Azure Active Directory Preview: Web Sign-On with Windows Azure Active Directory – PHP Walkthrough

Abstract: This document provides instructions on setting up a PHP web application and configuring it to leverage Windows Azure Active Directory to accept users from Office365 customers (the “WebSSO” initiative).

Table of Contents

[Overview 1](#_Toc320550438)

[Prerequisites 2](#_Toc320550439)

[Assets 2](#_Toc320550440)

[Generic Environment Requisites 2](#_Toc320550441)

[PHP-specific Requisites 2](#_Toc320550442)

[Walkthrough 2](#_Toc320550443)

[ISV - Creating a Simple PHP Application 2](#_Toc320550444)

[Customers - Provisioning a New Application in the Directory Tenant 7](#_Toc320550445)

[ISV - Protect the Application via WS-Federation and Onboard the First Customer 9](#_Toc320550446)

[ISV - Configuring the Application to Handle Sign On with Multiple Tenants 11](#_Toc320550447)

[Document History 15](#_Toc320550448)

# Overview

This document provides instructions on setting up an ASP.NET MVC web application and configuring it to leverage Azure Active Directory (AAD) to accept users from Office365 customers.  
The code and instructions are based on the features of the “WebSSO” initiative as it exists today and are meant to accomplish a short-term goal: helping the Azure Active Directory Preview participations to quickly get up to speed with AAD and the use of WS-Federation in their development platform of choice.

The walkthrough accomplishes the task by guiding the reader thru four distinct tasks, performed alternatively by two roles:

* Fabrikam, an ISV exposing the web application
* Awesome Computers and Trey Research, playing the role of Office365 customers

The four tasks are:

* ISV: Create one simple PHP web application
* Customer: Provision Fabrikam’s web application in their tenant
* ISV: Protect the application via WS-Federation and onboard the first customer
* ISV: Modify the application to handle sign on with multiple tenants

## Prerequisites

The walkthrough relies on various prerequisites being met by the software environment offered by the target machine.

### Assets

The walkthrough comes with some scripts and sample code which helps with some of the most tedious tasks. All the assets are available in the archive WAAD.WebSSO.PHP.ZIP.  
All the source code is available for the developer to study and tweak at will.

### Generic Environment Requisites

* Internet Information Services (IIS) 7.5 (SSL enabled)
* Windows Powershell
* Office 365 PS CmdLets ( <http://onlinehelp.microsoft.com/en-us/office365-enterprises/hh124998.aspx> )

### PHP-specific Requisites

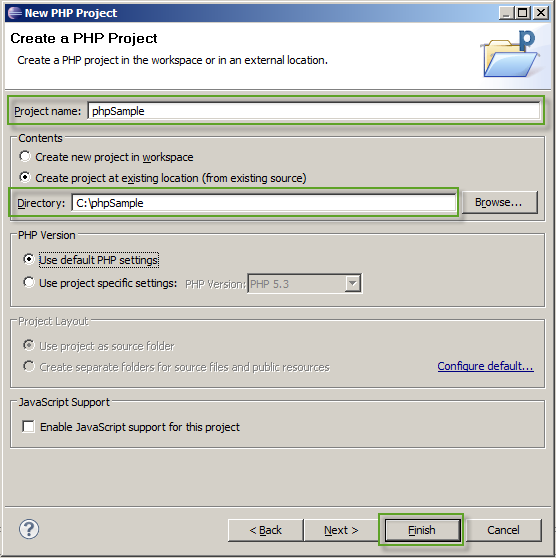
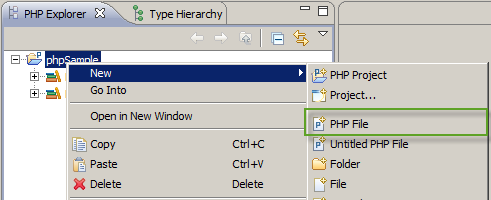
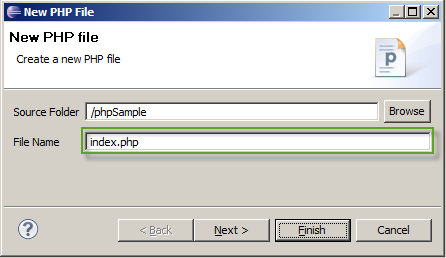
* PHP 5.3.1 (through Web Platform Installer)
* Eclipse PDT 3.0.x All In Ones ( <http://www.eclipse.org/pdt/downloads/> )

