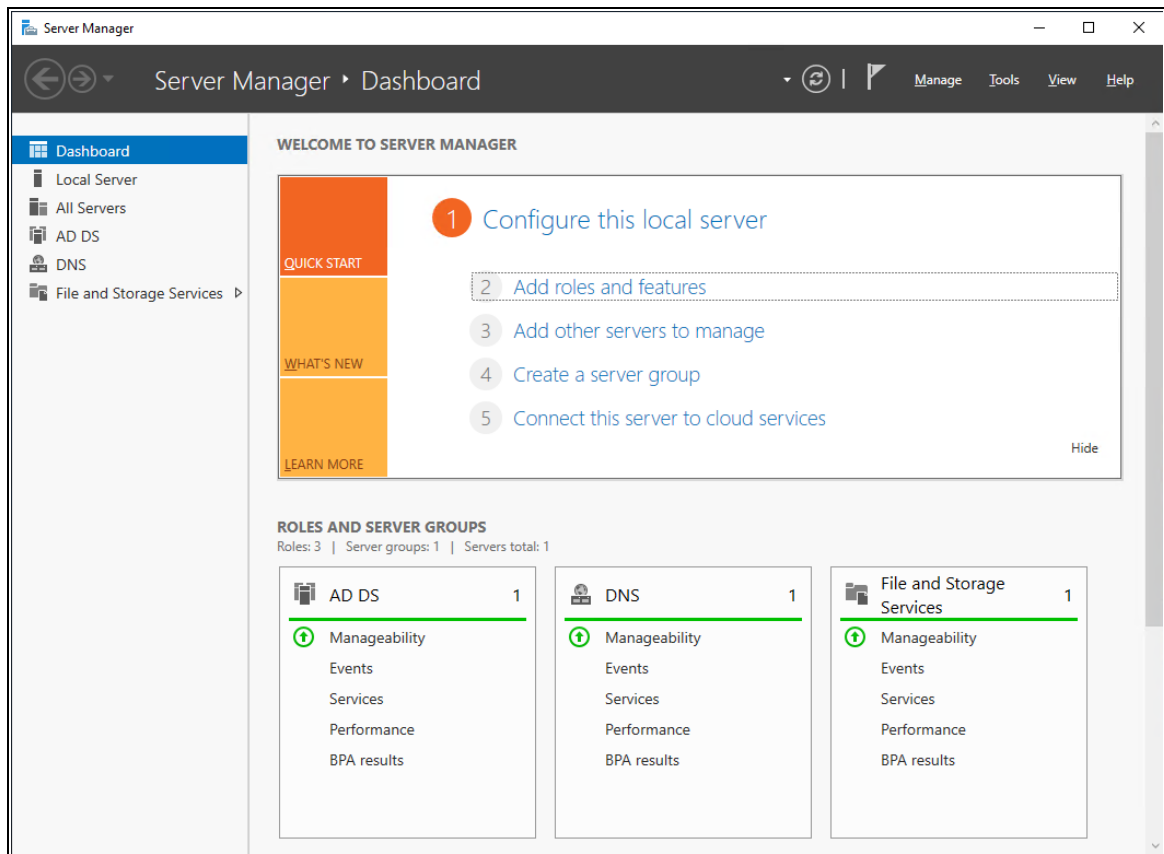
1. [Install Active Directory Federation Services.](#)
2. [Configure Active Directory Federation Services.](#)
3. [Verify that Active Directory Federation Services is running.](#)
4. [Verify that OAuth 2.0 is Enabled.](#)
5. [Create an Application Group for Polaris LeapWebApp.](#)
6. [Configure the AD FS Web Application: Claims and Permitted Scopes.](#)
7. [Enable CORS on AD FS to accept requests from Polaris APIs.](#)
8. [Set up web services and applications.](#)
9. [Enable session storage for LeapWebApp.](#)
10. [Customize the AD FS pages.](#)
11. [Change the access token lifetime.](#)
12. [Bind a new SSL certificate.](#)
13. [Troubleshoot.](#)

After you complete these tasks, [Add a URL rewrite rule for LeapWebApp.](#)

Install Active Directory Federation Services

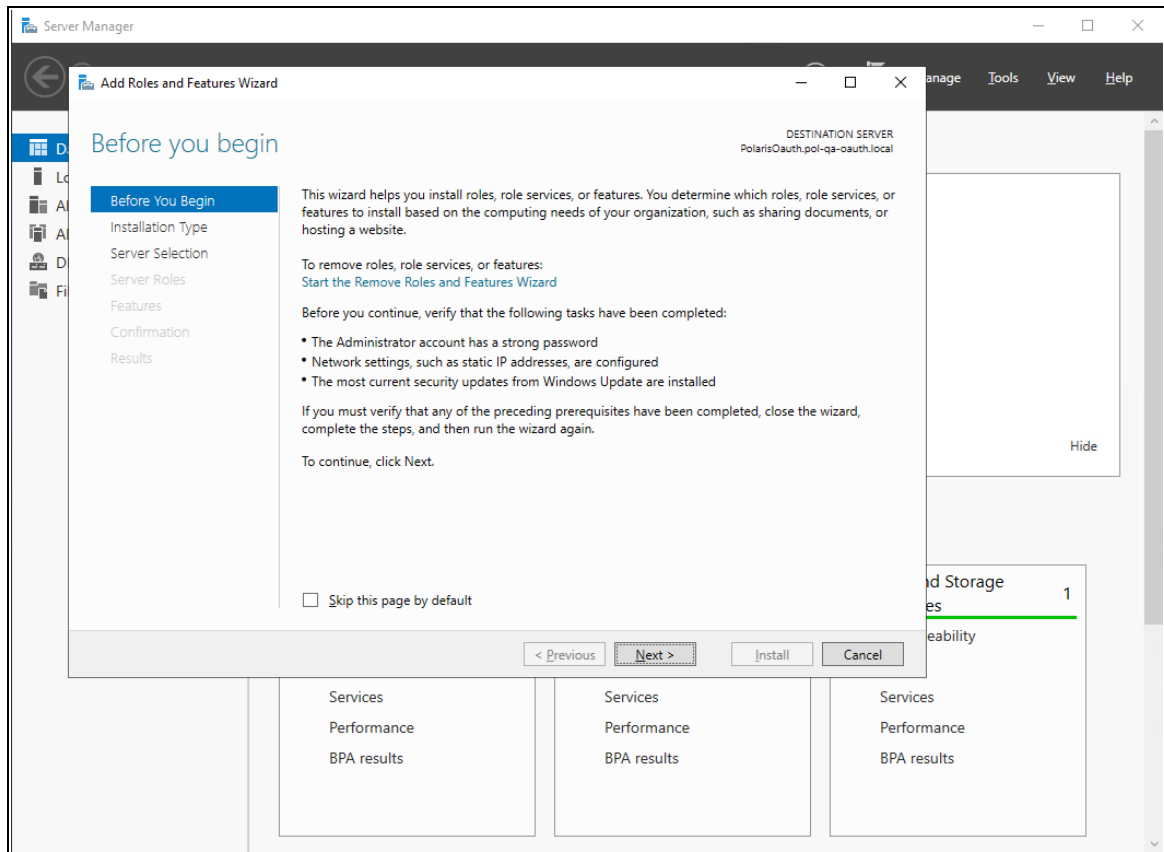
To install AD FS

1. Sign in to Windows Server 2019 with administrative privileges.
2. Start the Server Manager desktop application.

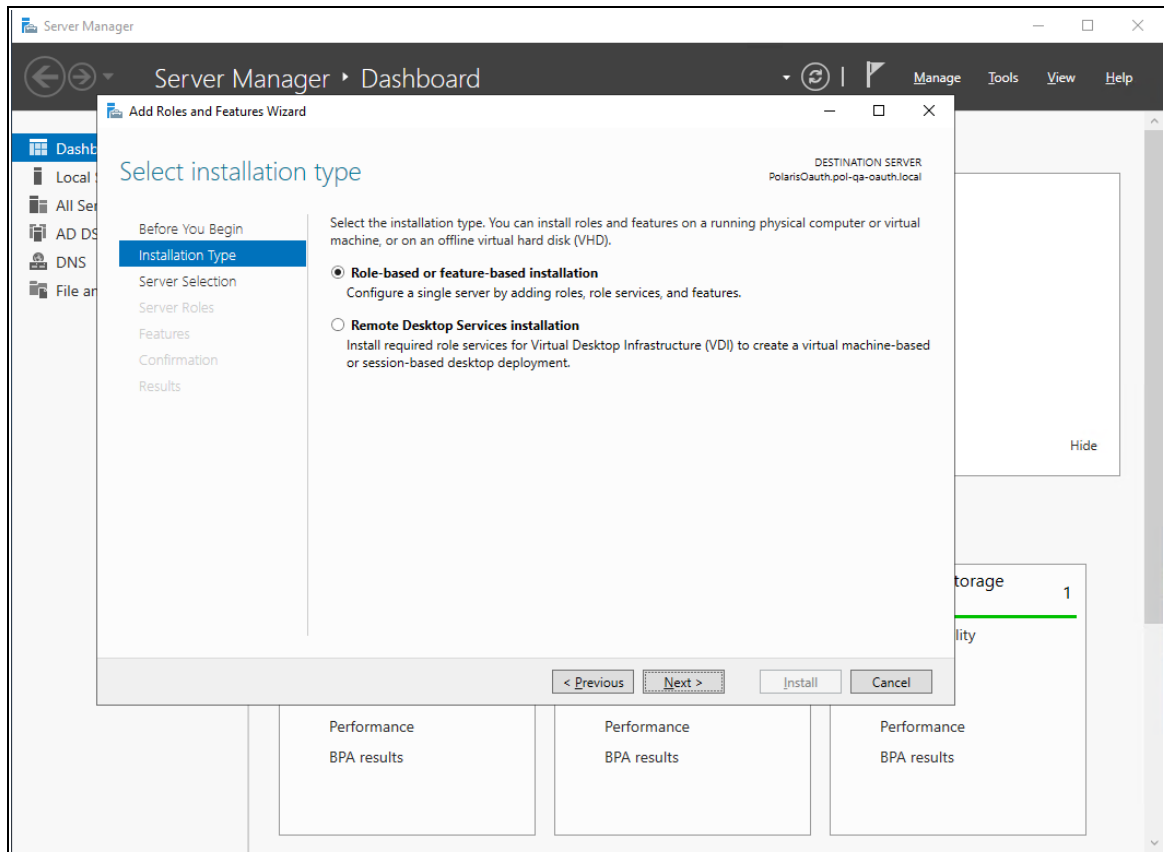


3. On the **Server Manager Dashboard** view, select **Add roles and features**.
The Add Roles and Features Wizard opens.

Polaris and OAuth 2.0 with OpenID Connect Integration Guide (7.3)

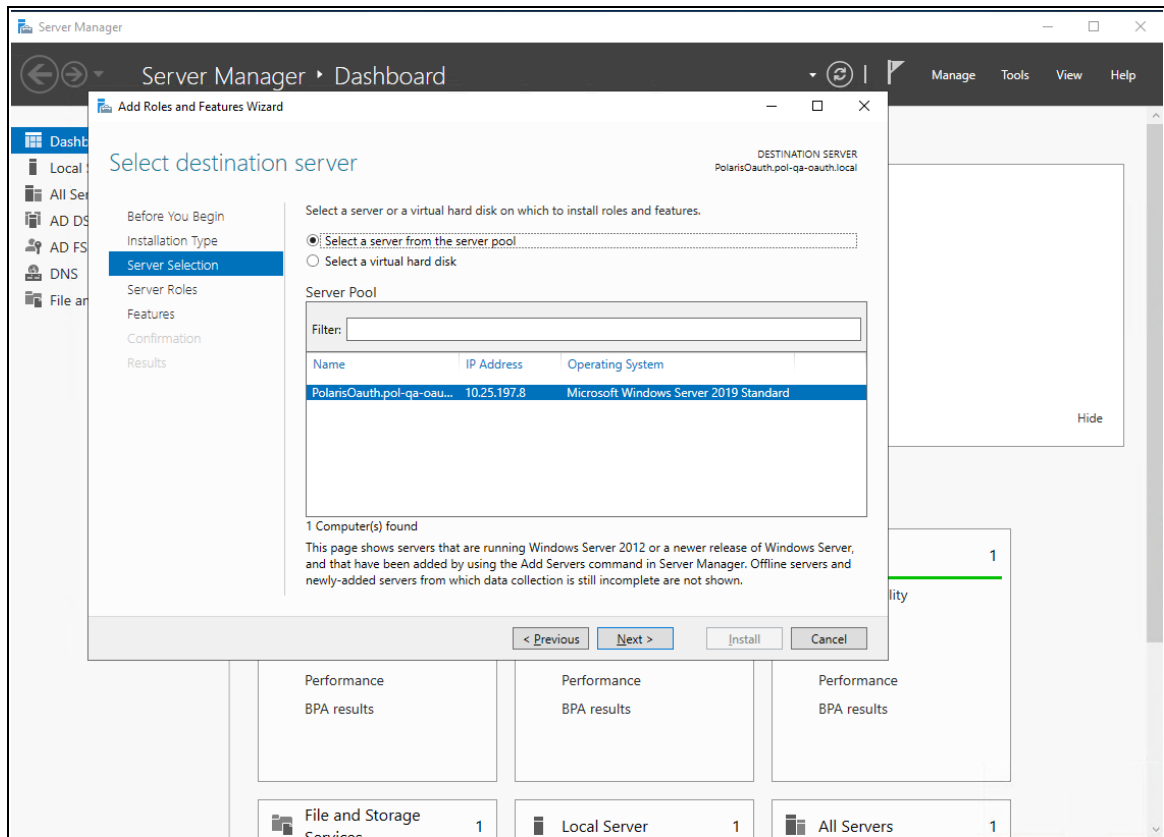


4. On the **Before You Begin** tab, select **Next**.

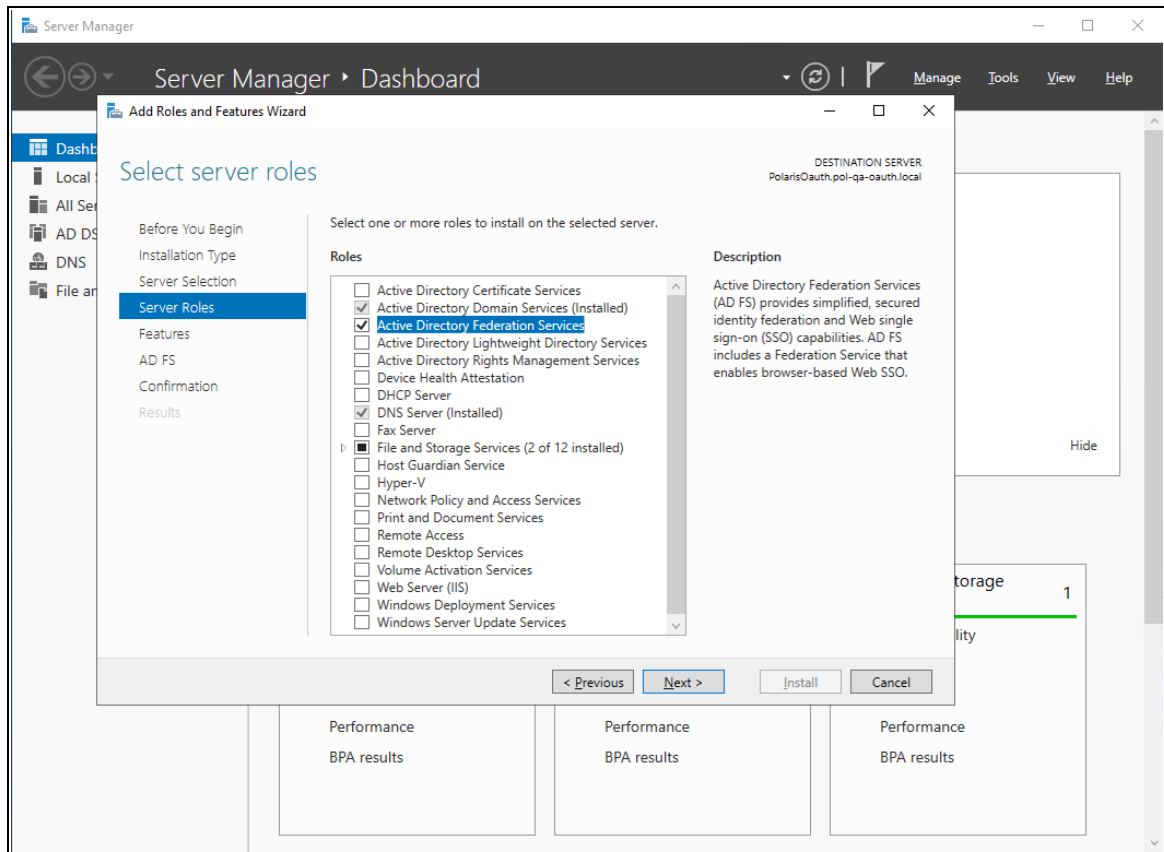


5. On the **Installation Type** tab, select **Role-based or feature-based installation**, and then select **Next**.

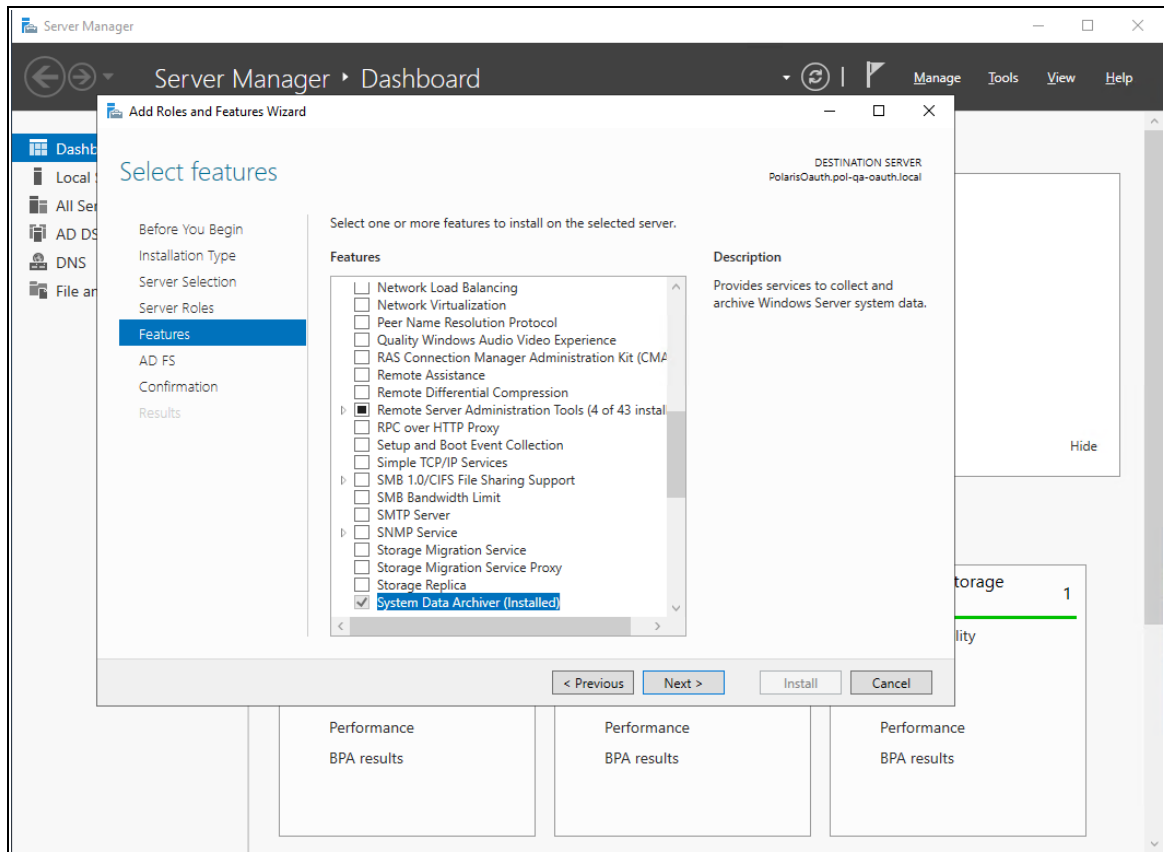
Polaris and OAuth 2.0 with OpenID Connect Integration Guide (7.3)



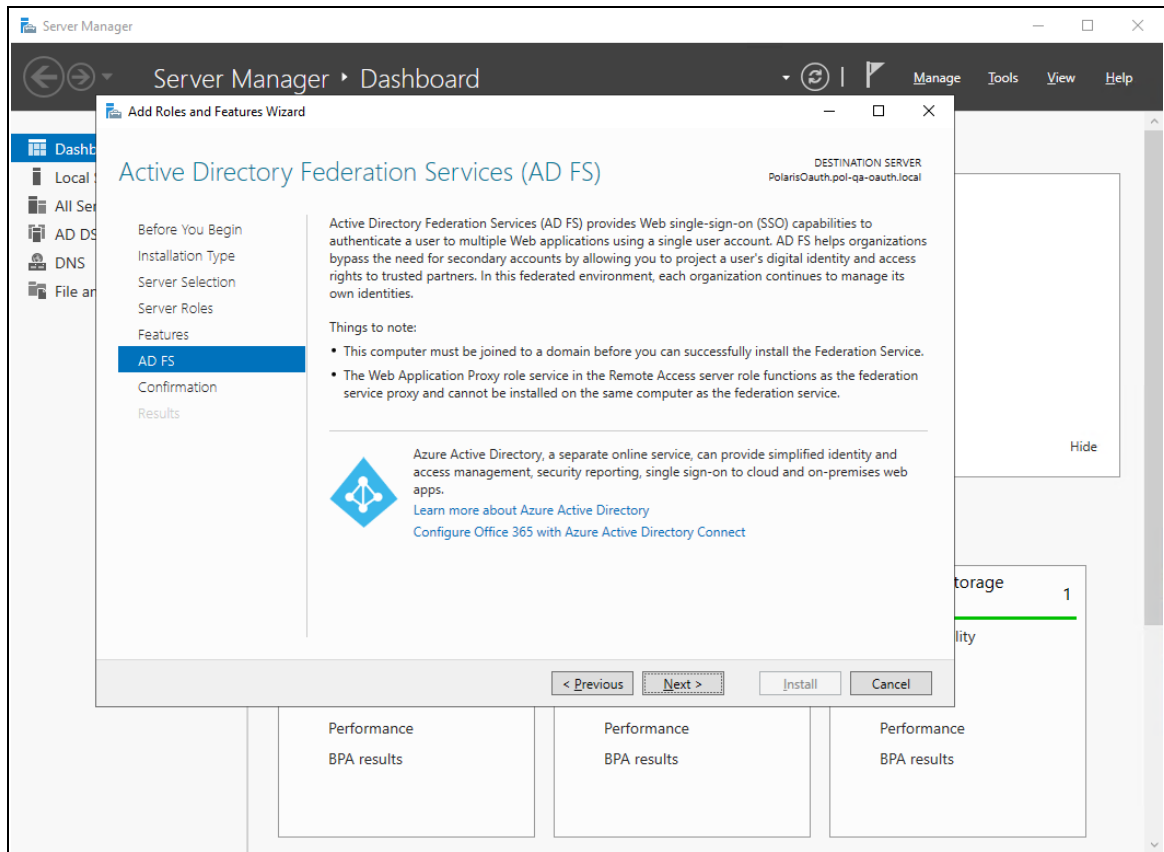
6. On the **Server Selection** tab, select the server, and then select **Next**.



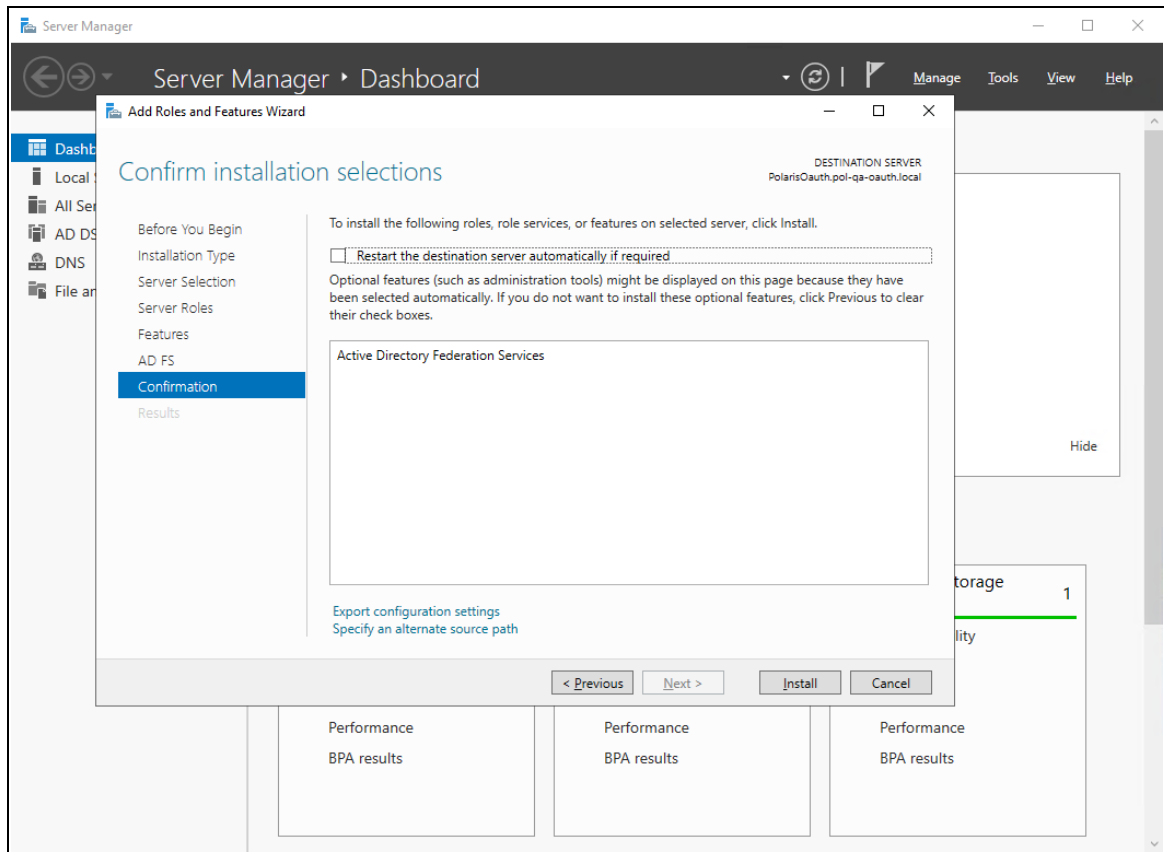
7. On the **Server Roles** tab, do the following:
 - a. Verify that **Active Directory Domain Services** are installed.
 - b. Select the **Active Directory Federation Services** role.
 - c. Select **Next**.



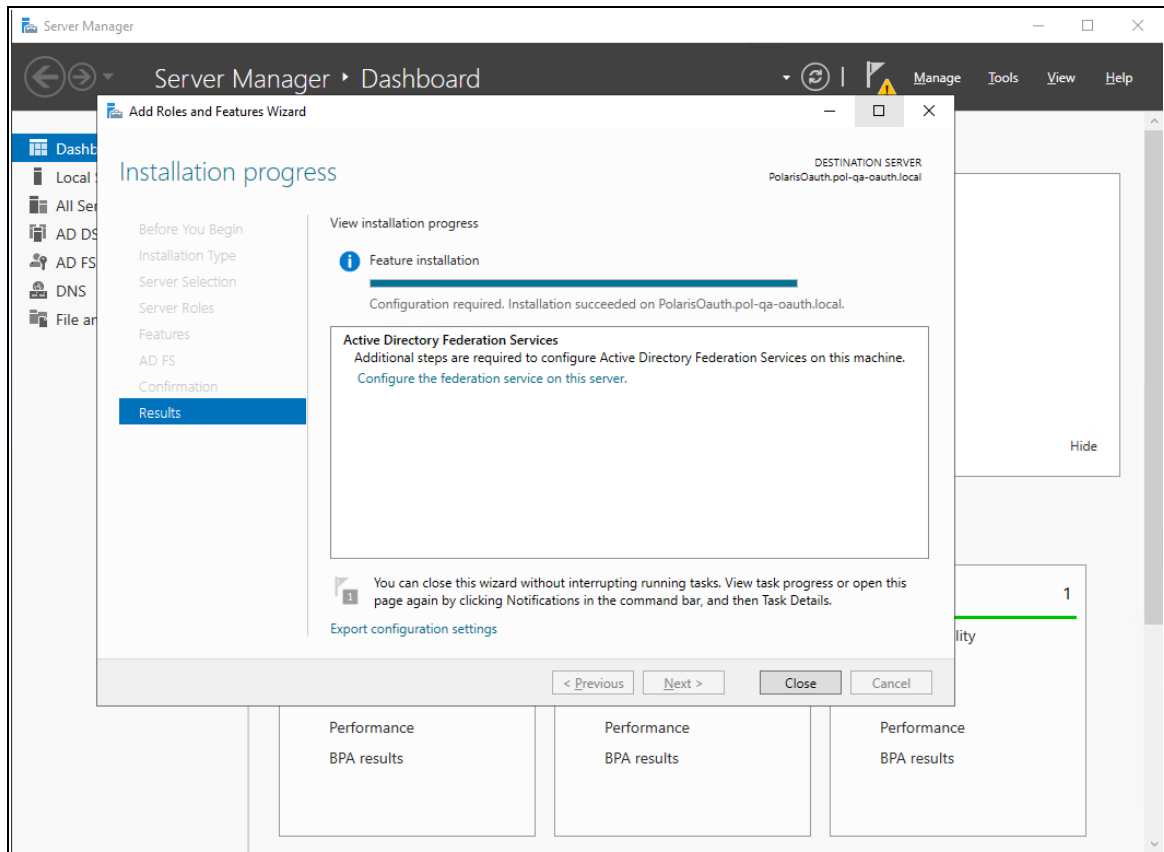
8. On the **Features** tab, select **Next**.



