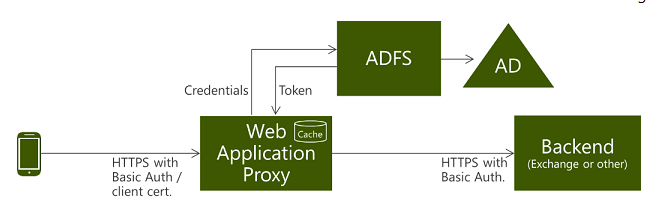
Changing from Active Directory to Active Directory Federated Services.

# How ADFS Works

AD FS is a claims-based authentication solution. It relies on claims about users. Claims are made up of user data such as name, email, etc. When users authenticate AD FS gathers information about the user from Active Directory then AD FS, acting as the issuer, issues a token to the user. That token is sent on to the claims based web application. The web application can then query the claims to get information about the user.



# Implementing the ADFS sample application

This is a step by step process documenting how we moved the OST application from an internal Active Directory application to an external web application authenticating through ADFS. The first step was to configure a claims aware sample application that Microsoft provides just for this purpose.

## Download and open the sample application

You can download the sample application from htpps://msdnshared.blob.core.windows.net/media/TNSBlogsFS/prod.evol.blogs.technet.com/Telligent.evolution.components.attachments/01/8598/00/00//03//64/54/88/SampApp%20and%20Rules.zip.

Extract the sample application. Open up visual studio and navigate to the solution.

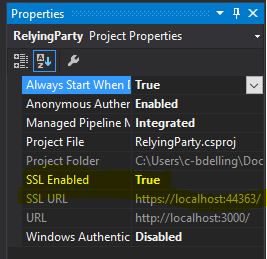
## Get ADFS information from your server admin.

You will need to reach out to the server administrator for your adfs server and get the following.

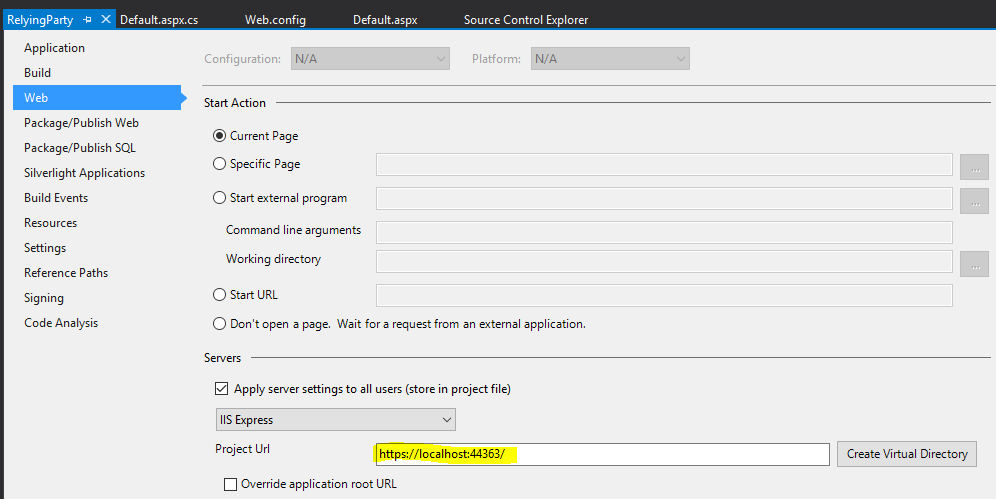
1. Your federation meta data file (ours was <https://www.fedsvc3copa.beta.pa.gov/federationmetadata/2007-06/FederationMetadata.xml>)
2. The name of your adfs server. (ours was <https://www.fedsvc3copa.beta.pa.gov/adfs>)
3. The thumbprint of the adfs certificate

## Configure your local application to run using https.

In order to run your application locally against adfs you will need to run as https. On your project go to properties (F4 key) and change ssl to enabled and write down the https address you are given.



Also update your project url on web tab of the properties



Finally contact your adfs server admin and ask them to add this url to their claims.

(Mitra Nayan Could you add <https://localhost:44363/>  to the claim configuration. When you have an opportunity. Thanks Bryan)

## Update the sample app web config.

You are now ready to update the web config and see if you can get the sample application to run, authenticate, and show you claims.

