Azure Active Directory Preview: Web Sign-On with Windows Azure Active Directory – ASP.NET Walkthrough

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

Table of Contents

[Overview 1](#_Toc320523395)

[Prerequisites 2](#_Toc320523396)

[Assets 2](#_Toc320523397)

[Generic Environment Requisites 2](#_Toc320523398)

[ASP.NET-specific Requisites 2](#_Toc320523399)

[Walkthrough 2](#_Toc320523400)

[ISV - Creating a Simple MVC Application 2](#_Toc320523401)

[ISV - Creating a simple MVC application 2](#_Toc320523402)

[Customers - Provisioning a New Application in the Directory Tenant 4](#_Toc320523403)

[ISV - Protect the Application via WS-Federation and Onboard the First Customer 7](#_Toc320523404)

[ISV - Configuring the Application to Handle Sign On with Multiple Tenants 9](#_Toc320523405)

[Document History 13](#_Toc320523406)

# 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 ASP.NET MVC 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.ASPNET.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

### ASP.NET-specific Requisites

* Visual Studio 2010 SP1
* .Net Framework 4.0
* ASP.Net MVC 3 (<http://www.asp.net/mvc/mvc3>)
* Windows Identity Foundation 1.0 Runtime
* Windows Identity Foundation SDK 4.0

# Walkthrough

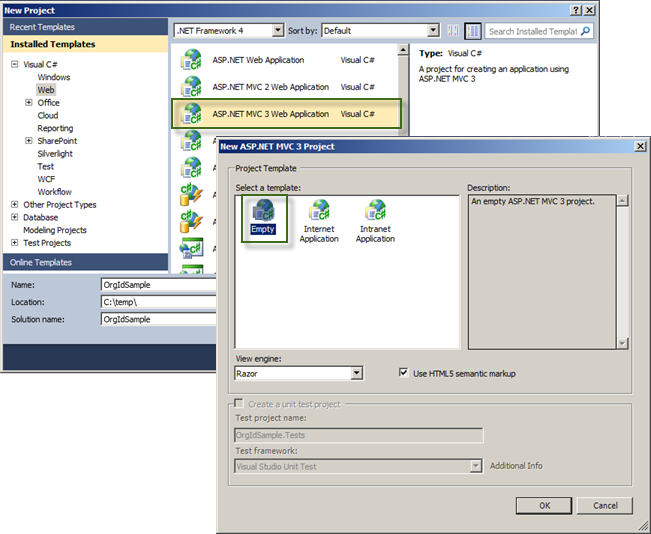
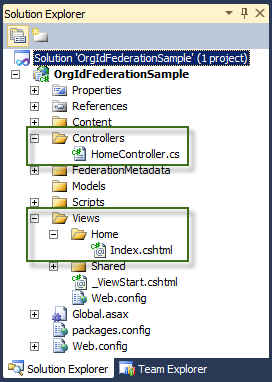
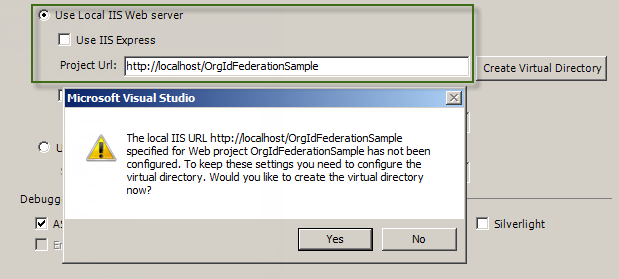
## ISV - Creating a Simple MVC Application

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

Fabrikam creates an MVC application in order to provide a service for future customers. They will grant access to users through a federated mechanism that will rely on customers’ STS. In this scenario we will see how two customers (and their members) access Fabrikam’s service. As these customers don’t have their own STS, they will rely on the STS provided by Office 365.

### ISV - Creating a simple MVC application

In order to provide a service, Fabrikam’s developers create a new MVC empty application in IIS by performing the following steps:

1. Open a new instance of Visual Studio 2010 using the *“Run as administrator…”* option.
2. Create a new project: ***File -> New Project -> Web -> ASP.NET MVC 3 Web Application -> Empty****.*  
   
3. Create a controller/view (**HomeController**/**Index**) which will be the main page of the sample website.  
   
4. **Right-click** the **OrgIdFederationSample** project in the Solution Explorer. Select ***Properties -> Web*** and then the *“Use Local IIS Web server”* option. Click **Yes** when the dialog box is displayed.  
     
