

How to Add Azure AD Authentication to KBVault

Revision: 1.2 | 04.28.15

Author: Jeff Blakely

Contents

Purpose	3
Prerequisites	3
Initial Steps.....	3
Create Azure App	4
Create Classes	5
Enable Authorization	7
Set Website to Use HTTPS	8
Edit the Web.config	10
Add AD User(s) to the Database	11
Edit Settings Table in the Database	11
Edit Detail.cshtml.....	12
Test.....	12

Purpose

This guide is intended to help you integrate KBVault with AD authentication using Azure/Office365. This guide does not show how to setup the initial Federation with Azure/Office365, for that you will need to do a fair amount of work not described in this document. If you need consulting services to do so, please feel free to contact me. This guide assumes you have a basic understanding of IIS, MSSQL, Visual Studio, AD, and Azure

Prerequisites

- IIS Website (KBVault in this example)
- MSSQL or MSSQL Express
- Visual Studio 2013
- Admin access to Azure with the ability to add an application
- Existing Azure/AD integration with your organization
- KBVault source code (<https://kbvault.codeplex.com>)
- Time and patience

Initial Steps

1. Launch Visual Studio 2013
2. Open the KBVault project
3. Open Package Manger (Tools -> NuGet Package Manager -> Package Manager Console)
4. Run The (4) four commands below to install the packages needed for AAD Authentication.
 - 4.1. `Install-Package Microsoft.Owin.Security.OpenIdConnect -Pre`
 - 4.2. `Install-Package Microsoft.Owin.Security.Cookies -Pre`
 - 4.3. `Install-Package Microsoft.Owin.Host.SystemWeb -Pre`
 - 4.4. `Install-Package jQuery -Pre`

Create Azure App

1. Create App in Azure AD (the example uses and app called AADAuth)

The screenshot displays the Microsoft Azure portal interface for configuring an application. The left-hand navigation pane is highlighted with a blue background and contains several icons. An orange arrow points to the 'AADAuth' application icon in this pane. The main content area is titled 'aadauth' and includes tabs for 'DASHBOARD', 'CONFIGURE', and 'OWNERS'. The 'CONFIGURE' tab is active, showing the 'properties' section. This section contains fields for 'NAME' (set to 'AADAuth'), 'SIGN-ON URL' (set to 'https://localhost:44304'), and 'LOGO' (a blue square with a white cube icon). Below these fields is a section for 'APPLICATION IS MULTI-TENANT' with 'YES' and 'NO' radio buttons, where 'NO' is selected. At the bottom, the 'CLIENT ID' field is shown with a long alphanumeric string and a copy icon. An orange arrow points to the 'CLIENT ID' label, and another orange arrow points to the copy icon.

Microsoft Azure | v

aadauth

DASHBOARD CONFIGURE OWNERS

properties

NAME AADAuth

SIGN-ON URL https://localhost:44304

LOGO

APPLICATION IS MULTI-TENANT YES NO

CLIENT ID 1179461a-0000-4000-a000-000000000000

Create Classes

2. In the root of the Project, create a new Class called Startup.cs (Right Click KBVault.Web -> Add -> Class). Clear out all of the default code and paste this code into it:

```
using Microsoft.Owin;  
using Owin;  
[assembly: OwinStartup(typeof(OldFashionOWIN.Startup))]  
namespace OldFashionOWIN  
{  
    public partial class Startup  
    {  
        public void Configuration(IAppBuilder app)  
        {  
            ConfigureAuth(app);  
        }  
    }  
}
```

