Den Europæiske Union ved Den Europæiske Fond for Udvikling af Landdistrikter og Ministeriet for Fødevarer, Landbrug og Fiskeri har deltaget i finansieringen af projektet.

VIDENCENTRET FOR LANDBRUG

# Configuring ASP.NET application to use AD FS 2.0

## A step-by-step guide

# Indhold



# 1 Dokumentinformation

## 1.1 Status

| Status | Beskrivelse |
|---|---|
| Released | Document is final. |

## 1.2 Versionshistorik

| Dato | Version | Initialer | Beskrivelse |
|---|---|---|---|
| 19-12-2012 | 1.0 | MCM | Initial version. |
| 10-01-2013 | 1.1 | MCM | Entity Framework has been removed. |

# 2 Configure ASP.NET application to use AD FS 2.0 - A step-by-step guide

This guide assumes that the following components are installed and configured on the development server:

1. IIS version 7.0 or later
2. ASP.NET 4.0
3. Visual Studio 2010 or later
4. Windows Identity Foundation 3.5 and Windows Identity Foundation SDK 4.0 (elaborated in step 2)
5. NuGet Package Manager

If the ASP.NET web application is using the deprecated DLI-SSO federation service as a claims provider, this configuration must be removed from the configuration file and any reference to DLI-SSO assemblies must be removed from the C# project before the application can be AD FS 2.0 enabled.

## 2.1 Step 1 – Create IIS web site and configure it to use SSL

Note: If a web site with host name “localhost.vfltest.dk” and configured with SSL certificate “*.vfltest.dk” already exists, this step can be skipped.

Prerequisites:

*.vfltest.dk SSL certificate file (and password)

To ease installation of SSL certificate, creation of IIS application pool and web site, a PowerShell script is available (TFS: $/DLBRLogin/DLBRLogin/trunk/Tools/Scripts/CreateWebsite.ps1).

The script does the following:

1. Install SSL certificate.
2. Create IIS Application Pool named “localhost.vfltest.dk”, supporting .NET Framework Version 4.0 running in Integrated Pipeline Mode with identity “NetworkService”.
3. Create a web site named “localhost.vfltest.dk” supporting HTTPS bindings.
4. Attaches certificate “*.vfltest.dk” to port 0.0.0.0:443 (HTTPS binding). If another certificate is already bound to this port, the definition will be overridden.

Note: It is important that PowerShell is executed with administrator privileges. This can be accomplished by using the “RunAs” command, where the selected user is member of “Administrators” group on the machine, e.g. runas /user:mcm powershell.

To execute the script, go to the Windows start menu and type PowerShell and select any version of Windows PowerShell:

1. If this is the first time a PowerShell script is executed on the computer, an error will occur saying that “File C:\ CreateWebsite.ps1 cannot be loaded because the execution of scripts is disabled on this system…”. The following PowerShell command must be executed in the PowerShell window: “Set-ExecutionPolicy Unrestricted”.
2. Execute the script by typing the full path to “CreateWebsite.ps1”.
3. If the SSL certificate is already installed on the computer, the script outputs “SSL certifikat 'CN=*.vfltest.dk, OU=Domain Control Validated, C=DK' er allerede installeret i store LocalMachine\My (Local Computer – Personal). If not, you will be prompted for the path to the PFX file and the password to the private key.
4. Next you will be prompted for a name for the web site. Default value is “localhost.vfltest.dk”. Note that “.vfltest.dk” part of the domain name is required to comply with the SSL certificate subject name “*.vfltest.dk”.
5. If no errors occurred during script execution, the web site is now ready and running.

## 2.2 Step 2 - Installation of Windows Identity Foundation (WIF) 3.5 and WIF SDK 4.0

Before development of a claims aware ASP.NET web application, Windows Identity Foundation 3.5 (WIF) and the WIF SDK 4.0 must be installed.

WIF 3.5 is part of .NET 3.5 and can be downloaded from http://www.microsoft.com/en-us/download/details.aspx?id=17331.

Follow the instructions on the download page. WIF 3.5 must be installed prior to WIF SDK 4.0.

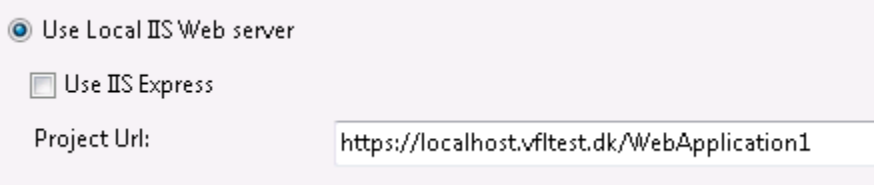
WIF SDK 4.0 can be downloaded from http://www.microsoft.com/en-us/download/details.aspx?id=4451. Choose the SDK for .NET 4.0. Note that side by side installation of the WIF SDK 3.5 and WIF SDK 4.0 is not recommended.