   
5. Run the application (F5) and you will see the Index view.
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  ----  7829c758-2bef-43df-a685-717089474505 |

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

## 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:

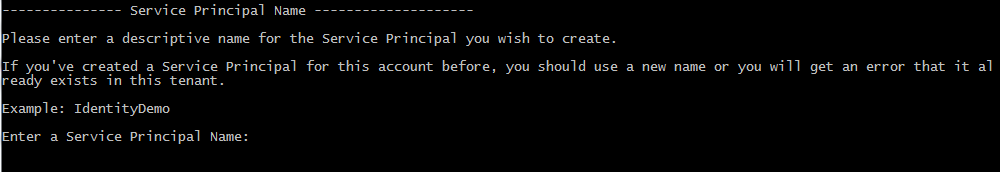
* The value of the **ServicePrincipalName** (*OrgIdFederationSample/localhost*)
* the **AppPrincipalId** (*7829c758-2bef-43df-a685-717089474505*)
* the **ReplyUrl**

Awesome Computers’ administrator performs the following steps:

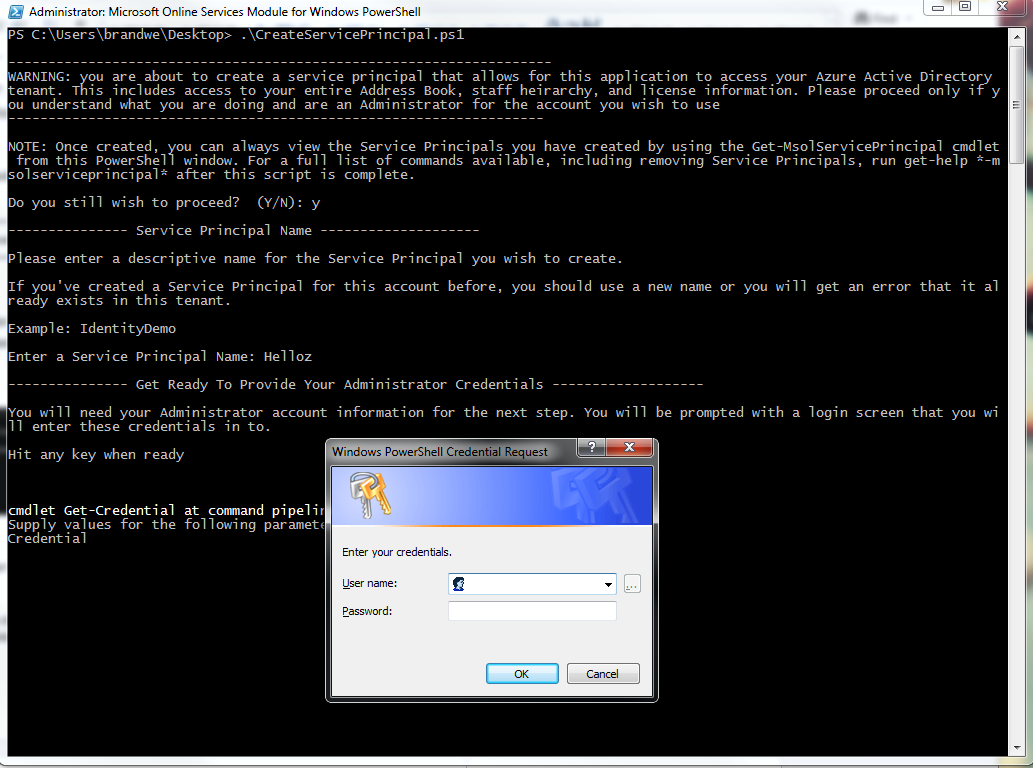
1. If you didn’t do it yet, download and install a set of Powershell scripts from Office 365’s online help [page](http://onlinehelp.microsoft.com/en-us/office365-enterprises/hh124998.aspx).
2. Locate the **CreateServicePrincipal.ps1** script in this code example set under WAAD.WebSSO.ASPNET/Scripts
3. 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

