Demo - Securing Application with Azure Active Directory

Lab version: 1.0

Last updated: 1/16/2015

Estimated demo delivery time: **10 to 15 minutes**.



Contents

[Overview 3](#_Toc408998110)

[Prerequisites 3](#_Toc408998111)

[Setup 3](#_Toc408998112)

[Exercise 1: Securing Service with Azure Active Directory 4](#_Toc408998113)

[Task 1: Creating a Directory 4](#_Toc408998114)

[Task 2: Creating a Demo User 5](#_Toc408998115)

[Task 3: Creating a Windows Azure Active Directory Service Application 8](#_Toc408998116)

[Task 4: Securing Expenses Application with Azure Active Directory 10](#_Toc408998117)

# Overview

In this demo, we will show how to use Azure Active Directory to help secure the Expenses application.

# Prerequisites

The following are required to complete this demo:

* [Microsoft Visual Studio 2013](http://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx) (tested with Update 4)
* Internet connection
* [Microsoft Azure](http://azure.microsoft.com/en-us/pricing/free-trial/) subscription
* Expenses.Mvc codebase

# Setup

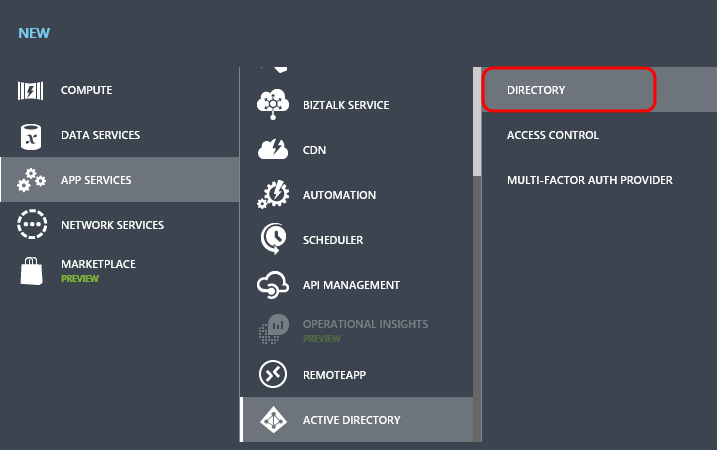
* **Use a fresh copy of the starting Expenses codebase for this demo** (where we run the application and database locally). Although it is certainly possible to continue where previous demos left off, this allows this demo to be delivered free of unnecessary pre-requisites and moving parts.
* Load and build the Expenses.Mvc solution to ensure that it builds correctly.
* Log into the Microsoft Azure subscription that you will be using for demonstration.

# Exercise 1: Securing Application with Azure Active Directory

In this exercise, we will setup and configure Azure Active Directory for the purposes of securing the Expenses application.

## Task 1: Creating a Directory

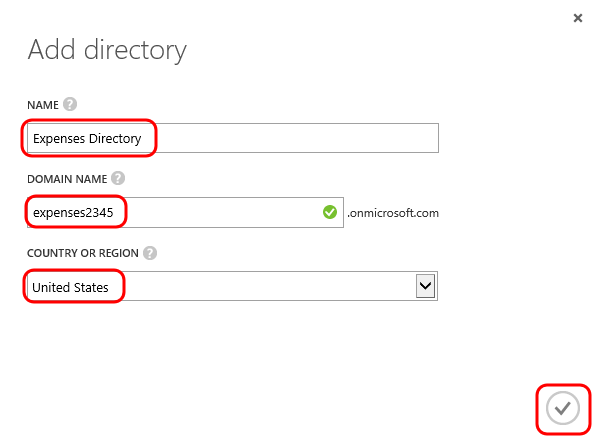
1. Log into the full Microsoft Azure portal [here](https://manage.windowsazure.com) (or if you are currently in the Preview portal at portal.azure.com you can click on your user name in the top-right corner and select the Full Azure Portal link).
2. Click New the new button in the bottom-left corner and then select App Services | Active Directory | Directory.



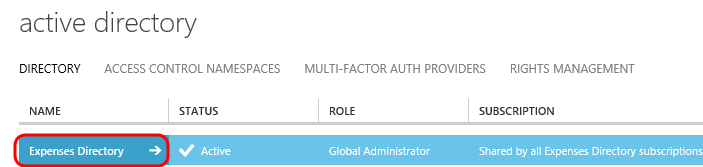
1. Click Custom Create.