3. Locate your Client ID (from step 1) and Website hostname (the name people will type into their browsers)

4. In the App_Start folder create a new Class called Startup.Auth.cs (Right Click App_Start -> Add -> Class). Clear out all of the default code and paste the code below into it. NOTE: Change the items in RED (ClientID, Authority) to reflect your organizations ClientID and Authority URL. You can find your ClientID on the Configuration Page for the application created previously AADAuth.

```
using Microsoft.Owin.Security;
using Microsoft.Owin.Security.Cookies;
using Microsoft.Owin.Security.OpenIdConnect;
using Owin;
namespace OldFashionOWIN
{
    public partial class Startup
    {
        public void ConfigureAuth(IAppBuilder app)
        {
            app.SetDefaultSignInAsAuthenticationType(CookieAuthenticationDefaults.AuthenticationType);

            app.UseCookieAuthentication(new CookieAuthenticationOptions());
            app.UseOpenIdConnectAuthentication(
                new OpenIdConnectAuthenticationOptions
                {
                    ClientId = "cp78fakee-6915-47ek-m2g5-z6496321f8sl",
                    Authority = "https://login.windows.net/yourwebsite.onmicrosoft.com"
                });
        }
    }
}
```

Enable Authorization

5. To enable Authorization edit the following files located under KBVault.Web/Controllers

Go to **Controllers/HomeController.cs** and add the following bits shown in RED

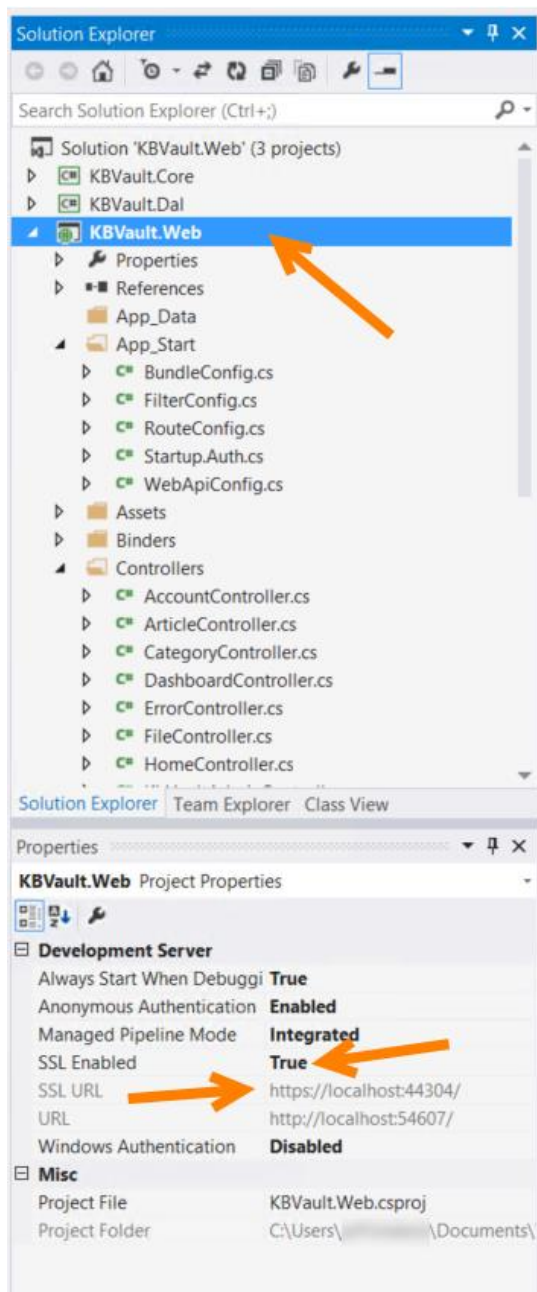
```
[Authorize]
public class HomeController : KbVaultPublicController
{
```

Go to **Controllers/KbVaultAdminController.cs** and add the following bits shown in RED

```
[Authorize]
public class KbVaultAdminController : Controller
{
```

Set Website to Use HTTPS

6. Make sure to set the project to use HTTPS:



KBVault.Web* × HomeController.cs Startup.Auth.cs Startup.cs

Application Configuration: N/A Platform: N/A

Build

Web*

Package/Publish Web

Package/Publish SQL

Silverlight Applications

Build Events

Resources

Settings

Reference Paths

Signing

Code Analysis

Start Action

☐ Current Page

☒ Specific Page

☐ Start external program

Command line arguments

Working directory

☐ Start URL

☐ Don't open a page. Wait for a request from an external application.