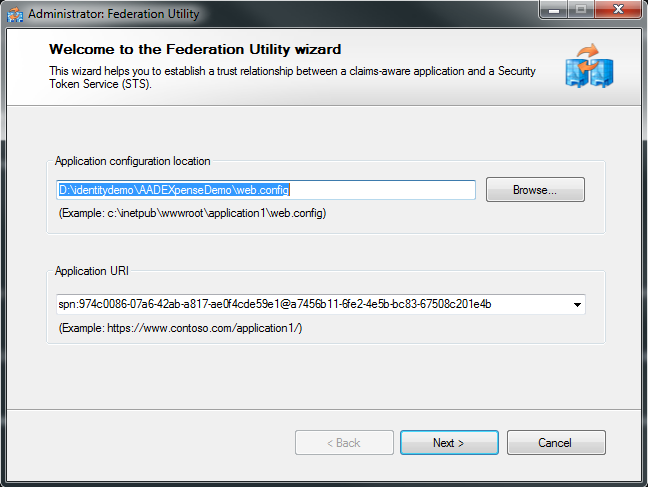
Let’s get back to Fabrikam. The application created in step1 is still unfinished: we need to add support for federated login. We’ll do it by using the **federation** and **simpleSAML.php** libraries and adding some extra artifacts, like a login page. With the app ready to authenticate requests using the WS-Federation protocol, we’ll add the WAAD tenant of Awesome Computers as a trusted provider. The initial setup of the federation will be done manually using FedUtil. The parameters required in the wizard will be: the audience Uri (spn:AppPrincipalId@realm) and the federation metadata (https://accounts.accesscontrol.windows.net/Federation……xml?realm=domain).

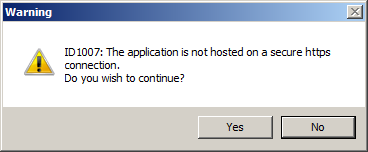
The audienceURI was already generated for you from the PowerShell script above when the admin of Awesome Computers authorized Fabrikam’s application. It should look like the following:

Example: **appid@realm** or 7829c758-2bef-43df-a685-717089474505**@**e4073280-196b-408f-9d40-0be89978fda0

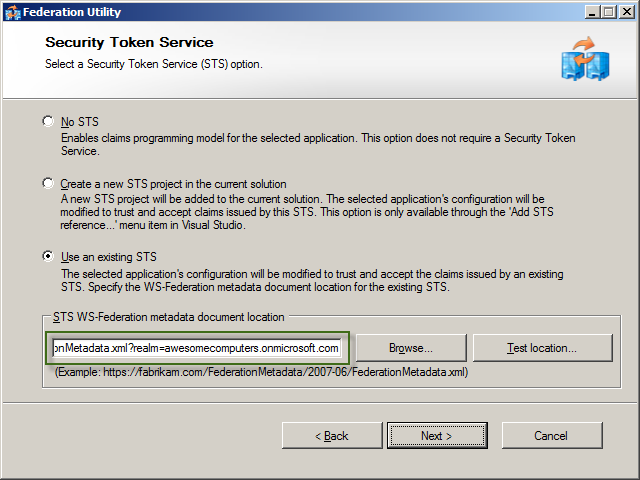
We will be using this Audience URI to add the first customer to our solution:

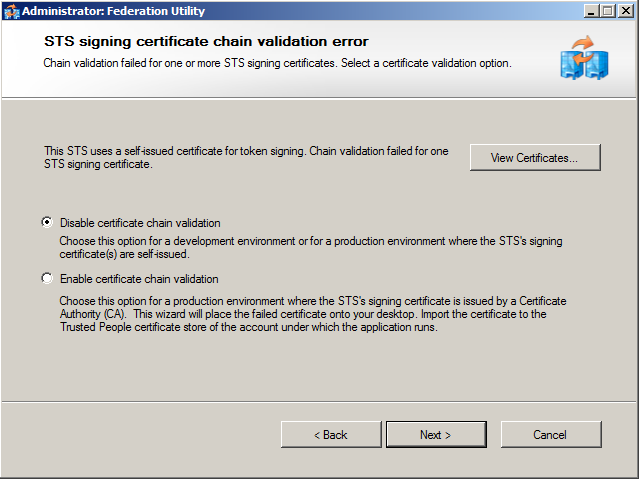
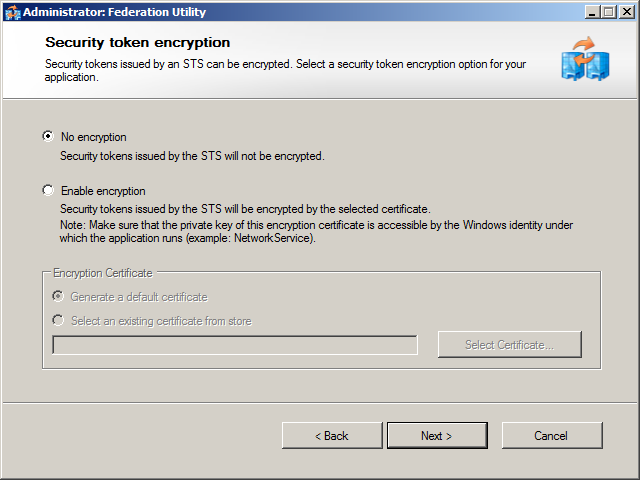
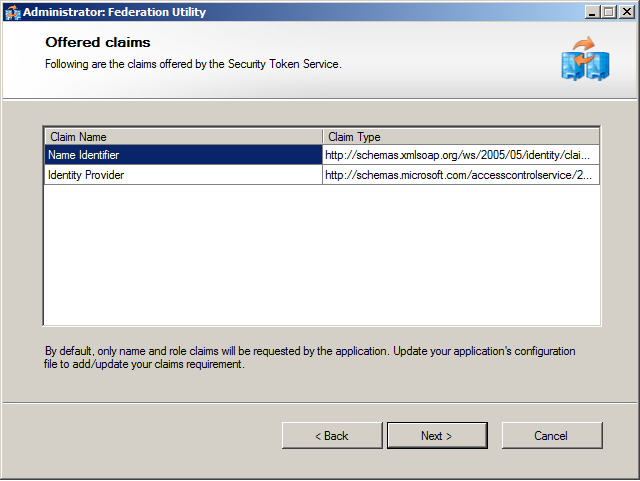
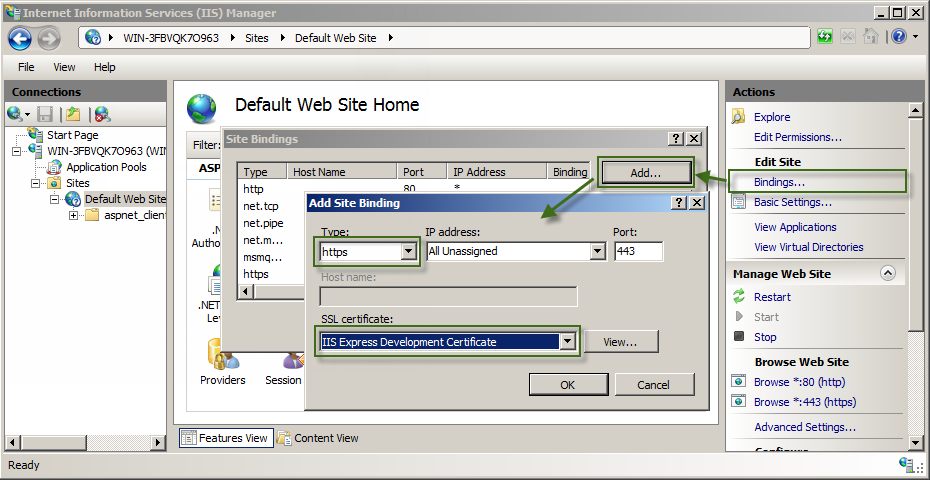
1. Right-click the project node in the Solution Explorer and select *“Add STS reference…”*.
2. Provide the Application URI based on the values obtained after creating the Service Principal. The URI must begin with **spn:** which indicates the URI is a service principal name. The **"appId@realm"** values are set as follows:

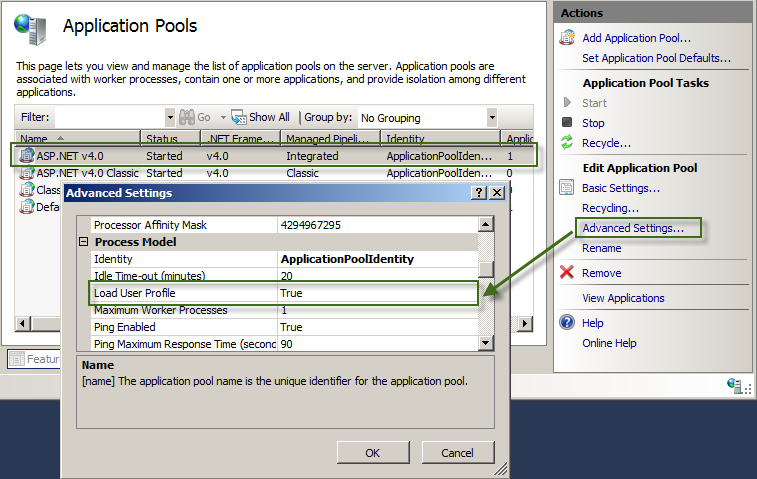
* **appId** is the **AppPrincipalId** generated by the application.
* **realm** is the GUID that you retrieved from the Federation Metadata Endpoint in Step 1.  
    
   

When you enter the SPN and select **Next**, you will see a Warning dialog: *“The application is not hosted on a secure https connection. Do you wish to continue?”*. Select **Yes** and continue. The Federation Utility does not recognize the spn: format but will accept the change, and the application will then use a secure https connection.   
  


1. The next page of the wizard allows you to configure the STS for your web application. Select the option, *“Use an existing STS*,*”* and then enter the location for the WS-Federation metadata document (<https://accounts.accesscontrol.windows.net/FederationMetadata/2007-06/FederationMetadata.xml?realm=awesomecomputers.onmicrosoft.com>).

**NOTE:** This is the same URL you used in Step 1 to get the correct realm to create the Audience URI.  
  


1. Click **Next** and select the *“Disable certificate chain validation”* option.  
     
   
2. After configuring the existing STS for your realm, the next page of the wizard configures the Security token encryption. Use the default value *“No encryption*.*”*   
     
   
3. Next, the Federation Utility shows the Offered claims provided by the STS. Click **Next** to continue.  
     
   
4. Open the IIS manager and configure the *“Default Web Site”* to support SSL (https): select the *“Default Web Site”* item in the left menu, click the*“Bindings…”* option in the right pane and when the *“Site Bindings”* dialog box appears click **Add** to set up https binding.  
     
   
5. From the IIS manager, click the *“Application Pools”* item in the left menu, select the *“ASP.NET 4.0”* application pool and click *“Advanced Settings…”* to set the *“Load User Profile”* property to **true**.



1. Return to Visual Studio and open the **web.config** file to perform the following changes:
   1. Find the **<wsFederation>** section and add a new attribute with the reply Url; the node will look like this:

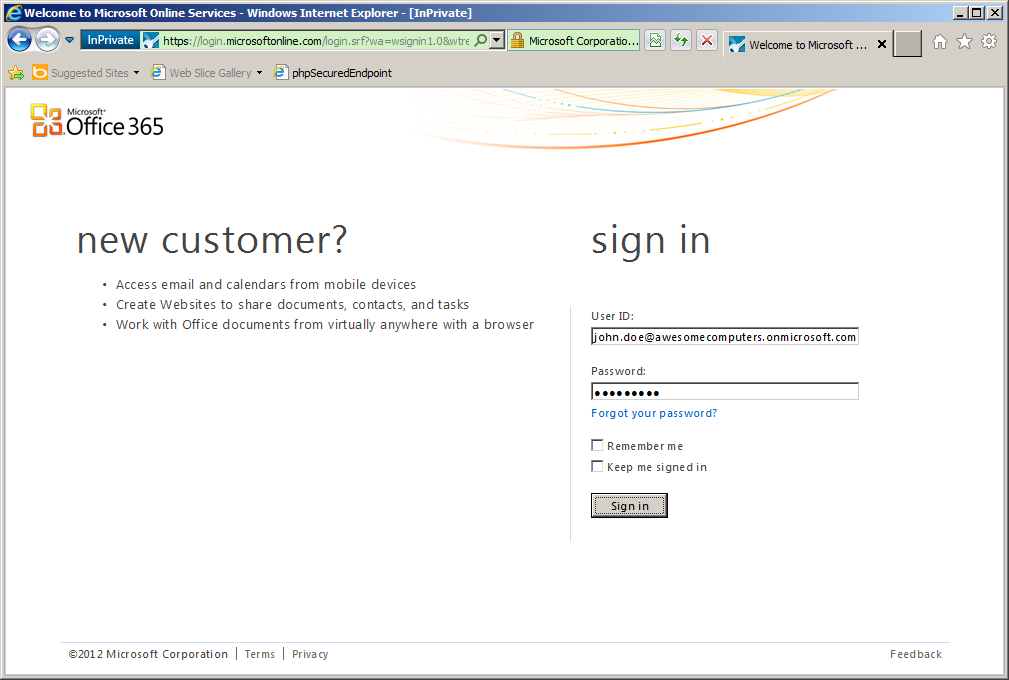
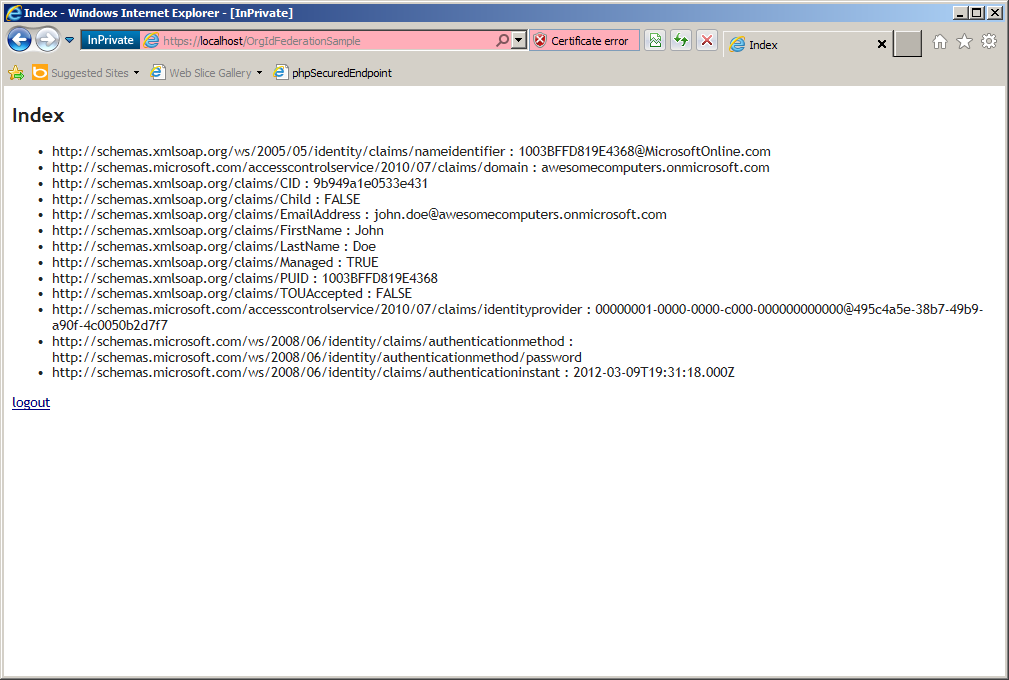
|  |
| --- |
| <wsFederation passiveRedirectEnabled="true" issuer="https://accounts.accesscontrol.windows.net/v2/wsfederation" realm="spn: 7829c758-2bef-43df-a685-717089474505@awesomecomputers.onmicrosoft.com" requireHttps="false" **reply="https://localhost/OrgIdFederationSample"** /> |

* 1. Add the **<httpRuntime>** node inside the **<system.web>** section and set the **requestValidationMode** attribute to *“2.0”*.