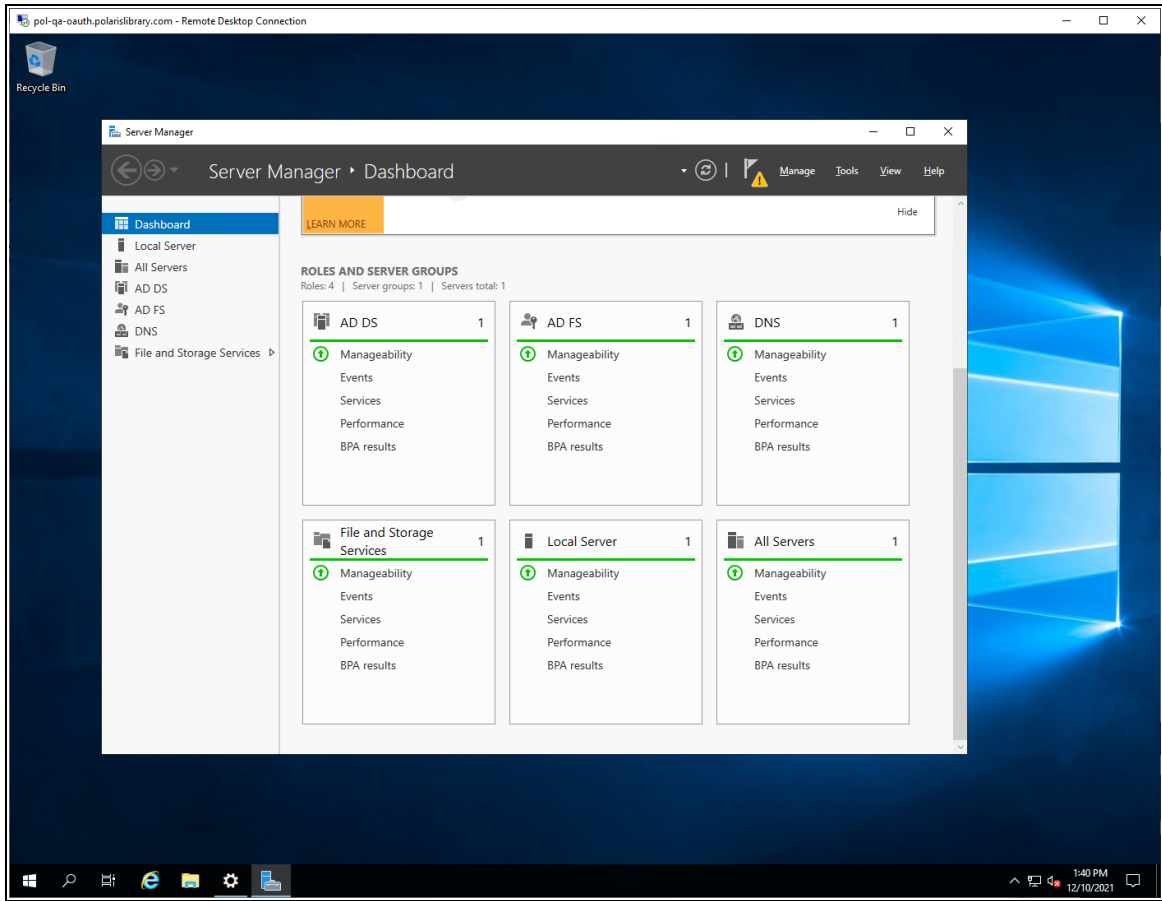
9. On the **AD FS** tab, read the Active Directory Federation Services (AD FS) information, and then select **Next**.



10. On the **Confirmation** tab, confirm your selections, and then select **Install**.



11. On the **Results** tab, select **Close** when the installation is complete.



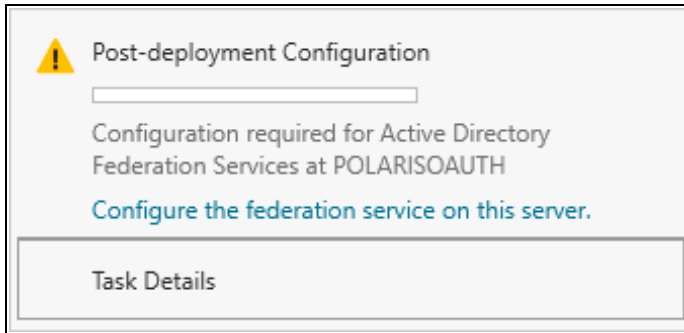
12. On the Server Manager dashboard, verify that AD FS is an installed role.
13. Restart the server.

Configure Active Directory Federation Services

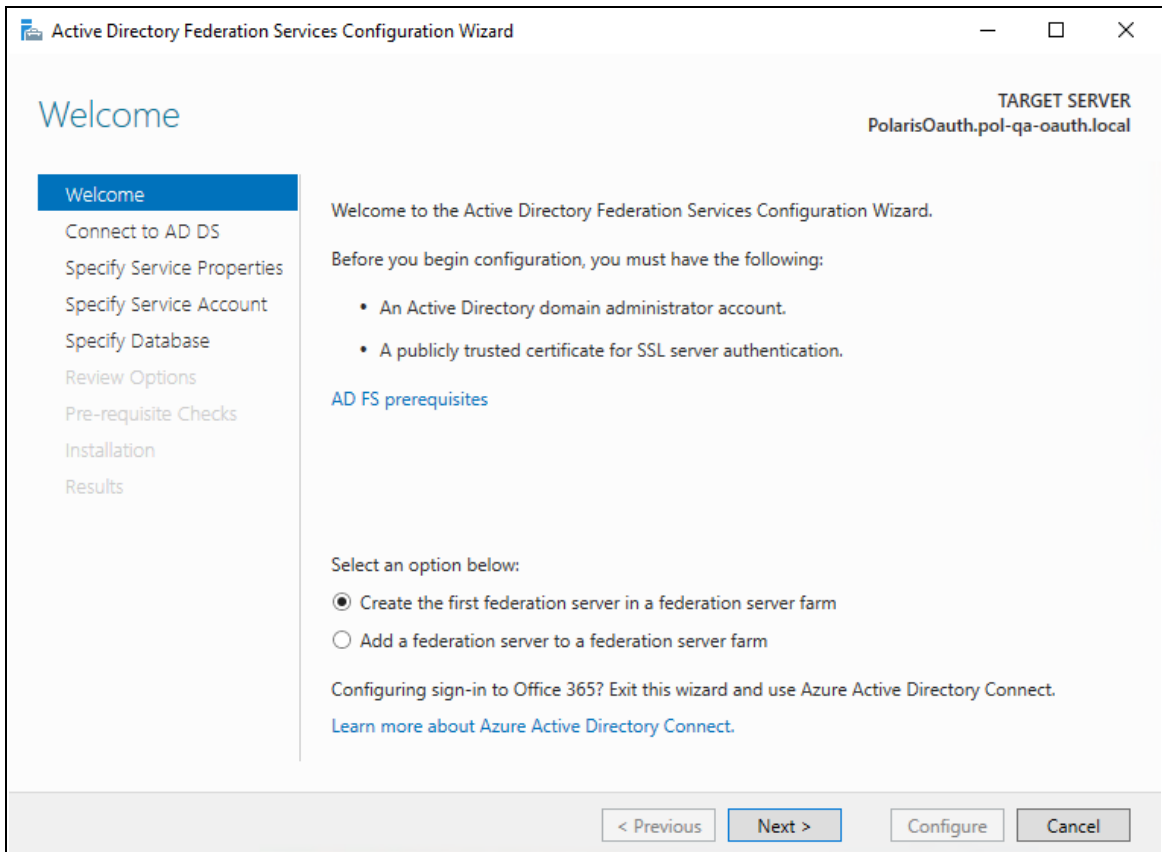
To configure Active Directory Federation Services

1. Start the Server Manager desktop application.

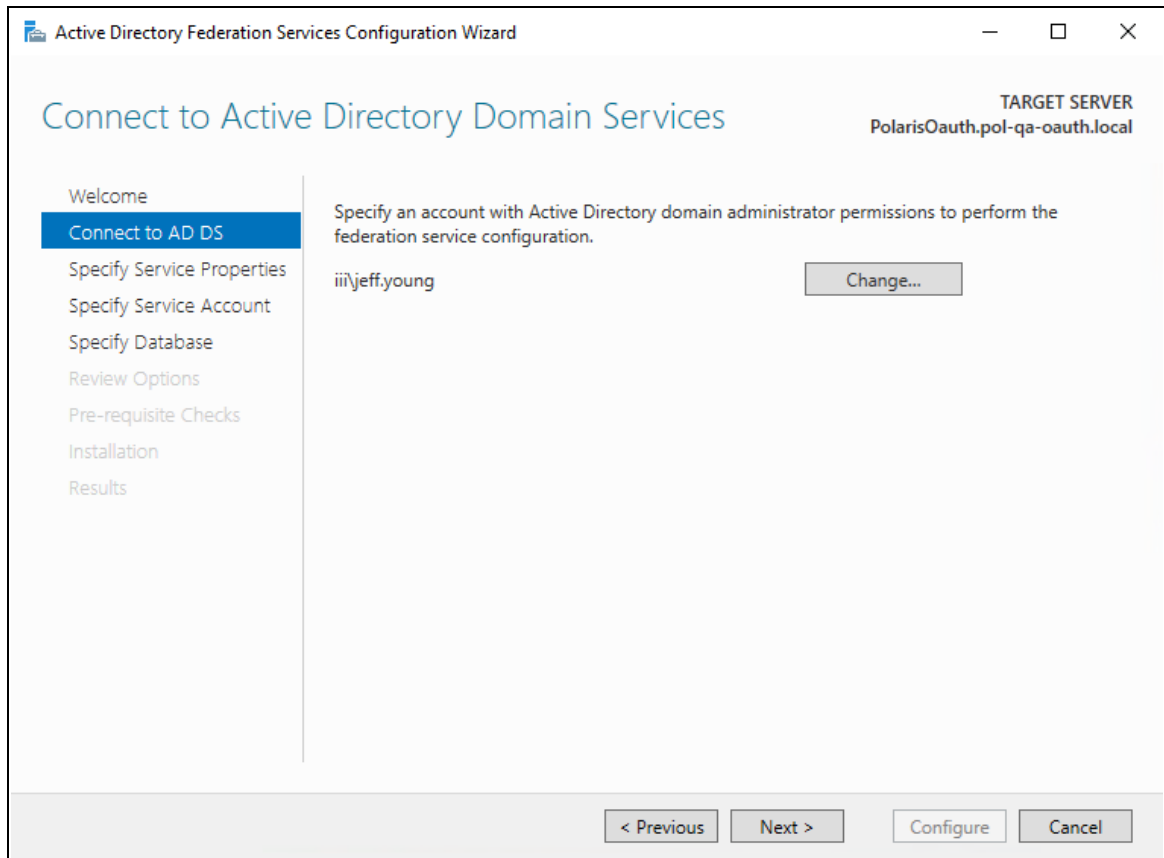
The system generates a configuration notification.



2. Open the notification, and select **Configure the federation service on this server**.
The Active Directory Federation Services Configuration wizard opens.



3. On the Welcome tab, select **Next**.



4. Select **Change**, and provide an administrator account. Then select **Next**.

The screenshot shows the 'Active Directory Federation Services Configuration Wizard' window. The title bar includes the Windows icon, the text 'Active Directory Federation Services Configuration Wizard', and standard window controls (minimize, maximize, close). The main window has a light blue header with the title 'Specify Service Properties' on the left and 'TARGET SERVER PolarisOAuth.pol-qa-oauth.local' on the right. A left-hand navigation pane contains a list of steps: 'Welcome', 'Connect to AD DS', 'Specify Service Properties' (highlighted with a blue bar), 'Specify Service Account', 'Specify Database', 'Review Options', 'Pre-requisite Checks', 'Installation', and 'Results'. The main content area is enclosed in a dashed border and contains three configuration fields: 'SSL Certificate:' with a dropdown menu showing '*.polarislibrary.com' and an 'Import...' button; 'Federation Service Name:' with a dropdown menu showing 'dev-fs.polarislibrary.com' and an example 'fs.contoso.com'; and 'Federation Service Display Name:' with a text box containing 'Polaris R&D Federation Service' and a note 'Users will see the display name at sign in.' with an example 'Contoso Corporation'. At the bottom of the window, there are four buttons: '< Previous', 'Next >', 'Configure', and 'Cancel'.

5. If not already installed on the server, select **Import** to install an SSL certificate.

The screenshot shows the 'Active Directory Federation Services Configuration Wizard' window. The title bar includes the Windows icon, the text 'Active Directory Federation Services Configuration Wizard', and standard window controls. The main window has a light blue header with the title 'Specify Service Properties' on the left and 'TARGET SERVER PolarisOAuth.pol-qa-oauth.local' on the right. A left-hand navigation pane contains a list of steps: 'Welcome', 'Connect to AD DS', 'Specify Service Properties' (highlighted with a blue bar), 'Specify Service Account', 'Specify Database', 'Review Options', 'Pre-requisite Checks', 'Installation', and 'Results'. The main content area is enclosed in a dashed border and contains three configuration fields: 'SSL Certificate:' with a dropdown menu showing '*.polarislibrary.com' and an 'Import...' button; 'Federation Service Name:' with a dropdown menu showing 'dev-fs.polarislibrary.com' and an example 'Example: fs.contoso.com'; and 'Federation Service Display Name:' with a text box containing 'Polaris R&D Federation Service' and a note 'Users will see the display name at sign in.' with an example 'Example: Contoso Corporation'. At the bottom of the window, there are four buttons: '< Previous', 'Next >' (highlighted with a blue border), 'Configure', and 'Cancel'.

6. Enter the following, and then select **Next**:

- Federation Service Name
- Federation Service Display Name

The screenshot shows the 'Active Directory Federation Services Configuration Wizard' window. The title bar includes the Windows logo, the text 'Active Directory Federation Services Configuration Wizard', and standard window controls (minimize, maximize, close). The main window has a title 'Specify Service Account' in blue text. In the top right corner, it says 'TARGET SERVER' followed by 'PolarisOAuth.pol-qa-oauth.local'. A yellow warning banner at the top states: 'Group Managed Service Accounts are not available because the KDS Root Key has not been set. Use the foll... Show more X'. On the left is a navigation pane with the following items: 'Welcome', 'Connect to AD DS', 'Specify Service Properties', 'Specify Service Account' (highlighted in blue), 'Specify Database', 'Review Options', 'Pre-requisite Checks', 'Installation', and 'Results'. The main area contains the text 'Specify a domain user account or group Managed Service Account.' followed by two radio button options. The first option, 'Create a Group Managed Service Account', is unselected. The second option, 'Use an existing domain user account or group Managed Service Account', is selected. Below these options are three fields: 'Account Name' with the value 'POL-QA-OAUTH\', 'Account Name' with the value 'POL-QA-OAUTH\Po...' and a 'Clear' button, and 'Account Password' with a masked password '.....'. To the right of the 'Account Name' field is a 'Select...' button. At the bottom of the window are four buttons: '< Previous', 'Next >', 'Configure', and 'Cancel'.

Active Directory Federation Services Configuration Wizard

Specify Service Account

TARGET SERVER
PolarisOAuth.pol-qa-oauth.local

Group Managed Service Accounts are not available because the KDS Root Key has not been set. Use the foll... Show more X

Welcome
Connect to AD DS
Specify Service Properties
Specify Service Account
Specify Database
Review Options
Pre-requisite Checks
Installation
Results

Specify a domain user account or group Managed Service Account.

☐ Create a Group Managed Service Account

Account Name: POL-QA-OAUTH\

☒ Use an existing domain user account or group Managed Service Account

Account Name: POL-QA-OAUTH\Po... Clear Select...

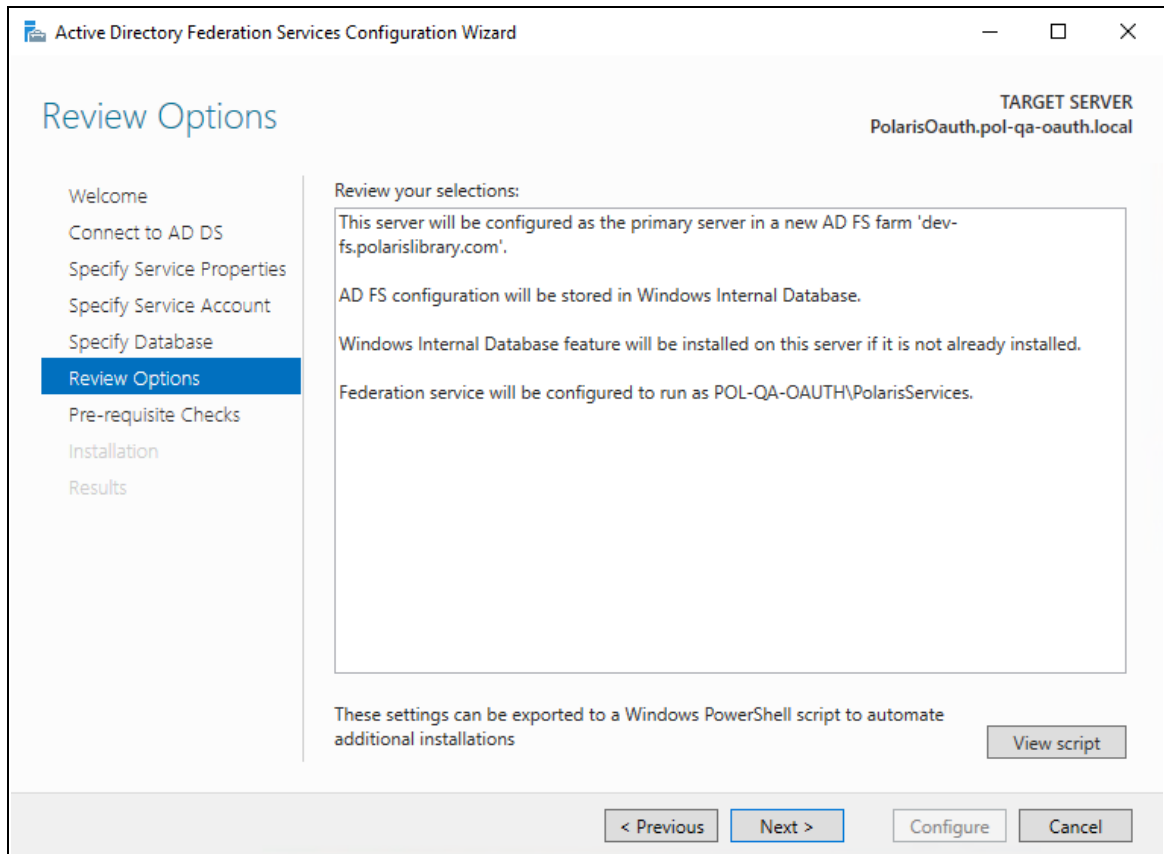
Account Password:

< Previous Next > Configure Cancel

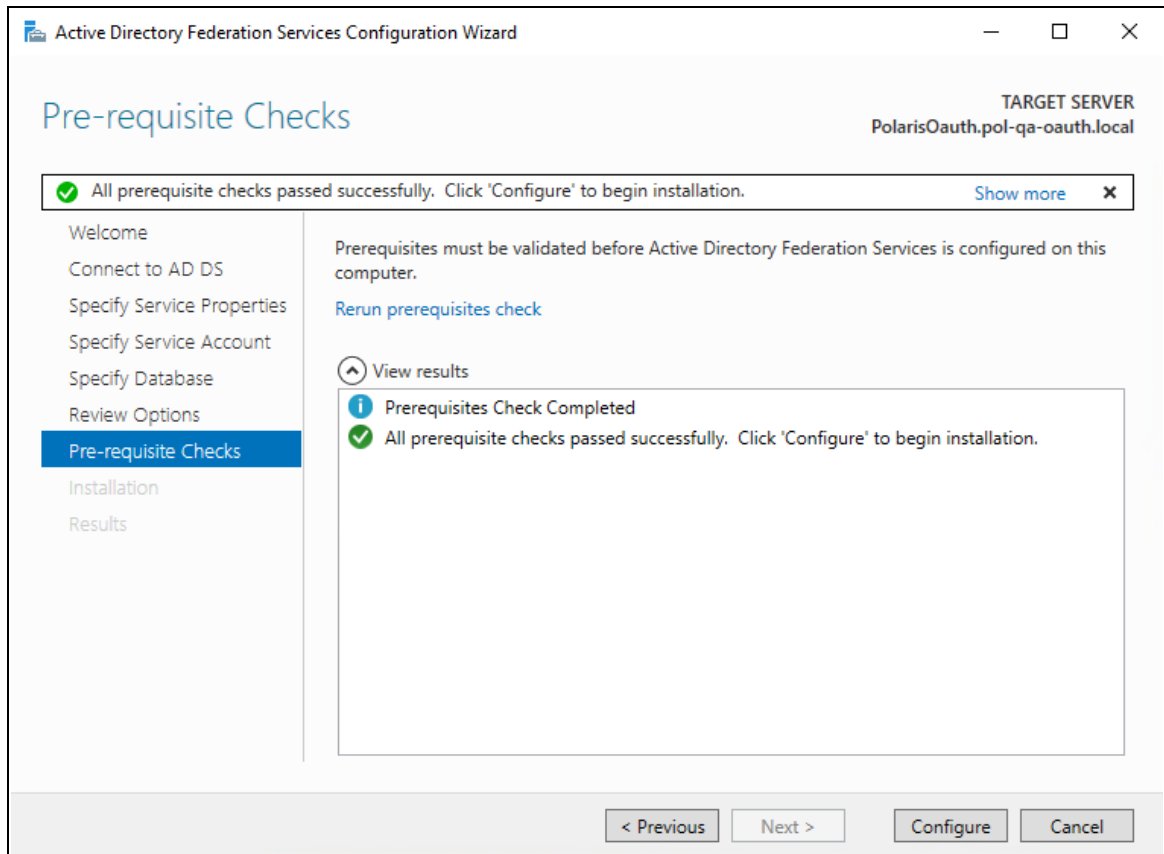
7. Specify a service account, and then select **Next**.

The screenshot shows the 'Active Directory Federation Services Configuration Wizard' window. The title bar includes the Windows icon, the text 'Active Directory Federation Services Configuration Wizard', and standard window controls (minimize, maximize, close). The main window has a light blue header area with the title 'Specify Configuration Database' on the left and 'TARGET SERVER PolarisOAuth.pol-qa-oauth.local' on the right. A vertical sidebar on the left contains a list of steps: 'Welcome', 'Connect to AD DS', 'Specify Service Properties', 'Specify Service Account', 'Specify Database' (highlighted in blue), 'Review Options', 'Pre-requisite Checks', 'Installation', and 'Results'. The main content area has the heading 'Specify a database to store the Active Directory Federation Service configuration data.' followed by two radio button options: 'Create a database on this server using Windows Internal Database.' (which is selected) and 'Specify the location of a SQL Server database.' Below these are two text input fields: 'Database Host Name:' and 'Database Instance:'. A note below the second field states 'To use the default instance, leave this field blank.' At the bottom of the window is a grey bar containing four buttons: '< Previous', 'Next >', 'Configure', and 'Cancel'.

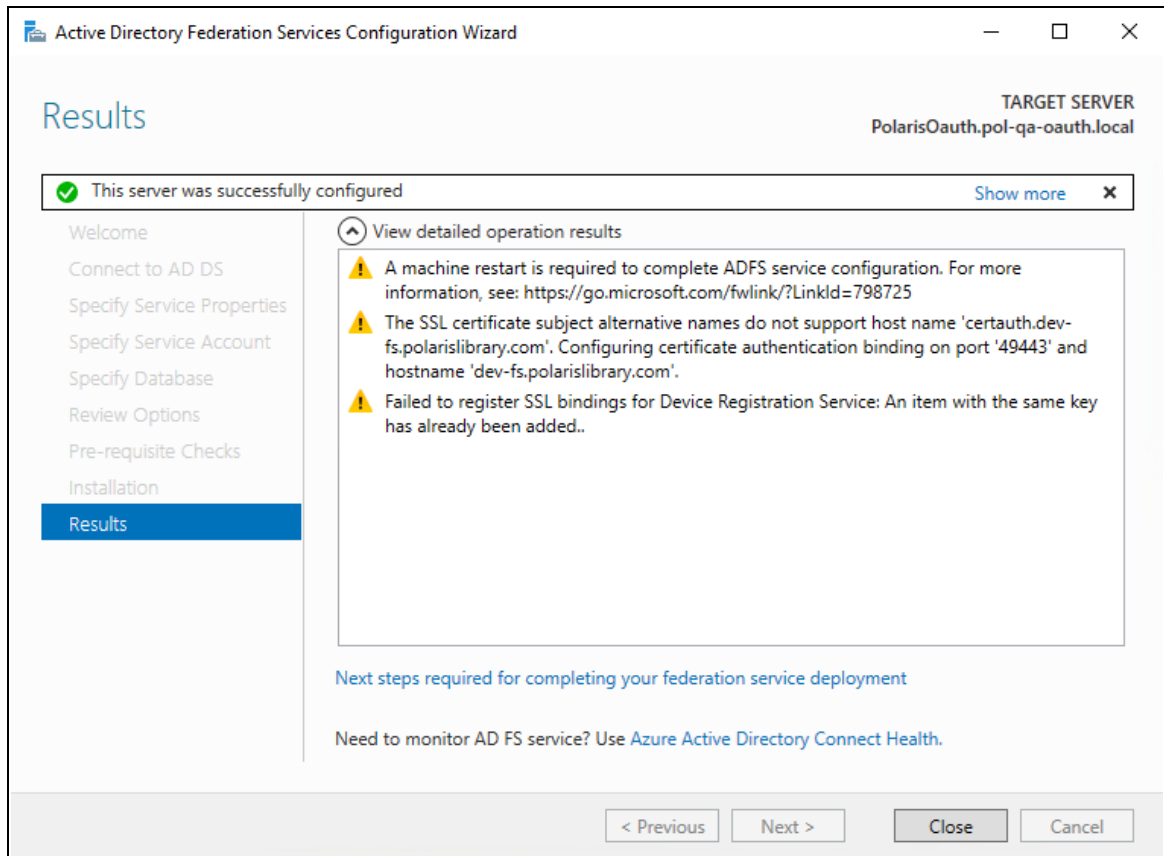
8. Specify the location of the AD FS configuration database, and then select **Next**.
For simple scenarios, creating the local database is acceptable.