## 2.3 Step 3 – Create a claims aware ASP.NET 4.0 web application

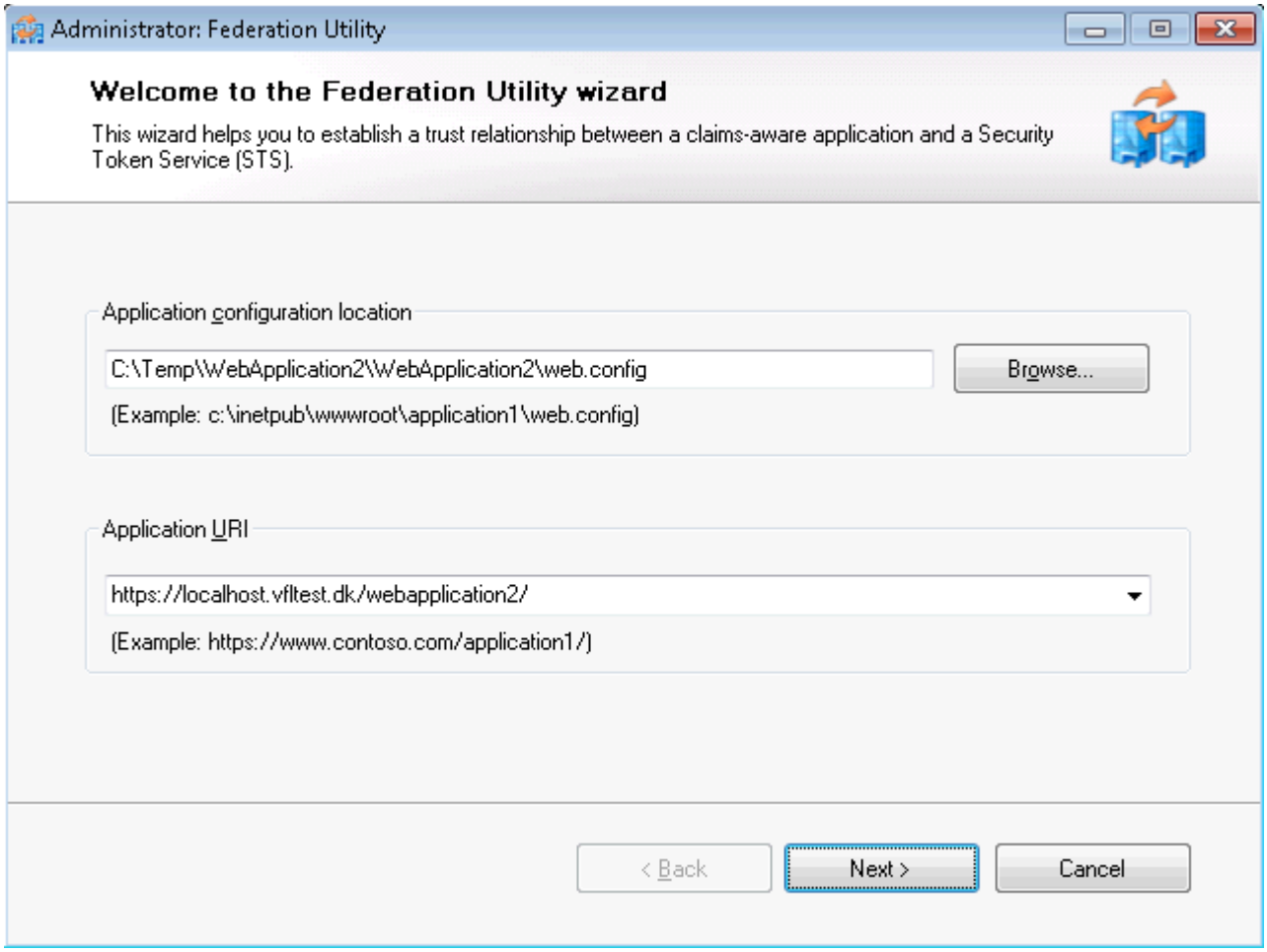
First a discussion about which version of Visual Studio to use. Visual Studio 2010 and Visual Studio 2012 both has support for .NET 4.0 applications, but only Visual Studio 2010 has built-in support for integrating an ASP.NET 4.0 web application with WIF 3.5. This integration is surfaced by the "Add STS Reference…" command, that is available when you in Solution Explorer right-click the web application project file. If you prefer to use Visual Studio 2012, you have to use the external WIF SDK 4.0 tool "FedUtil" (in Visual Studio 2010 "FedUtil" is a built-in extension that is invoked by "Add STS Reference…" command). FedUtil can be found in the path C:\Program Files (x86)\Windows Identity Foundation SDK\v4.0\FedUtil.exe.

This guide assume that the project configuration file (web.config) is empty.