|  |
| --- |
| <system.web>  **<httpRuntime requestValidationMode="2.0" />** |

1. From now on the site will require authentication, so we will change the **Index** page to show the authenticated user information (claims). Open the **Index** view and add the following code snippet at the end of the page:

|  |
| --- |
| <p>  @if (User.Identity.IsAuthenticated)  {  <ul>  @foreach (string claim in ((Microsoft.IdentityModel.Claims.IClaimsIdentity)this.User.Identity).Claims.Select(c => c.ClaimType + " : " + c.Value))  {  <li>@claim</li>  }  </ul>  }  </p> |
|  |
| ***Note****: this* ***Index*** *page will act as the secured service provided by Fabrikam.* |

1. Press **F5** to run the application and you will be redirected to the Office 365 identity provider page where you can log in using your **awesomecomputers.onmicrosoft.com** credentials (e.g. john.doe@awesomecomputers.onmicrosoft.com).  
     
   
2. Finally, if the login process is successful you will be redirected to the secured page (**Home**/**Index**) as an authenticated user.  
     
   

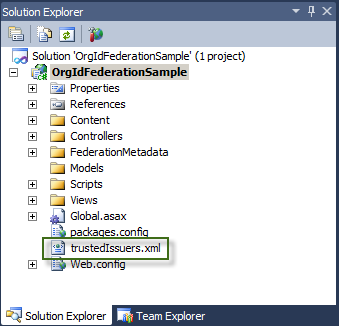
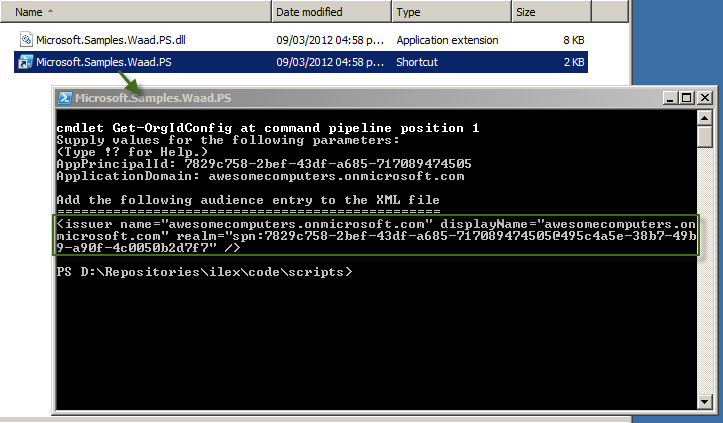
If for some reason your application is meant to work with a single Office365 tenant, for example if you are writing a LoB application, you can stop here. What you have seen so far enables you to offer to one arbitrary app the same single sign on experience you enjoy with Office365. If instead you are 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 Visual Studio, add an empty XML file to the application root called *“trustedIssuers.xml”*. 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:  
     
   

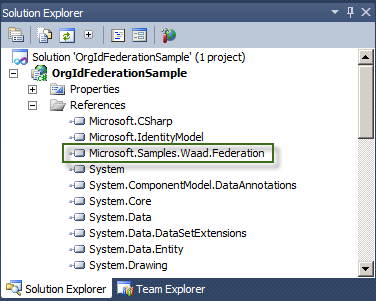
|  |
| --- |
| ***Note****: Behind the scenes the script will retrieve the federation metadata to get the issuer identifier for generating the realm’s SPN value.* |

1. Open the XML file and include the generated node:

|  |
| --- |
| <trustedIssuers>  <issuer name="awesomecomputers.onmicrosoft.com" displayName="awesomecomputers.onmicrosoft.com" realm="spn:7829c758-2bef-43df-a685-717089474505@495c4a5e-38b7-49b9-a90f-4c0050b2d7f7" />  </trustedIssuers> |

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

|  |
| --- |
| <trustedIssuers>  <issuer name="awesomecomputers.onmicrosoft.com" displayName="Awesome Computers" realm=" spn:7829c758-2bef-43df-a685-717089474505@495c4a5e-38b7-49b9-a90f-4c0050b2d7f7" />  <issuer name=" treyresearchinc.onmicrosoft.com" displayName="Trey Research Inc." realm="spn:7829c758-2bef-43df-a685-717089474505@13292593-4861-4847-8441-6da6751cfb86" />  </trustedIssuers> |

1. Add a reference to **Microsoft.Samples.Waad.Federation** assembly.  
     
   
2. Go to the **web.config** and add the following snippet inside **<service>** under **<microsoft.IdentityModel>**. This is the security token handler that will validate the audience Uri dynamically using the *“trustedIssuers.xml”* repository.