Here is the original web config



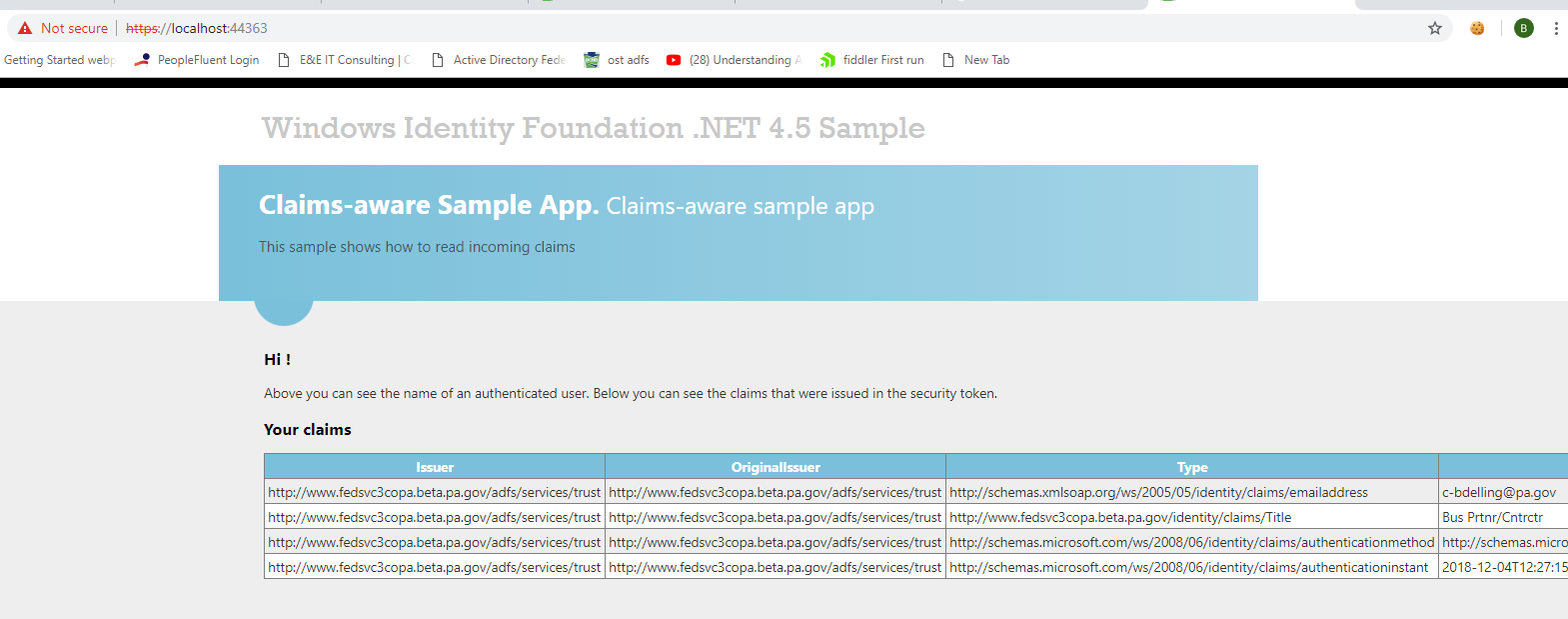
Update the web config

1. change all instances of <https://sts.contoso.com/adfs> to the name of your adfs server. Change the name
2. change all instances of <https://app1.contoso.com/sampapp/> to the url for your local host (for example <https://localhost:44363/>
3. all instances of the thumbprint to the thumbprint you got from the server admin.

Here what my updated config file looked like.



If everything went well, you should be directed to enter your credentials and then be redirected back to your local application and see the claims aware application



## Deploy the application to a web server.

You are now ready to deploy the sample application to a web server. Do a build and deploy your files to your web server making sure to change the local host url to the url of the actual application.

Ask the server admin to add your url to the claims configuration

(Mitra Nayan Could you add <https://costa.beta.pa.gov/> to the claim configuration. When you have an opportunity. Thanks Bryan)

(for example change all occurences of <https://localhost:44363/>  to <https://costa.beta.pa.gov/> in the web config file).

Attempt to navigate to the url and make sure you can see the claims aware application.

## Troubleshooting.

If you end up with the error

Server Error in '/' Application.

*WIF10201: No valid key mapping found for securityToken: 'System.IdentityModel.Tokens.X509SecurityToken' and issuer: 'http://www.fedsvc3copa.beta.pa.gov/adfs/services/trust'.*

Try replacing the files in wwwroot with all the files from your local solution. (open the folder up in file explorer copy all the files into wwwroot and run the application as a web page instead of a web site. We had to do this for some reason and it took us a long time to figure it out.)

# Updating your Active Directory Web Application to Use ADFS.

Now that the sample application is up and running you are ready to update your actual application to use ADFS.

Open up your application in visual studio and once again follow the steps above to change the url to run as https instead of http.

Update your web config to set requireHttps to true.

Once again ask your adfs server admin to add the claim for your url.

You will then need to update your web config to add in all the relevant adfs information. I am just providing a before and after web config as an example.

Original Web Config file before using ADFS.



Here is the web config file after it has been updated to use adfs.



Finally you may need to change portions of your application to use claims if it can not find the information from active directory. For our application this occurred in the base controller. Here is the relevant code. It looks for the username from active directory. If it can not find it it checks if there is a claim for username. If it still can’t find it it looks for an email claim and then finds the username from the user database table. (see code below)

public static string UserName

{

get

{

if (System.Web.HttpContext.Current != null && System.Web.HttpContext.Current.User != null)

{

var name = string.Empty;

if (System.Web.HttpContext.Current.User.Identity.Name != null)

{

name = System.Web.HttpContext.Current.User.Identity.Name;

}

else

{

// we did not find a current user in httpcontext so we are probably using ADFS instead of AD look at the claims to find the username

var claimsPrincipal = System.Web.HttpContext.Current.User as ClaimsPrincipal;

name = claimsPrincipal.Claims.Where(c => c.Type == ClaimTypes.Name).Select(c => c.Value).SingleOrDefault();

// if we can not find the username from the Name Claim Type use the Email Claim to get the name from the user table.

if (name == null || string.IsNullOrEmpty(name) || string.IsNullOrWhiteSpace(name))

{

var email = claimsPrincipal.Claims.Where(c => c.Type == ClaimTypes.Email).Select(c => c.Value).SingleOrDefault();

ApplicationDataContext db = new ApplicationDataContext();

name = db.Users.FirstOrDefault(x => x.Email == email).UserName;

}

}

if (name == null || string.IsNullOrEmpty(name) || string.IsNullOrWhiteSpace(name))

{

throw new ApplicationException("unable to locate a valid username");

}

return name;

}

else

{

return BaseControllerStrings.System;

}

}

}