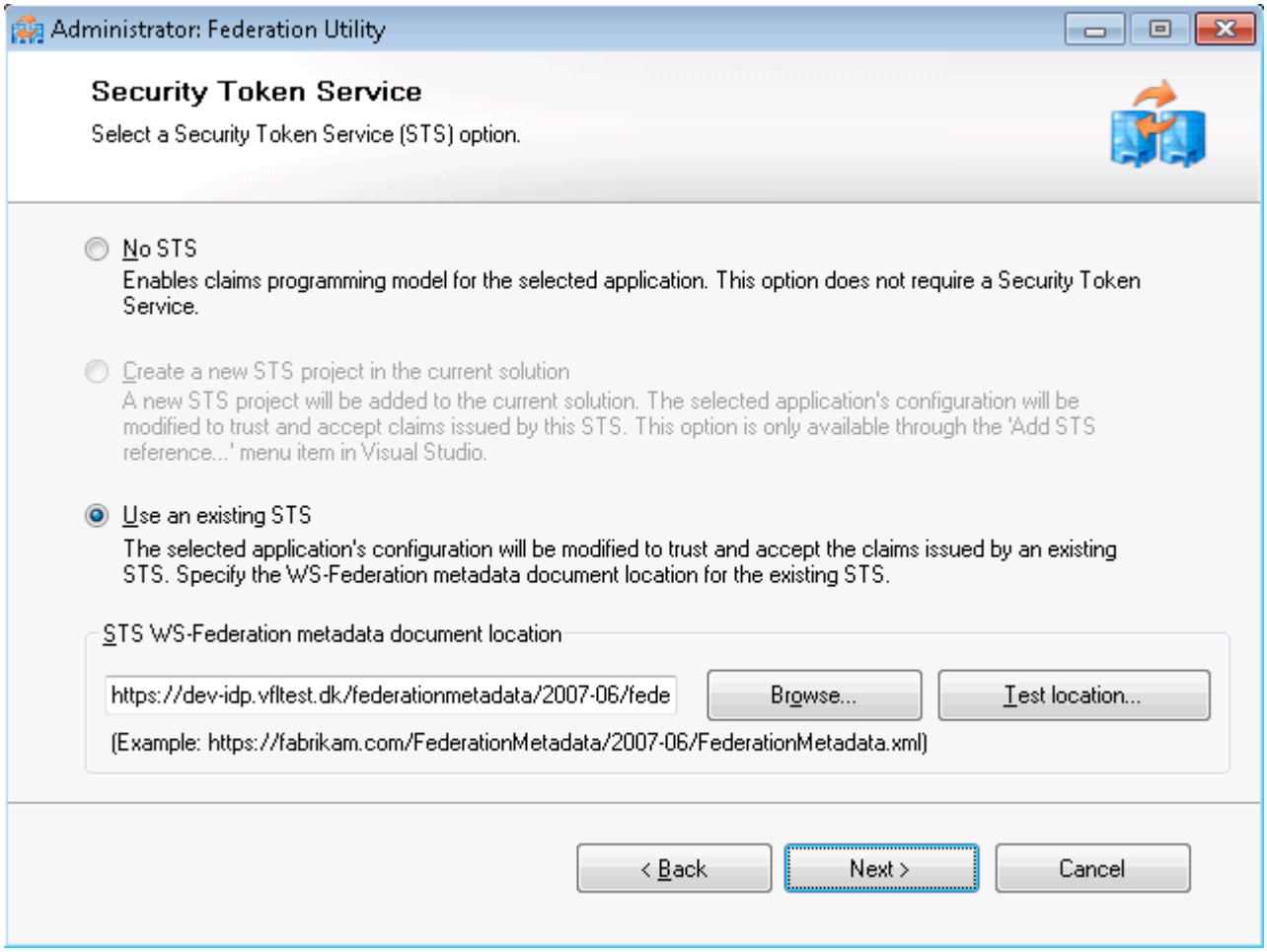
1. In Visual Studio, create a new empty ASP.NET 4.0 web application.
2. In Solution Explorer, right-click the project and select "Properties".
3. In the left pane click "Web".
4. In the "Servers" section select "Use Local IIS Web Server. Make sure that "Use IIS Express is not selected, we want to use the IIS web site we created in the previous step. It is important that the domain part of the project url is "localhost.vfltest.dk" and the URI scheme is https. Choose a relevant application name instead of "Webapplication1". Click "Create Virtual Directory". This will create a new IIS web application hosted in the "localhost.vfltest.dk" web site.



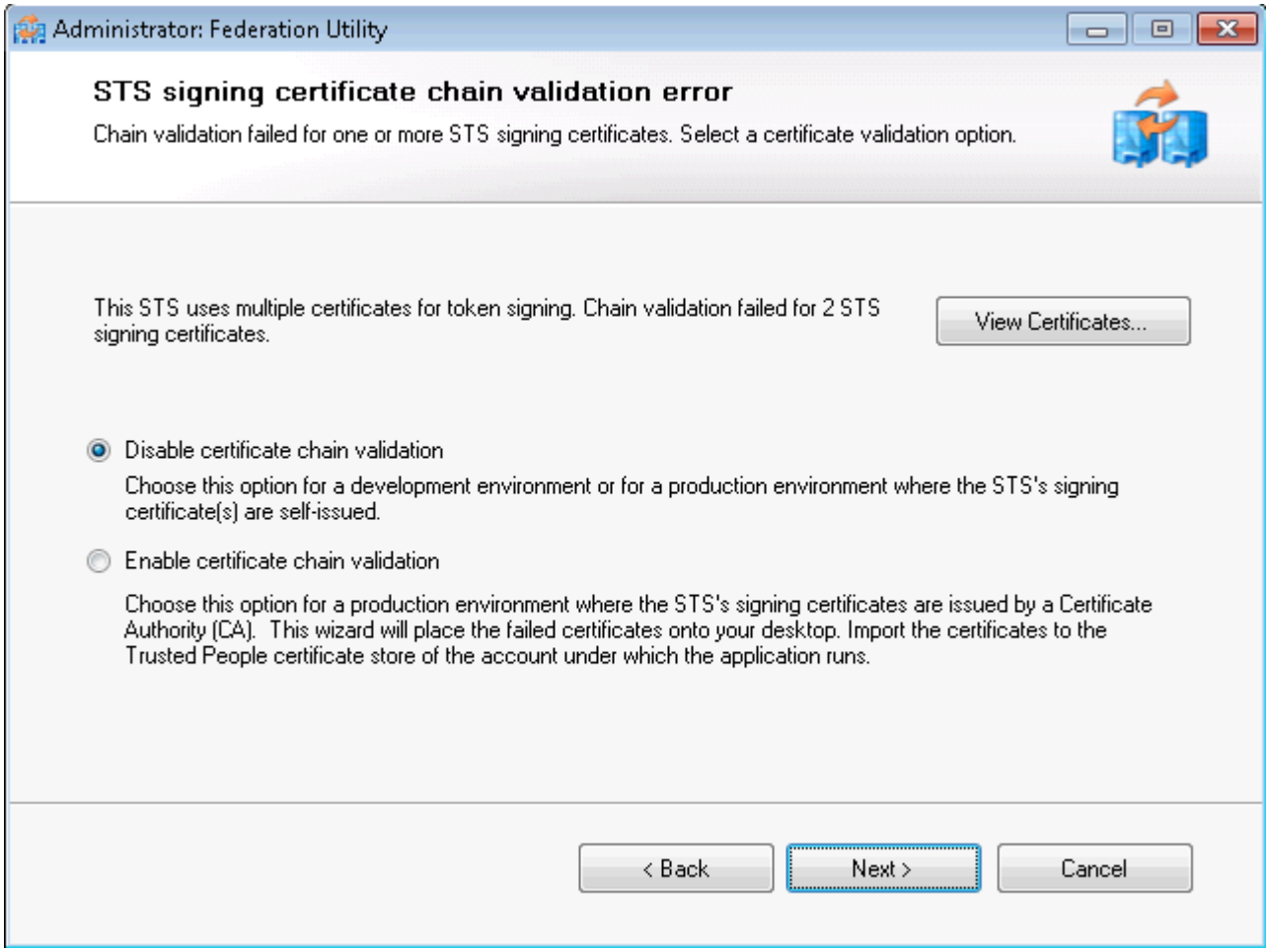
5. If Visual Studio 2010, in Solution Explorer, right-click the project file and choose menu item "Add STS Reference…". If Visual Studio 2012, execute the external application "C:\Program Files (x86)\Windows Identity Foundation SDK\v4.0\FedUtil.exe". Add the path to the web application configuration file (web.config) and the application URI. Note the trailing slash (/) in "Application URI".