Servers

☒ Apply server settings to all users (store in project file)

IIS Express

Project Url

☐ Override application root URL

Debuggers

☒ ASP.NET ☐ Native Code ☐ SQL Server ☐ Silverlight

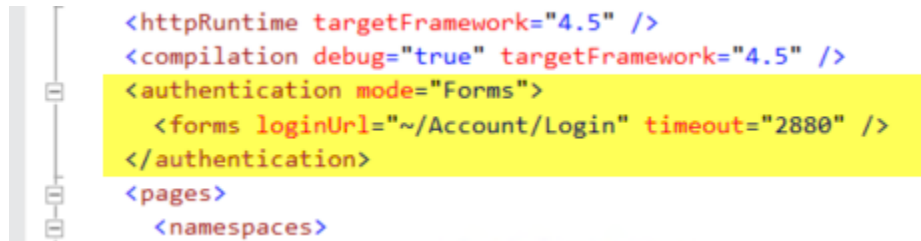
☐ Enable Edit and Continue

Edit the Web.config

7. Open the Web.config file
8. Change the Database Connection String (if needed)

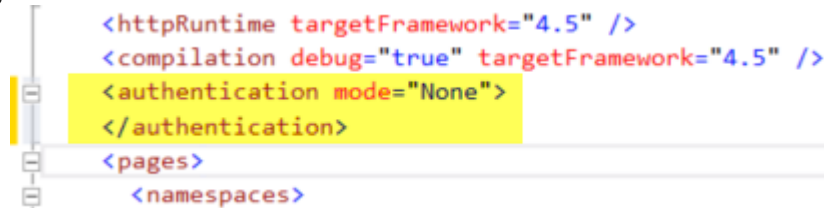
```
<connectionStrings>
  <add name="KbVaultEntities"
  connectionString="metadata=res://*/KbVaultEntityModel.csdl|res://*/KbVaultEntityModel.ssdl|res://*/KbVaultEntityModel.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=.\\dev;initial catalog=kbvault;integrated security=True;multipleactive resultsets=True;application name=EntityFramework&quot;;providerName=System.Data.EntityClient" />
</connectionStrings>
```

9. Locate the lines shown below



```
<httpRuntime targetFramework="4.5" />
<compilation debug="true" targetFramework="4.5" />
<authentication mode="Forms">
  <forms loginUrl="~/Account/Login" timeout="2880" />
</authentication>
<pages>
  <namespaces>
```

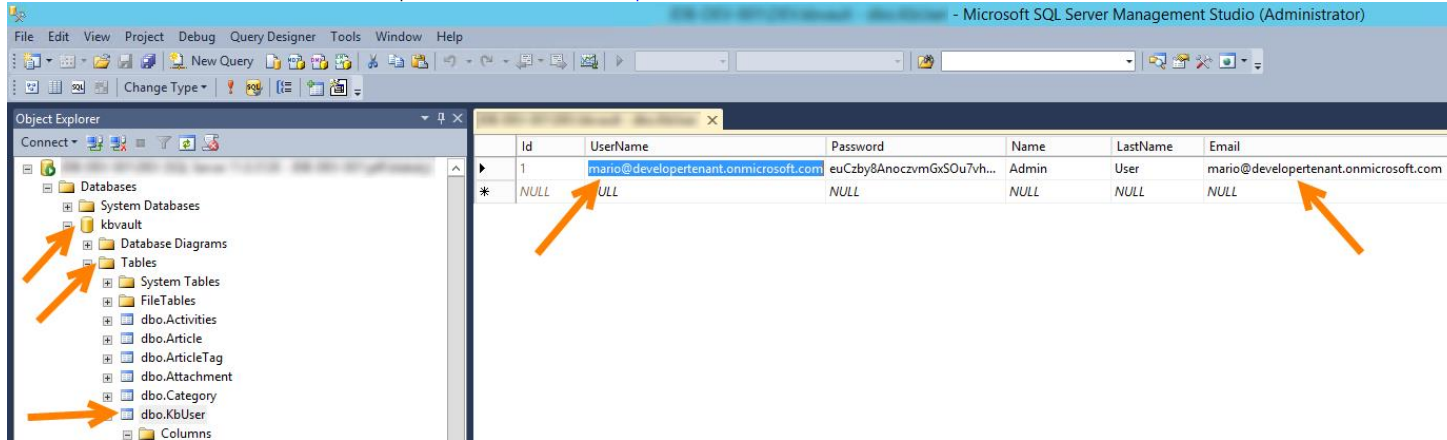
- 9.1. Change the code to look like this:



```
<httpRuntime targetFramework="4.5" />
<compilation debug="true" targetFramework="4.5" />
<authentication mode="None">
</authentication>
<pages>
  <namespaces>
```