# Walkthrough

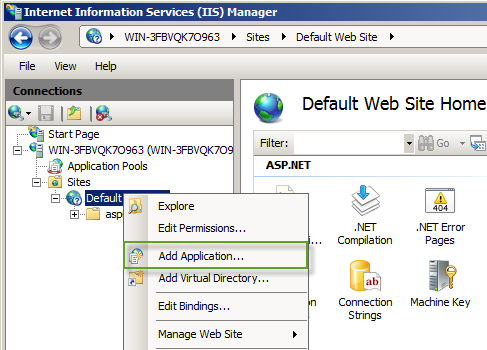
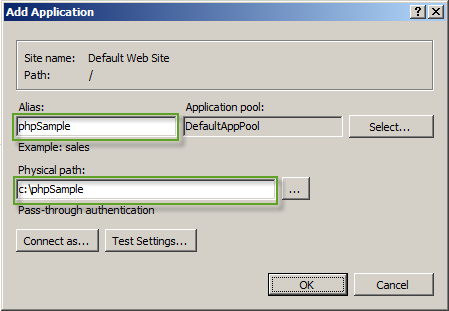
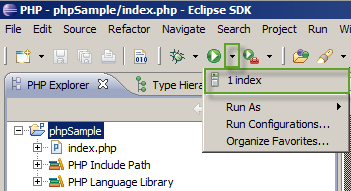
## ISV - Creating a Simple PHP Application

In this first task you will create a simple PHP application, which will become our mockup protected resource.

Fabrikam’s developers create a new **PHP Project** using Eclipse by performing the following steps:

1. Open a new instance of Eclipse.
2. Create a new project: ***File -> New Project -> PHP Project***
3. In the first wizard window provide a project name and a folder where the project will be created and click **Finish**.  
     
   
4. Select the sample project, **Right-click** and select ***New -> PHP File***.  
     
   
5. Provide a name for the page and click **Finish**.  
     
   
6. Replace the generated code with the following:

|  |
| --- |
| <!DOCTYPE>  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>  <title>Index</title>  </head>  <body>  <h2>Index Page</h2>  </body>  </html> |

1. Open **Internet Information Service** **(IIS) Manager**.
2. Right-click the **Default Web Site** item and choose **Add Application…**  
     
   
3. Provide an alias and the path where the **phpSample** project is created and click **OK**.  
     
   
4. Go back to Eclipse, click the arrow beside the play button on the toolbar and select **Index**.  
     
   
5. At this point you should be able to browse the index page.  
     
   
6. Open a Powershell console and run the following command to generate a new GUID for this application:

|  |
| --- |
| PS C:\Windows\system32> [guid]::NewGuid()  Guid  ----  **d184f6dd-d5d6-44c8-9cfa-e2d630dea392** |

|  |
| --- |
| ***Note****: This identifier will be the* ***AppPrincipalId*** *used in further customer provisioning that want to access the application using their Office 365 accounts.* |

## Customers - Provisioning a New Application in the Directory Tenant

Awesome Computers, an Office 365 customer, acquired the right[[1]](#footnote-1) to access Fabrikam’s application (for example: they bought a license from Fabrikam). In this task we show how Awesome Computer’s administrators provision the application in their tenant and communicate back to Fabrikam the data they need in order to set up single sign on for Awesome Computers’ employees. The desired result is allowing Awesome Computers’ employees to authenticate with Fabrikam’s web application exactly in the same way in which they authenticate with their Office365 applications.

|  |
| --- |
| ***Note****: If you don’t have access to an Office365 tenant for this walkthrough, you can obtain one by applying for a FREE TRIAL subscription on Office 365’s Sign-up* [*page*](http://www.microsoft.com/en-us/office365/online-software.aspx#fbid=8qpYgwknaWN)*.* |

In a nutshell, in order to register for Fabrikam’s application Awesome Computers’ administrator needs to create a new Service Principal for it in their tenant.