1. In the Add Directory window, provide a Name, Domain Name, and Country or Region. Click the Complete button to create the directory.



1. Navigate to the new directory in the portal.

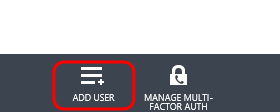


## Task 2: Creating a Demo User

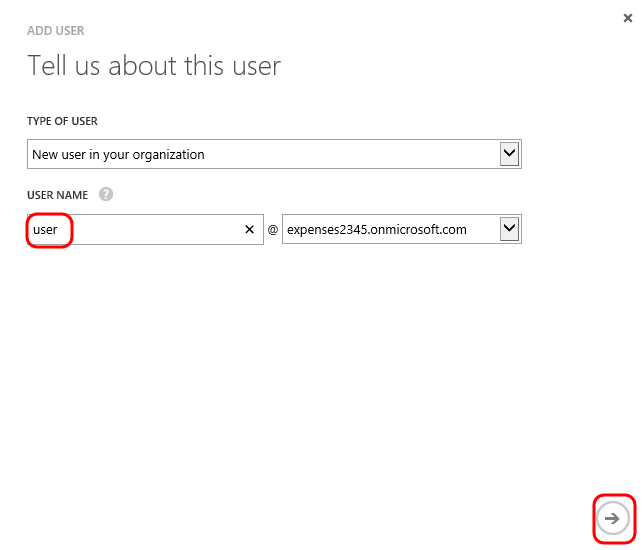
1. Select the Users tab.



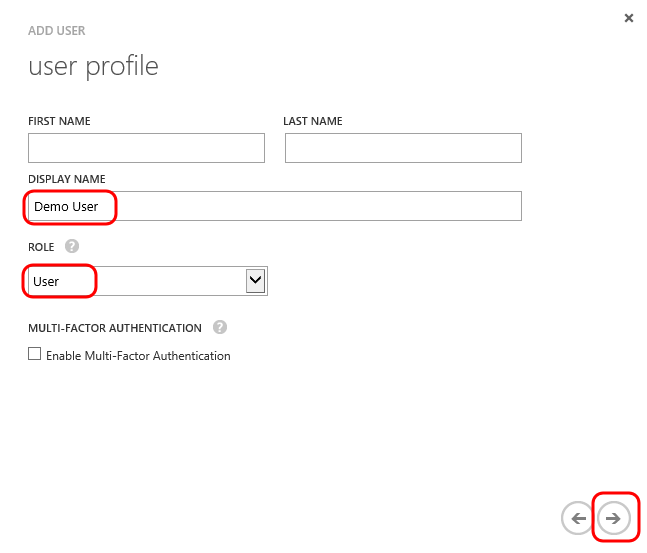
1. Click the Add User button at the bottom.



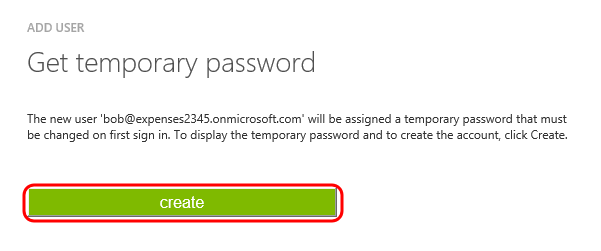
1. Use the default for the Type of User option, which is “New user in your organization”.
2. Create a user in the directory named “user” for demonstration purposes.
3. Click the Next button.



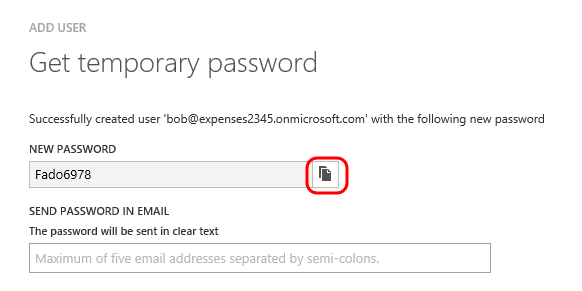
1. Provide a display name, use the default role of User, and then click the Next button.



1. Click the Create button to generate a temporary password.



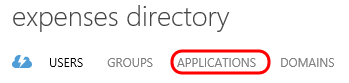
1. Copy the temporary password to Notepad or other location so that we can use it to login with.