9. Review your selections, and then select **Next**.



10. After you complete all pre-requisite checks, select **Configure**.

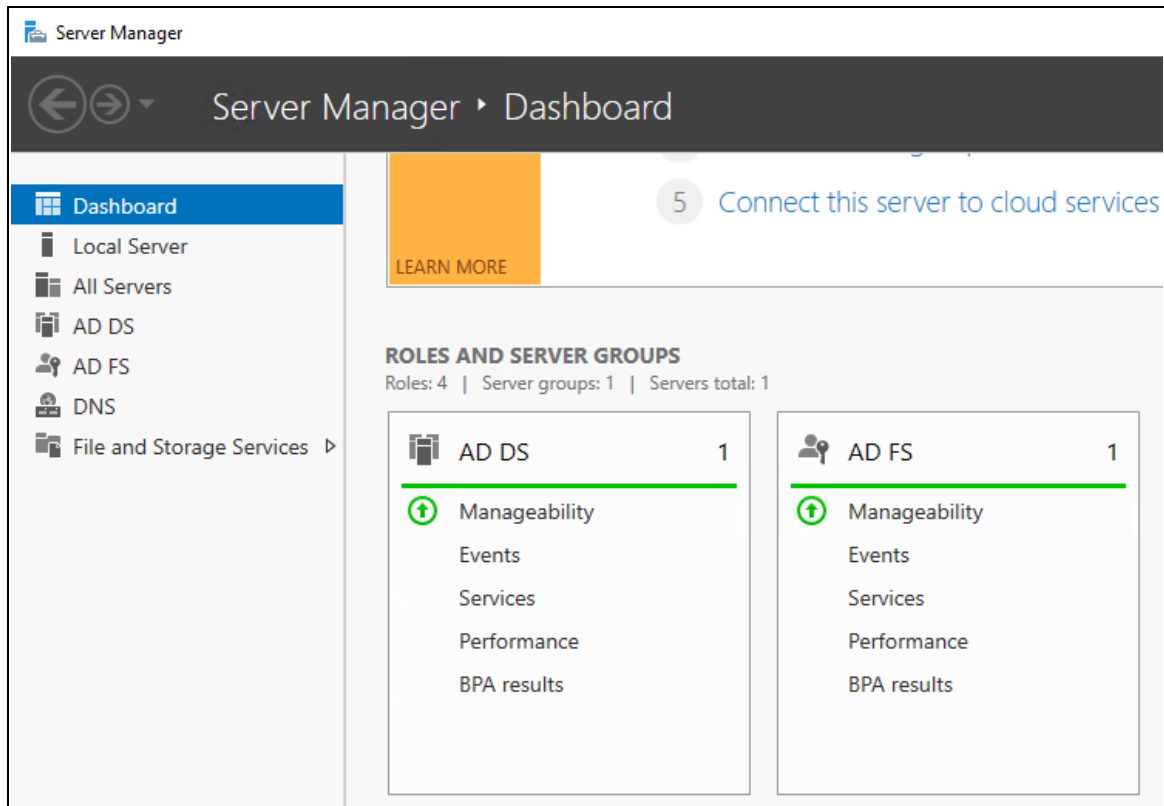


11. When the configuration wizard has completed successfully, select **Close**, and then restart the server.

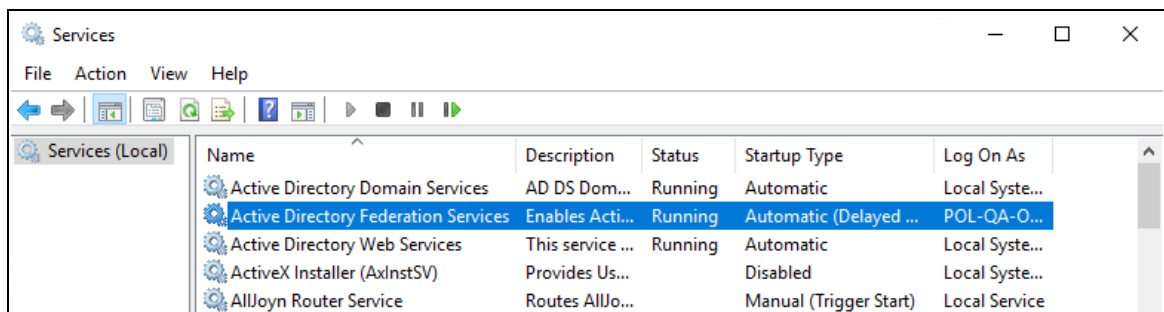
Verify Active Directory Federation Services Is Running

To verify that Active Directory Federation Services is running

1. Start the Server Manager desktop application.
AD FS should be green.



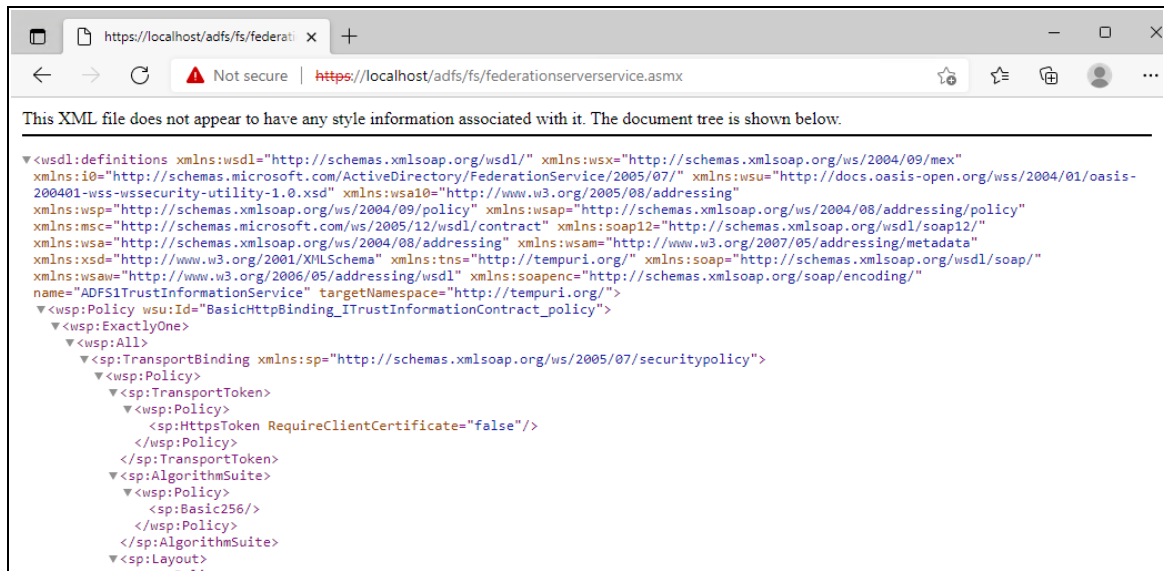
2. Start the Services application and check the status.



3. Open the Edge (or Chrome) web browser and go to <https://localhost/adfs/fs/federationsservice.asmx>
 - If you want to ignore certificate errors, select **Advanced**.

A page similar to the following image opens:

Polaris and OAuth 2.0 with OpenID Connect Integration Guide (7.3)

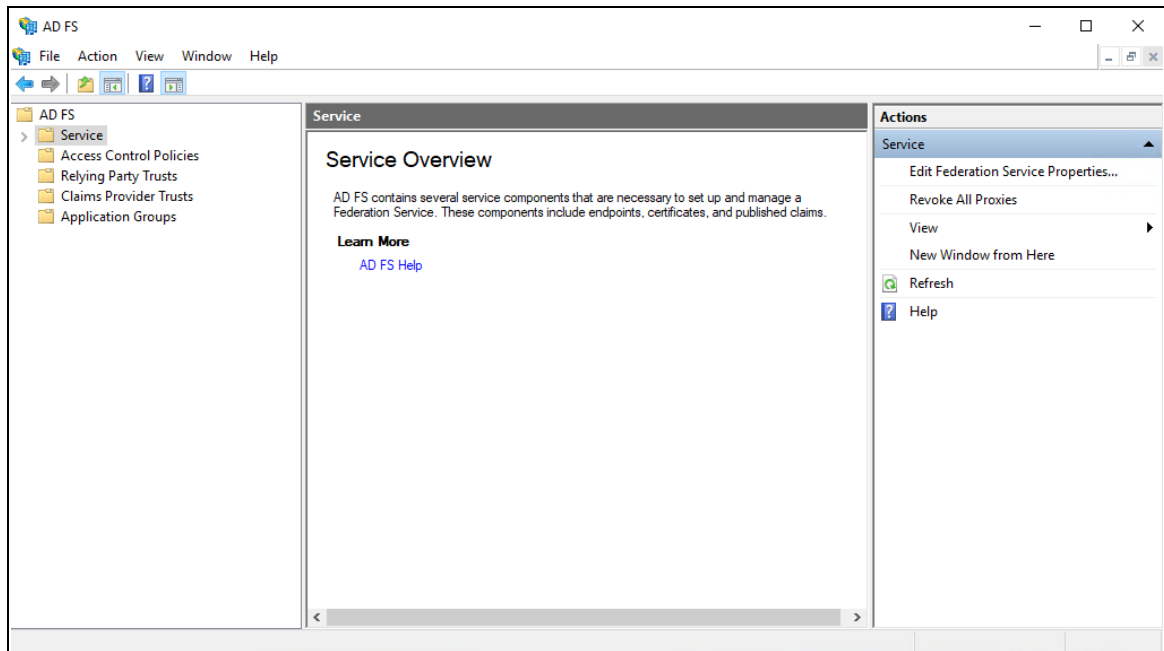


This indicates that Active Directory Federation Services is running.

Verify that OAuth 2.0 is Enabled

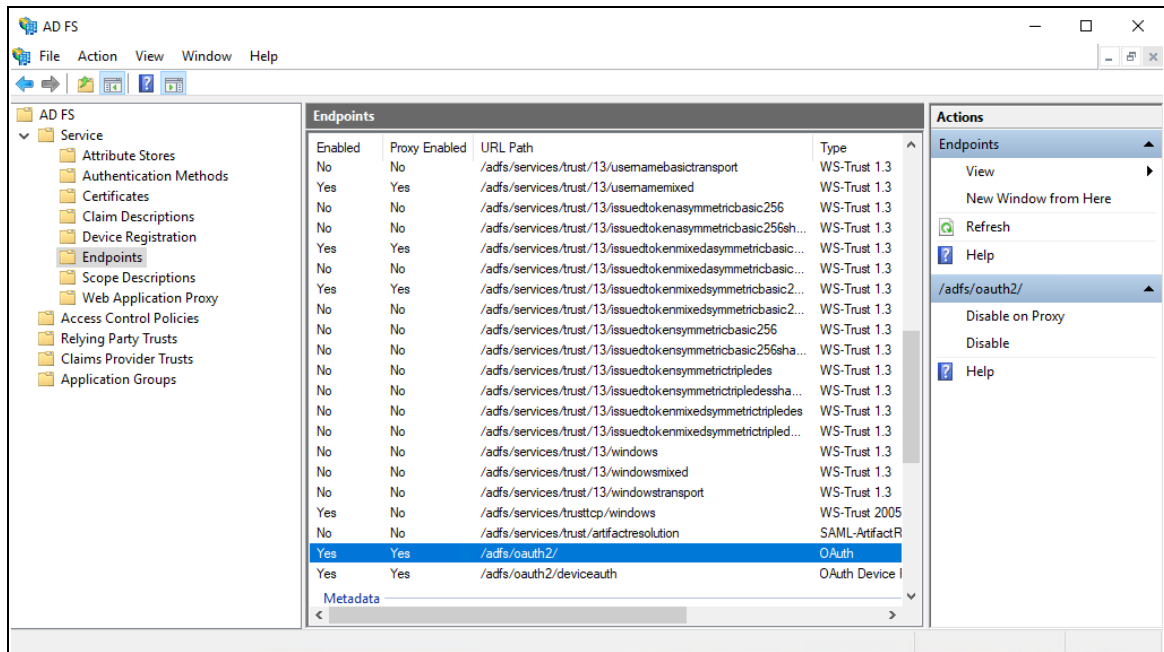
To verify that OAuth 2.0 is enabled

1. Open the AD FS Management desktop application.



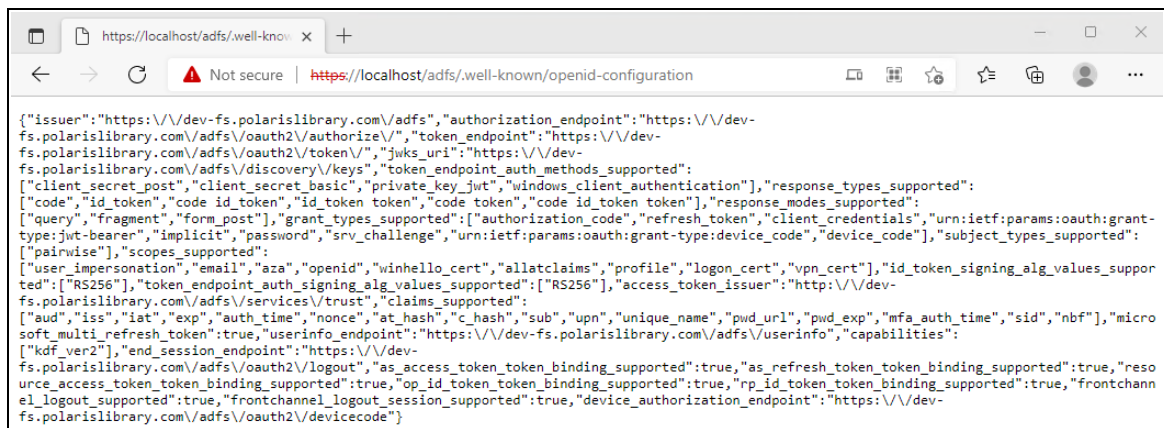
2. Open the **Service** folder, and then select the **Endpoint** folder.

Polaris and OAuth 2.0 with OpenID Connect Integration Guide (7.3)



3. Search for the oauth2 path.
4. In either the Edge or Chrome web browser, go to <https://localhost/adfs/.well-known/openid-configuration>
 - If you want to ignore certificate errors, select **Advanced**.

A page similar to the following image opens:

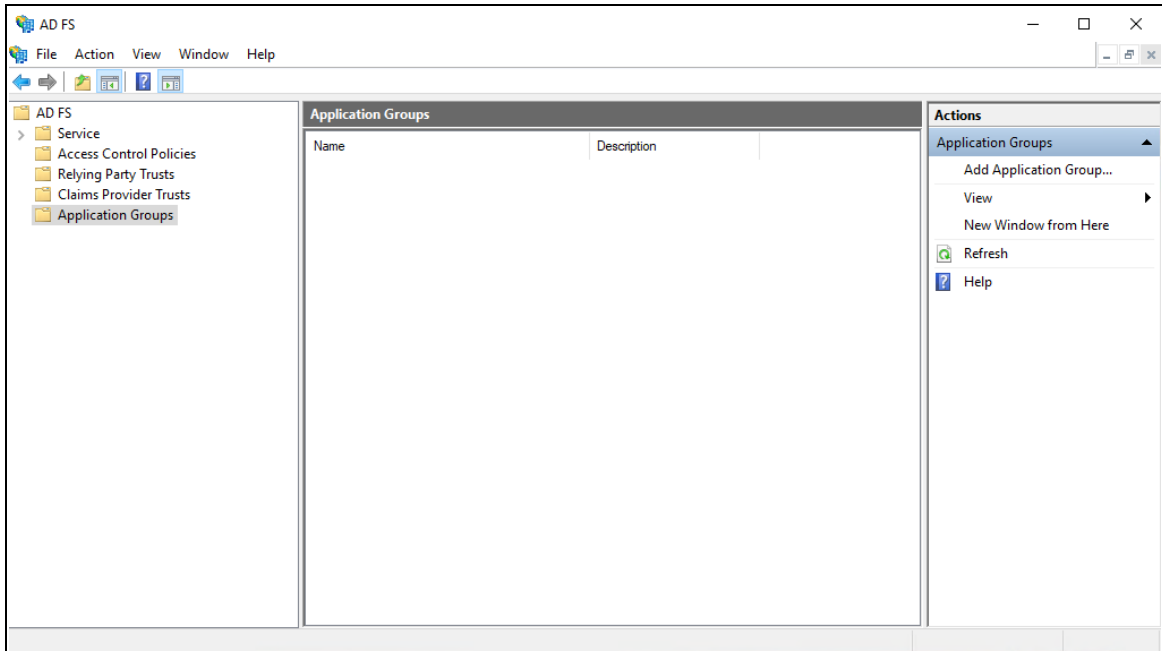


This indicates that OAuth 2.0 is available.

Create an Application Group

To create an application group for use with Polaris Admin and LeapWebApp

1. Open the AD FS Management desktop application.



2. Select the **Application Groups** folder.
3. Under **Actions**, select **Add Application Group**.

The Add Application Group wizard opens.

The screenshot shows the 'Add Application Group Wizard' dialog box with the 'Welcome' tab selected. The 'Steps' pane on the left lists: Welcome (selected), Native application, Apply Access Control Policy, Summary, and Complete. The main area contains three input fields: 'Name:' with the text 'Polaris', 'Description:' with the text 'Polaris web applications', and 'Template:' which displays a list of application types. Under 'Client-Server applications', 'Web browser accessing a web application' is selected. Under 'Standalone applications', 'Native application', 'Server application', and 'Web API' are listed. At the bottom are buttons for '< Previous', 'Next >' (highlighted), and 'Cancel'. A 'More information...' link is also present.

Add Application Group Wizard

Welcome

Steps

- Welcome
- Native application
- Apply Access Control Policy
- Summary
- Complete

Name:

Polaris

Description:

Polaris web applications

Template:

Client-Server applications

- Native application accessing a web API
- Server application accessing a web API
- Web browser accessing a web application**

Standalone applications

- Native application
- Server application
- Web API

[More information...](#)

< Previous **Next >** Cancel

4. On the **Welcome** tab, do the following:
 - a. In the **Name** box, enter **Polaris**.
 - b. In the **Description** box, enter **Polaris web applications**.
 - c. From the Template section, select **Web browser accessing a web application**.
5. Select **Next**.

Add Application Group Wizard

Native application

Steps

- Welcome
- Native application
- Apply Access Control Policy
- Summary
- Complete

Name:
Polaris - Native application

Client Identifier:
0a586b1e-eeb0-4c8a-8381-50e9cafec240

Redirect URI:
Example: https://Contoso.com Add

https://rd-polaris.polarislibrary.com/PolarisAdmin/login
https://rd-polaris.polarislibrary.com/PolarisAdmin/oauth-success
https://rd-polaris.polarislibrary.com/Polaris.AdminServices/swagger/oauth2-redirect.html Remove

Description:

< Previous **Next >** Cancel

6. On the **Native application** tab, in the **Redirect URI** box, enter the following URIs:
- https://server address/PolarisAdmin/
 - https://server address/PolarisAdmin/login
 - https://server address/PolarisAdmin/oauth-success
 - https://server address/Polaris.AdminServices/swagger/oauth2-redirect.html
 - https://server address/LeapWebApp/signin-oidc
 - https://server address/LeapWebApp/signin-override-oidc
 - https://server address/LeapWebApp/signout-callback-oidc
 - https://server address/Polaris.ApplicationServices/swagger/oauth2-redirect.html

Note:

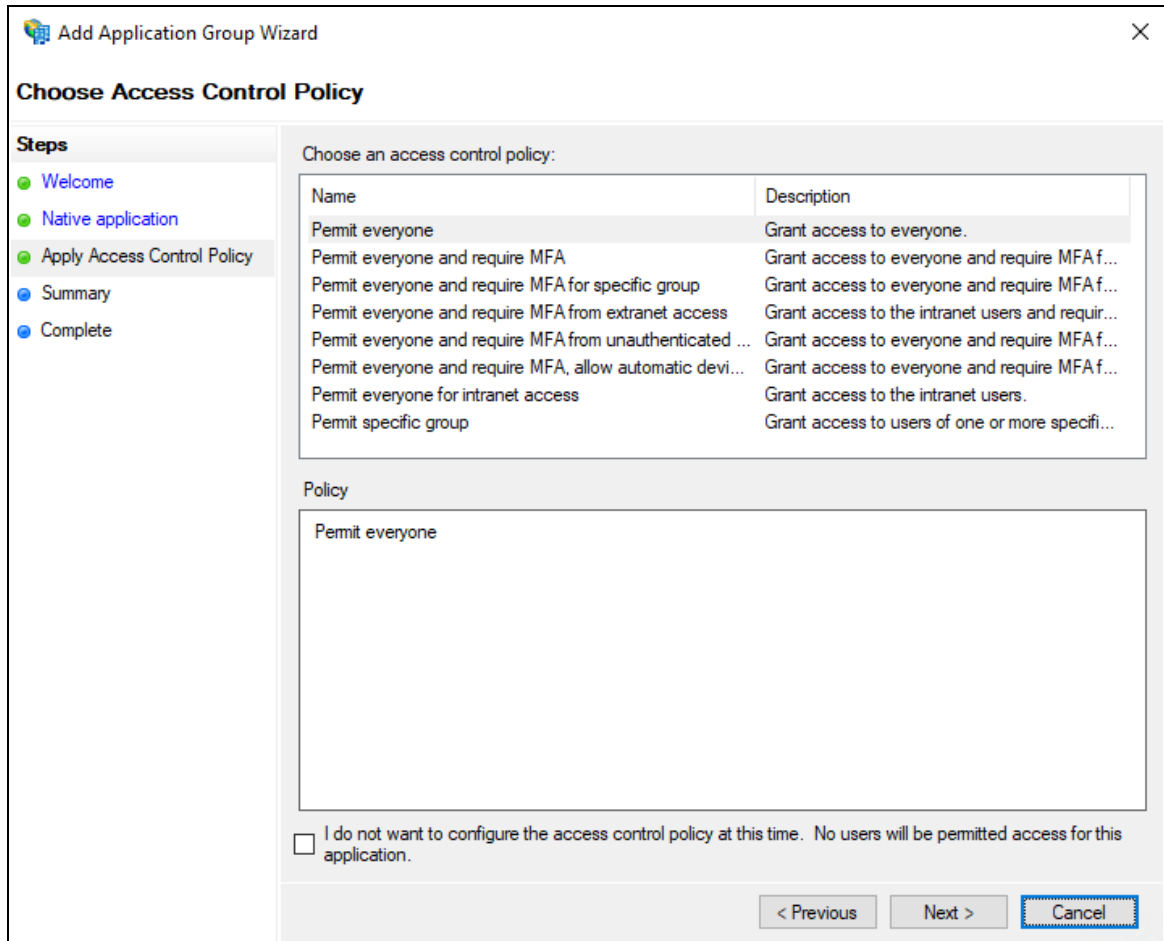
Replace *server address* with the FQDN that matches your Polaris

System Administration (web-based) or Leap server name and certificate.

7. Copy the value in the **Client Identifier** box to Notepad.

You'll need this when you set up PolarisAdmin's appsettings.user.json.

8. Select **Next**.



Add Application Group Wizard

Choose Access Control Policy

Steps

- Welcome
- Native application
- Apply Access Control Policy
- Summary
- Complete

Choose an access control policy:

Name	Description
Permit everyone	Grant access to everyone.
Permit everyone and require MFA	Grant access to everyone and require MFA f...
Permit everyone and require MFA for specific group	Grant access to everyone and require MFA f...
Permit everyone and require MFA from extranet access	Grant access to the intranet users and requir...
Permit everyone and require MFA from unauthenticated ...	Grant access to everyone and require MFA f...
Permit everyone and require MFA, allow automatic devi...	Grant access to everyone and require MFA f...
Permit everyone for intranet access	Grant access to the intranet users.
Permit specific group	Grant access to users of one or more specifi...

Policy

Permit everyone

☐ I do not want to configure the access control policy at this time. No users will be permitted access for this application.

< Previous Next > Cancel

9. On the **Apply Access Control Policy** tab, select an access control policy, and then select **Next**.

The screenshot shows the 'Add Application Group Wizard' window with the 'Summary' tab selected. The window has a title bar with a close button. On the left, a 'Steps' pane lists: Welcome, Native application, Apply Access Control Policy, Summary (highlighted), and Complete. The main area contains a summary of the application settings. At the top, it says 'Review the following settings and click 'Next' to create the application.' The settings are organized into three sections: 'Application Group', 'Native application', and 'Web application'. The 'Application Group' section shows Name: Polaris and Description: Polaris web applications. The 'Native application' section shows Name: Polaris - Native application, Identifier: 0a586b1e-eeb0-4c8a-8381-50e9cafec240, and four Redirect URIs. The 'Web application' section shows Name: Polaris - Web application, Identifier: 0a586b1e-eeb0-4c8a-8381-50e9cafec240, Access control policy: Permit everyone, and Application permissions. At the bottom right are buttons for '< Previous', 'Next >' (which is highlighted with a blue border), and 'Cancel'.

Add Application Group Wizard

Summary

Steps

- Welcome
- Native application
- Apply Access Control Policy
- Summary**
- Complete

Review the following settings and click 'Next' to create the application.