Fabrikam somehow[[2]](#footnote-2) provides to Awesome Computers:

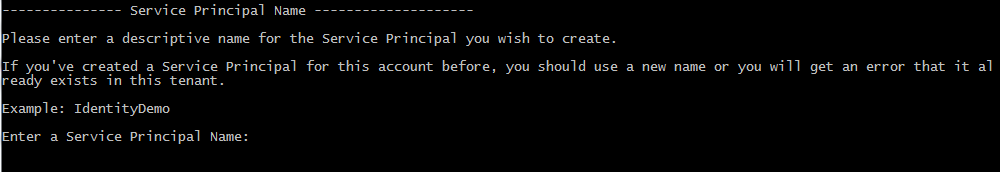
* The value of the **ServicePrincipalName** (*phpSample/localhost*)
* the **AppPrincipalId** (*d184f6dd-d5d6-44c8-9cfa-e2d630dea392*)
* the **ReplyUrl**

Awesome Computers’ administrator performs the following steps:

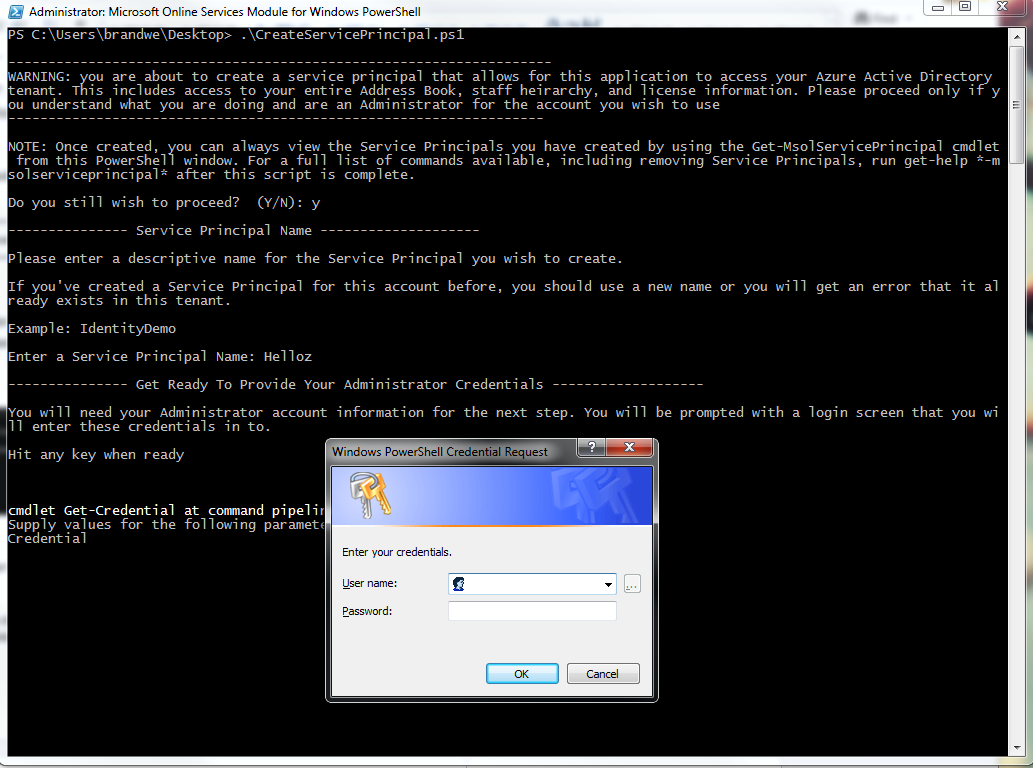
1. If you didn’t do it yet, download and install a set of Powershell scripts from Office 365’s online help [page](http://onlinehelp.microsoft.com/en-us/office365-enterprises/hh124998.aspx).
2. Locate the **CreateServicePrincipal.ps1** script in this code example set under WAAD.WebSSO.PHP/Scripts
3. From the **Start Menu**, Open the *“Microsoft Online Services Module for Windows PowerShell”* console.
4. Run the SampleAppServicePrincipal.ps1 command from the Microsoft Online Services Module for Windows PowerShell Console

|  |
| --- |
| PS C:\Windows\system32> ./CreateServicePrincipal.ps1 |

You will be asked to pick a name for your Service Principal. This just needs to be a descriptive name that you can remember in case you wish to inspect or remove the Service Principal later on.



1. You will be prompted to enter your administration credentials for your Office365 tenant:



1. If the script is successful, you will get output like indicated below. **You will want to capture these items for use later in the walkthrough**:

|  |
| --- |
| Setting permissions to allow the Service Principal to have Read Only access to your Azure Active Directory tenant. See the P  owerShell script to see how this is done.  --------------- Script is complete ----------------------  Company ID: eda7500a-ad60-4648-9019-89dddd37673b  **AppPrincipal ID: a4ba298c-0d1c-4332-b85f-7ab34743d377**  App Principal Secret: 0guCAQnwLuTtpa+jWkam1Y/gcGsRzEOOOA2+Illt5bQ=  **Audience URI: a4ba298c-0d1c-4332-b85f-7ab34743d377@eda7500a-ad60-4648-9019-89dddd37673b**  Usage : Verify |
|  |
| ***Note****: In the command shown here,* ***AppPrincipalId*** *values are those provided by Fabrikam.* |

That’s it: Fabrikam’s application has been provisioned in the directory tenant of Awesome Computer.