6. Choose “Use an existing STS” and type “https://dev-idp.vfltest.dk/federationmetadata/2007-06/federationmetadata.xml“ or “https://devtest-idp.vfltest.dk/federationmetadata/2007-06/federationmetadata.xml“ as STS WS-Federation metadata document location, depending on whether the application must federate with DEV or DEVTEST identity provider.

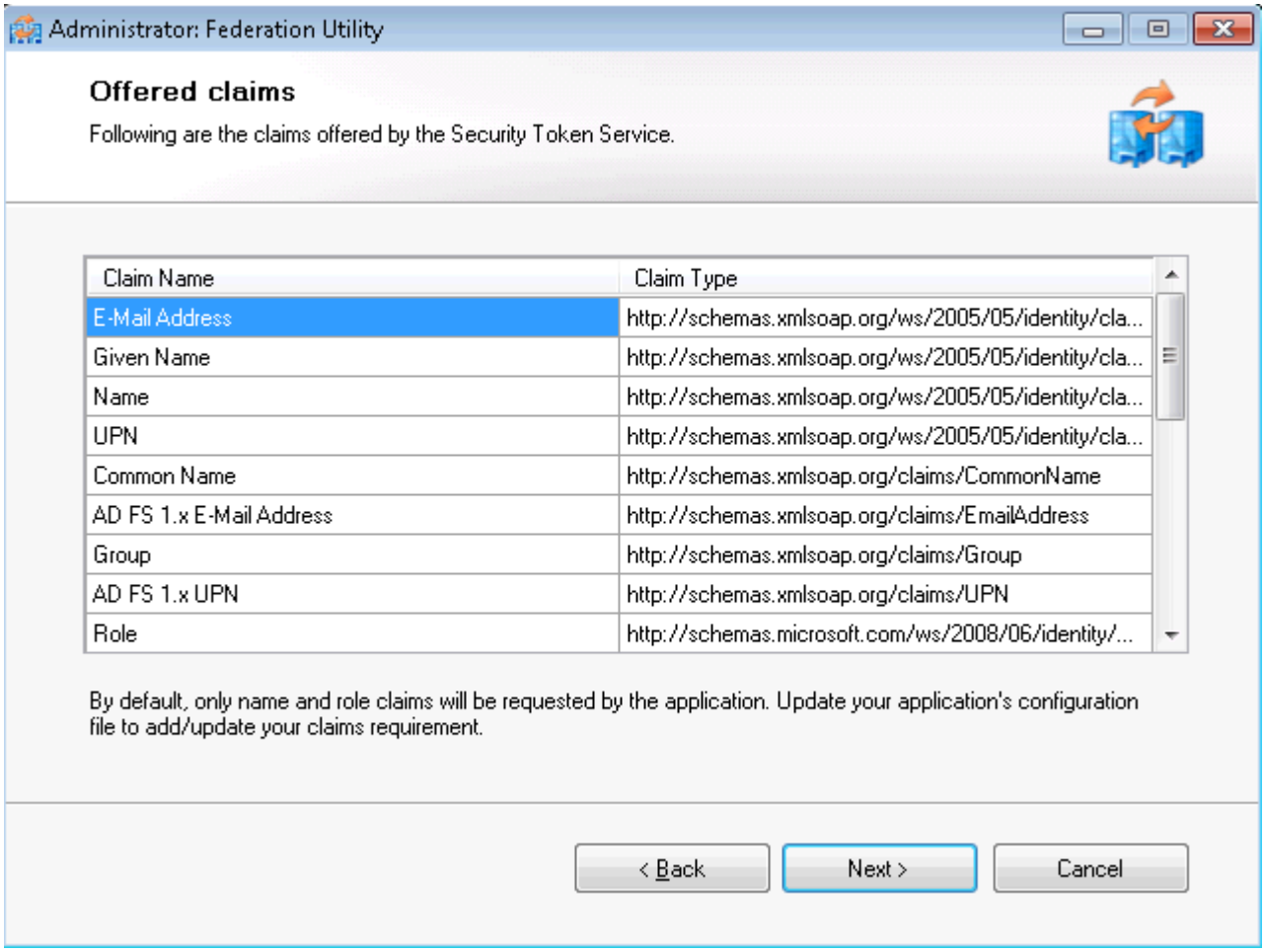


