**Steps to configure Azure AD authentication:**

* Build Base Code and publish in local to verify code is working without issues.
* Change .net framework version to 4.6 for every project in the solution.
* Re-Build projects individually based on their dependencies.
* Fix project reference issues (if any).
* Verify the third party dll controls. There should not be an impact of framework change.
* In web.config, modify the <authentication> tag under <system.web> section as below:

**<authentication mode="None"/>**

* Under < appSettings> tag in web.config, add the following keys:

**<add key="ClientValidationEnabled" value="true" />**

**<add key="UnobtrusiveJavaScriptEnabled" value="true" />**

**<add key="ida:ClientId" value="……." />**

**<add key="ida:Tenant" value="……." />**

**<add key="ida:AADInstance" value="https://login.microsoftonline.com/{0}" />**

**<add key="ida:PostLogoutRedirectUri" value=" https://[App Service Name].azurewebsites.net”/>**

Note:

1. Go to ApplicationResourceGroup🡪 Access Control(IAM)🡪 find AD name(similar to ‘SPN.(ogrname/AppName)Prod.internet.Prod’)
2. ida:Tenant is the AAD tenant name In Azure Active Directories-> go to properties-> Directory Id here is the required Tenant ID.
3. ida:ClientId is the AAD Application ID from the Azure portal Azure Active Directories🡪 go to ‘App Registrations’🡪 search for the AD name🡪 AppId for this AD name is ClientId
4. ida:PostLogoutRedirectUri is the new SSL URL of the application; this must match with the *Reply Url* of the Azure AD Application ex:https://xx.xxx.xxx.x

* Add the following ASP.Net OWIN middleware NuGets to the web application:
  + Microsoft.IdentityModel.Protocol.Extensions (version 1.0.3.3)
  + System.IdentityModel.Tokens.Jwt (make sure 4.0.3 version is installed and not 5.0.0)
  + Microsoft.Owin.Security.OpenIdConnect (version 3.0.1)
  + Microsoft.Owin.Security.Cookies (version 3.0.1)
  + Microsoft.Owin.Host.SystemWeb (version 3.0.1)
* Re-Build the solution and publish in local to verify the build works fine.
* Add new class file to the root of Web Application (Startup.cs).
* Add system.configuration dll in the project references if doesn’t exist.
* Add the following code to the file:

**using Microsoft.Owin;**

**using Owin;**

**using Microsoft.Owin.Security;**

**using Microsoft.Owin.Security.Cookies;**

**using Microsoft.Owin.Security.OpenIdConnect;**

**using System.Configuration;**

**using System.Globalization;**

**using System.Threading.Tasks;**

**public class Startup**

**{**

**private static string clientId = ConfigurationManager.AppSettings["ida:ClientId"];**

**private static string aadInstance = ConfigurationManager.AppSettings["ida:AADInstance"];**

**private static string tenant = ConfigurationManager.AppSettings["ida:Tenant"];**

**private static string postLogoutRedirectUri = ConfigurationManager.AppSettings["ida:PostLogoutRedirectUri"];**

**string authority = String.Format(CultureInfo.InvariantCulture, aadInstance, tenant);**

**public void Configuration(IAppBuilder app)**

**{**

**app.SetDefaultSignInAsAuthenticationType(CookieAuthenticationDefaults.AuthenticationType);**

**app.UseCookieAuthentication(new CookieAuthenticationOptions());**

**app.UseOpenIdConnectAuthentication(**

**new OpenIdConnectAuthenticationOptions**

**{**

**ClientId = clientId,**

**Authority = authority,**

**PostLogoutRedirectUri = postLogoutRedirectUri,**

**RedirectUri = postLogoutRedirectUri,**

**Notifications = new OpenIdConnectAuthenticationNotifications**

**{**

**AuthenticationFailed = context =>**

**{**

**context.HandleResponse();**

**context.Response.Redirect("/Error?message=" + context.Exception.Message);**

**return Task.FromResult(0);**

**}**

**}**

**});**

**}**

**}**

* Open or create ‘Global.asax’ file to include the following code:

**void Session\_Start(object sender, EventArgs e)**

**{**

**if (!Request.IsAuthenticated)**

**{**

**System.Web.HttpContext.Current.GetOwinContext().Authentication.Challenge(new AuthenticationProperties { RedirectUri = "/" }, OpenIdConnectAuthenticationDefaults.AuthenticationType);**

**}**

**Else**

**{**

**Already existing code**

**}**

**}**

* Place a similar code in the Page\_Load method of Master Page, if exists.
* Also, place this code in start page of the application where CPassId is getting filled in Session variable.

For Eg: [Home.aspx.cs](http://Home.aspx.cs), Index.aspx.cs, etc.

>>!

* Comment code (if exists) from the start page of application.

For Eg: Index.asp.

**Note: By doing this, we are preventing UserId Session variable to populate in Client Side. We have to fill Session variable in Server Side.**

**<!--Response.Buffer = True**

**Response.Expires = 0**

**response.expiresabsolute = Now() -1**

**Response.addHeader "pragma","no-cache"**

**response.addHeader "cache-control","private"**

**response.addHeader "cache-control","no-cache"**

**Response.addHeader "cache-control","no-store"**

**response.cacheControl = "no-cache"-->**

**<!--'from E-pass page**

**DIM strUserID**

**strUserID = LCase(EPass\_ID())**

**<input type='hidden' id='RSA\_ID' name='RSA\_ID' value='=strUserID'>-->**

* Look for the code where UserId Session variable is filled up. Most probably, it should be start page of application or Global.ascx file. Comment the existing code and get EpassId from the following method.

**private string GetEpassId()**

**{**

**string EPassId = string.Empty;**

**if (!string.IsNullOrWhiteSpace(HttpContext.Current.Request.ServerVariables["AUTH\_USER"]))**

**{**

**EPassId = HttpContext.Current.Request.ServerVariables["AUTH\_USER"];**

**if (EPassId.Contains("@"))**

**EPassId = EPassId.Trim().Split('@')[0];**

**}**

**Else // Else block varies from application to application.**

**{**

**EPassId = "input\_not\_allowed";**

**}**

**Session["EPASS\_ID"] = EPassId; //Modify Session variable name as per application.**

**return EPassId;**

**}**

* Re-Build the application and publish to local path to verify the code.