Now Fabrikam needs to provision Awesome Computers as a customer of the application: all it takes is for Fabrikam to know that users from the Office365 tenant with domain *awesomecomputers.onmicrosoft.com* should be granted access. How that information reaches Fabrikam will depend on how subscriptions are handled, in this walkthrough we just assume it happened without detailing how.

## ISV - Protect the Application via WS-Federation and Onboard the First Customer

Let’s get back to Fabrikam. The application created in step1 is still unfinished: we need to add support for federated login. We’ll do it by using the **federation** and **simpleSAML.php** libraries and adding some extra artifacts, like a login page. With the app ready to authenticate requests using the WS-Federation protocol, we’ll add the WAAD tenant of Awesome Computers as a trusted provider.

1. Create a **federation.ini** file in the project’s root and provide the following information:

**NOTE:** audienceuri= and realm= are the values you retrieved from the PowerShell command above. Remember that you must add **spn:** to be beginning of this value. Use the audienceuri for both values below.

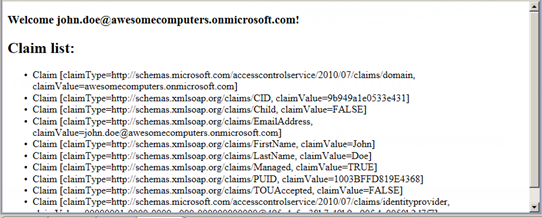
|  |
| --- |
| **federation.trustedissuers.issuer**=https://accounts.accesscontrol.windows.net/v2/wsfederation  **federation.trustedissuers.thumbprint**=3f5dfcdf4b3d0eab9ba49befb3cfd760da9cccf1  **federation.trustedissuers.friendlyname**=Awesome Computers  **federation.audienceuris**=spn:d184f6dd-d5d6-44c8-9cfa-e2d630dea392@495c4a5e-38b7-49b9-a90f-4c0050b2d7f7  **federation.realm**=spn:**d184f6dd-d5d6-44c8-9cfa-e2d630dea392@495c4a5e-38b7-49b9-a90f-4c0050b2d7f7**  **federation.reply**=https://localhost/phpSample/index.php |

1. Create a new PHP file called **secureResource.php**, include the following code, and replace the highlighted path according to the library’s folder location (the **federation** and the **simpleSAMLphp** folders should be inside the phpLibraries folder ):

|  |
| --- |
| <?php  ini\_set('include\_path', ini\_get('include\_path').';c:\phpLibraries\;');  **require\_once** ('federation/FederatedLoginManager.php');  session\_start();  $token = $\_POST['wresult'];  $loginManager = **new** FederatedLoginManager();  **if** (!$loginManager->isAuthenticated()) {  **if** (**isset** ($token)) {  **try** {  $loginManager->authenticate($token);  } **catch** (Exception $e) {  print\_r($e->getMessage());  }  } **else** {  $returnUrl = "https://" . $\_SERVER['HTTP\_HOST'] . $\_SERVER['PHP\_SELF'];  header('Pragma: no-cache');  header('Cache-Control: no-cache, must-revalidate');  header("Location: " . FederatedLoginManager :: *getFederatedLoginUrl*($returnUrl), **true**, 302);  **exit**();  }  }  ?> |

1. Open the **index.php** page to secure the page by adding the **require\_once** sentence and the code for listing the claims for the authenticated user.

|  |
| --- |
| <?php  **require\_once** (dirname(\_\_FILE\_\_) . '/secureResource.php');  ?>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>  <title>Index Page</title>  </head>  <body>  <h2>Index Page</h2>  <h3>Welcome <strong><?php print\_r($loginManager->getPrincipal()->getName()); ?></strong>!</h3>    <h4>Claim list:</h4>  <ul>  <?php  **foreach** ($loginManager->getClaims() **as** $claim) {  print\_r('<li>' . $claim->toString() . '</li>');  }  ?>  </ul>  </body>  </html> |

1. Run the project. You should be automatically redirected to the Office365 page where you can log in using your **awesomecomputers.onmicrosoft.com** credentials (e.g. [john.doe@awesomecomputers.onmicrosoft.com](mailto:john.doe@awesomecomputers.onmicrosoft.com)).  
   