7. Choose “Disable certificate chain validation”.



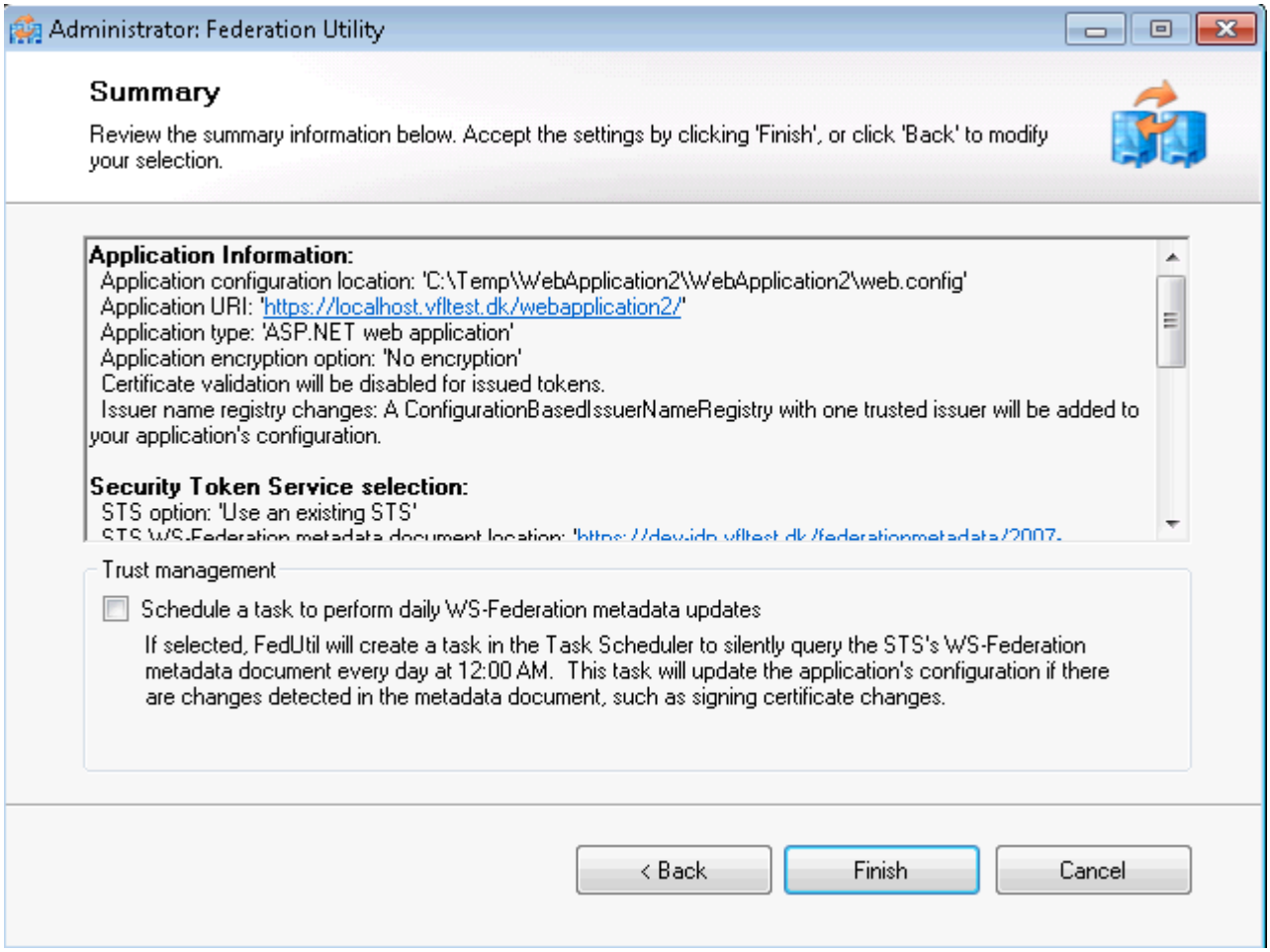
8. Choose “No encryption”.

9. A list of claims offered by the STS is presented.

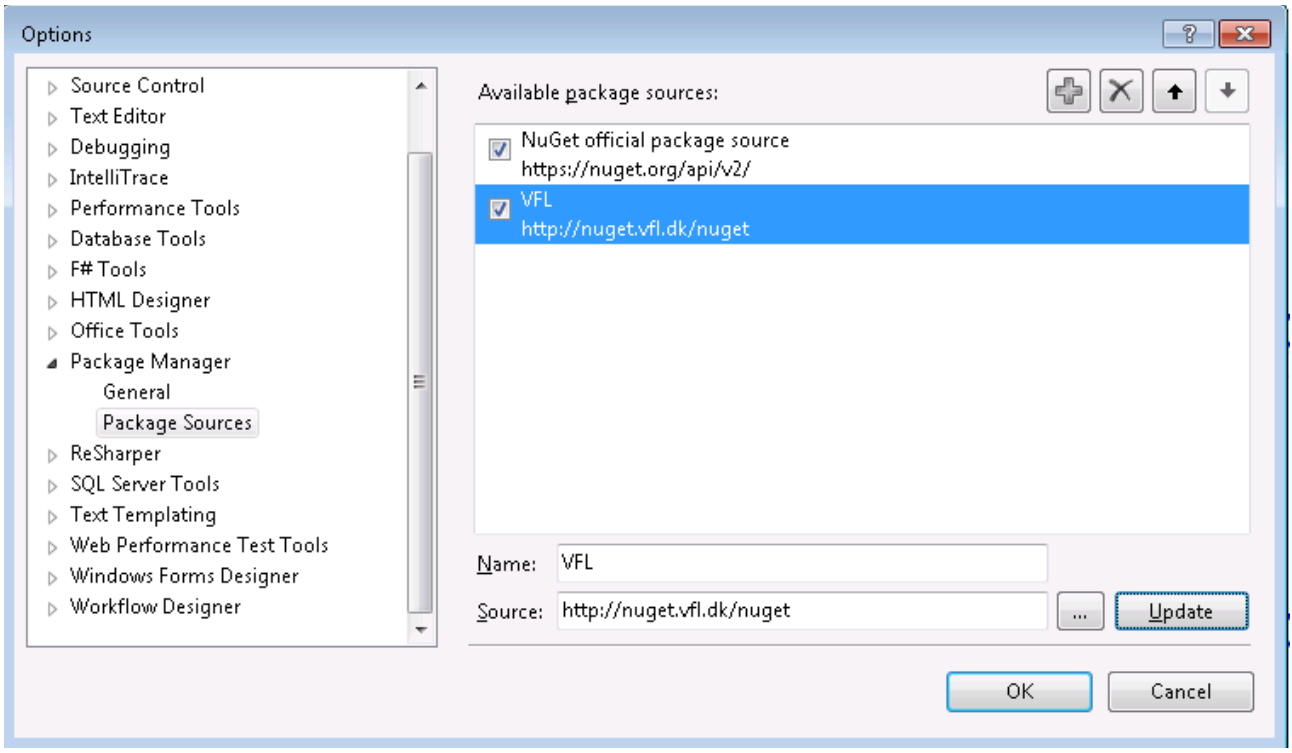



Note that this list is not maintained in the DLBR Common Login Federation, so the list is of little use (and has no bearing on the claims actually issued to the RP).

