Configuring MVCForum for SSO on Windows Identity Foundation (WIF)

The first thing you need for SSO on WIF is an Issuer. The issuer can be thought as a separate website that handles authentication for RelyingParty's that use it. A Relying Party is any website configured to use the issuer site for WIF Passive Redirect SSO. In this context MVCForum will be the relying party and the issuer is a login site you need to build (or use one already in your use).

The issuer site can authenticate any way you want, ADFS, NTLM, Basic, Forms, CasDotNetClient (uses forms), etc etc. What's important is what the issuer will do with url requests from a relying party, once it's authenticated.

The SSOLogin example in the MVCForum source code handles authentication via Forms Auth, and processesing the WIF Signin and Signout requests and should be a pretty good example to get everything you need setup.

In it's simplest form, the Issuer site needs to handle the following,

1. Unauthenticated Login

This means that when a relying party redirects to the Issuer, the issuer should present them with some form of authentication process, then authenticate.

2. Authenticated Redirect back to the RelyingParty

Once the Issuer has authenticated the user (via insert method here), it needs to redirect back to the relying party with a sign in url request. You don't need to worry about the details of the request becase WIF handles building it for you, you just need to call a method...

FederatedPassiveSecurityTokenServiceOperations.ProcessRequest(System.Web.HttpContext.Current.Request, User as ClaimsPrincipal, WIF.ExampleSecurityTokenServiceConfiguration.Current.CreateSecurityTokenService(), System.Web.HttpContext.Current.Response);

3. A custom SecurityTokenServiceConfiguration implementation

There are many examples of these on the internet, but going off the SSOLogin example in MVCForum, all you really need to implement is a Current Static Singleton method to get it's instance, and it's constructor with an IssuerName.

4. A custom SecurityTokenService implementation

There are many examples of these as well. But you need to implement some X509SigningCredentials and X509EncryptingCredentials, Override GetScope, GetOutputClaimsIdentity, and create a method to validate incoming url requests.

There is a powershell script in the SSOLogin site to generate some test certificates. However, once that script is run you need to go into the certificate snap in in the MMC console and grant permissions for all the service accounts to access the private keys.

Start -> Run -> MMC -> Add Certificates Snap in -> Trusted People -> Right Click TokenSigningCert -> All Tasks -> Manage Private Keys -> Add Users for full control

Repeat for TokenEncryptingCert.

Going into depth on the SecurityTokenService

* GetScope

This method is designed to match up certificates to specific relying parties based on their urls. In the SSOLogin implementation, we assume all relying parties (sites using SSO) will use the same certificate, TokenSigningCert

* GetOutputClaimsIdentity

This method is used to add additional claim objects to the principal that is being authenticated and having a session set for.

MVCForum will listen to a claim type of

schemas.mvcforum.com/claims/isforumadmin

If a user has this claim and it's value is "true" that user will be added to the admin mvcforum on login (every time, even if removed). You could use the method to add said claim. However you can also create a ClaimsAuthenticationManager implementation and do the same thing there, but that is not included in the scope of this tutorial (but is simple to do and add to the web.config).

* ValidateAppliesTo
* This method should check the request.Url and scope to determine if the relying part has access to use the Issuer Site. If not, throw an InvalidRequestException to deny the use of the issuer for authentication.

The SSOLogin implementation doesn't do anything, but it does match up the certficates so a RelyingParty can't use the Issuer without having the certificate in it's certificate store and configured in the web.config.

5. MVCForum Web.Config Settings to use Issuer

Configure the following app setting Values (all required)

<add key="IsWIFRelyingParty" value="true"/>  
IsWIFRelyingParty tells MVCForum to bypass forms authentication and use WIF SSO

<add key="WIFAdminUser" value="rmann" />  
WIFAdminUser tells MVCForum to treat the specific user as an admin when logged in from WIF SSO, otherwise you won't be able to do anything to admin MVCForum when logged in with WIF for the first time. The value should be the value of the name claim "<http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name>"  
  
If your Issuer is using Forms Authentication, the name claim is the MembershipUser.UserName value. If you are using NTLM it will be the windows user name or active directory samAccountName (the old version, not the new principalName with domain tacked on it), etc.  
  