2. Finally, if the login process is successful you will be redirected to the secured page (**phpSample**/**index.php**) as an authenticated user.  
     
   

If for some reason your application is meant to work with a single Office365 tenant, for example if you are writing a LoB application, you can stop here. What you have seen so far enables you to offer to one arbitrary app the same single sign on experience you enjoy with Office365. If instead you are developing applications that need to be accessed by multiple tenants, the next section will help you to modify the code to accommodate your scenario.

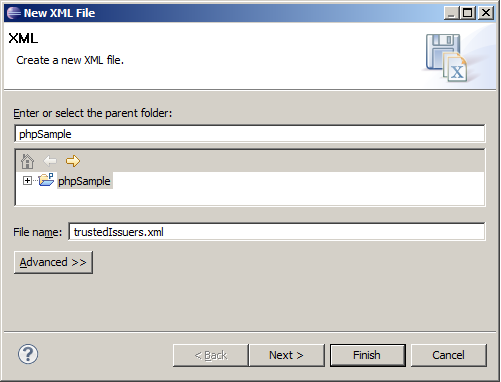
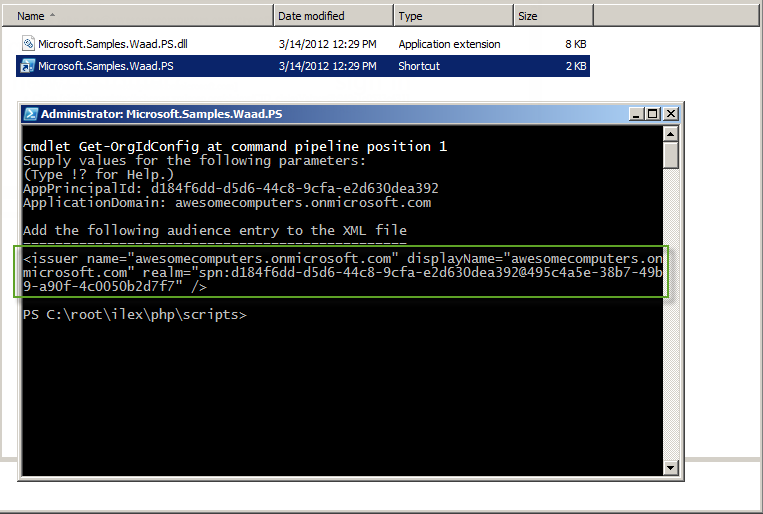
## ISV - Configuring the Application to Handle Sign On with Multiple Tenants

Let’s say that Fabrikam wants to sell access to its application to multiple customers. The technique demonstrated so far works with only one trusted provider: Fabrikam’s developers will need to make some changes in the application to accommodate sign on from both Awesome Computers and its future new customers. The main new features needed are:

* Support for multiple identity providers in the login page
* Maintain a list of the trusted providers and the audienceURI they’ll send to the application; use that list to determine how to validate incoming tokens

For the sake of this example we will add another fictitious customer, Trey research. Trey Research Inc. will register the application in its tenant as shown for Awesome Computers in the Customers task earlier in the walkthrough.

Here there’s the list of tweaks that need to happen in the app to make it enable multi-tenant sign-on, intertwined with the provisioning of Trey Research.

1. From Eclipse, **Right-click** the **phpSample** project, select **New -> Xml File** and provide *“trustedIssuers.xml”* as the file name. This file will contain a list of the trusted issuers for the application (in this case with Awesome Computers and Trey Research Inc.) which will be used by the dynamic audience Uri validator.  
     
   
2. Go to the scripts folder and open the **Microsoft.Samples.Waad.Federation.PS** link to generate the trusted issuers’ nodes to add to the XML repository. It will ask you for the **AppPrincipalId** and the **AppDomain** name to generate the issuer node as depicted below:  
     
   

|  |
| --- |
| ***Note****: Behind the scenes the script will retrieve the federation metadata to get the issuer identifier for generating the realm’s SPN value.* |

1. Open the XML file, create an **issuers** root node and include the output node:

|  |
| --- |
| <issuers>  <issuer name=*"awesomecomputers.onmicrosoft.com"* displayName=*"awesomecomputers.onmicrosoft.com "* realm=*"spn:d184f6dd-d5d6-44c8-9cfa-e2d630dea392@495c4a5e-38b7-49b9-a90f-4c0050b2d7f7"* />  </issuers> |