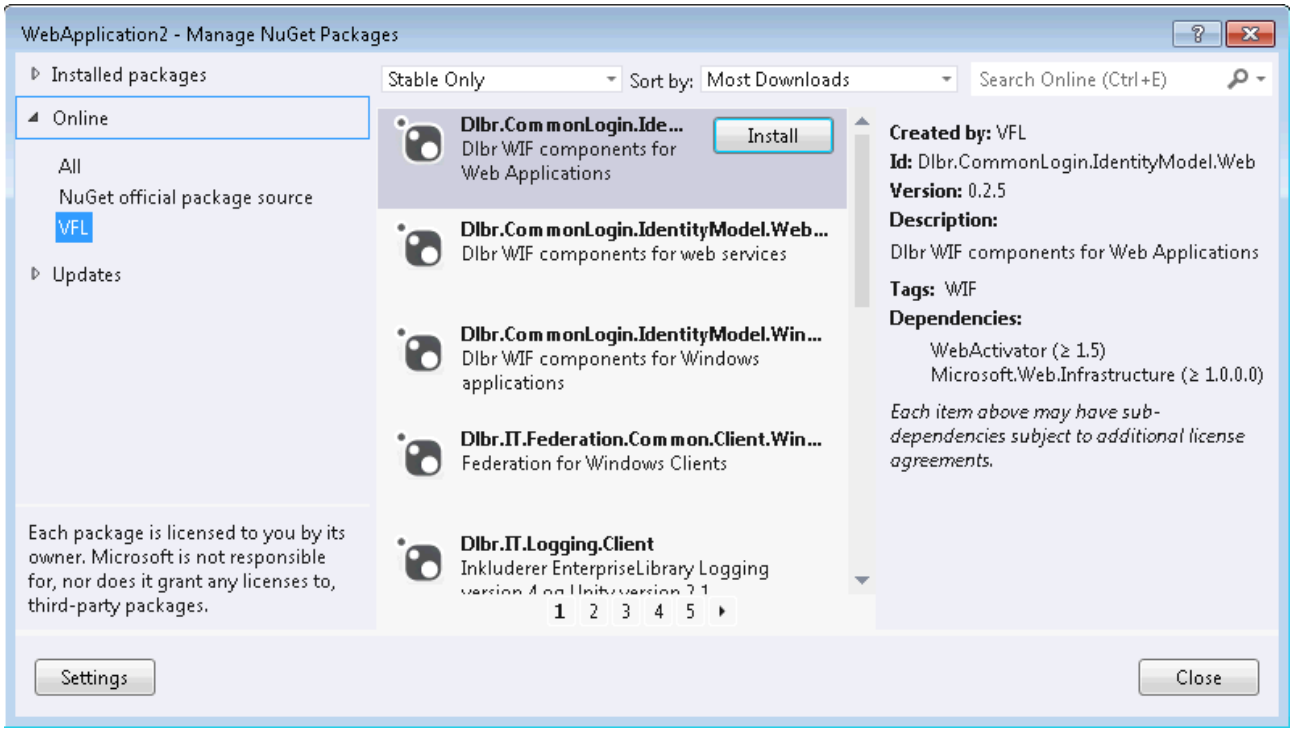
10. A summary is shown as the last step in the “FedUtil” wizard.



11. To simplify integration between the web application and the AD FS 2.0 identity/claims provider, components have been developed. To facilitate the initial plumbing, the components are available as NuGet packages in the VFL NuGet repository (http://nuget.vfl.dk/nuget). To install the packages, right-click the project in Solution Explorer and select "Manage NuGet Packages…". In the left pane choose "VFL", select package "Dlbr.CommonLogin.IdentityModel.Web" and click "Install".
    If VFL package source has been enabled as available package source in the NuGet Package Manager, continue to next step. Otherwise click "Settings", add "VFL" and "http://nuget.vfl.dk/nuget".



12. In Solution Explorer, right-click project file and choose menu item “Manage NuGet Package…”. From the VFL store select “Dlbr.CommonLogin.IdentityModel.Web” and click “Install”.




13. After adding package “Dlbr.CommonLogin.IdentityModel.Web” to the project, a connection string is added to web.config, referencing a local SQLEXPRESS instance (`"Data Source=.\SQLEXPRESS;…")`. This purpose of this database is caching security tokens issued by AD FS 2.0. The database must be created manually or an existing can be used if preferred. Creating the database can be done using a script found in TFS: $/DLBRLogin/DLBRLogin/trunk/Source/Dlbr.CommonLogin.IdentityModel.Web/Database/db.sql. The name of the database is optional and must be synchronized with the value in the connection string section in web.config. Note that the table name is not optional and must be “SecurityTokenCacheEntries”. To help avoiding the “SecurityTokenCacheEntries” table growing too large, a script containing a definition for a database job can also be found in TFS: $/DLBRLogin/DLBRLogin/trunk/Source/Dlbr.CommonLogin.IdentityModel.Web/Database/job.sql. The job delete rows older than 7 days from table “SecurityTokenCacheEntries”. Note that the database name must be changed accordingly to be in sync with the value in the connection string.