1. Click the Complete button.

## Task 3: Creating a Windows Azure Active Directory Service Application

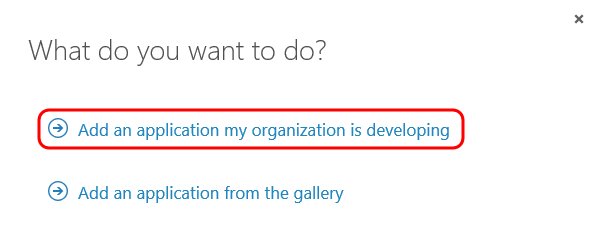
1. Click the Applications tab.



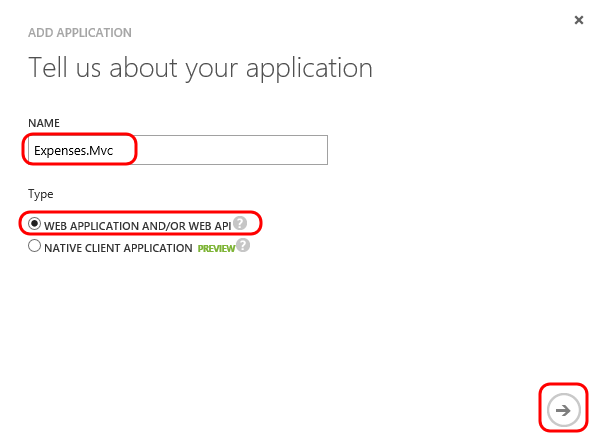
1. Click the Add button near the bottom.



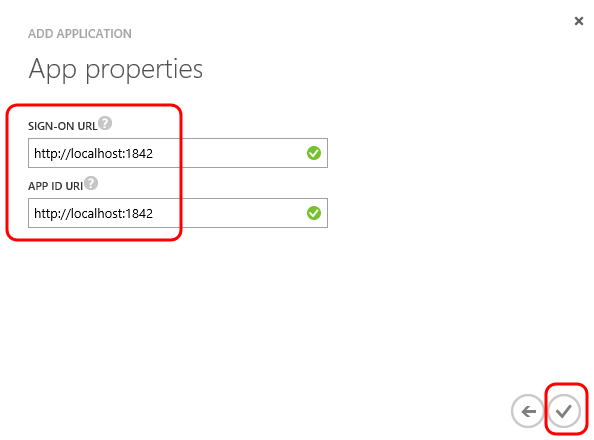
1. Click the “Add an application my organization is developing” link.



1. Enter “Expenses.Mvc” for the name and select “Web Application and/or Web API” for the type. Click the Next button to continue.



1. Since we are in the development phase, provide a Sign-On URL of http://localhost:1842.
2. For the App ID URI, provide a unique logical identifier. This does not need to resolve to an Internet address, the only requirement is that it is in valid URI format.
3. Click the Complete button.



## Task 4: Securing Expenses Application with Azure Active Directory

1. Open the Expenses.Mvc.sln solution in Visual Studio.
2. Select Tools | NuGet Package Manager | Package Manager Console from the main menu.
3. In the Package Manager Console window, ensure that the Expenses.Web project is selected.



1. Use the following command to install the necessary .NET Framework 4.5 identity protocol extensions (OpenId Connect):

Install-Package Microsoft.IdentityModel.Protocol.Extensions -Version 1.0.1

1. Use the following command to install the needed OWIN middleware for OpenId Connect:

Install-Package Microsoft.Owin.Security.OpenIdConnect -Version 3.0.0

1. Use the following command to install the OWIN middleware to use cookies:

Install-Package Microsoft.Owin.Security.Cookies -Version 3.0.0

1. Use the following command to install the bits necessary to enable OWIN-based applications to run on IIS using the ASP.NET request pipeline:

Install-Package Microsoft.Owin.Host.SystemWeb -Version 3.0.0

1. Open the Web.config file in the editor and add the following app settings keys:

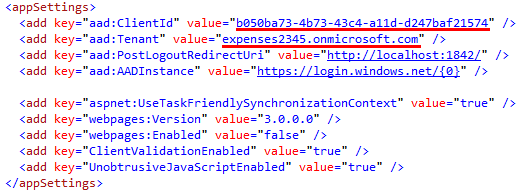
<add key="aad:ClientId" value="[Enter client ID as obtained from Azure Portal, e.g. 82692da5-a86f-44c9-9d53-2f88d52b478b]" />