1. Repeat Step 2 to generate Trey Research Inc. node. Notice that you can change the display name to show a user-friendly name.

|  |
| --- |
| <issuers>  <issuer name=*"awesomecomputers.onmicrosoft.com"* displayName=*"Awesome Computers"* realm=*"spn:d184f6dd-d5d6-44c8-9cfa-e2d630dea392@495c4a5e-38b7-49b9-a90f-4c0050b2d7f7"* />  <issuer name=*"treyresearchinc.onmicrosoft.com"* displayName=*"Trey Research Inc."* realm=*"spn:d184f6dd-d5d6-44c8-9cfa-e2d630dea392@13292593-4861-4847-8441-6da6751cfb86"* />  </issuers> |

1. Create a new **login.php** file and include the following code:

|  |
| --- |
| <?php  ini\_set('include\_path', ini\_get('include\_path').';c:\phpLibraries\;');  **require\_once** ('waad-federation/TrustedIssuersRepository.php');  $repository = **new** TrustedIssuersRepository();  $trustedIssuers = $repository->getTrustedIdentityProviderUrls();  ?>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>  <title>Login Page</title>  </head>  <body>  <h3>Login Page</h3>    <ul>  <?php  **foreach** ($trustedIssuers **as** $trustedIssuer) {  print\_r('<li><a href="' . $trustedIssuer->getLoginUrl("https://localhost/phpSample/index.php") . '">' . $trustedIssuer->displayName . '</a></li>');  }  ?>  </ul>  </body>  </html> |

|  |
| --- |
| ***Note****: Replace the highlighted code with the path where the libraries are located (****federation****,* ***waad-federation*** *and* ***simpleSAMLphp*** *should be inside this folder )* |

1. Open the **secureResource.php** file and replace the required library **“federation/FederatedLoginManager.php”** with **“waad-federation/ConfigurableFederatedLoginManager.php”**

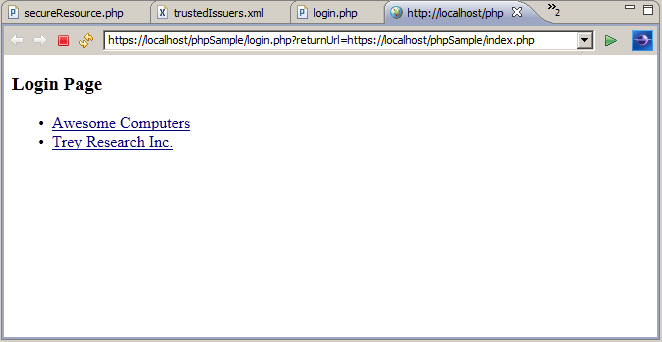
|  |
| --- |
| **require\_once** ('waad-federation/ConfigurableFederatedLoginManager.php'); |

1. Replace the class name **FederatedLoginManager** with **ConfigurableFederatedLoginManager** when instantiating the object.

|  |
| --- |
| $loginManager = **new** ConfigurableFederatedLoginManager(); |

1. Also replace the Location header with the following:

|  |
| --- |
| header("Location: https://" . $\_SERVER['HTTP\_HOST'] . dirname($\_SERVER['SCRIPT\_NAME']) . "/login.php?returnUrl=" . $returnUrl, **true**, 302); |

1. Run the project and you will be able to see a list with the links for each trusted identity provider retrieved from the *“trusted.issuers.xml”* repository.  
     
   

|  |
| --- |
| ***Note****: The home realm discovery strategy of presenting an explicit list of trusted providers is not always feasible in practice. Here it is used for the sake of simplicity* |

Once you see the list in your browser, you can navigate to either providers: the authentication flow will unfold in the same way described in the former section. The application will validate the incoming token accordingly. You can try to delete entries in trusted.issuers.xml, as it would happen for example once a subscription expire, and verify that the application will reject authentication attempts from the corresponding provider.

1. Please note that in this walkthrough we don’t give detailed indications on how Fabrikam makes its application available to potential customers. We also don’t specify how the customers circle back information to Fabrikam after provisioning. In actual settings the flow would likely be driven by some kind of marketplace or would follow whatever licensing and provisioning system the ISV has in place. [↑](#footnote-ref-1)
2. See note 1. The data listed here are the same for all prospect customers, hence depending on the subscription model those might even be simply publicly available. [↑](#footnote-ref-2)