Application Group

Name: Polaris

Description: Polaris web applications

Native application

Name: Polaris - Native application

Identifier: 0a586b1e-eeb0-4c8a-8381-50e9cafec240

Redirect URIs:

- <https://rd-polaris.polarislibrary.com/PolarisAdmin/>
- <https://rd-polaris.polarislibrary.com/PolarisAdmin/login>
- <https://rd-polaris.polarislibrary.com/PolarisAdmin/oauth-success>
- <https://rd-polaris.polarislibrary.com/Polaris.AdminServices/swagger/oauth2-redirect.html>

Web application

Name: Polaris - Web application

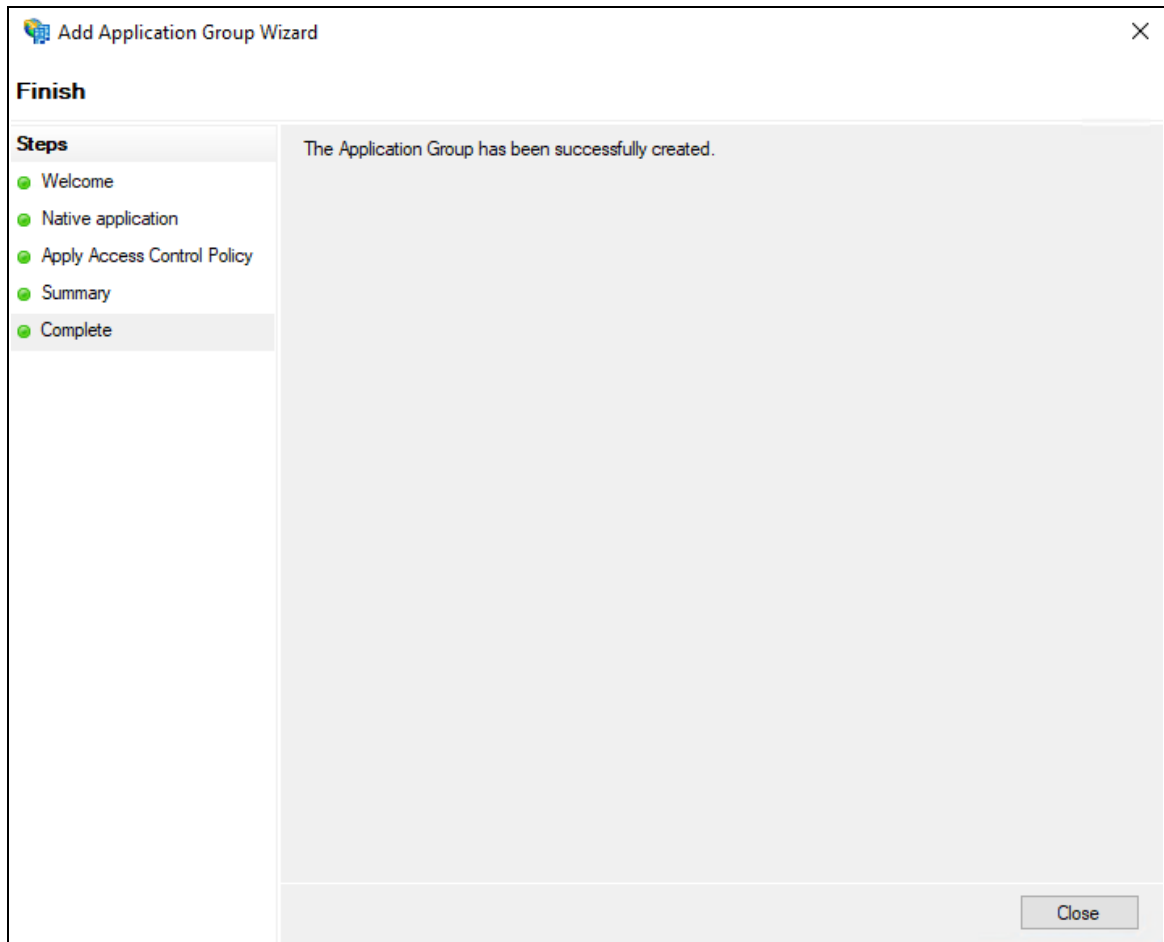
Identifiers: 0a586b1e-eeb0-4c8a-8381-50e9cafec240

Access control policy: Permit everyone

Application permissions:

< Previous **Next >** Cancel

10. On the **Summary** tab, review the settings and then select **Next**.

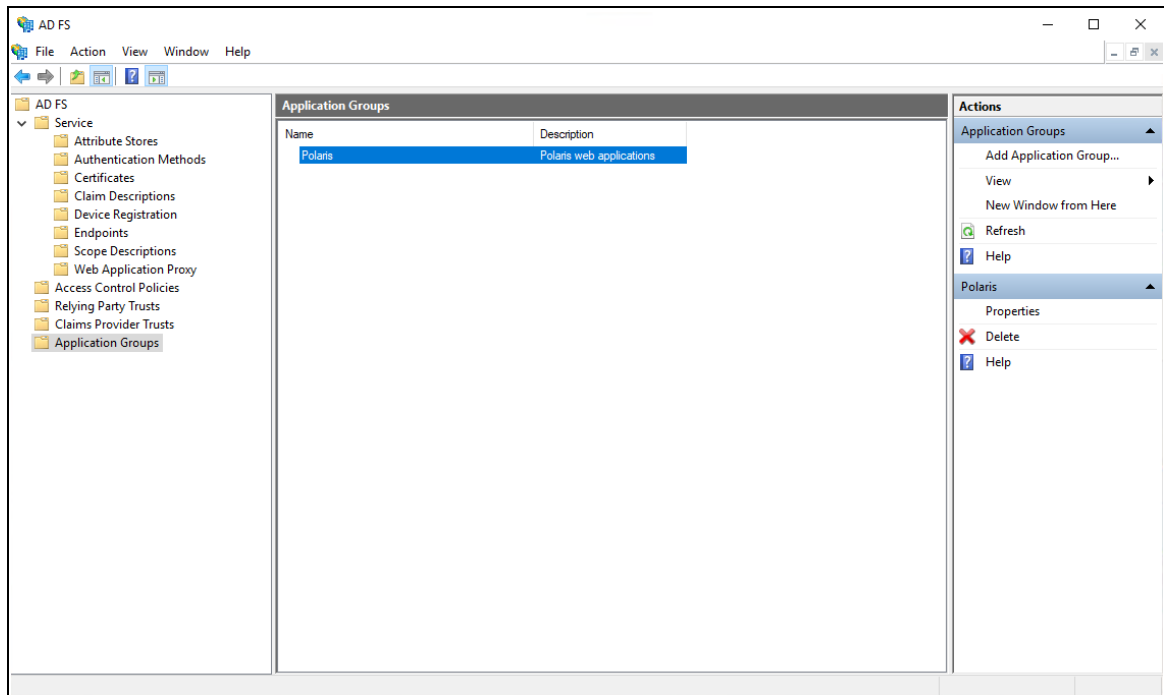


11. On the **Complete** tab, select **Close**.

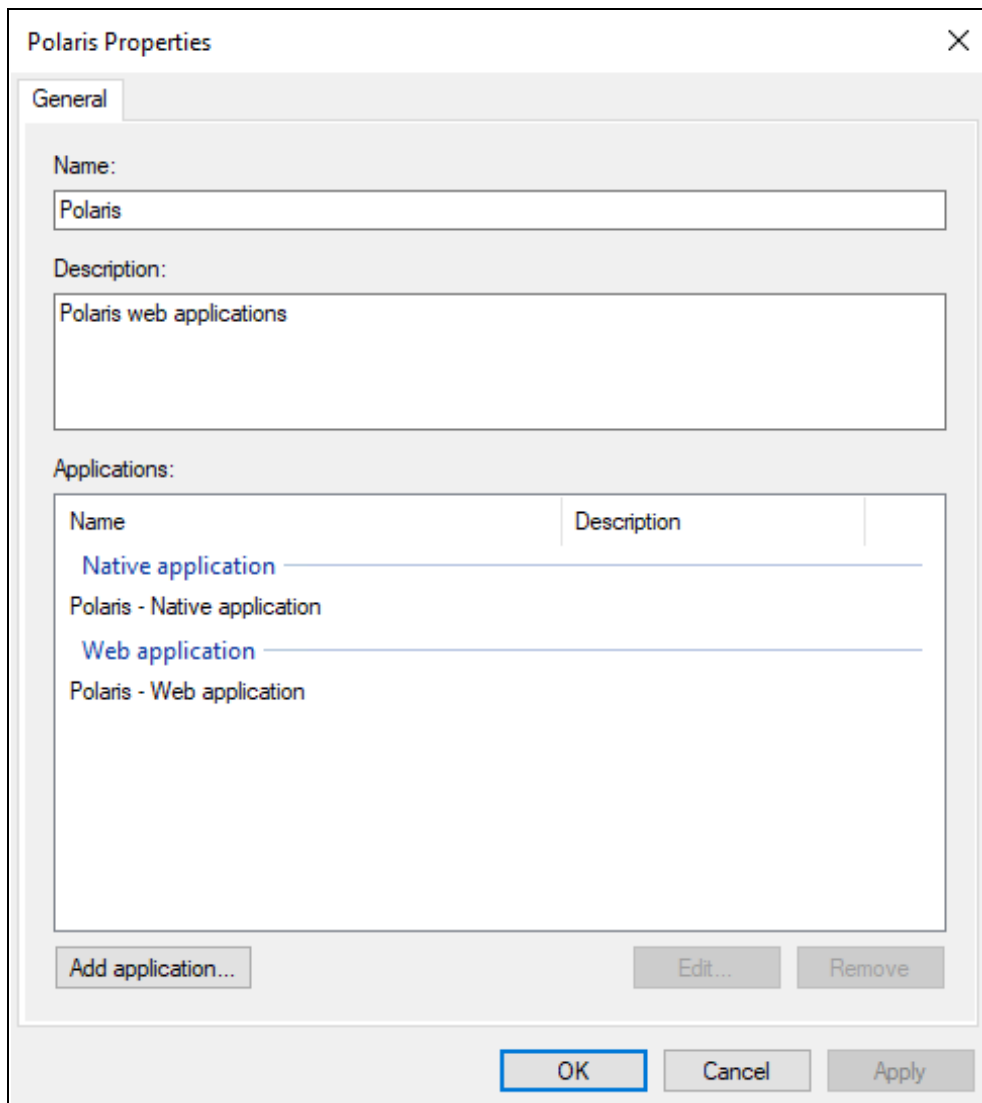
Configure the AD FS Web Application: Claims and Permitted Scopes

To configure the AD FS web application

1. Open the AD FS Management desktop application.



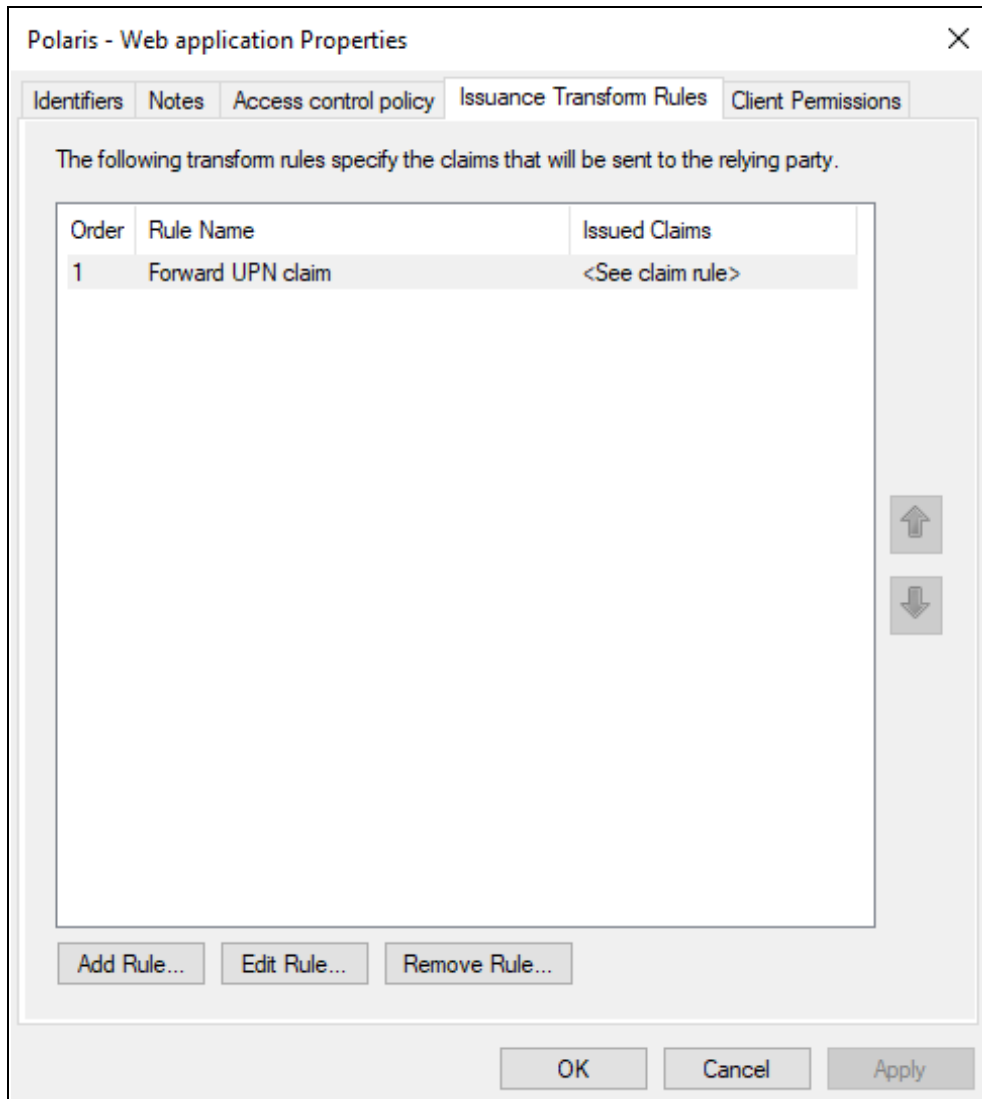
2. Select the **Application Groups** folder.
3. Select the **Polaris** application group, and then select **Properties**.



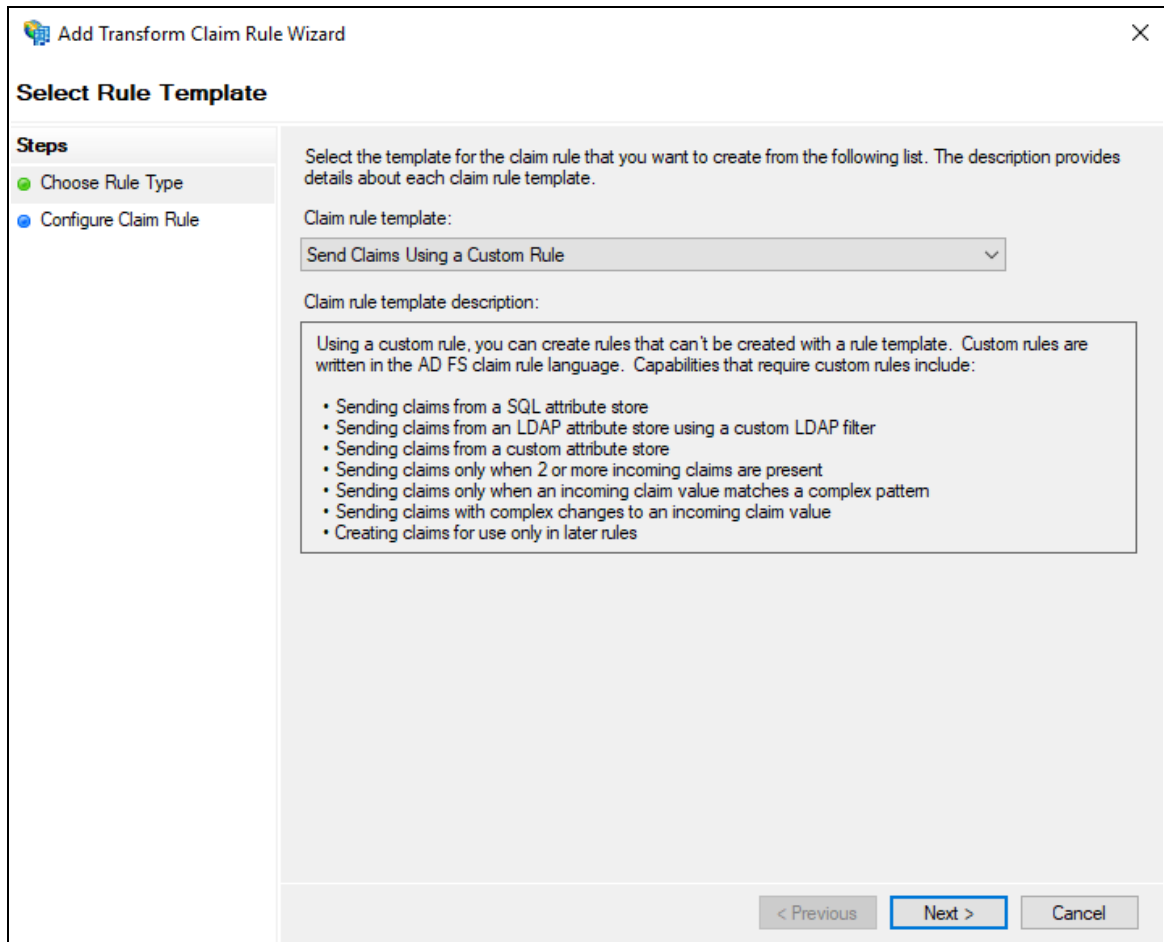
The image shows a 'Polaris Properties' dialog box with a 'General' tab. It contains fields for 'Name' (Polaris) and 'Description' (Polaris web applications). Below these is a table of applications with two columns: 'Name' and 'Description'. The table lists 'Native application' and 'Web application' as blue links, followed by 'Polaris - Native application' and 'Polaris - Web application'. At the bottom are buttons for 'Add application...', 'Edit...', 'Remove', 'OK', 'Cancel', and 'Apply'.

Name	Description
Native application	
Polaris - Native application	
Web application	
Polaris - Web application	

4. Select **Polaris - Web application**, and then select **Edit**.



5. Select the **Issuance Transform Rules** tab, and then select **Add Rule**.



- On the Add Transform Claim Rule Wizard, select **Send Claims Using a Custom Rule** from the **Claim rule template** list, and then select **Next**.

Add Transform Claim Rule Wizard

Configure Rule

Steps

- Choose Rule Type
- Configure Claim Rule

You can configure a custom claim rule, such as a rule that requires multiple incoming claims or that extracts claims from a SQL attribute store. To configure a custom rule, type one or more optional conditions and an issuance statement using the AD FS claim rule language.

Claim rule name:
Forward UPN Claim

Rule template: Send Claims Using a Custom Rule

Custom rule:

```
c:[Type == "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn"]
=> issue(claim = c);
```

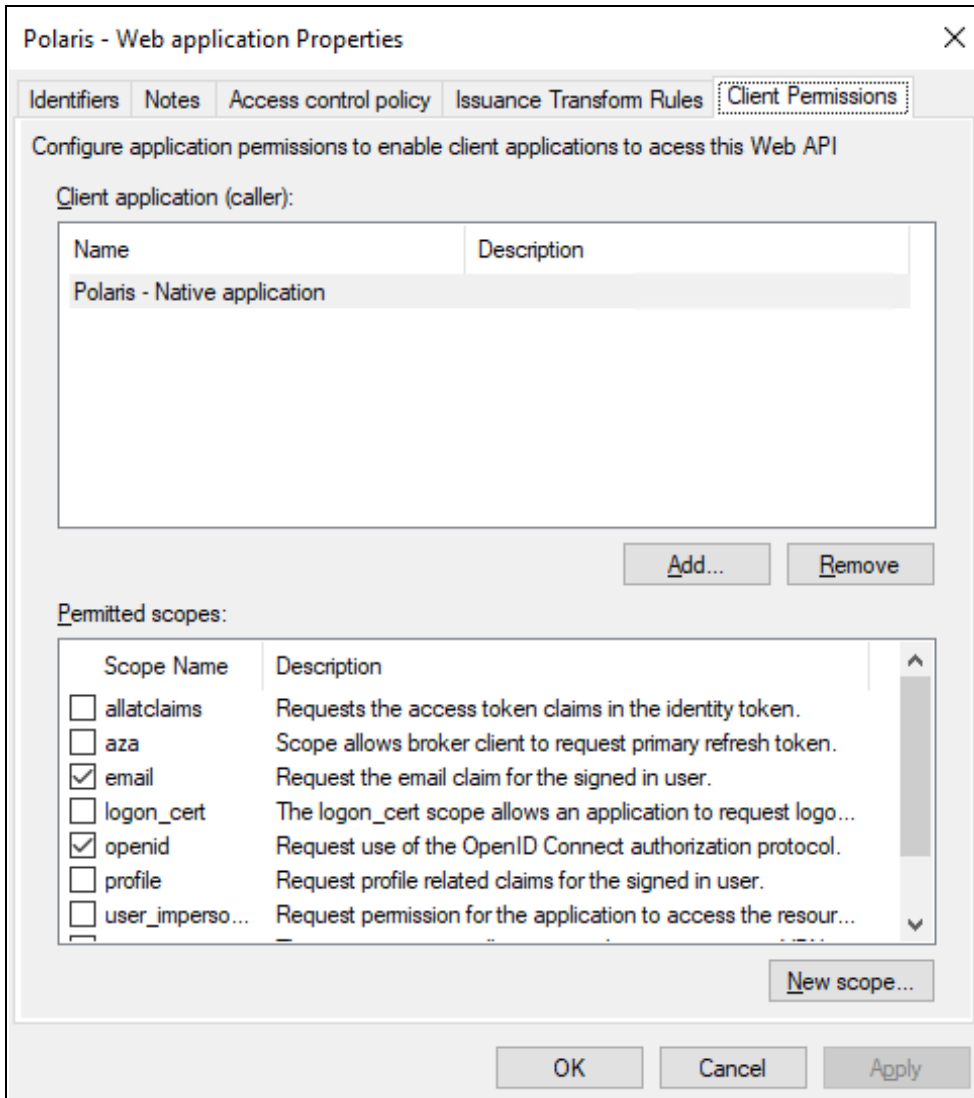
< Previous **Finish** Cancel

- In the **Claim rule name** box, enter **Forward UPN Claim**.
- In the **Custom rule** box, enter the following rule:


```
c:[Type ==
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn"]
=> issue(claim = c);
```
- Select **Finish**.
- On the **Issuance Transform Rules** tab, select **Add Rule**.
- On the Add Transform Claim Rule Wizard, select **Send Claims Using a Custom Rule** from the **Claim rule template** list, and then select **Next**.
- In the **Claim rule name** box, enter **Add TenantId**.
- In the **Custom rule** box, enter the following rule:

```
=> issue(Type =  
"http://schemas.microsoft.com/identity/claims/tenantid",  
Value = "polaris");
```

14. Select **Finish**.



Polaris - Web application Properties

Identifiers Notes Access control policy Issuance Transform Rules **Client Permissions**

Configure application permissions to enable client applications to access this Web API

Client application (caller):

Name	Description
Polaris - Native application	

Add... Remove

Permitted scopes:

Scope Name	Description
<input type="checkbox"/> allatclaims	Requests the access token claims in the identity token.
<input type="checkbox"/> aza	Scope allows broker client to request primary refresh token.
<input checked="" type="checkbox"/> email	Request the email claim for the signed in user.
<input type="checkbox"/> logon_cert	The logon_cert scope allows an application to request logo...
<input checked="" type="checkbox"/> openid	Request use of the OpenID Connect authorization protocol.
<input type="checkbox"/> profile	Request profile related claims for the signed in user.
<input type="checkbox"/> user_imperso...	Request permission for the application to access the resour...

New scope...

OK Cancel Apply

15. On the **Client Permissions** tab, verify that **email** and **openid** are selected.
16. Select **OK** to close the Web application Properties dialog.
17. Select **OK** to close the Polaris properties dialog.
18. Using the services applet, restart the Active Directory Federation Services service.

Enable CORS on AD FS To Accept Requests from Polaris APIs

To enable CORS on AD FS to accept requests from Polaris APIs

1. Refer to the information on the following page:
 - <https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/operations/customize-http-security-headers-ad-fs#cross-origin-resource-sharing-cors-headers>
2. Use the following commands:
 - `Set-AdfsResponseHeaders -EnableCORS $true`
 - `Set-AdfsResponseHeaders -CORSTrustedOrigins https://rd-polaris.polarislibrary.com,https://example2.com`

Note:

Replace `https://rd-polaris.polarislibrary.com` and `https://example2.com` with your own URL or list of URLs.

Set Up Web Services and Applications

To set up each of the following web services and applications, you must configure a .json file for each of the following:

- Polaris.AdminServices (the API service)
- PolarisAdmin (the web-based Polaris System Administration application)
- Polaris.ApplicationServices (Leap's API service)
- LeapWebApp (Leap)

The four .json files are all named appsettings.user.json, but they reside in different directories:

- C:\Program Files\Polaris\7.3\Polaris.AdminServices
- C:\Program Files\Polaris\7.3\PolarisAdmin\assets
- C:\Program Files\Polaris\7.3\Polaris.ApplicationServices
- C:\Program Files\Polaris\7.3\LeapWebApp

This section contains the following topics:

- [Set Up Polaris.AdminServices](#)
- [Set Up PolarisAdmin](#)
- [Set Up Polaris.ApplicationServices](#)
- [Set Up LeapWebApp](#)

Set Up Polaris.AdminServices

To set up Polaris.AdminServices

Note:

When you edit the appsettings.user.json file, you must run the editing application (for example, Notepad) as administrator.

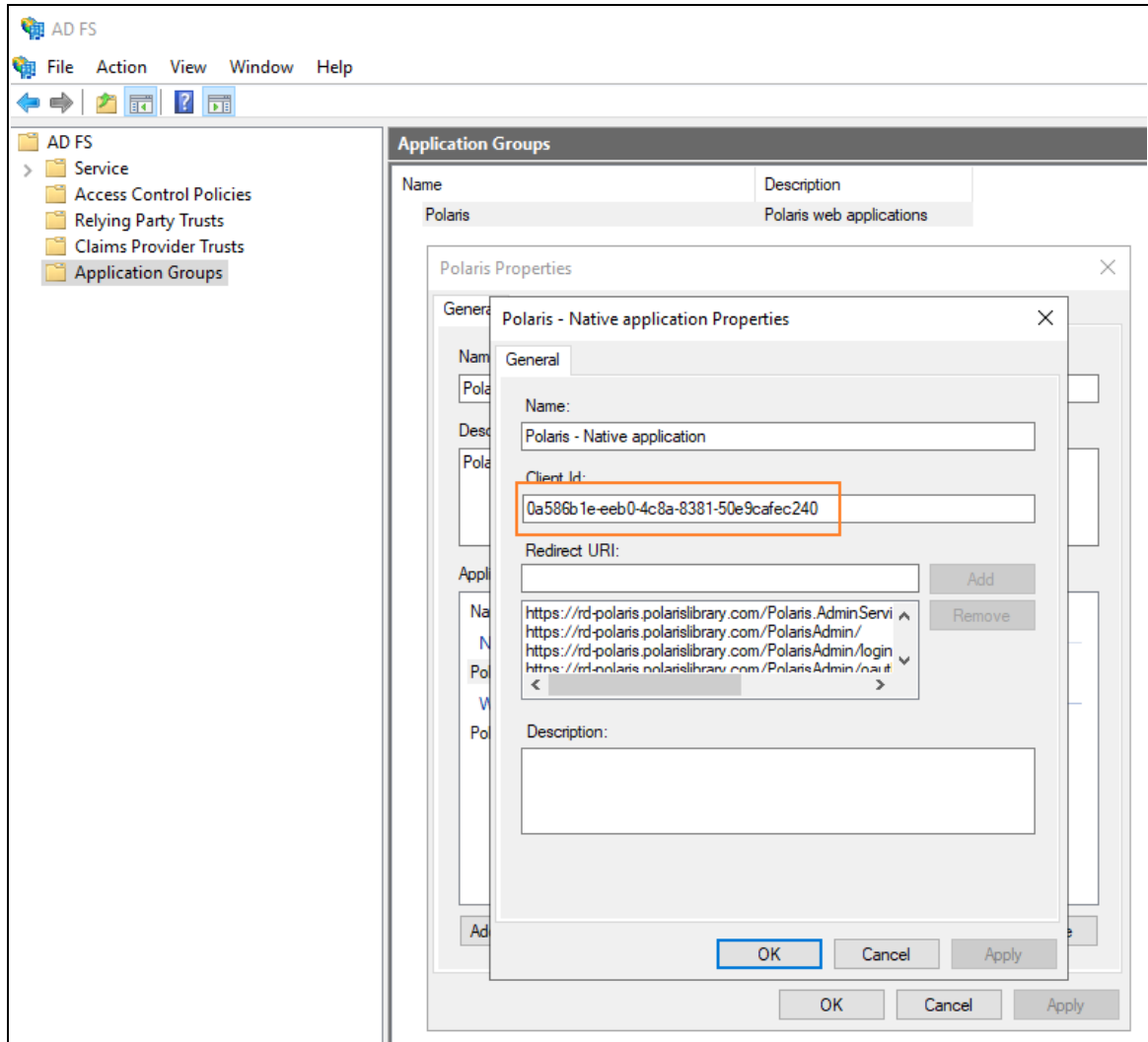
Verify that OAuth is Enabled

- Open C:\Program Files\Polaris\7.3\Polaris.AdminServices\appsettings.user.json and verify `Polaris.OAuth.Enabled` is set to `true`.

```
"Polaris": {  
  "CachePermissions": true,  
  "CORS": {  
    "AllowedHosts": "https://rd-polaris.polarislibrary.com"  
  },  
  "BasicAuth": {  
    "Enabled": false  
  },  
  "OAuth": {  
    "Enabled": true,  
    "ClientID": "0a586b1e-eeb0-4c8a-8381-50e9cafec240",  
    "Authority": "https://dev-fs.polarislibrary.com/adfs/",  
    "Audience": "microsoft:identityserver:0a586b1e-eeb0-4c8a-8381-50e9cafec240",  
    "ValidIssuer": "http://dev-fs.polarislibrary.com/adfs/services/trust",  
    "ValidAudience": "microsoft:identityserver:0a586b1e-eeb0-4c8a-8381-50e9cafec240",  
    "AuthorizationUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/authorize",  
    "TokenUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/token"  
  },  
}
```

Update the Client ID

1. On the AD FS server, open AD FS Management desktop application.



2. Copy the client ID from the Polaris - Native application properties dialog.
3. Paste the copied client ID into the appsettings.user.json file.
4. If you started from the template, replace `[client-id-that-might-look-like-a-guid]` with the copied client ID.

It should look like the following image when complete (your client ID will be different):

```

"Polaris": {
  "CachePermissions": true,
  "CORS": {
    "AllowedHosts": "https://rd-polaris.polarislibrary.com"
  },
  "BasicAuth": {
    "Enabled": false
  },
  "OAuth": {
    "Enabled": true,
    "ClientID": "0a586b1e-eeb0-4c8a-8381-50e9cafec240",
    "Authority": "https://dev-fs.polarislibrary.com/adfs/",
    "Audience": "microsoft:identityserver:0a586b1e-eeb0-4c8a-8381-50e9cafec240",
    "ValidIssuer": "http://dev-fs.polarislibrary.com/adfs/services/trust",
    "ValidAudience": "microsoft:identityserver:0a586b1e-eeb0-4c8a-8381-50e9cafec240",
    "AuthorizationUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/authorize",
    "TokenUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/token"
  },
},

```

Update the AD FS Server Location

1. If you started from the template, replace *[my-adfs-server-domain-name]* with the AD FS server address.
2. It should look like the following when complete (your AD FS server address will be different):

```

"Polaris": {
  "CachePermissions": true,
  "CORS": {
    "AllowedHosts": "https://rd-polaris.polarislibrary.com"
  },
  "BasicAuth": {
    "Enabled": false
  },
  "OAuth": {
    "Enabled": true,
    "ClientID": "0a586b1e-eeb0-4c8a-8381-50e9cafec240",
    "Authority": "https://dev-fs.polarislibrary.com/adfs/",
    "Audience": "microsoft:identityserver:0a586b1e-eeb0-4c8a-8381-50e9cafec240",
    "ValidIssuer": "http://dev-fs.polarislibrary.com/adfs/services/trust",
    "ValidAudience": "microsoft:identityserver:0a586b1e-eeb0-4c8a-8381-50e9cafec240",
    "AuthorizationUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/authorize",
    "TokenUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/token"
  },
},

```

Set Up PolarisAdmin

To set up PolarisAdmin

Note:

When you edit the appsettings.user.json file, you must run the editing application (for example, Notepad) as administrator.