<add key="aad:Tenant" value="[Enter tenant name, e.g. expenses.onmicrosoft.com]" />

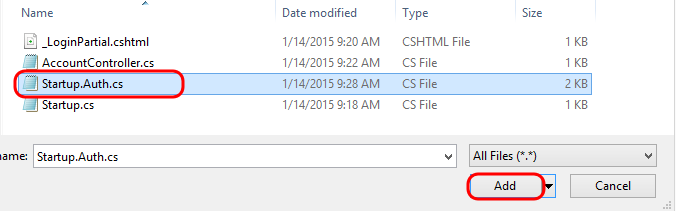
<add key="aad:PostLogoutRedirectUri" value="http://localhost:1842/" />

<add key="aad:AADInstance" value="https://login.windows.net/{0}" />

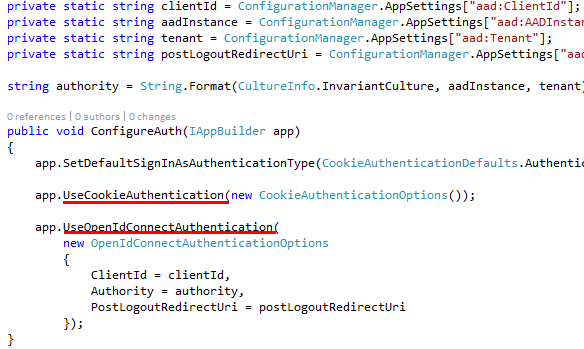
1. Replace the placeholder values with your Client ID and Tenant values from the portal. Your updated web.config should look similar to the following:



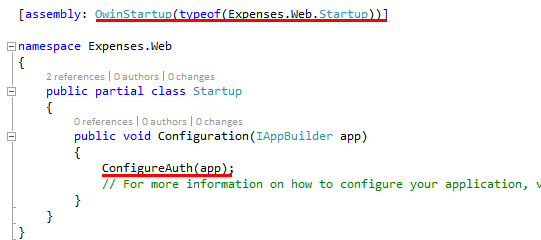
1. Right-click on the App\_Start folder and select Add | Existing Item. Select the Startup.Auth.cs file from the DemoFiles folder (should be one level up from the project) and then click Add.



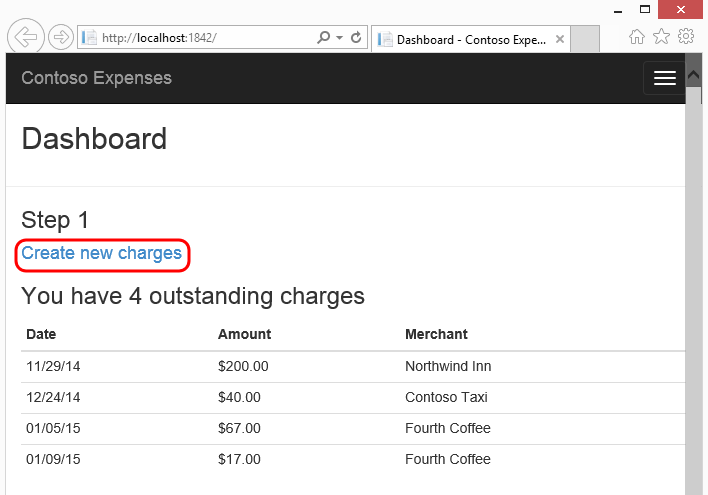
1. Open Startup.Auth.cs in the editor. This class provides a ConfigureAuth method that sets up the use of cookies to maintain session with the browser client and configures the use of OpenID Connect with the OWIN runtime. It grabs the necessary configuration from the app settings from configuration.