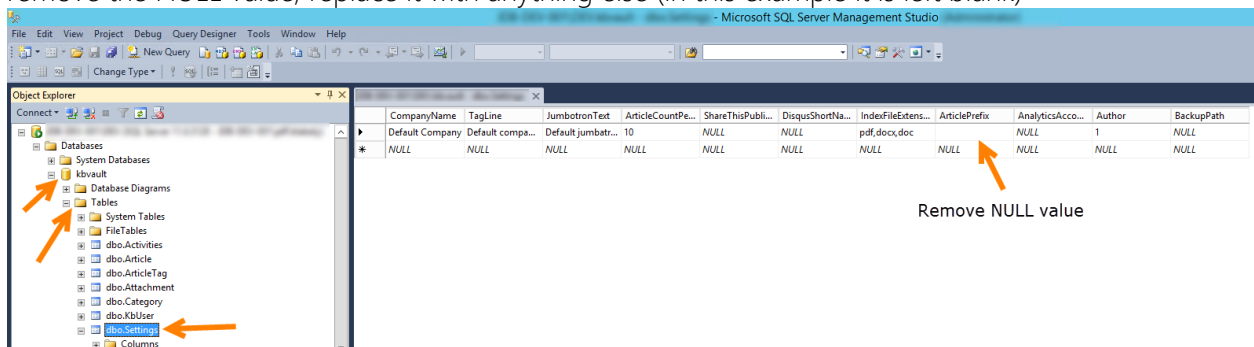
Add AD User(s) to the Database

10. In order to log in, you will need to manually add (at least 1) user to SQL DB for Admin backend access. Make sure the UserName and Email fields are exactly the same as the Azure AD username (in this example mario@developertenant.onmicrosoft.com)



Edit Settings Table in the Database

11. To fix an issue with the Search feature edit the ArticlePrefix column in the dbo.Settings Table and remove the NULL value, replace it with anything else (in this example it is left blank)



Edit Detail.cshtml.

12. Since Forms Based authentication isn't being used, remove the following line of code in the Detail.cshtml file located in KBVault.Web\Views\Home

BEFORE:

```
<!--  
    RIGHT SIDEBAR  
-->  
<div class="col-xs-2">  
    <div class="row clearfix">  
        @Html.AntiForgeryToken()  
        <a href="javascript:void(0)">  
    </div>
```

AFTER:

```
<!--  
    RIGHT SIDEBAR  
-->  
<div class="col-xs-2">  
    <div class="row clearfix">  
        <a href="javascript:void(0)">  
    </div>
```

Test

13. In Visual Studio Press F5 (debug) and you should see:



Sign in

Sign in with your organizational account

mario@developertenant.onmicrosoft.com

.....

☐ Keep me signed in

Sign in

Cancel

[Can't access your account?](#)