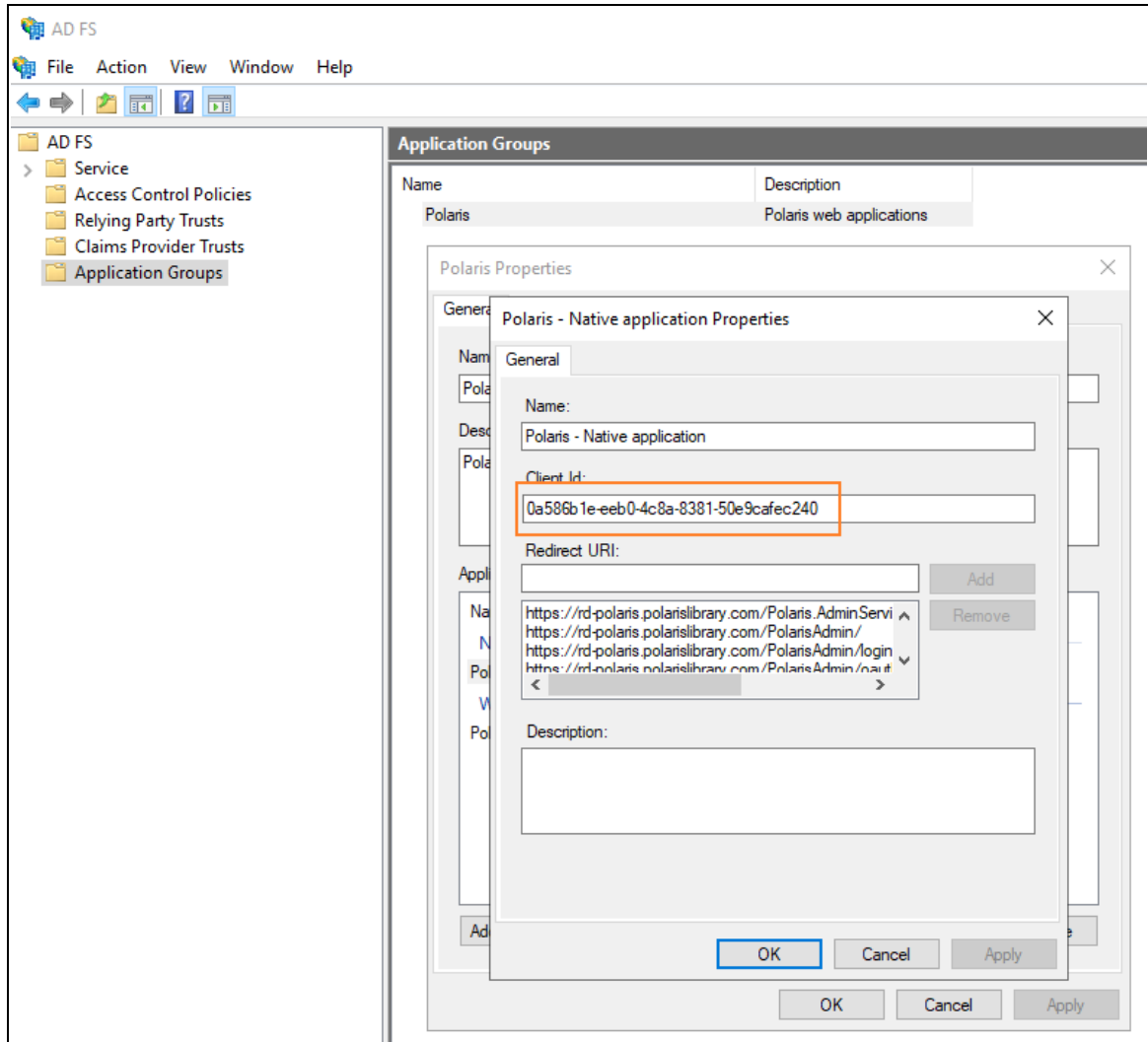
Verify that OAuth is Enabled

- Open C:\Program Files\Polaris\7.3\PolarisAdmin\assets\appsettings.user.json and verify that `oauthEnabled` is set to `true`.

```
{
  "apiUrlRoot": "https://rd-polaris.polarislibrary.com/polaris.adminservices/api/",
  "oauthEnabled": true,
  "msal": {
    "auth": {
      "clientId": "0a586b1e-eeb0-4c8a-8381-50e9cafec240",
      "authority": "https://dev-fs.polarislibrary.com/adfs/",
      "knownAuthorities": ["dev-fs.polarislibrary.com"],
      "redirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin/oauth-success",
      "postLogoutRedirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin",
      "protocolMode": "OIDC",
      "navigateToLoginRequestUrl": false
    },
    "cache": {
      "cacheLocation": "localStorage",
      "storeAuthStateInCookie": false,
      "secureCookies": true
    },
    "guard": {
      "interactionType": "redirect",
      "authRequest": {
        "scopes": ["openid", "profile", "email", "urn:microsoft:userinfo"]
      },
      "loginFailedRoute": "/login-failed"
    },
    "interceptor": {
      "interactionType": "redirect",
      "protectedResourceMap": [
        ["https://rd-polaris.polarislibrary.com/Polaris.AdminServices/api/protected/", ["email"]]
      ]
    }
  }
}
```


Update the Client ID

1. On the AD FS server, open AD FS Management desktop application.



2. Copy the client ID from the Polaris - Native application Properties dialog.
3. Paste the copied client ID into the appsettings.user.json file.
4. If you started from the template, replace `[CLIENTID-ASSIGNED-IN-ADFS]` with the copied client ID.

It should look like the following when complete (your client ID will be different):

```
{
  "apiUrlRoot": "https://rd-polaris.polarislibrary.com/polaris.adminservices/api/",
  "oauthEnabled": true,
  "msal": {
    "auth": {
      "clientId": "0a586b1e-eeb0-4c8a-8381-50e9cafec240",
      "authority": "https://dev-fs.polarislibrary.com/adfs/",
      "knownAuthorities": ["dev-fs.polarislibrary.com"],
      "redirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin/oauth-success",
      "postLogoutRedirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin",
      "protocolMode": "OIDC",
      "navigateToLoginRequestUrl": false
    },
    "cache": {
      "cacheLocation": "localStorage",
      "storeAuthStateInCookie": false,
      "secureCookies": true
    },
    "guard": {
      "interactionType": "redirect",
      "authRequest": {
        "scopes": ["openid", "profile", "email", "urn:microsoft:userinfo"]
      },
      "loginFailedRoute": "/login-failed"
    },
    "interceptor": {
      "interactionType": "redirect",
      "protectedResourceMap": [
        ["https://rd-polaris.polarislibrary.com/Polaris.AdminServices/api/protected/", ["email"]]
      ]
    }
  }
}
```

Update AD FS Server Location

- If you started from the template, replace `[ADFS-SERVER-ADDR]` with the AD FS server address.

It should look like the following when complete (your AD FS server address will be different):

```
{
  "apiUrlRoot": "https://rd-polaris.polarislibrary.com/polaris.adminservices/api/",
  "oauthEnabled": true,
  "msal": {
    "auth": {
      "clientId": "0a586b1e-eeb0-4c8a-8381-50e9cafec240",
      "authority": "https://dev-fs.polarislibrary.com/adfs/",
      "knownAuthorities": ["dev-fs.polarislibrary.com"],
      "redirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin/oauth-success",
      "postLogoutRedirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin",
      "protocolMode": "OIDC",
      "navigateToLoginRequestUrl": false
    },
    "cache": {
      "cacheLocation": "localStorage",
      "storeAuthStateInCookie": false,
      "secureCookies": true
    },
    "guard": {
      "interactionType": "redirect",
      "authRequest": {
        "scopes": ["openid", "profile", "email", "urn:microsoft:userinfo"]
      },
      "loginFailedRoute": "/login-failed"
    },
    "interceptor": {
      "interactionType": "redirect",
      "protectedResourceMap": [
        ["https://rd-polaris.polarislibrary.com/Polaris.AdminServices/api/protected/", ["email"]]
      ]
    }
  }
}
```

Update Polaris Admin Server Location

- If you started from the template, replace `[POLADMIN-SERVER-ADDR]` with the AD FS server address.

It should look like the following image when complete (your AD FS server address will be different):

```
{
  "apiUrlRoot": "https://rd-polaris.polarislibrary.com/polaris.adminservices/api/",
  "oauthEnabled": true,
  "msal": {
    "auth": {
      "clientId": "0a586b1e-eeb0-4c8a-8381-50e9cafec240",
      "authority": "https://dev-fs.polarislibrary.com/adfs/",
      "knownAuthorities": ["dev-fs.polarislibrary.com"],
      "redirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin/oauth-success",
      "postLogoutRedirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin",
      "protocolMode": "OIDC",
      "navigateToLoginRequestUrl": false
    },
    "cache": {
      "cacheLocation": "localStorage",
      "storeAuthStateInCookie": false,
      "secureCookies": true
    },
    "guard": {
      "interactionType": "redirect",
      "authRequest": {
        "scopes": ["openid", "profile", "email", "urn:microsoft:userinfo"]
      },
      "loginFailedRoute": "/login-failed"
    },
    "interceptor": {
      "interactionType": "redirect",
      "protectedResourceMap": [
        ["https://rd-polaris.polarislibrary.com/Polaris.AdminServices/api/protected/", ["email"]]
      ]
    }
  }
}
```

Update Polaris Admin Services (API) Server Location

- If you started from the template, replace `[POLADMINSVCSERVER-ADDR]` with the AD FS server address.

It should look like the following image when complete (your AD FS server address will be different):

```
{
  "apiUrlRoot": "https://rd-polaris.polarislibrary.com/polaris.adminservices/api/",
  "oauthEnabled": true,
  "msal": {
    "auth": {
      "clientId": "0a586b1e-eeb0-4c8a-8381-50e9cafec240",
      "authority": "https://dev-fs.polarislibrary.com/adfs/",
      "knownAuthorities": ["dev-fs.polarislibrary.com"],
      "redirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin/oauth-success",
      "postLogoutRedirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin",
      "protocolMode": "OIDC",
      "navigateToLoginRequestUrl": false
    },
    "cache": {
      "cacheLocation": "localStorage",
      "storeAuthStateInCookie": false,
      "secureCookies": true
    },
    "guard": {
      "interactionType": "redirect",
      "authRequest": {
        "scopes": ["openid", "profile", "email", "urn:microsoft:userinfo"]
      },
      "loginFailedRoute": "/login-failed"
    },
    "interceptor": {
      "interactionType": "redirect",
      "protectedResourceMap": [
        ["https://rd-polaris.polarislibrary.com/Polaris.AdminServices/api/protected/", ["email"]]
      ]
    }
  }
}
```

Set Up Polaris.ApplicationServices

To set up Polaris.ApplicationServices

Note:

When you edit the appsettings.user.json file, you must run the editing application (for example, Notepad) as administrator.

Verify that OAuth Is Enabled

- Open C:\Program Files\Polaris\7.3\Polaris.ApplicationServices\appsettings.user.json and verify that `OAuth.Enabled` is set to `true`.

```

"OAuth": {
  "Enabled": true,
  "Authorities": [
    {
      "Name": "ADFS",
      "Authority": "https://dev-fs.polarislibrary.com/adfs/",
      "Audience": "microsoft:identityserver:3eb2a79f-db5a-4ba0-b22f-e7d16a616d4a",
      "MetaAddress": "https://dev-fs.polarislibrary.com/adfs/.well-known/openid-configuration",
      "RequireHttpsMetadata": true,
      "RequireSignedTokens": true,
      "ValidateIssuer": true,
      "ValidIssuers": [
        "https://dev-fs.polarislibrary.com/adfs",
        "http://dev-fs.polarislibrary.com/adfs/services/trust"
      ],
      "ValidateAudience": true,
      "ValidAudiences": [
        "3eb2a79f-db5a-4ba0-b22f-e7d16a616d4a",
        "microsoft:identityserver:3eb2a79f-db5a-4ba0-b22f-e7d16a616d4a"
      ],
      "ClaimTypeUPN": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn"
    }
  ],
  "Swagger": {
    "ClientID": "3eb2a79f-db5a-4ba0-b22f-e7d16a616d4a",
    "AppName": "Polaris.ApplicationServices",
    "AuthorizationUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/authorize",
    "TokenUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/token",
    "RefreshTokenUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/token",
    "LogoutUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/logout"
  }
}

```

Update the AD FS Server Location

- If you started from the template, replace *adfs-server-address* with the AD FS server address.

It should look like the following when complete (your AD FS server address will be different):

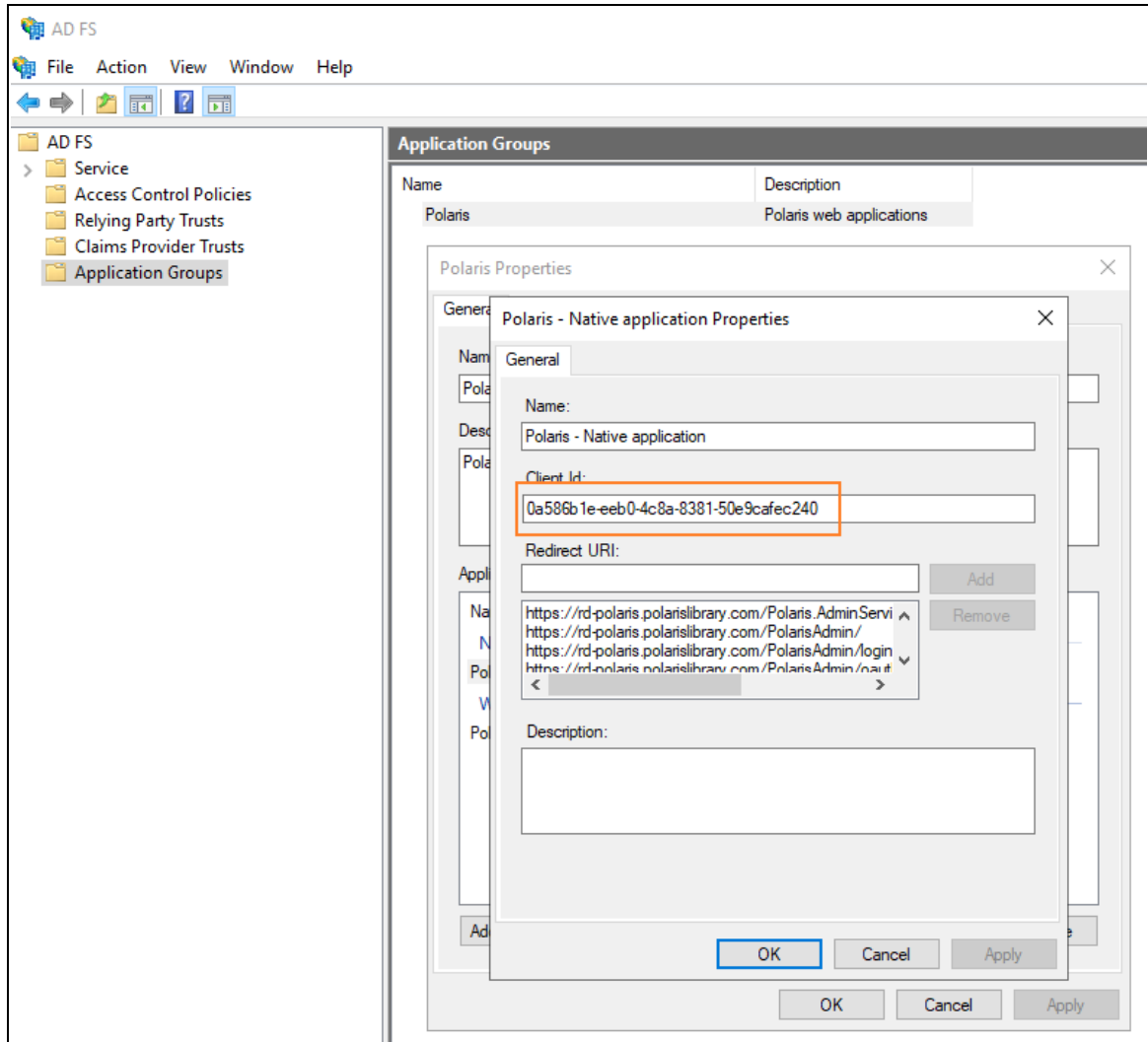
```

"OAuth": {
  "Enabled": true,
  "Authorities": [
    {
      "Name": "ADFS",
      "Authority": "https://dev-fs.polarislibrary.com/adfs/",
      "Audience": "microsoft:identityserver:3eb2a79f-db5a-4ba0-b22f-e7d16a616d4a",
      "MetaAddress": "https://dev-fs.polarislibrary.com/adfs/.well-known/openid-configuration",
      "RequireHttpsMetadata": true,
      "RequireSignedTokens": true,
      "ValidateIssuer": true,
      "ValidIssuers": [
        "https://dev-fs.polarislibrary.com/adfs",
        "http://dev-fs.polarislibrary.com/adfs/services/trust"
      ],
      "ValidateAudience": true,
      "ValidAudiences": [
        "3eb2a79f-db5a-4ba0-b22f-e7d16a616d4a",
        "microsoft:identityserver:3eb2a79f-db5a-4ba0-b22f-e7d16a616d4a"
      ],
      "ClaimTypeUPN": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn"
    }
  ],
  "Swagger": {
    "ClientID": "3eb2a79f-db5a-4ba0-b22f-e7d16a616d4a",
    "AppName": "Polaris.ApplicationServices",
    "AuthorizationUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/authorize",
    "TokenUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/token",
    "RefreshTokenUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/token",
    "LogoutUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/logout"
  }
},

```

Update the Client ID

1. On the AD FS server, open the AD FS Management desktop application.



2. Copy the client ID from the Polaris - Native application Properties dialog.
3. Paste the copied client ID into the appsettings.user.json file.
4. If you started from the template, replace `client-id-configured-in-adfs` with the copied client ID.

It should look like the following when complete (your client ID will be different):


```

"OAuth": {
  "Enabled": true,
  "Authorities": [
    {
      "Name": "ADFS",
      "Authority": "https://dev-fs.polarislibrary.com/adfs/",
      "Audience": "microsoft:identityserver:3eb2a79f-db5a-4ba0-b22f-e7d16a616d4a",
      "MetaAddress": "https://dev-fs.polarislibrary.com/adfs/.well-known/openid-configuration",
      "RequireHttpsMetadata": true,
      "RequireSignedTokens": true,
      "ValidateIssuer": true,
      "ValidIssuers": [
        "https://dev-fs.polarislibrary.com/adfs/",
        "http://dev-fs.polarislibrary.com/adfs/services/trust"
      ],
      "ValidateAudience": true,
      "ValidAudiences": [
        "3eb2a79f-db5a-4ba0-b22f-e7d16a616d4a",
        "microsoft:identityserver:3eb2a79f-db5a-4ba0-b22f-e7d16a616d4a"
      ],
      "ClaimTypeUPN": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn"
    }
  ],
  "Swagger": {
    "ClientID": "3eb2a79f-db5a-4ba0-b22f-e7d16a616d4a",
    "AppName": "Polaris.ApplicationServices",
    "AuthorizationUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/authorize",
    "TokenUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/token",
    "RefreshTokenUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/token",
    "LogoutUrl": "https://dev-fs.polarislibrary.com/adfs/oauth2/logout"
  }
},

```

Set Up LeapWebApp

To set up LeapWebApp

Note:

When you edit the appsettings.user.json file, you must run the editing application (for example, Notepad) as administrator.

Verify that OAuth Is Enabled

- Open C:\Program Files\Polaris\7.3\LeapWebApp\appsettings.user.json and verify that OAuthEnabled is set to true.

```
"OAuthEnabled": true,
"OAuth": {
  "Authority": "https://dev-fs.polarislibrary.com/adfs/",
  "ClientSecret": null,
  "MetadataAddress": "https://dev-fs.polarislibrary.com/adfs/.well-known/openid-configuration",
  "KnownAuthorities": [ "dev-fs.polarislibrary.com" ],
  "CallbackPath": "/signin-oidc",
  "SignedOutCallbackPath": "/signout-callback-oidc",
  "SignedOutRedirectUri": "/login",
  "RemoteAuthenticationTimeout": 15,
  "RemoteFailureRedirectUri": "/leapwebapp/logout",
  "ResponseMode": "form_post",
  "ResponseType": "code id_token token",
  "UsePkce": true
},
```

Update the AD FS Server Location

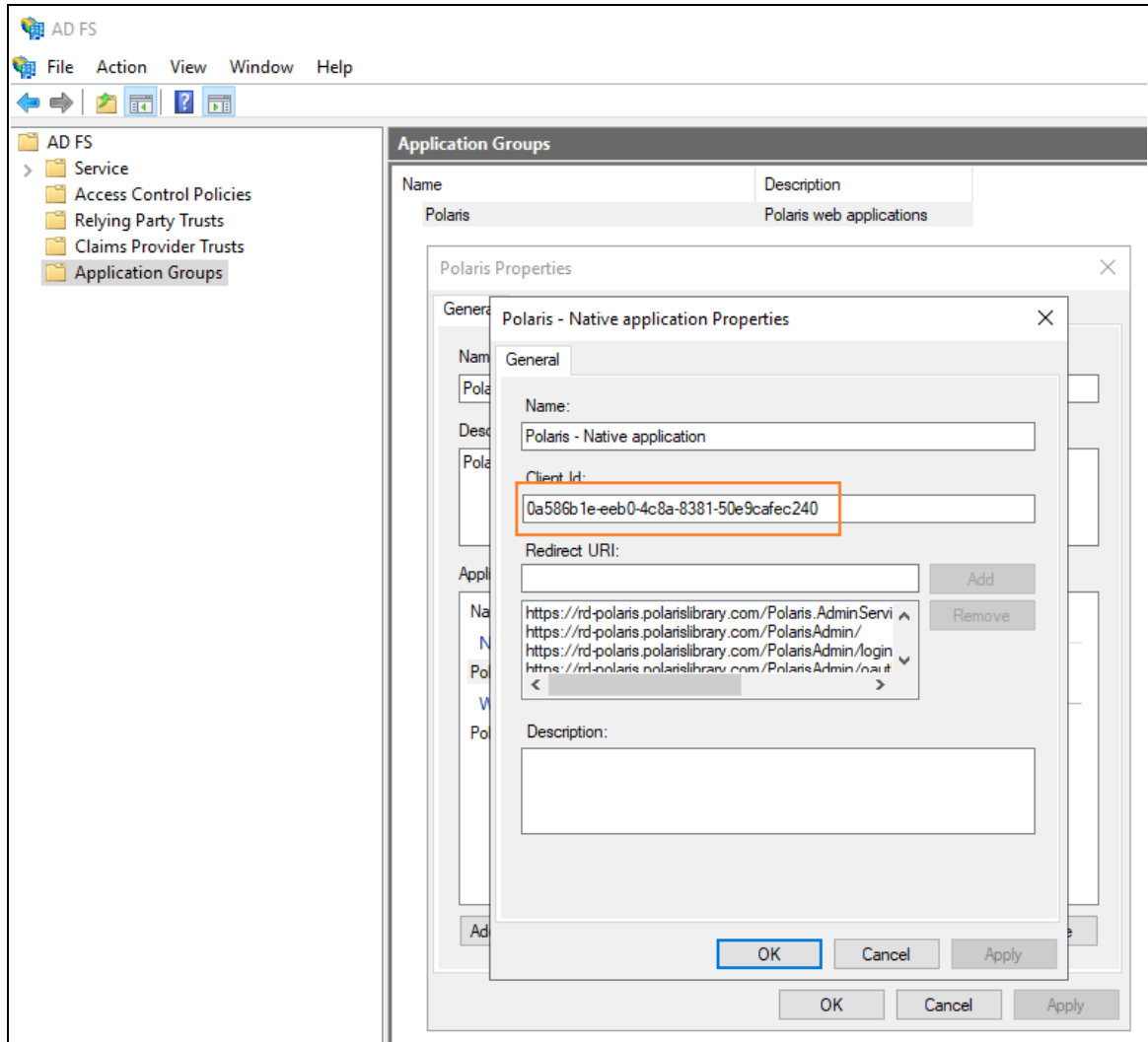
- If you started from the template, replace `[adfs-server-address]` with the AD FS server address.

It should look like the following when complete (your AD FS server address will be different):

```
"OAuthEnabled": true,
"OAuth": {
  "Authority": "https://dev-fs.polarislibrary.com/adfs/",
  "ClientId": "0a586b1e-eeb0-4c8a-8381-50e9cafec240",
  "ClientSecret": null,
  "MetadataAddress": "https://dev-fs.polarislibrary.com/adfs/.well-known/openid-configuration",
  "KnownAuthorities": [ "dev-fs.polarislibrary.com" ],
  "CallbackPath": "/signin-oidc",
  "SignedOutCallbackPath": "/signout-callback-oidc",
  "SignedOutRedirectUri": "/login",
  "RemoteAuthenticationTimeout": 15,
  "RemoteFailureRedirectUri": "/leapwebapp/logout",
  "ResponseMode": "form_post",
  "ResponseType": "code id_token token",
  "UsePkce": true
},
```

Update the Client ID

1. On the AD FS server, open the AD FS Management desktop application.



2. Copy the client ID from the Polaris - Native application Properties dialog.
3. Paste the copied client ID into the appsettings.user.json file.
4. If you started from the template, replace `client-id-configured-in-adfs` with the copied client ID.

It should look like the following when complete (your client ID will be different):

```

"OAuthEnabled": true,
"OAuth": {
  "Authority": "https://dev-fs.polarislibrary.com/adfs/",
  "ClientId": "0a586b1e-eeb0-4c8a-8381-50e9cafec240",
  "ClientSecret": null,
  "MetadataAddress": "https://dev-fs.polarislibrary.com/adfs/.well-known/openid-configuration",
  "KnownAuthorities": [ "dev-fs.polarislibrary.com" ],
  "CallbackPath": "/signin-oidc",
  "SignedOutCallbackPath": "/signout-callback-oidc",
  "SignedOutRedirectUri": "/login",
  "RemoteAuthenticationTimeout": 15,
  "RemoteFailureRedirectUri": "/leapwebapp/logout",
  "ResponseMode": "form_post",
  "ResponseType": "code id_token token",
  "UsePkce": true
},

```

(Optional) Disable Permission Overrides in Leap

By default, permission overrides are enabled in Leap.

To disable permission overrides

- Add the following value to the root level object in the .json file:

```
"client_permissionoverride_enabled": "false",
```

Note:

If you disable permission overrides in Leap, we recommend that you also update the language string that controls the title of the Permission Override dialog. For more information, search for the SW_CL_PERMISSION_OVRRD_TTL language string in the Polaris Web Admin Tool (Language Editor) Guide.

Enable Session Storage for LeapWebApp

Enable session storage for the best user experience when using OAuth and OIDC.

Microsoft SQL Server Express 2019 (or a newer version) must be installed to use session storage. You install SQL Server Express separately. It is not part of the Leap installation.

To enable session storage

- Open C:\Program Files\Polaris\7.3\LeapWebApp\appsettings.user.json and set `SessionStore.Enabled` to `true`.

Note:

You must run the editing application (for example, Notepad) as administrator.

```
"SessionStore": {
  "Enabled": true,
  "ConnectionString": "Data Source=.\Polaris; Initial Catalog=PolarisCache; Integrated Security=True;",
  "SessionTimeoutMinutes": "1440",
  "SchemaName": "dbo",
  "TableName": "Sessions"
},
```

Customize the AD FS Pages

Use the following resources to customize AD FS pages:

- [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn280950\(v=ws.11\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn280950(v=ws.11))
 - Get-AdfsGlobalWebContent
 - Set-AdfsGlobalWebContent

Examples:

Customize the examples below to suit your library's needs.

```
PS C:\Windows\system32> Set-AdfsGlobalWebContent -
SignOutPageDescriptionText "You have successfully signed
out.<br>If you have been directed here immediately after
signing in, your session may have timed out."
```

```
PS C:\Windows\system32> Set-AdfsWebTheme -TargetName
default -Logo @{path="c:\ADFS Custom\leap_logo.png"}
```

```
PS C:\Windows\system32> Set-AdfsGlobalWebContent -
CompanyName "Polaris R&D"
```

- Advanced customization:
 - [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn636121\(v=ws.11\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn636121(v=ws.11))

Change the Access Token Lifetime

The default token lifetime for both access and ID tokens is 60 minutes. Execute the following command to increase the expiration time to 24 hours:

```
Set-AdfsWebApiApplication -TokenLifetime 1440 -TargetIdentifier  
"0a586b1e-eeb0-4c8a-8381-50e9cafec240"
```

Note:

Replace *TargetIdentifier* with the Polaris Application Group native application client ID.