14. In Solution Explorer, right-click project file, choose menu item “Add – New Item”. In the left pane select “Web” and choose “Global Application Class”. From web.config find the following section:

```
        <!--
      To enable session mode, uncomment the following section and add the
following to Global.asax.cs:
      #############
(…….)
        -->
```

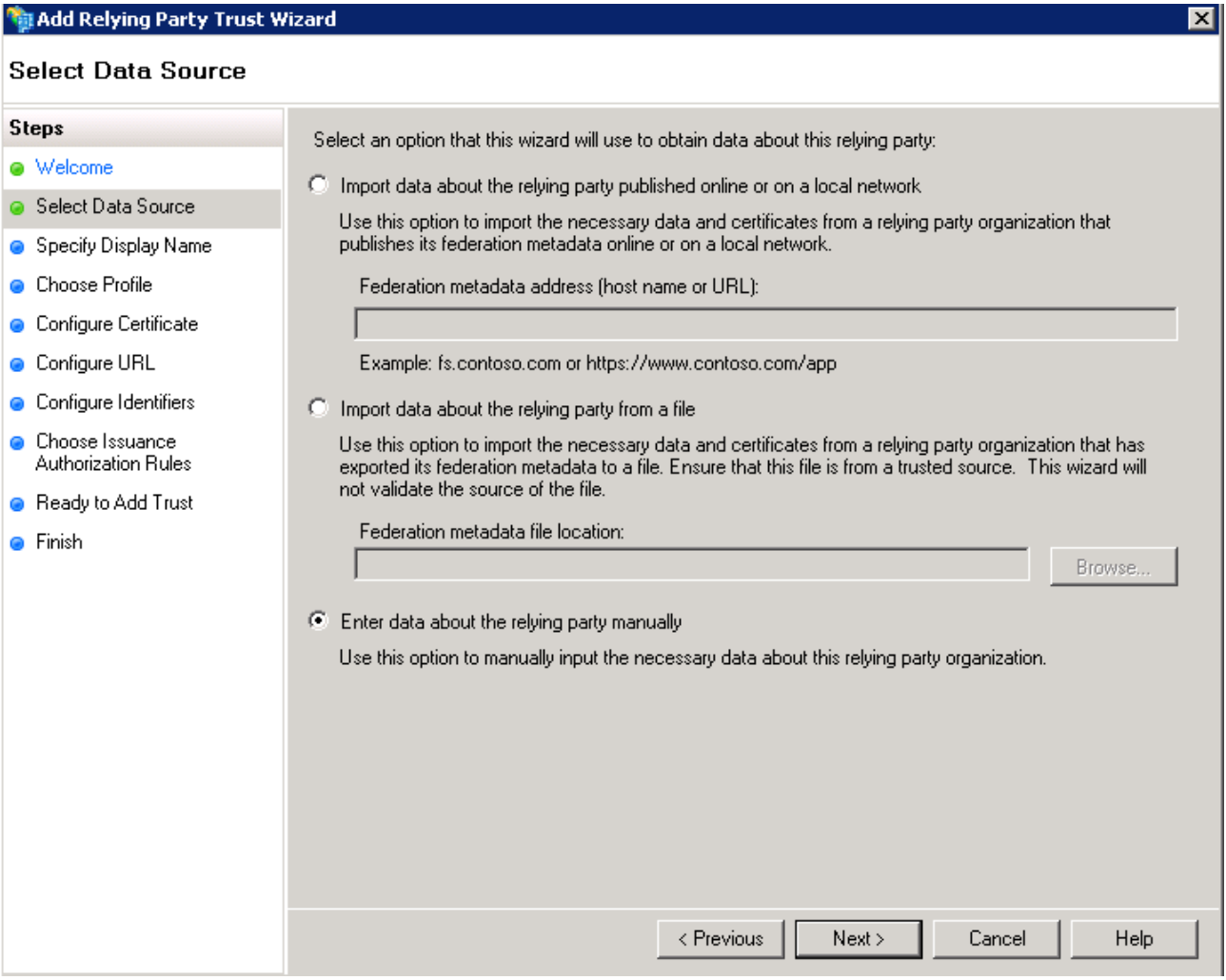
Move the C# function from web.config and add it to Global.asax. In web.config, uncomment the section:

```
<remove type="Microsoft.IdentityModel.Tokens.SessionSecurityTokenHandler……
<add type="Microsoft.IdentityModel.Tokens.SessionSecurityTokenHandler…..
```

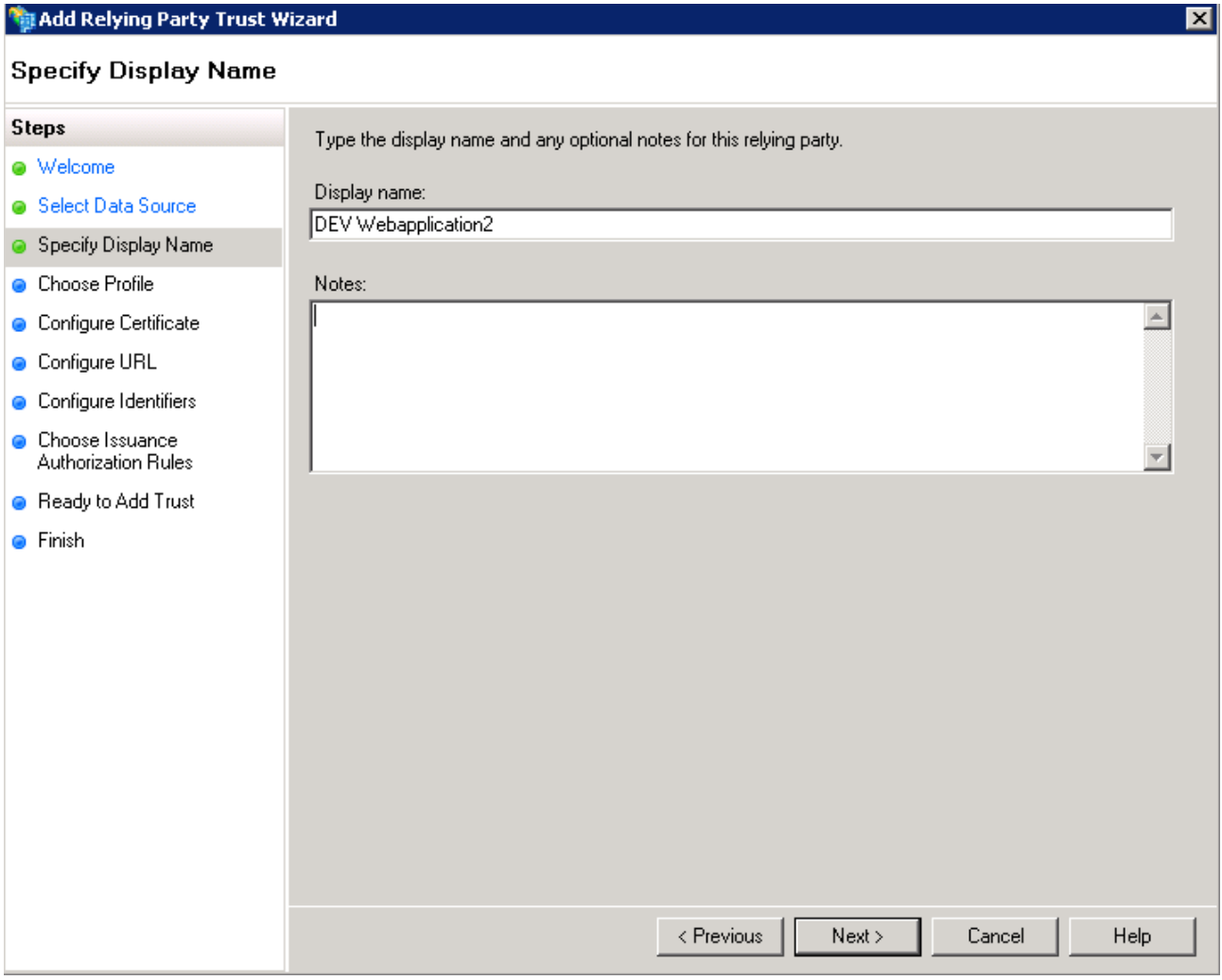
## 2.4 Step 4 – Add STS Relying Party Trust