Configure the following sections

<system.identityModel>   
 <identityConfiguration saveBootstrapContext="true">  
 <issuerNameRegistry type="System.IdentityModel.Tokens.ConfigurationBasedIssuerNameRegistry, System.IdentityModel, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">  
 <trustedIssuers>  
 <add thumbprint="97f983a05587253b6835d1bd0062000c5d1f398d" name="TokenSigningCert" />  
 </trustedIssuers>  
 </issuerNameRegistry>  
 <audienceUris mode="never" />  
 </identityConfiguration>  
 </system.identityModel>

The trusted issuer thumbprint needs to be looked up using the MMC -> Certificates snapin mentioned earlier. Thumbprint is a property on the certification and the name is the Certificates Subject. The thumbprint has spaces in it in Certificates, make sure you remove those.

Audience Urls are all of the sites allowed to use the Issuer (SSOLogin doesn't really use this information as it's ValidateAppliesTo method doesn't do anything, just returns void and let's them in. As such it's mode is set to none here. If you are checking them though, then don't set it to none and add entries for the Uri's.

<system.identityModel.services>  
 <federationConfiguration identityConfigurationName="">  
 <serviceCertificate>  
 <certificateReference x509FindType="FindBySubjectName" findValue="TokenSigningCert" storeLocation="LocalMachine" storeName="TrustedPeople"/>  
 </serviceCertificate>  
 <wsFederation passiveRedirectEnabled="true" issuer="http://localhost:10000" realm="http://localhost:10001" requireHttps="false" />  
 <cookieHandler requireSsl="false" mode="Default">  
 <chunkedCookieHandler chunkSize="2000"/>  
 </cookieHandler>  
 </federationConfiguration>  
 </system.identityModel.services>

What you need to configure is the certificateReference, which is the certificate your SecurityTokenService will use for the credentials on scope configuration.

Now for the design of the implementation on MVCForum, you should leave passiveRedirectEnabled set to true, but you need to change the issuer to be the Url of your Issuer site. In the SSOLogin example, it was running with no dns binding on port 10000.

The realm, is the site the web.config is on E.g. (MVCForum) and in the source examples, MVCForum was running on port 10001.   
  
If your sites are using SSL, then set requireHttps to true, otherwise leave it false. If you set it to true, I will assume you have the SSL Certificate etc because you'll likely need to use that for the certificate configuration above.

The cookie handler handles how WIF will persist the session. The above settings will not generate persistent cookies and will instead create session cookies, meaning if the user closes their browser their logged in status is gone and they will need to log back in on next browser page load. To make it persistent, add the persistentLifeTime attribute and give it a value of how long to keep the persisted cookie. Also set requireSSL here if you are using SSL. Mode should be left on Default (which will be chunked) otherwise you could run out of room for the claims if your implementation implements a lot of claims. The claims for the logged in user are encrypted and stored in the cookie. So if it's not chunked and you have a TON of claims, it will fail (and I'm not sure how gracefully). Chunksize should also be set lower than max cookie size because say it's 40 bytes from cap and needs to encrypt one more claim that is 60 bytes in size, ERROR overflow.. So basically max cookie size 4096 bytes, so if you set it to 2000 then the largest a claim string can be is 2096 bytes. If you set the chunk size to 3500, then the max claim size would need to exceed 596 bytes. And by claim size, I'm referring to the total encoded string length of a claim issued by a SecurityTokenService.

\*important notes to not forget\*

If using these default certificates... You must add the network/service accounts to have permission to access their private keys with the MMC -> Certificates snapin, or error. You must also configure the above trustedissuers section to match the certificates subject name and thumbprint (which will be different on your machine, thumbprint wise). You must also configure the certificateReference web.config entry as listed above (to match your certificate).

Rough Draft (Ryan)