Bind a New SSL Certificate

If your web server certificate expires, use the instructions below to bind a new SSL certificate.

To bind a new SSL certificate

1. Install the certificate using Certificates Management.
2. Set the service communications certificate using the AD FS Management Console:
 - a. Expand the Services folder.
 - b. Select a new certificate.
 - c. Restart the AD FS service.
3. Attach the certificate to AD FS using PowerShell:
 - a. Get the certificate's thumbprint by viewing the certificate.

```
c:\> Set-AdfsSslCertificate -Thumbprint  
e8fd5016542796214e94f72d76095f9fc587c731
```
 - b. Restart the AD FS service.

Troubleshoot

Force a logout

- <https://AD FS server address/adfs/oauth2/logout>

Note:

Replace *AD FS server address* with your library's AD FS server address.

AD FS in one-way trust

Problem: Only local accounts are authenticating

Solution: Make sure the account running the AD FS service is a parent domain account and not a local account.

Receiving "User is not a valid Polaris user." error

- Check the setting Polaris.OAuth.ValidIssuer in the Polaris.AdminServices appsettings.user.json file.

Example value: `http://AD FS server address/adfs/services/trust`

Note:

Replace *AD FS server address* with your library's AD FS server address.

- Verify a domain is attached to AD user accounts so the UPN claim can be added to the ID token's claims.

The UPN claim should look like `user@mydomain.com`.

Troubleshoot Redirect URIs

Redirect URIs are case-sensitive.

Configuring Azure AD

Important:

The mechanism used to connect an Azure AD user to a Polaris user is the user principal name (UPN) in the format of an email address. For example, `user@mydomain.com`.

During the account verification process, we use the `openid` and `profile` scopes, which triggers Azure AD to return the `upn` claim or the `preferred_username` claim (or both). These must be returned in the `name@domain` format. The Polaris.ApplicationServices (API) can then use that information to map the Azure AD user to a Polaris user. If the `preferred_username` is a generic name, phone number, or other value, you can choose to apply the `email` scope to return the email.

See [Configure LeapWebApp for Use with Azure AD](#) for more information.

To configure Polaris OAuth support with Azure AD, perform the following tasks:

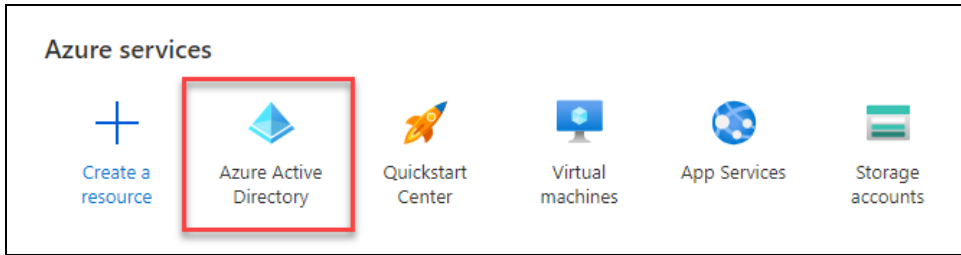
1. [Register LeapWebApp with Azure AD](#).
2. [Create client credentials](#).
3. [Add authentication redirect URIs](#).
4. [Expose the Polaris.ApplicationServices API](#).
5. [Configure an ID token](#).
6. [Set up users and groups](#).
7. [Control access to LeapWebApp using Azure AD](#).
8. [Set up web services and applications](#).

After you complete these tasks, [Add a URL rewrite rule for LeapWebApp](#).

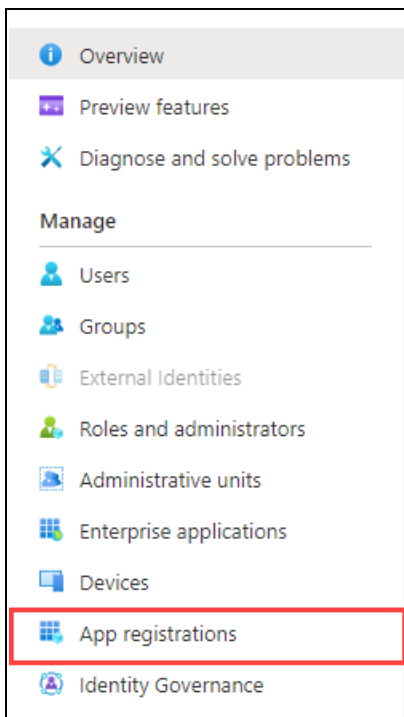
Register LeapWebApp with Azure AD

To register LeapWebApp with Azure AD

1. Sign in to the Azure portal:
<https://portal.azure.com/>
2. In the **Azure services** list, select **Azure Active Directory**.



3. In the list of options at the left side of the screen, select **App registrations**.



The App registrations page appears.

4. Select **New registration**.

The Register an application dialog appears.

Register an application ...

*** Name**

The user-facing display name for this application (this can be changed later).

Supported account types

Who can use this application or access this API?

☒ Accounts in this organizational directory only (Jeffrey Young only - Single tenant)
☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant)
☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
☐ Personal Microsoft accounts only

[Help me choose...](#)

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

5. Enter "LeapWebApp" in the **Name** field.
6. Select an option from the **Supported account types** list.
7. Add a redirect URI:
 - a. Select the **Web** URI type.
 - b. Enter an address that uses the following format:
https://[FQDN]/leapwebapp/signin-oidc

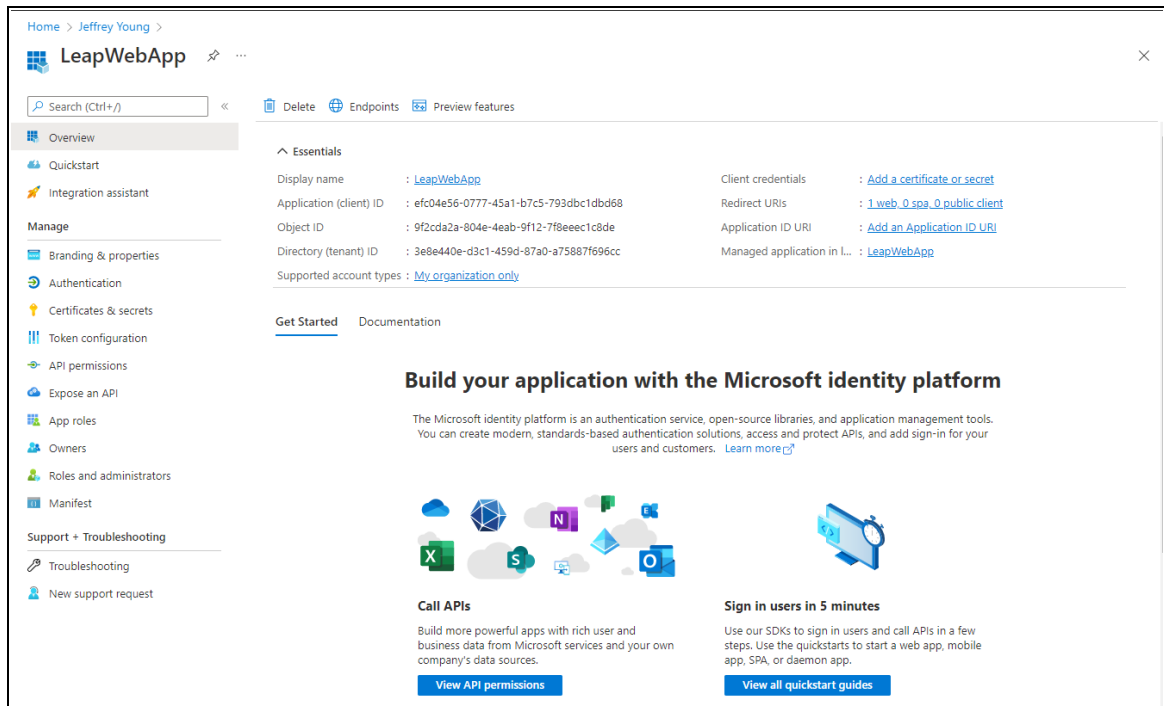
Notes:

- Replace [FQDN] with the fully-qualified domain name of your LeapWebApp server.
- Example: https://leap.mylibrary.org/leapwebapp/signin-oidc

8. Select **Register**.

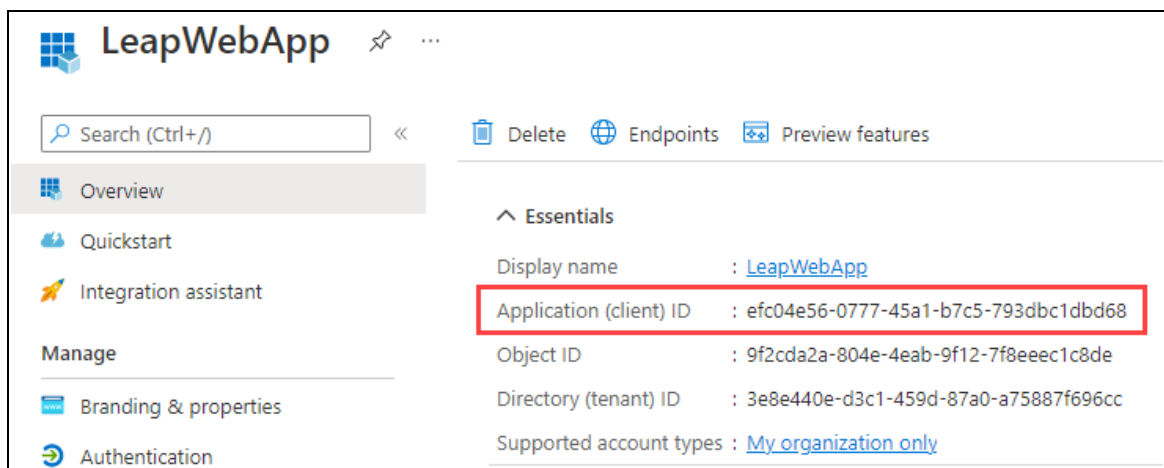
The page for your new LeapWebApp application appears.

Polaris and OAuth 2.0 with OpenID Connect Integration Guide (7.3)



- Copy the application (client) ID and paste it into Notepad (or a similar text editor) and save the file. You must have this value to complete several procedures later in the Azure AD configuration process.

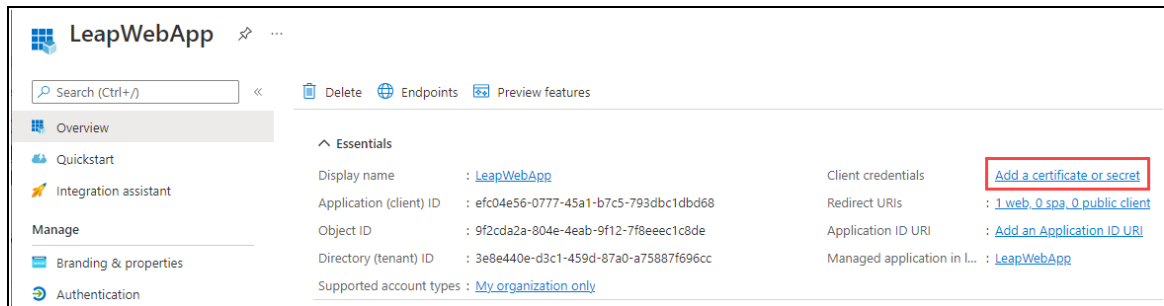
In the example below, the application (client) ID is "efc04e56-0777-45a1-b7c5-793dbc1dbd68".



Create Client Credentials

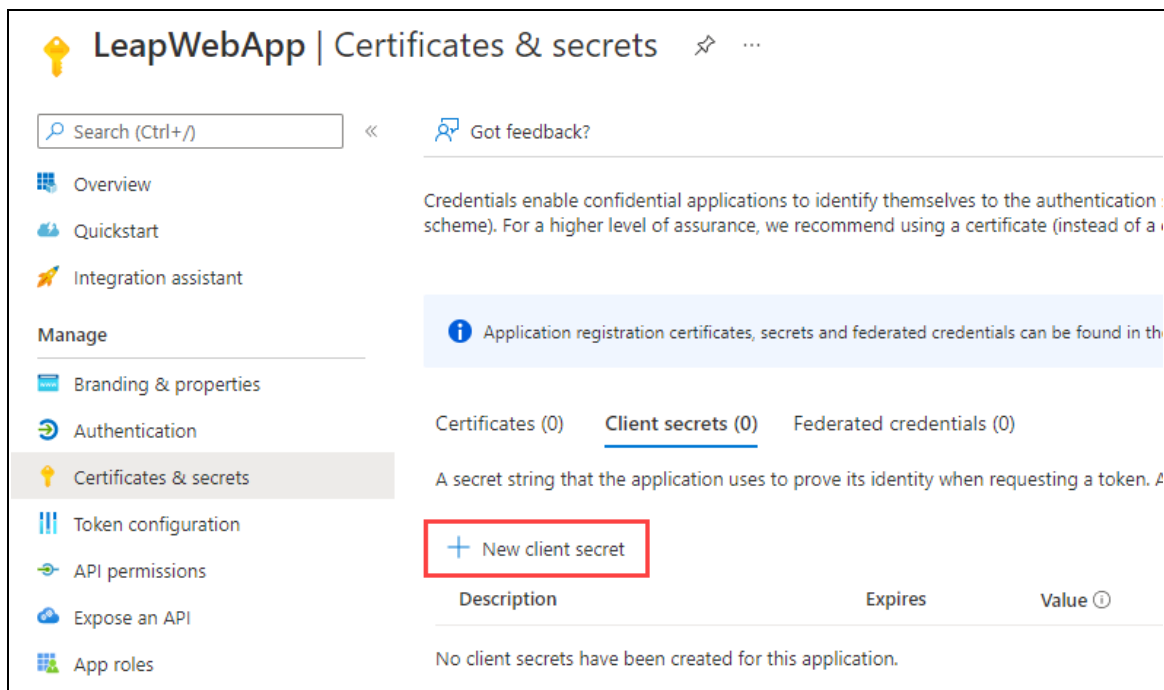
To create client credentials for LeapWebApp

1. On the LeapWebApp page, select **Add a certificate or secret**.



The Certificates & secrets page appears.

2. Select the **Client secrets** tab.
3. Select **New client secret**.



The Add a client secret dialog appears.

4. Enter a description in the **Description** field.
5. Select an option from the **Expires** list to specify when the client credentials expire.
6. Select **Add**.

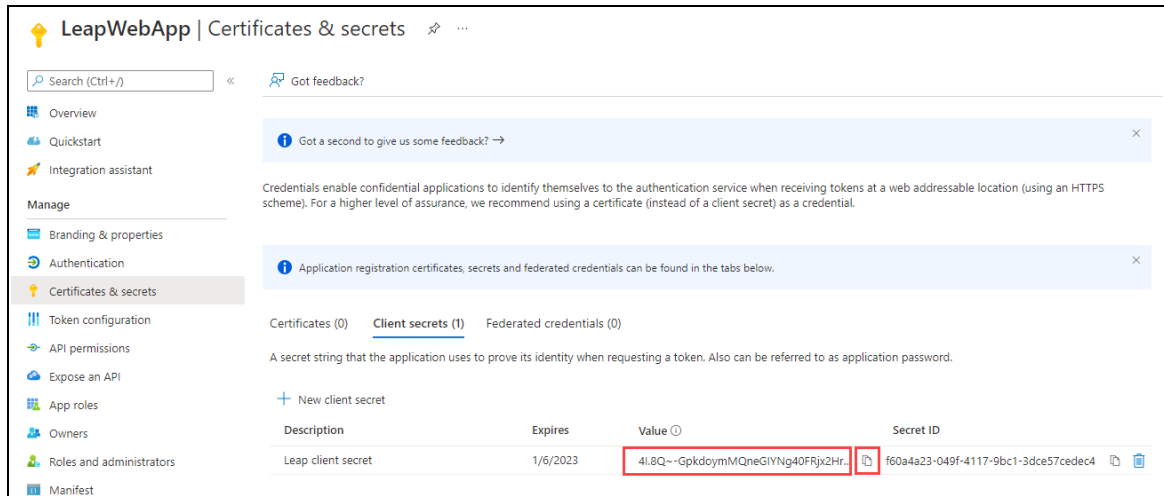
The Add a client secret dialog closes. The client secret for LeapWebApp appears on the **Client secrets** tab of the Certificates & secrets page.

7. Copy the text in the **Value** column, then paste it into Notepad (or a similar text editor) and save the file. You must have this value to complete the [Configure LeapWebApp for Use with Azure AD](#) procedure.

Important:

- You must save this value now. Only a portion of the value appears when you return to the page later.
- Use the copy icon to be sure that you are copying the entire value.

In the example below, the value is "4l.8Q~-GpkdoymMQneGIYNg40FRjx2Hr1wWLDcbr".



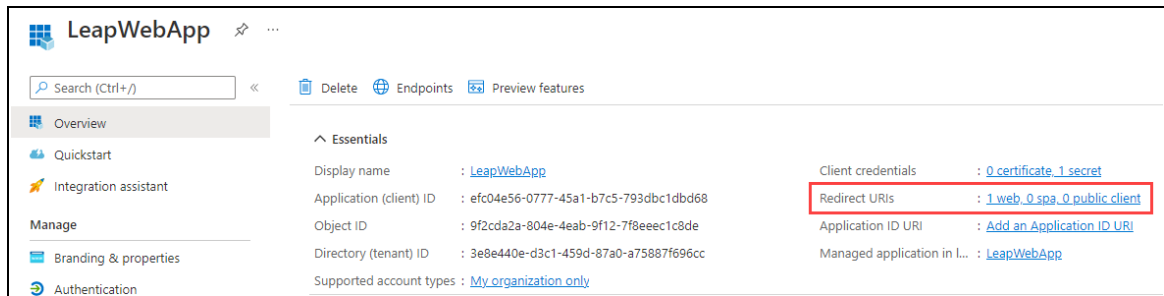
The screenshot shows the 'LeapWebApp | Certificates & secrets' page. The left sidebar contains navigation links: Overview, Quickstart, Integration assistant, Manage (Branding & properties, Authentication, Certificates & secrets), Token configuration, API permissions, Expose an API, App roles, Owners, Roles and administrators, and Manifest. The main content area is titled 'Certificates & secrets' and has tabs for 'Certificates (0)', 'Client secrets (1)', and 'Federated credentials (0)'. The 'Client secrets (1)' tab is active, showing a table with one entry: 'Leap client secret'. The table has columns for 'Description', 'Expires', 'Value', and 'Secret ID'. The 'Value' column for the 'Leap client secret' entry contains the text '4l.8Q~-GpkdoymMQneGIYNg40FRjx2Hr1wWLDcbr', which is highlighted with a red box. A copy icon is visible next to the value.

Description	Expires	Value	Secret ID
Leap client secret	1/6/2023	4l.8Q~-GpkdoymMQneGIYNg40FRjx2Hr1wWLDcbr	f60a4a23-049f-4117-9bc1-3dce57cedec4

Add Authentication Redirect URIs

To add authentication redirect URIs

1. On the LeapWebApp page, select the link beside **Redirect URIs**.



The Authentication page appears.

2. Select **Add a platform**.

The Configure platforms dialog appears.

3. Select the **Single-page application** tile.

The Configure single-page application dialog appears.

4. In the **Redirect URIs** field, enter an address that uses the following format:

`https://[FQDN]/leapwebapp/signin-override-oidc`

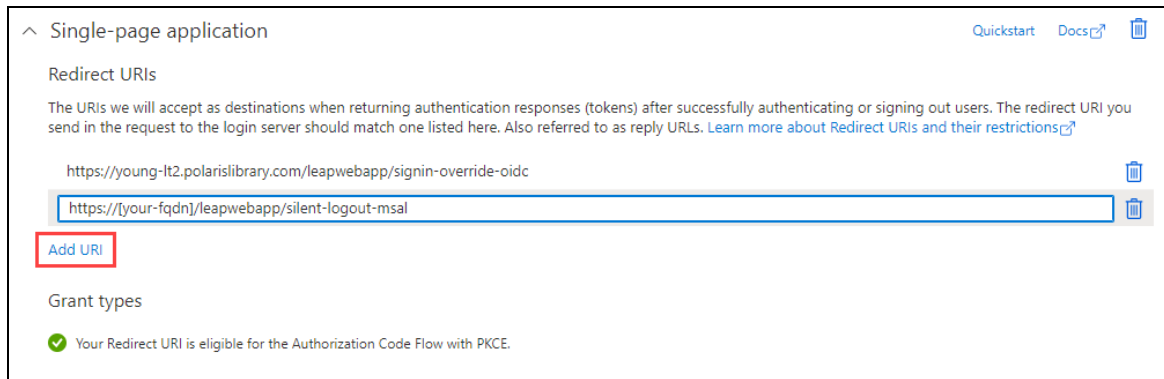
Notes:

- Replace *[FQDN]* with the fully-qualified domain name of your LeapWebApp server.
- Example: `https://leap.mylibrary.org/leapwebapp/signin-override-oidc`

5. Select **Configure**.

The new redirect URI appears on the Authentication page in the **Single-page application** list.

6. Select **Add URI**.



Single-page application

Redirect URIs

The URIs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating or signing out users. The redirect URI you send in the request to the login server should match one listed here. Also referred to as reply URLs. [Learn more about Redirect URIs and their restrictions](#)

<https://young-it2.polarislibrary.com/leapwebapp/signin-override-oidc>

[https://\[your-fqdn\]/leapwebapp/silent-logout-msal](https://[your-fqdn]/leapwebapp/silent-logout-msal)

Add URI

Grant types

✓ Your Redirect URI is eligible for the Authorization Code Flow with PKCE.

7. Enter an address that uses the following format:

`https://[FQDN]/leapwebapp/silent-logout-msal`

Notes:

- Replace `[FQDN]` with the fully-qualified domain name of your LeapWebApp server.
- Example: `https://leap.mylibrary.org/leapwebapp/silent-logout-msal`

8. Select **Add URI** again.
9. Enter an address that uses the following format:

`https://[FQDN]/Polaris.ApplicationServices/swagger/oauth2-redirect.html`

Notes:

- Replace `[FQDN]` with the fully-qualified domain name of your LeapWebApp server.
- Example:
`https://leap.mylibrary.org/Polaris.ApplicationServices/swagger/oauth2-redirect.html`

10. Select **Add URI** again.
11. Enter an address that uses the following format:

`https://[FQDN]/PolarisAdmin/oauth-success`

Notes:

- Replace *[FQDN]* with the fully-qualified domain name of your LeapWebApp server.
- Example: `https://leap.mylibrary.org/PolarisAdmin/oauth-success`

12. Select **Add URI** again.

13. Enter an address that uses the following format:

`https://[FQDN]/Polaris.AdminServices/swagger/oauth2-redirect.html`

Notes:

- Replace *[FQDN]* with the fully-qualified domain name of your LeapWebApp server.
- Example:
`https://leap.mylibrary.org/Polaris.AdminServices/swagger/oauth2-redirect.html`

14. Select **Save** to save the five redirect URIs.

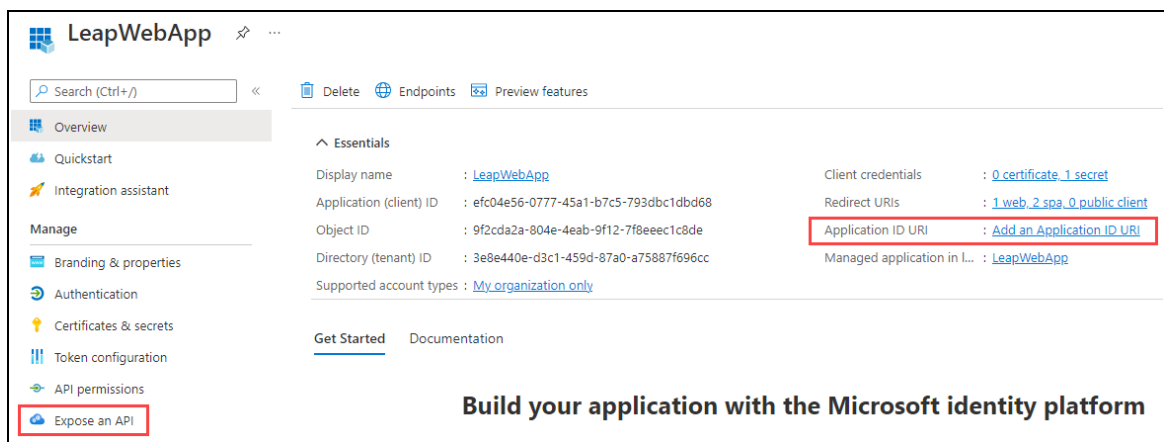
Expose the Polaris.ApplicationServices API

To expose the Polaris.ApplicationServices API

1. On the LeapWebApp page, select **Add an Application ID URI**.

Note:

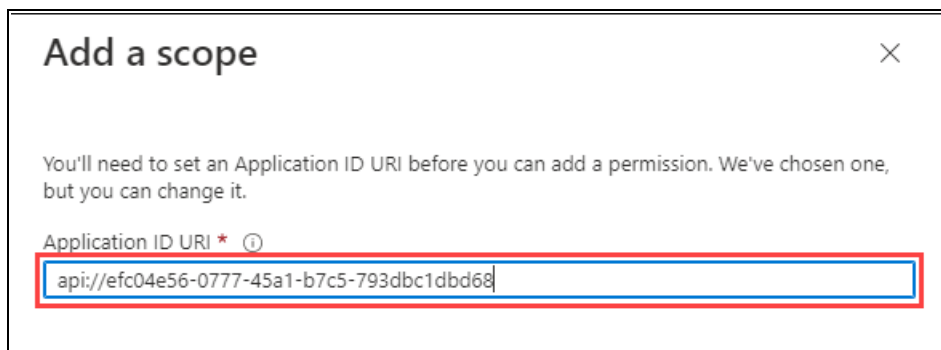
You can also select **Expose an API** in the list of options at the left side of the screen.



The Expose an API page appears.

2. Select **Add a scope**.

The Add a scope dialog appears. The **Application ID URI** field contains an automatically-generated URI.



3. Select **Save and continue**.

The Add a scope dialog refreshes.

Add a scope ✕

Scope name * ⓘ
pas ✓
api://efc04e56-0777-45a1-b7c5-793dbc1dbd68/pas

Who can consent? ⓘ
☒ Admins and users ☐ Admins only

Admin consent display name * ⓘ
Access Polaris.ApplicationServices ✓

Admin consent description * ⓘ
Allows the app to access the Polaris.ApplicationServices web API. ✓

4. Enter "pas" in the **Scope name** field.
5. Enter "Access Polaris.ApplicationServices" in the **Admin consent display name** field.
6. Enter "Allows the app to access the Polaris.ApplicationServices web API." in the **Admin consent description** field.
7. Select **Add scope**.

The Azure portal saves the scope and closes the Add a scope dialog.

8. On the Expose an API page, select **Add a client application**.

The Add a client application dialog appears.

Add a client application [X]

Client ID ⓘ
f9f429e3-8355-4eb3-876b-0e45b6ddd295 ✓

Authorized scopes ⓘ
☒ api://efc04e56-0777-45a1-b7c5-793dbc1dbd68/pas

9. In the **Client ID** field, enter the Application (client) ID that you copied and saved during the [Register LeapWebApp with Azure AD](#) procedure.
10. Select the **Authorized scopes** checkbox.
11. Select **Add application**.

The Azure portal saves your changes and closes the Add a client application dialog.

12. On the Expose an API page, copy the new scope, then paste it into Notepad (or a similar text editor) and save the file. Your value will be similar to this one:

api://efc04e56-0777-45a1-b7c5-793dbc1dbd68/pas

You must have this value to complete two procedures later in the Azure AD configuration process.

Configure an ID Token

To allow Leap to sign out of specific accounts, you must add an ID token that contains the login_hint claim.

To configure an ID token

1. On the LeapWebApp Overview page, select **Token configuration** from the list of options at the left side of the screen.