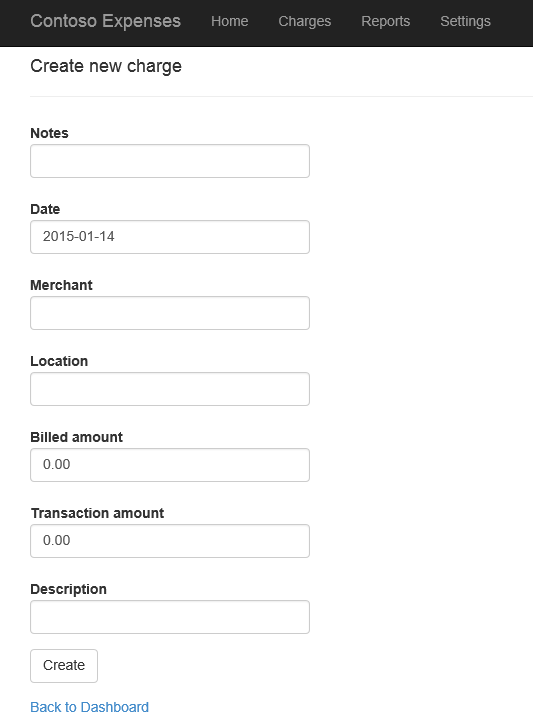
1. Add the Startup.cs file from the DemoFiles folder to the root of the project.
2. Open Startup.cs in the editor. The Startup class and the Configuration method will be called when the OWIN runtime starts up. Note that it calls the ConfigureAuth method.



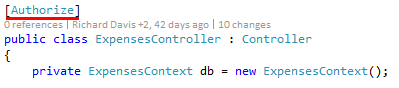
1. Press F5 to launch the Expenses web application.
2. Click on the “Create new charges” link.



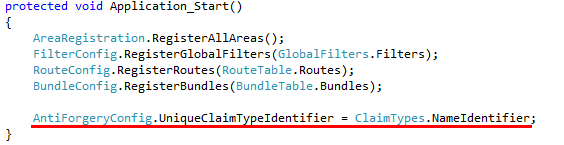
1. Note that this still works as it did before we started adding in the necessary changes to authenticate with Azure Active Directory. We were not required to log in.



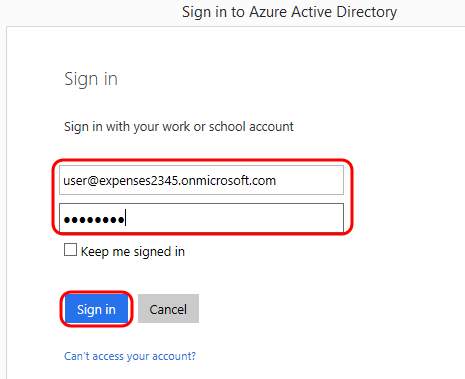
1. Close the browser window to return to Visual Studio and stop debugging.
2. We could now perform the work to add in a “log in” link to the UI and manually issue an authentication challenge back to the client when appropriate, but let’s keep things simple for this demonstration and instead make use of the Authorize attribute to do this for us. Open ExpensesController.cs in the editor and add the “[Authorize]” attribute to the ExpensesController class definition.



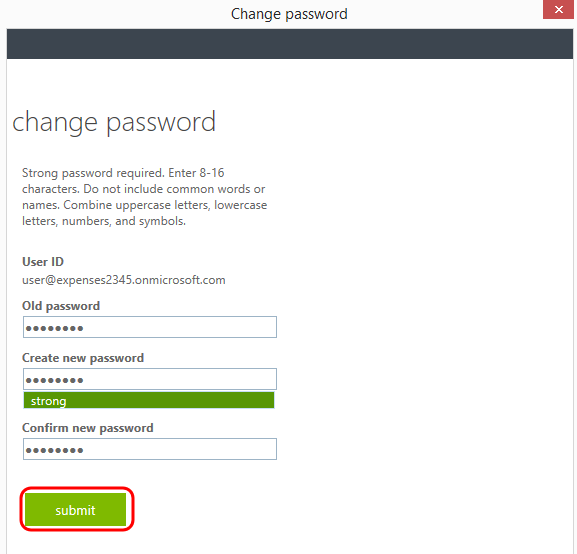
1. Open Global.asax and set the claim type identifier to be ClaimTypes.NameIdentifier. This will aid in helping to prevent cross site request forgery in posts to some of the Expenses controller methods (see the ValidateAntiForgeryToken attributes on some of the controller methods).



1. Press F5 to launch the Expenses application once again. Note that you are re-directed to sign into your AAD tenant. Sign in with the demo user you created earlier.



1. Since this is your first log in, you will be required to change the password to one of your choosing. Go ahead and do this and then click Submit.



1. Close the browser window and return to Visual Studio to stop debugging.