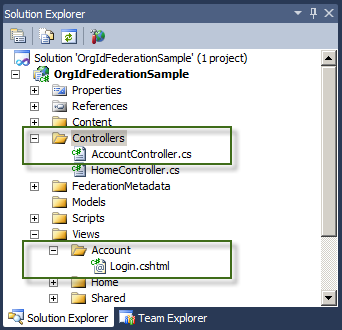
|  |
| --- |
| <microsoft.identityModel>  <service>  **<securityTokenHandlers>**  **<remove type="Microsoft.IdentityModel.Tokens.Saml2.Saml2SecurityTokenHandler, Microsoft.IdentityModel, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>**  **<add type="Microsoft.Samples.Waad.Federation.ConfigurationBasedSaml2SecurityTokenHandler, Microsoft.Samples.Waad.Federation"/>**  **</securityTokenHandlers>** |

1. As we are going to create a custom login page to support both organizations, we’ll disable the automatic redirection. Locate the **<wsFederation>** node and set the attribute **passiveRedirectEnabled** to **false**.

|  |
| --- |
| <wsFederation **passiveRedirectEnabled="false"** issuer="https://accounts.accesscontrol.windows.net/v2/wsfederation" realm="spn:7829c758-2bef-43df-a685-717089474505@awesomecomputers.onmicrosoft.com" requireHttps="false" reply="https://localhost/OrgIdFederationSample" /> |

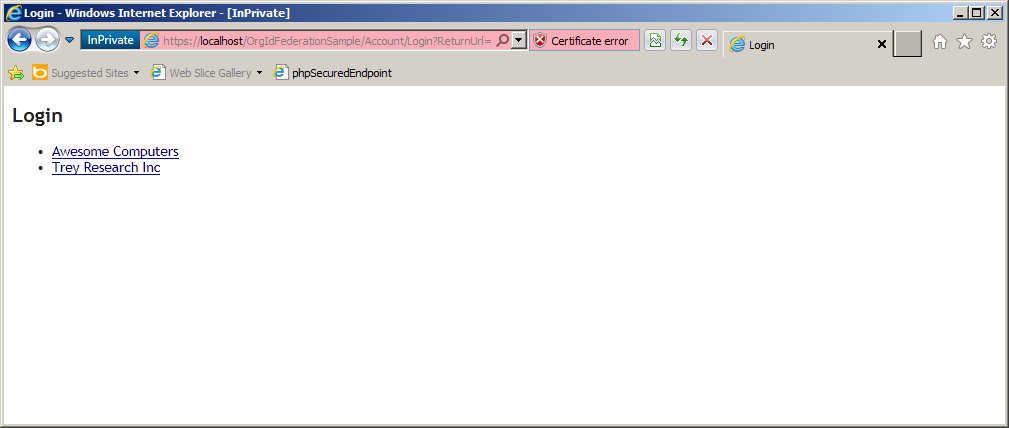
1. Under **<system.web>** replace the **<authentication>** node using the following snippet. This login page will display a list of trusted providers allowing users to perform the login process with their organization credentials.

|  |
| --- |
| <authentication mode="Forms">  <forms loginUrl="~/Account/Login" timeout="2880" />  </authentication> |

1. Create a new controller/view (**AccountController**/**Login**)  
     
   
2. Open the **Login** view and add the following code snippet (at the end) to list the available trusted providers:

|  |
| --- |
| <ul>  @foreach (var trustedIssuer in new Microsoft.Samples.Waad.Federation.TrustedIssuersRepository().GetTrustedIdentityProviderUrls())  {  <li><a href="@trustedIssuer.LoginUrl">@trustedIssuer.DisplayName</a></li>  }  </ul> |

1. Run the application (F5) and you will be able to see a list with the links for each trusted identity provider retrieved from the *“trustedIssuers.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)