The Token configuration page appears.

2. Select **Add optional claim**.

The Add optional claim dialog appears.

Add optional claim ×

Once a token type is selected, you may choose from a list of available optional claims.

*** Token type**

Access and ID tokens are used by applications for authentication. [Learn more](#)

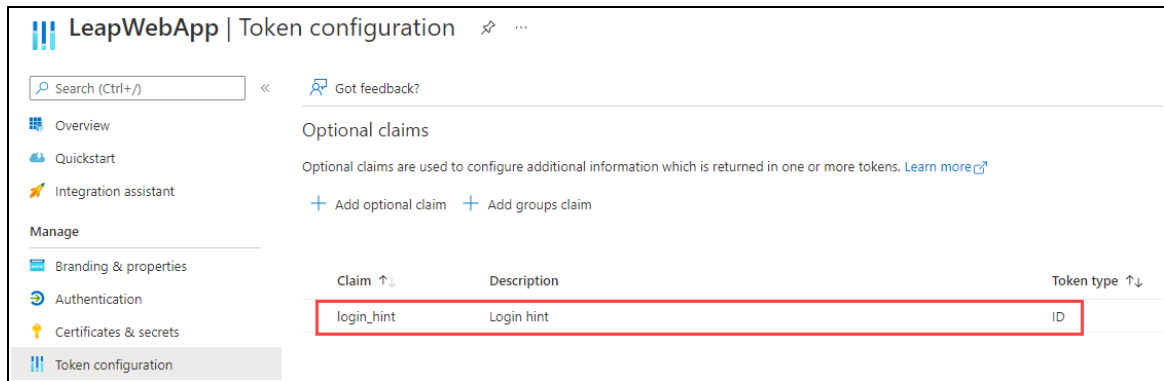
☒ ID
☐ Access
☐ SAML

<input checked="" type="checkbox"/> Claim ↑↓	Description
<input type="checkbox"/> acct	User's account status in tenant
<input type="checkbox"/> auth_time	Time when the user last authenticated; See OpenID Con...
<input type="checkbox"/> ctry	User's country/region
<input type="checkbox"/> email	The addressable email for this user, if the user has one
<input type="checkbox"/> family_name	Provides the last name, surname, or family name of the ...
<input type="checkbox"/> fwd	IP address
<input type="checkbox"/> given_name	Provides the first or "given" name of the user, as set on ...
<input type="checkbox"/> in_corp	Signals if the client is logging in from the corporate net...
<input type="checkbox"/> ipaddr	The IP address the client logged in from
<input checked="" type="checkbox"/> login_hint	Login hint
<input type="checkbox"/> onprem_sid	On-premises security identifier

- Set the **Token type** setting to the **ID** option.
- Select the **login_hint** checkbox.
- Select **Add**.

The Azure portal saves the token and closes the Add optional claim dialog.

- Verify that the new login_hint claim appears on the Token configuration page.



Set Up Users and Groups

To set up users and groups

1. On the Azure AD Overview page, select the **Enterprise applications** option from the list at the left side of the screen.

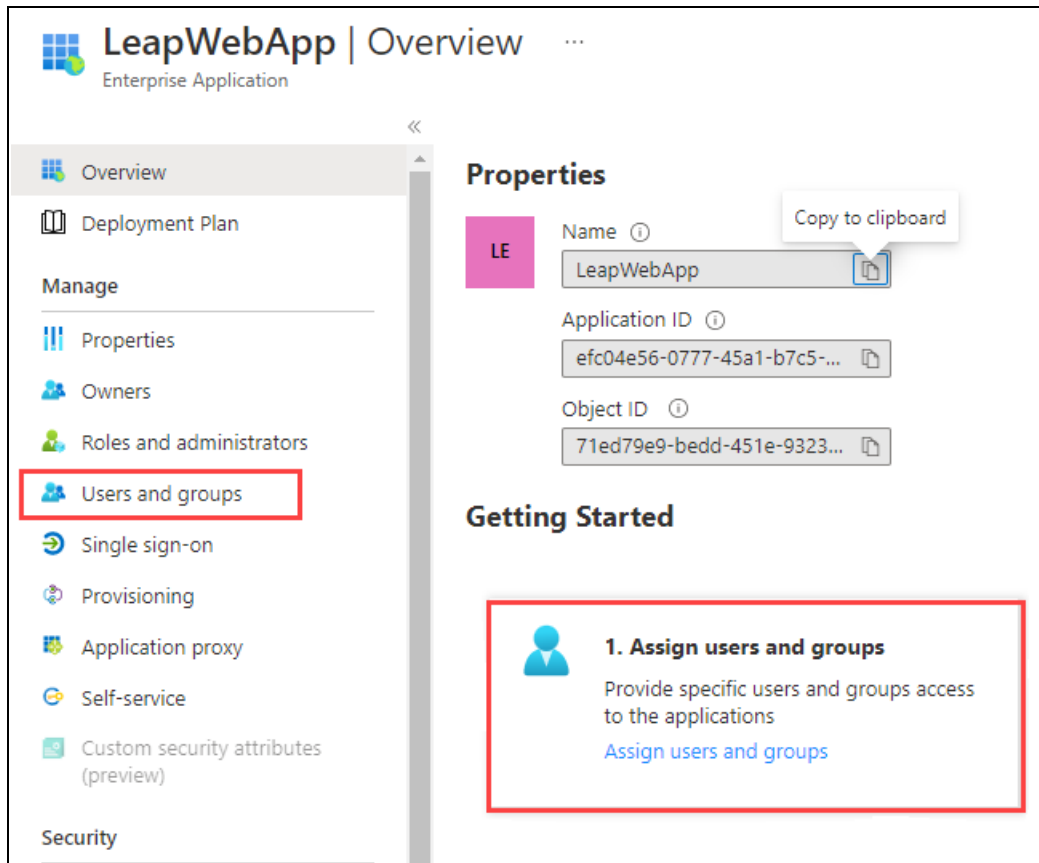
Note:

You can also select **Enterprise applications** from the list of services on the Azure portal home page.

The All applications page appears.

2. Select the **LeapWebApp** link.

The LeapWebApp Overview page appears.



3. Select the **Users and groups** option from the list at the left side of the screen. You can also select the **Assign users and groups** tile.

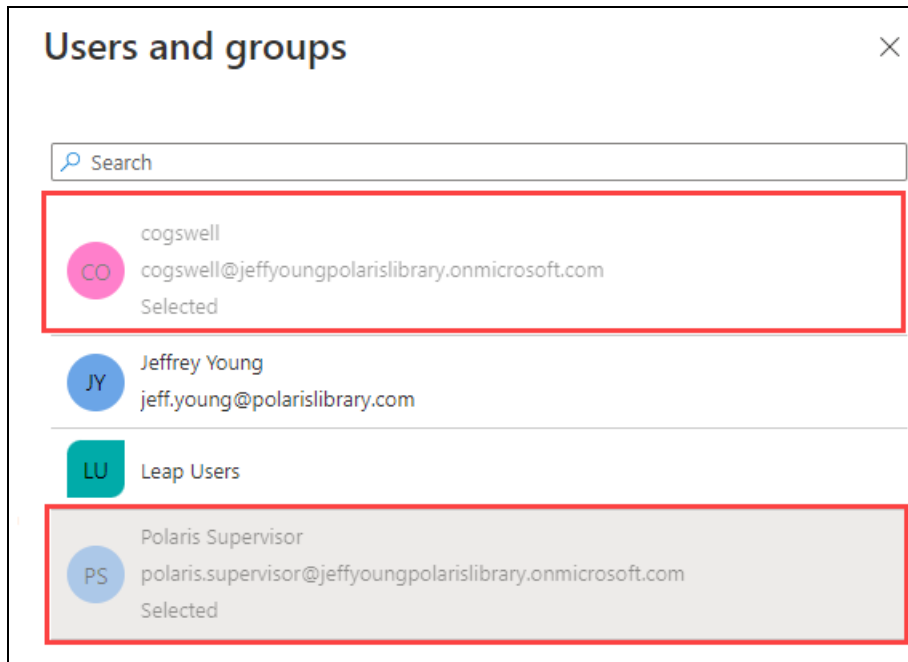
The Users and groups page appears.

4. Select **Add user/group**.

The Add Assignment page appears.

5. Select the **None Selected** link.

The Users and groups dialog appears.



6. Select the users and groups that you want to allow access to LeapWebApp.
7. Click **Select**.

The Users and groups dialog closes.

8. On the Add Assignment page, select **Assign**.

The Azure portal saves the user and group assignments.

Control Access to LeapWebApp Using Azure AD

To control access to LeapWebApp using Azure AD

1. On the Azure AD Overview page, select the **Enterprise applications** option from the list at the left side of the screen.

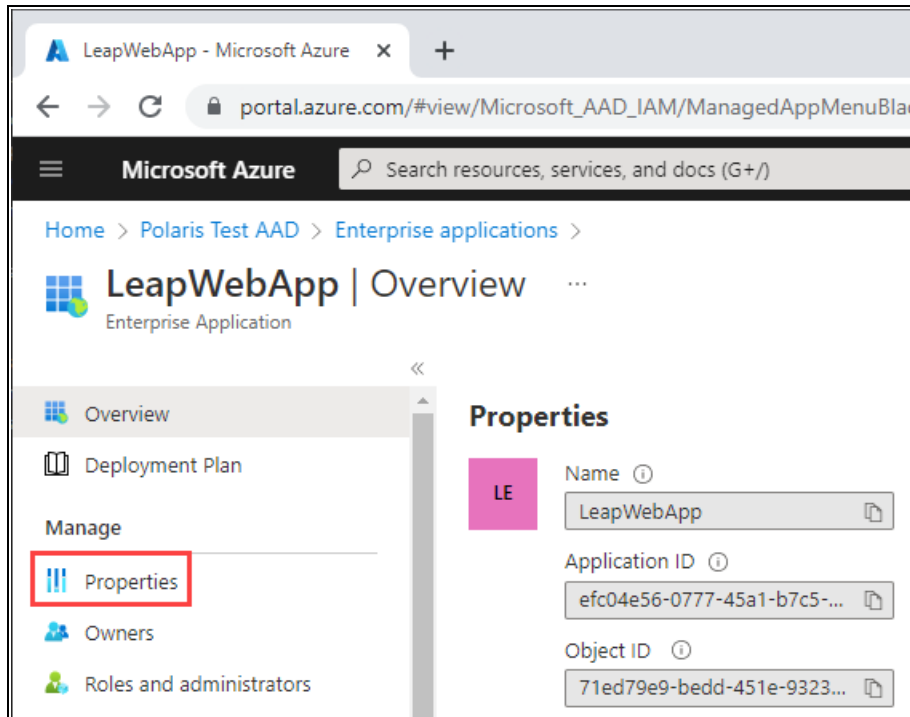
Note:

You can also select **Enterprise applications** from the list of services on the Azure portal home page.

The All applications page appears.

2. Select the **LeapWebApp** link.

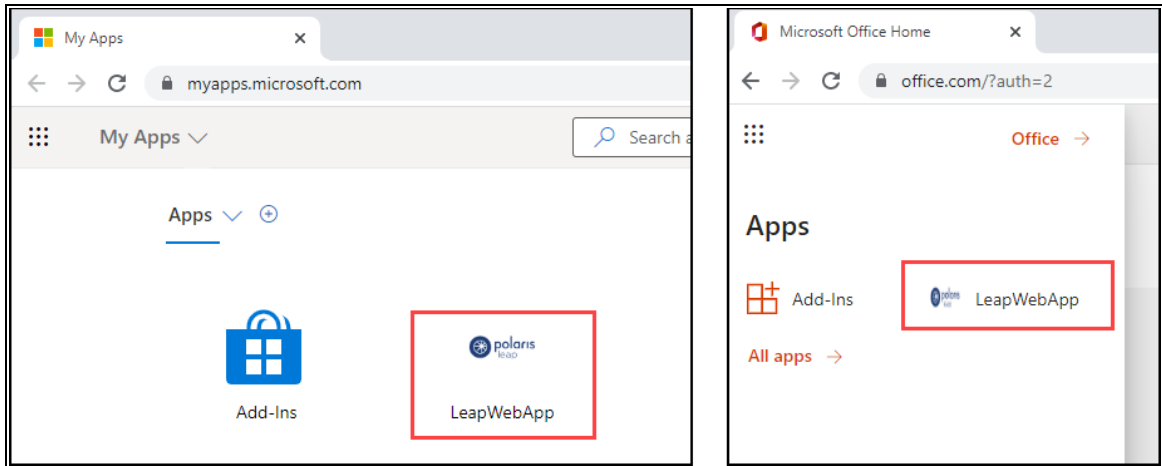
The LeapWebApp Overview page appears.



3. Select the **Properties** option from the list at the left side of the screen.

The Properties page appears.

4. Set the **Assignment requirement?** setting to **Yes**. This allows access to be controlled by the users and groups assigned to the LeapWebApp enterprise application. (When it is set to **No**, all users can sign in.)
5. Set the **Visible to users?** setting to **Yes**. This makes the LeapWebApp application visible to users in their Microsoft My Apps portal and on their Office 365 page.



Important:

When a user accesses LeapWebApp from the Microsoft My Apps portal or their Office 365 page, they might have to click the Polaris Leap Sign In button. This is because cookies are a part of the Leap authentication process.

6. Select **Save**.

The Azure portal saves your changes.

Set Up Web Services and Applications

To set up each of the following web services and applications, you must configure a .json file for each of the following:

- LeapWebApp (Leap)
- Polaris.ApplicationServices (Leap's API service)
- PolarisAdmin (the web-based Polaris System Administration application)
- Polaris.AdminServices (the API service)

The four .json files are named appsettings.user.json, but they reside in different directories:

- C:\Program Files\Polaris\7.3\LeapWebApp
- C:\Program Files\Polaris\7.3\Polaris.ApplicationServices

- C:\Program Files\Polaris\7.3\PolarisAdmin\assets
- C:\Program Files\Polaris\7.3\Polaris.AdminServices

This section contains the following topics:

- [Configure LeapWebApp for Use with Azure AD](#)
- [Configure Polaris.ApplicationServices for Use with Azure AD](#)
- [Configure PolarisAdmin for Use with Azure AD](#)
- [Configure Polaris.AdminServices for Use with Azure AD](#)

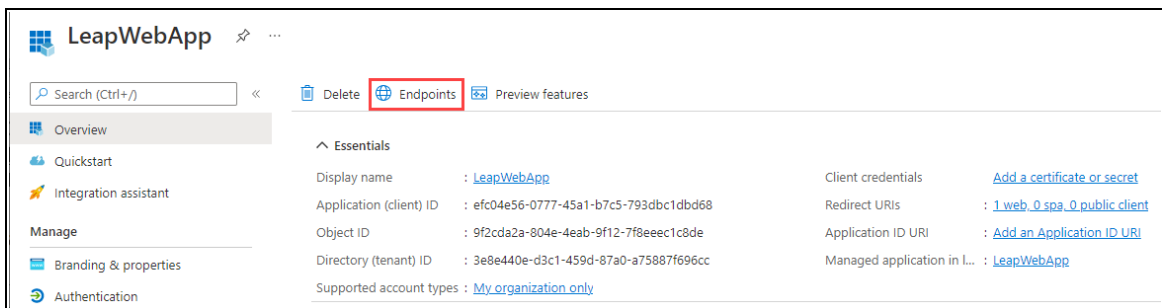
Configure LeapWebApp for Use with Azure AD

To configure LeapWebApp, you will update the C:\Program Files\Polaris\7.3\LeapWebApp\appsettings.user.json file using the following information:

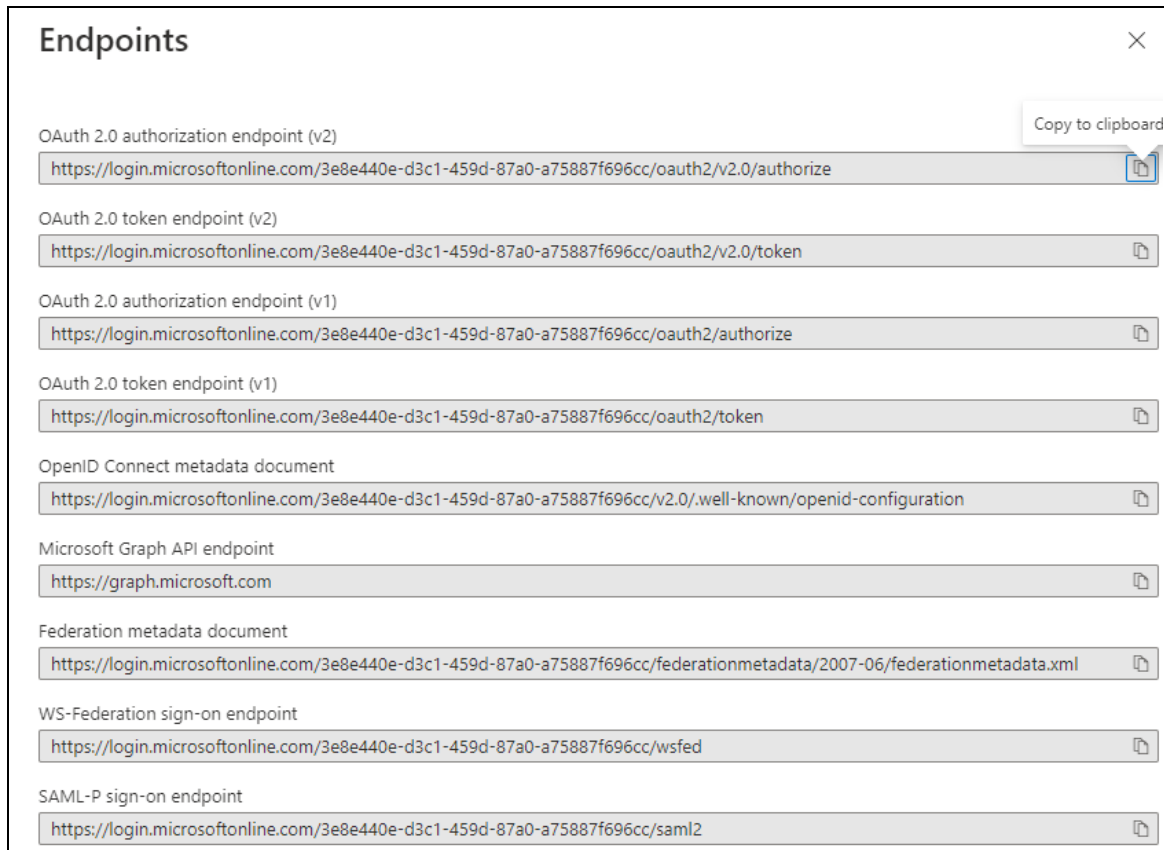
- Endpoint URIs copied from the Azure portal
- Values that you copied and saved during earlier steps in the Azure AD configuration process

To configure LeapWebApp

1. In the Azure portal, select **App registrations** from the list of options at the left side of the screen.
2. Select **LeapWebApp**.
3. On the LeapWebApp page, select **Endpoints**.



The Endpoints dialog appears. Leave this browser tab open so that you can copy endpoint URIs from it and paste them into the LeapWebApp appsettings.user.json file.



4. Open the C:\Program Files\Polaris\7.3\LeapWebApp\appsettings.user.json file.

Note:

You must run the editing application (for example, Notepad) as administrator.

5. On the Endpoints dialog, copy the root value and tenant ID from the **OAuth 2.0 authorization endpoint (v2)** box and paste it into the `Authority` property in the .json file. Your value will be similar to this one:

`https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/`

Note:

The value must include the trailing slash character.

6. In the .json file, replace the `ClientId` property value with the application (client) ID you copied during the [Register LeapWebApp with Azure AD](#) step. Your value will be similar to this one:

```
efc04e56-0777-45a1-b7c5-793dbc1dbd68
```

7. In the .json file, replace the `ClientSecret` property value with the client secret you copied and saved during the [Create Client Credentials](#) step. Your value will be similar to this one:

```
4I.8Q~-GpkdoymMQneGIYNg40FRjx2Hr1wWLDcbr
```

8. On the Endpoints dialog, copy the value from the **OpenID Connect metadata document** box and paste it into the `MetadataAddress` property in the .json file. Your value will be similar to this one:

```
https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/v2.0/.well-known/openid-configuration
```

9. On the Endpoints dialog, copy the FQDN of the Microsoft server from the **OpenID Connect metadata document** box and paste it into the `KnownAuthorities` property in the .json file. The value will be identical to this one:

```
login.microsoftonline.com
```

10. In the .json file, update the `Scopes` property to add the scope you copied and saved during the [Expose the Polaris.ApplicationServices API](#) step. Your value will be similar to this one:

```
api://efc04e56-0777-45a1-b7c5-793dbc1dbd68/pas
```

Note:

If you want Azure AD to return the user's email address, add "email" to the `Scopes` property. You might choose to do this if the `preferred_username` is a generic name, phone number, or other value.

11. In the .json file, set the `AlternateUpnClaimType` property to "preferred_username".

Note:

You can also set this property to "email", if you choose.

12. (Optional) Disable permission overrides in Leap. By default, permission overrides are enabled in Leap. If you want to disable overrides:

- Add the following value to the root level object in the .json file:

```
"client_permissionoverride_enabled": "false",
```

Note:

If you disable permission overrides in Leap, we recommend that you also update the language string that controls the title of the Permission Override dialog. For more information, search for the SW_CI_PERMISSION_OVRD_TTL language string in the Polaris Web Admin Tool (Language Editor) Guide.

13. Save the .json file. Your updated file should look similar to the example below.

```
"OAuth": {
  "Authority": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc",
  "ClientId": "efc04e56-0777-45a1-b7c5-793dbcd68",
  "ClientSecret": "4I.8Q~-GpkdoymMQneGIYNg40FRjx2HrlwWLDcbz",
  "MetadataAddress": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/v2.0/.well-known/openid-configuration",
  "KnownAuthorities": [ "login.microsoftonline.com" ],
  "CallbackPath": "/signin-oidc",
  "SignedOutCallbackPath": "/signout-callback-oidc",
  "SignedOutRedirectUri": "/login",
  "RemoteAuthenticationTimeout": 1,
  "RemoteFailureRedirectUri": "/leapwebapp/logout",
  "ResponseMode": "form_post",
  "ResponseType": "code",
  "SaveTokens": false,
  "Scopes": [ "openid", "profile", "api://efc04e56-0777-45a1-b7c5-793dbcd68/pas" ],
  "UsePkce": true,
  "AlternateUpnClaimType": "preferred_username",
  "AlternateLogoutUri": null
},
```

14. Leave the browser tab displaying the Endpoints dialog open, and continue to the [Configure Polaris.ApplicationServices for Use with Azure AD](#) procedure.

Configure Polaris.ApplicationServices for Use with Azure AD

To configure Polaris.ApplicationServices, you will update the C:\Program Files\Polaris\7.3\Polaris.ApplicationServices\appsettings.user.json file using the following information:

- Endpoint URLs copied from the Azure portal
- Values that you copied and saved during earlier steps in the Azure AD configuration process

To configure Polaris.ApplicationServices

1. Open the C:\Program Files\Polaris\7.3\Polaris.ApplicationServices\appsettings.user.json file.

Note:

You must run the editing application (for example, Notepad) as administrator.

2. In the `Authorities` array, set the `Name` property to "AzureAD".
3. In the .json file, update the `Authority` property:
 - a. On the Endpoints dialog of the Azure AD portal, copy the value from the **OAuth 2.0 authorization endpoint (v2)** box but omit the trailing *authorize*.
 - b. Paste this value into the `Authority` property in the .json file. Your value will be similar to this one:

```
https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/
```

4. In the .json file, update the `Audience` property:
 - a. Locate the application (client) ID you copied and saved during the [Register LeapWebApp with Azure AD](#) step.
 - b. Use it to construct a string with the following format:
`api://[application (client) ID]`
 - c. Paste this value into the `Audience` property. Your value will be similar to this one:

```
api://efc04e56-0777-45a1-b7c5-793dbc1dbd68
```

5. On the Endpoints dialog, copy the value from the **OpenID Connect metadata document** box and paste it into the `MetaAddress` property in the .json file. Your value will be similar to this one:

```
https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/well-known/openid-configuration
```

6. In the .json file, add a value to the `ValidIssuers` property:

- a. On the Endpoints dialog, copy the tenant ID from the **OAuth 2.0 authorization endpoint (v2)** box.

- b. Use it to construct a URI with the following format:

`https://sts.windows.net/[tenant ID]`

- c. Paste this value into the `ValidIssuers` property. Your value will be similar to this one:

`https://sts.windows.net/3e8e440e-d3c1-459d-87a0-a75887f696cc/`

7. In the .json file, add a second value to the `ValidIssuers` property:

- a. On the Endpoints dialog, copy the value from the **OAuth 2.0 authorization endpoint (v2)** box but omit the trailing *authorize*.

- b. Paste this value into the `ValidIssuers` property in the .json file. Your value will be similar to this one:

`https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0`

8. In the .json file, add a value to the `ValidAudiences` property:

- a. Locate the application (client) ID you copied and saved during the [Register LeapWebApp with Azure AD](#) step.

- b. Paste this value into the `ValidAudiences` property. Your value will be similar to this one:

`efc04e56-0777-45a1-b7c5-793dbc1dbd68`

9. In the .json file, add a second value to the `ValidAudiences` property:

- a. Locate the application (client) ID you copied and saved during the [Register LeapWebApp with Azure AD](#) step.

- b. Use it to construct a string with the following format:

`api://[application (client) ID]`

- c. Paste this value into the `ValidAudiences` property. Your value will be similar to this one:

`api://efc04e56-0777-45a1-b7c5-793dbc1dbd68`

10. In the `.json` file, update the `UPNClaimTypes` property to add `"upn"` and `"preferred_username"` if those values are not already present. Your updated values in the `Authorities` array should look similar to the example below.

```
{
  "Name": "AzureAD",
  "Authority": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/",
  "Audience": "api://efc04e56-0777-45a1-b7c5-793dbc1dbd68",
  "MetaAddress": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/v2.0/.well-known/openid-configuration",
  "RequireHttpsMetadata": true,
  "RequireSignedTokens": true,
  "ValidateIssuer": true,
  "ValidIssuers": [
    "https://sts.windows.net/3e8e440e-d3c1-459d-87a0-a75887f696cc/",
    "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/v2.0"
  ],
  "ValidateAudience": true,
  "ValidAudiences": [
    "efc04e56-0777-45a1-b7c5-793dbc1dbd68",
    "api://efc04e56-0777-45a1-b7c5-793dbc1dbd68"
  ],
  "UPNClaimTypes": [ "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn", "upn", "preferred_username" ]
}
```

11. In the `Swagger` property in the `.json` file, update the `ClientID` property:
- Locate the application (client) ID you copied and saved during the [Register LeapWebApp with Azure AD](#) step.

- Paste this value into the `ClientID` property. Your value will be similar to this one:

efc04e56-0777-45a1-b7c5-793dbc1dbd68

12. On the Endpoints dialog, copy the value from the **OAuth 2.0 authorization endpoint (v2)** box and paste it into the `AuthorizationUrl` property in the `.json` file. Your value will be similar to this one:

https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/authorize

13. In the `.json` file, update the `TokenUrl` property:
- On the Endpoints dialog, copy the value from the **OAuth 2.0 token endpoint (v2)** box but replace *authorize* with *token*.
 - Paste the value into the `TokenUrl` property in the `.json` file. Your value will be similar to this one:


```
https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/token
```

14. In the .json file, update the `RefreshTokenUrl` property:

- a. On the Endpoints dialog, copy the value from the **OAuth 2.0 token endpoint (v2)** box but replace *authorize* with *token*.
- b. Paste the value into the `RefreshTokenUrl` property in the .json file. Your value will be similar to this one:

```
https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/token
```

15. In the .json file, update the `LogoutUrl` property:

- a. On the Endpoints dialog, copy the value from the **OAuth 2.0 authorization endpoint (v2)** box but replace *authorize* with *logout*.
- b. Paste the value into the `LogoutUrl` property in the .json file. Your value will be similar to this one:

```
https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/logout
```

16. In the .json file, add a new scope `Name` and `Description`:

- a. Locate the scope you copied and saved during the [Expose the Polaris.ApplicationServices API](#) step.
- b. Copy the scope and use it to construct a new `Name` property. Your value will be similar to this one:

```
"Name": "api://efc04e56-0777-45a1-b7c5-793dbc1dbd68/pas"
```

- c. Paste the `Name` property into the .json file.
- d. Add a `Description` property that matches the example below:

```
"Description": "API Scope defined in AzureAD for LeapWebApp"
```

17. Save the .json file. Your updated values in the `Swagger` property should look similar to the example below.

```
"Swagger": {
  "ClientID": "efc04e56-0777-45a1-b7c5-793dbcd68",
  "ClientSecret": "",
  "AppName": "Polaris.ApplicationServices",
  "AuthorizationUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/authorize",
  "TokenUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/token",
  "RefreshTokenUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/token",
  "LogoutUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/logout",
  "Scopes": [
    {
      "Name": "openid",
      "Description": "Use OIDC to verify the user's identity"
    },
    {
      "Name": "email",
      "Description": "Optional to return user's email address"
    },
    {
      "Name": "urn:microsoft:userinfo",
      "Description": "urn:microsoft:userinfo"
    },
    {
      "Name": "api://efc04e56-0777-45a1-b7c5-793dbcd68/pas",
      "Description": "API Scope defined in AzureAD for LeapWebApp"
    }
  ]
}
```

Note:

Changes to the appsettings.user.json files do not take effect until the IIS application pools are restarted or IIS is reset.

Configure PolarisAdmin for Use with Azure AD

To configure PolarisAdmin, you will update the C:\Program Files\Polaris\7.3\PolarisAdmin\assets\appsettings.user.json file. You will use several values copied from the Azure AD portal.

Important:

By default, the appsettings.user.json template file contains configuration settings that apply to AD FS. The latest 7.3 patch includes a RELEASE-NOTES.md file that contains the template settings that apply to Azure AD.

To configure PolarisAdmin

1. Open the C:\Program Files\Polaris\7.3\PolarisAdmin\assets\appsettings.user.json file.

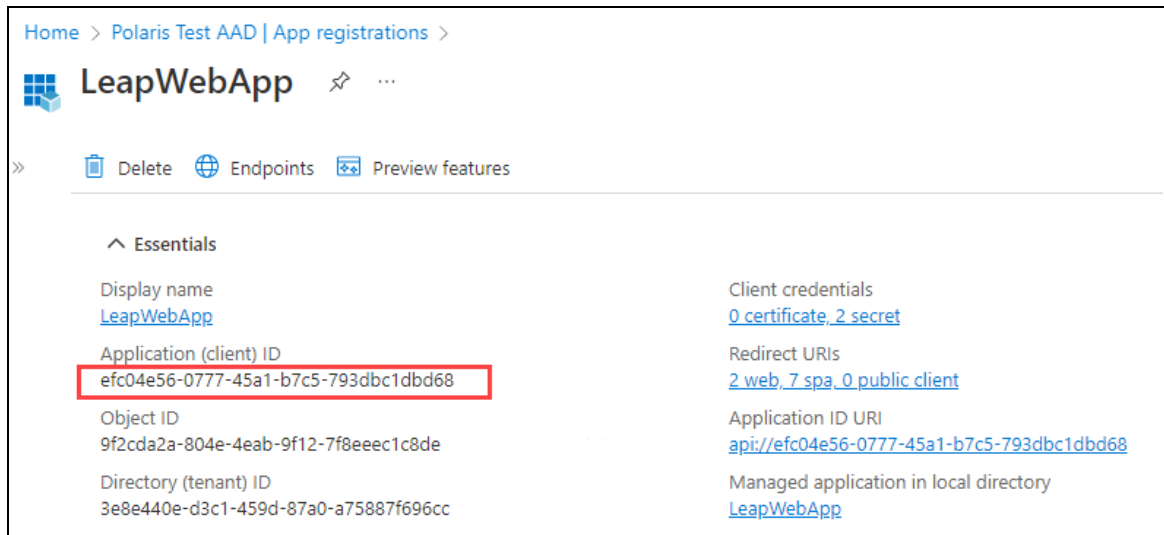
Note:

You must run the editing application (for example, Notepad) as administrator.

2. Verify that `oauthEnabled` is set to `true`.

```
{
  "apiUrlRoot": "https://rd-polaris.polarislibrary.com/polaris.adminservices/api/",
  "oauthEnabled": true,
  "msal": {
    "auth": {
      "clientId": "0a586b1e-eeb0-4c8a-8381-50e9cafec240",
      "authority": "https://dev-fs.polarislibrary.com/adfs/",
      "knownAuthorities": ["dev-fs.polarislibrary.com"],
      "redirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin/oauth-success",
      "postLogoutRedirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin",
      "protocolMode": "OIDC",
      "navigateToLoginRequestUrl": false
    },
    "cache": {
      "cacheLocation": "localStorage",
      "storeAuthStateInCookie": false,
      "secureCookies": true
    },
    "guard": {
      "interactionType": "redirect",
      "authRequest": {
        "scopes": ["openid", "profile", "email", "urn:microsoft:userinfo"]
      },
      "loginFailedRoute": "/login-failed"
    },
    "interceptor": {
      "interactionType": "redirect",
      "protectedResourceMap": [
        ["https://rd-polaris.polarislibrary.com/Polaris.AdminServices/api/protected/", ["email"]]
      ]
    }
  }
}
```

3. In the Azure portal, copy the Application (client) ID.




4. Paste the copied client ID into the appsettings.user.json file. If you started from the template settings provided in the RELEASE-NOTES.md file, replace *[CLIENTID-ASSIGNED-IN-AZUREAD]* with the copied client ID.




When complete, your file should look like the following example (your client ID will be different):

```
{
  "apiUrlRoot": "https://rd-polaris.polarislibrary.com/Polaris.AdminServices/api/",
  "oauthEnabled": true,
  "msal": {
    "auth": {
      "clientId": "efc04e56-0777-45a1-b7c5-793dbclbdb68",
      "authority": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/v2.0/",
      "knownAuthorities": [ "login.microsoftonline.com" ],
      "redirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin/oauth-success",
      "postLogoutRedirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin",
      "protocolMode": "OIDC",
      "navigateToLoginRequestUrl": false
    },
    "cache": {
      "cacheLocation": "localStorage",
      "storeAuthStateInCookie": false,
      "secureCookies": true
    },
    "guard": {
      "interactionType": "redirect",
      "authRequest": {
        "scopes": [ "openid", "profile" ]
      },
      "loginFailedRoute": "/login-failed"
    },
    "interceptor": {
      "interactionType": "redirect",
      "protectedResourceMap": [
        [ "https://rd-polaris.polarislibrary.com/Polaris.AdminServices/api/protected/",
          [ "openid", "profile", "api://efc04e56-0777-45a1-b7c5-793dbclbdb68/pas" ] ]
      ]
    }
  }
}
```

5. In the .json file, update the authority to use login.microsoftonline.com.
6. Update the knownAuthorities to use login.microsoftonline.com.
7. In the Azure portal, copy the Directory (tenant) ID.

Home > Polaris Test AAD | App registrations >

 **LeapWebApp** ✨ ...

»  Delete  Endpoints  Preview features

^ Essentials

Display name LeapWebApp	Client credentials 0 certificate, 2 secret
Application (client) ID efc04e56-0777-45a1-b7c5-793dbclbdb68	Redirect URIs 2 web, 7 spa, 0 public client
Object ID 9f2cda2a-804e-4eab-9f12-7f8eeec1c8de	Application ID URI api://efc04e56-0777-45a1-b7c5-793dbclbdb68
Directory (tenant) ID 3e8e440e-d3c1-459d-87a0-a75887f696cc	Managed application in local directory LeapWebApp

8. In the .json file, paste the copied tenant ID into the `authority`. If you started from the template settings provided in the RELEASE-NOTES.md file, replace `[TENANTID-ASSIGNED-IN-AZUREAD]` with the copied tenant ID.

When complete, your file should look like the following example (your tenant ID will be different):

```
{
  "apiUrlRoot": "https://rd-polaris.polarislibrary.com/Polaris.AdminServices/api/",
  "oauthEnabled": true,
  "msal": {
    "auth": {
      "clientId": "efc04e56-0777-45a1-b7c5-793dbclbdb68",
      "authority": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/v2.0/",
      "knownAuthorities": [ "login.microsoftonline.com" ],
      "redirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin/oauth-success",
      "postLogoutRedirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin",
      "protocolMode": "OIDC",
      "navigateToLoginRequestUrl": false
    },
    "cache": {
      "cacheLocation": "localStorage",
      "storeAuthStateInCookie": false,
      "secureCookies": true
    },
    "guard": {
      "interactionType": "redirect",
      "authRequest": {
        "scopes": [ "openid", "profile" ]
      },
      "loginFailedRoute": "/login-failed"
    },
    "interceptor": {
      "interactionType": "redirect",
      "protectedResourceMap": [
        [ "https://rd-polaris.polarislibrary.com/Polaris.AdminServices/api/protected/",
          [ "openid", "profile", "api://efc04e56-0777-45a1-b7c5-793dbclbdb68/pas" ] ]
      ]
    }
  }
}
```

9. In the .json file, update the server location in the `redirectUri` and the `postLogoutRedirectUri`. If you started from the template settings provided in the RELEASE-NOTES.md file, replace `[POLADMIN-SERVER-ADDR]` with the FQDN of the server that hosts the Polaris System Administration (web-based) user interface.

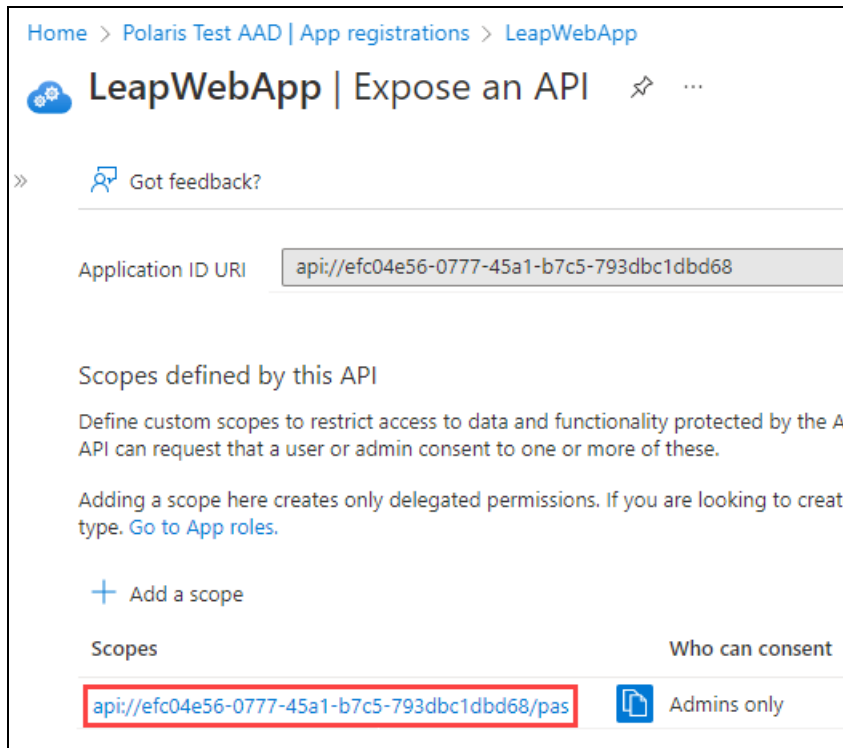
When complete, your file should look like the following example (your server address will be different):

```
{
  "apiUrlRoot": "https://rd-polaris.polarislibrary.com/Polaris.AdminServices/api/",
  "oauthEnabled": true,
  "msal": {
    "auth": {
      "clientId": "efc04e56-0777-45a1-b7c5-793dbclbdb68",
      "authority": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/v2.0/",
      "knownAuthorities": [ "login.microsoftonline.com" ],
      "redirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin/oauth-success",
      "postLogoutRedirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin",
      "protocolMode": "OIDC",
      "navigateToLoginRequestUrl": false
    },
    "cache": {
      "cacheLocation": "localStorage",
      "storeAuthStateInCookie": false,
      "secureCookies": true
    },
    "guard": {
      "interactionType": "redirect",
      "authRequest": {
        "scopes": [ "openid", "profile" ]
      },
      "loginFailedRoute": "/login-failed"
    },
    "interceptor": {
      "interactionType": "redirect",
      "protectedResourceMap": [
        [ "https://rd-polaris.polarislibrary.com/Polaris.AdminServices/api/protected/",
          [ "openid", "profile", "api://efc04e56-0777-45a1-b7c5-793dbclbdb68/pas" ] ]
      ]
    }
  }
}
```

10. In the .json file, update the server location in the `protectedResourceMap`. If you started from the template settings provided in the `RELEASE-NOTES.md` file, replace `[POLADMIN SVC-SERVER-ADDR]` with the FQDN of the server that hosts the API service for Polaris System Administration (web-based).

When complete, your file should look like the example shown in step 13 (your server address will be different).

11. In the Azure portal, LeapWebApp section of the App registrations dashboard and copy the API scope.



12. In the .json file, paste the copied scope into the `protectedResourceMap` array.

When complete, your file should look like the example shown in step 13.

13. In the `protectedResourceMap` array, verify that:

- The `openid` and `profile` scopes are listed.
- The `email` scope is not listed.

When complete, your file should look like the following example:


```
{
  "apiUrlRoot": "https://rd-polaris.polarislibrary.com/Polaris.AdminServices/api/",
  "oauthEnabled": true,
  "msal": {
    "auth": {
      "clientId": "efc04e56-0777-45a1-b7c5-793dbclbdb68",
      "authority": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/v2.0/",
      "knownAuthorities": [ "login.microsoftonline.com" ],
      "redirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin/oauth-success",
      "postLogoutRedirectUri": "https://rd-polaris.polarislibrary.com/PolarisAdmin",
      "protocolMode": "OIDC",
      "navigateToLoginRequestUrl": false
    },
    "cache": {
      "cacheLocation": "localStorage",
      "storeAuthStateInCookie": false,
      "secureCookies": true
    },
    "guard": {
      "interactionType": "redirect",
      "authRequest": {
        "scopes": [ "openid", "profile" ]
      },
      "loginFailedRoute": "/login-failed"
    },
    "interceptor": {
      "interactionType": "redirect",
      "protectedResourceMap": [
        [ "https://rd-polaris.polarislibrary.com/Polaris.AdminServices/api/protected/",
          [ "openid", "profile", "api://efc04e56-0777-45a1-b7c5-793dbclbdb68/pas" ] ]
      ]
    }
  }
}
```

14. Save the .json file.

Configure Polaris.AdminServices for Use with Azure AD

To configure Polaris.AdminServices, you will update the C:\Program Files\Polaris\7.3\Polaris.ApplicationServices\appsettings.user.json file. You will use several values copied from the Azure AD portal.

Important:

By default, the appsettings.user.json template file contains configuration settings that apply to AD FS. The latest 7.3 patch includes a RELEASE-NOTES.md file that contains the template settings that apply to Azure AD.

To configure Polaris.AdminServices

1. Open the C:\Program Files\Polaris\7.3\Polaris.AdminServices\appsettings.user.json file.

Note:

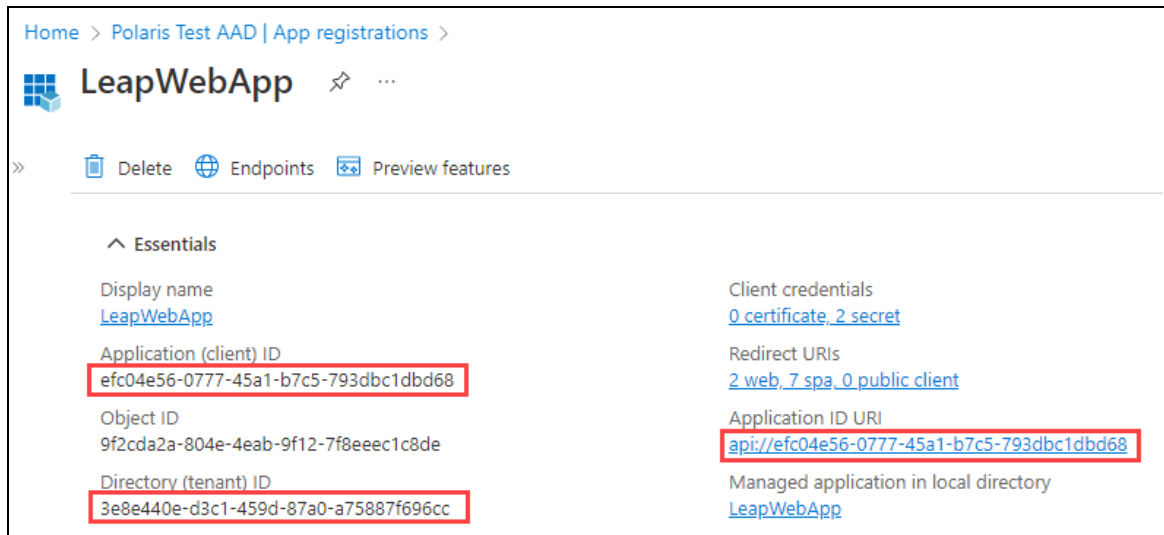
You must run the editing application (for example, Notepad) as administrator.

2. Verify that `Enabled` is set to `true` under the `OAuth` object.

```
"Polaris": {
  "CachePermissions": false,
  "CORS": {
    "AllowedHosts": "https://rd-polaris.polarislibrary.com"
  },
  "ShowPII": true,
  "SwaggerEnabled": true,
  "BasicAuth": {
    "Enabled": false
  },
  "OAuth": {
    "Enabled": true,
    "Authorities": [
      {
        "Name": "AzureAD",
```

3. In the Azure portal, go to the App registrations dashboard and retrieve the following values:
 - Application (client) ID
 - Directory (tenant) ID
 - Application ID URI

You will use these values in the steps below.



4. Paste the application (client) ID into the appsettings.user.json file. If you started from the template settings provided in the RELEASE-NOTES.md file, replace *[CLIENTID-ASSIGNED-IN-AZUREAD]* with the client ID from the Azure portal.

When complete, your file should look like the following example (your client ID will be different):

```
"Polaris": {
  "CachePermissions": false,
  "CORS": {
    "AllowedHosts": "https://rd-polaris.polarislibrary.com"
  },
  "ShowPII": true,
  "SwaggerEnabled": true,
  "BasicAuth": {
    "Enabled": false
  },
  "OAuth": {
    "Enabled": true,
    "Authorities": [
      {
        "Name": "AzureAD",
        "Authority": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/",
        "Audience": "api://efc04e56-0777-45a1-b7c5-793dbcd68",
        "MetaAddress": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/v2.0/.well-known/openid-configuration",
        "RequireHttpsMetadata": true,
        "RequireSignedTokens": true,
        "ValidateIssuer": true,
        "ValidIssuers": [
          "https://sts.windows.net/3e8e440e-d3c1-459d-87a0-a75887f696cc/",
          "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/v2.0"
        ],
        "ValidateAudience": true,
        "ValidAudiences": [
          "efc04e56-0777-45a1-b7c5-793dbcd68",
          "api://efc04e56-0777-45a1-b7c5-793dbcd68"
        ],
        "UPNClaimTypes": [ "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn", "upn", "preferred_username" ]
      }
    ],
    "Swagger": {
      "ClientID": "efc04e56-0777-45a1-b7c5-793dbcd68",
      "ClientSecret": "",
      "AppName": "Polaris.ApplicationServices",
      "AuthorizationUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/authorize",
      "TokenUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/token",
      "RefreshTokenUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/token",
      "LogoutUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/logout",
      "Scopes": [
        {
          "Name": "openid",
          "Description": "Use OIDC to verify the user's identity"
        },
        {
          "Name": "email",
          "Description": "Optional to return user's email address"
        },
        {
          "Name": "api://efc04e56-0777-45a1-b7c5-793dbcd68/pas",
          "Description": "API Scope defined in AzureAD"
        }
      ]
    }
  }
}
```

5. Paste the directory (tenant) ID into the appsettings.user.json file. If you started from the template settings provided in the RELEASE-NOTES.md file, replace *[TENANTID-ASSIGNED-IN-AZUREAD]* with the tenant ID from the Azure portal.

When complete, your file should look like the following example (your tenant ID will be different):

```
"Polaris": {
  "CachePermissions": false,
  "CORS": {
    "AllowedHosts": "https://rd-polaris.polarislibrary.com"
  },
  "ShowPII": true,
  "SwaggerEnabled": true,
  "BasicAuth": {
    "Enabled": false
  },
  "OAuth": {
    "Enabled": true,
    "Authorities": [
      {
        "Name": "AzureAD",
        "Authority": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/",
        "Audience": "api://efc04e56-0777-45a1-b7c5-793dbcd68",
        "MetaAddress": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/v2.0/.well-known/openid-configuration",
        "RequireHttpsMetadata": true,
        "RequireSignedTokens": true,
        "ValidateIssuer": true,
        "ValidIssuers": [
          "https://sts.windows.net/3e8e440e-d3c1-459d-87a0-a75887f696cc/",
          "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/v2.0"
        ],
        "ValidateAudience": true,
        "ValidAudiences": [
          "efc04e56-0777-45a1-b7c5-793dbcd68",
          "api://efc04e56-0777-45a1-b7c5-793dbcd68"
        ],
        "UPNClaimTypes": [ "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn", "upn", "preferred_username" ]
      }
    ],
    "Swagger": {
      "ClientID": "efc04e56-0777-45a1-b7c5-793dbcd68",
      "ClientSecret": "",
      "AppName": "Polaris.ApplicationServices",
      "AuthorizationUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/authorize",
      "TokenUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/token",
      "RefreshTokenUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/token",
      "LogoutUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/logout",
      "Scopes": [
        {
          "Name": "openid",
          "Description": "Use OIDC to verify the user's identity"
        },
        {
          "Name": "email",
          "Description": "Optional to return user's email address"
        },
        {
          "Name": "api://efc04e56-0777-45a1-b7c5-793dbcd68/pas",
          "Description": "API Scope defined in AzureAD"
        }
      ]
    }
  }
}
```

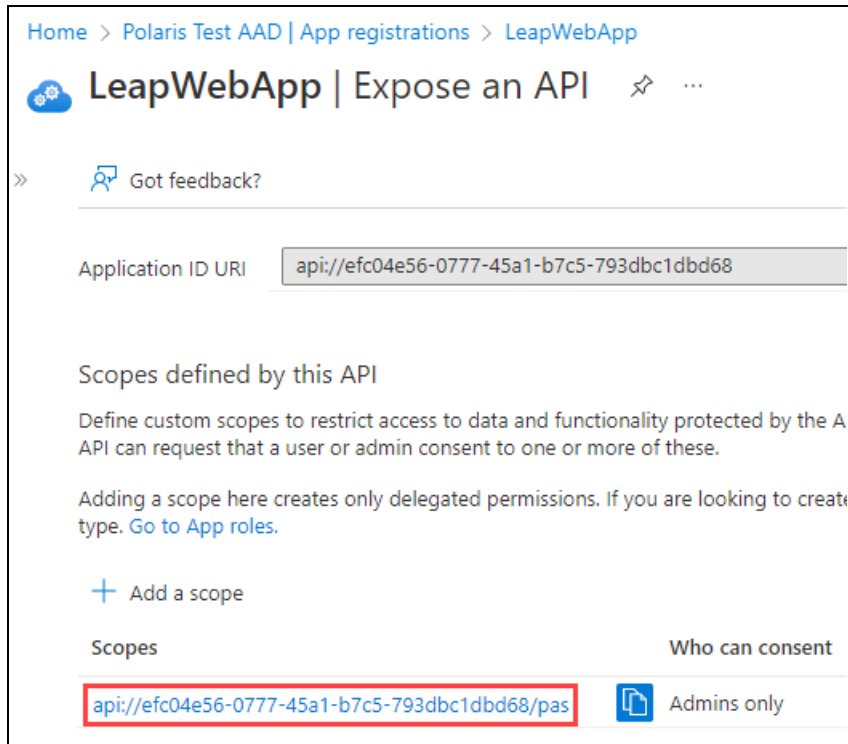
- Paste the application ID URI into the appsettings.user.json file. If you started from the template settings provided in the RELEASE-NOTES.md file, replace [APPID-URI-ASSIGNED-IN-AZUREAD] with the application ID URI from the Azure portal.

When complete, your file should look like the following example (your application ID URI will be different):

```
"Polaris": {
  "CachePermissions": false,
  "CORS": {
    "AllowedHosts": "https://rd-polaris.polarislibrary.com"
  },
  "ShowPII": true,
  "SwaggerEnabled": true,
  "BasicAuth": {
    "Enabled": false
  },
  "OAuth": {
    "Enabled": true,
    "Authorities": [
      {
        "Name": "AzureAD",
        "Authority": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/",
        "Audience": "api://efc04e56-0777-45a1-b7c5-793dbc1dbd68",
        "MetaAddress": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/v2.0/.well-known/openid-configuration",
        "RequireHttpsMetadata": true,

```

7. In the Azure portal, go to the LeapWebApp section of the App registrations dashboard and copy the API scope.



8. Paste the API scope into the appsettings.user.json file.

When complete, your file should look like the following example (your scope value will be different):

```
"Swagger": {
  "ClientID": "efc04e56-0777-45a1-b7c5-793dbc1dbd68",
  "ClientSecret": "",
  "AppName": "Polaris.ApplicationServices",
  "AuthorizationUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/authorize",
  "TokenUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/token",
  "RefreshTokenUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/token",
  "LogoutUrl": "https://login.microsoftonline.com/3e8e440e-d3c1-459d-87a0-a75887f696cc/oauth2/v2.0/logout",
  "Scopes": [
    {
      "Name": "openid",
      "Description": "Use OIDC to verify the user's identity"
    },
    {
      "Name": "email",
      "Description": "Optional to return user's email address"
    },
    {
      "Name": "api://efc04e56-0777-45a1-b7c5-793dbc1dbd68/pas",
      "Description": "API Scope defined in AzureAD"
    }
  ]
}
```

9. Save the .json file.

Add a URL Rewrite Rule for LeapWebApp

Adding a URL rewrite rule redirects incoming URLs to the correct address for the LeapWebApp. This must be done manually, since the library may already use other URL rewrite rules.

To add a URL rewrite rule, you must have the Microsoft IIS URL Rewrite 2.1 extension. For more information, see <https://www.iis.net/downloads/microsoft/url-rewrite>.

To add a URL rewrite rule

1. Open the root IIS web.config file, found in the following location:
C:\inetpub\wwwroot\web.config
2. Add a rewrite rule to the **system.webServer** node.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <rewrite>
      <rules>
        <rule name="UrlToLowercase" stopProcessing="true">
          <match url="(.*)" ignoreCase="true" />
          <action type="Redirect" url="https://{HTTP_HOST}{ToLower:{PATH_INFO}}" redirectType="Found" appendQueryString="true" />
          <conditions>
            <add input="{PATH_INFO}" pattern="^/LeapWebApp(.*)|^/Leapwebapp(.*)|^/LEAPWEBAPP(.*)" ignoreCase="false" />
          </conditions>
        </rule>
      </rules>
    </rewrite>
  </system.webServer>
</configuration>
```

Note:

For sample rewrite rule text that you can copy and paste, see [Sample Rewrite Rule Text](#).

In the example above, if the incoming URL includes a path that contains any of the following, the rewrite rule redirects to /leapwebapp:

- /LeapWebApp
 - /Leapwebapp
 - /LEAPWEBAPP
3. Save the web.config file.

Note:

When registering redirect URIs for LeapWebApp in AD FS, the URIs should

be lowercase. For example:

- <https://rd-polaris.polarislibrary.com/leapwebapp/signin-oidc>
- <https://rd-polaris.polarislibrary.com/leapwebapp/signin-override-oidc>
- <https://rd-polaris.polarislibrary.com/leapwebapp/signout-callback-oidc>

Sample Rewrite Rule Text

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <rewrite>
      <rules>
        <rule name="UrlToLowercase" stopProcessing="true">
          <match url="(.*)" ignoreCase="true" />
          <action type="Redirect" url="https://{HTTP_HOST}
            {ToLower:{PATH_INFO}}" redirectType="Found"
            appendQueryString="true" />
          <conditions>
            <add input="{PATH_INFO}" pattern="^/LeapWebApp
              (.*)|^/Leapwebapp(.*)|^/LEAPWEBAPP(.*)"
              ignoreCase="false" />
          </conditions>
        </rule>
      </rules>
    </rewrite>
  </system.webServer>
</configuration>
```

Additional URL Rewrite Resources

See Microsoft's [URL Rewrite Module Configuration Reference](#) for additional information:

- <https://docs.microsoft.com/en-us/iis/extensions/url-rewrite-module/url-rewrite-module-configuration-reference>