In this step we will add a new Relying Party Trust to the STS configuration.

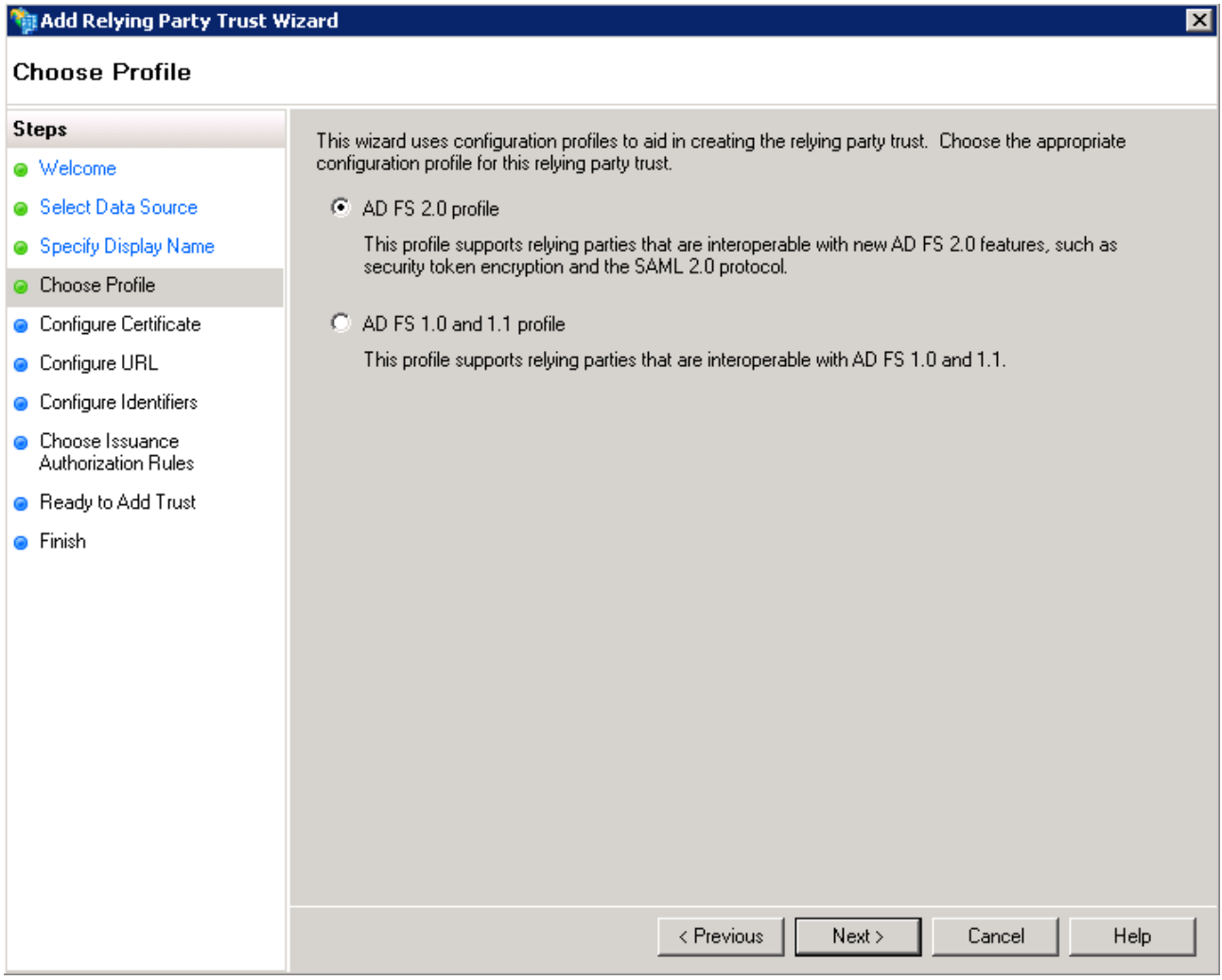
1. Login to dev-idp.vfltest.dk or devtest-idp.vfltest.dk server, depending on whether the application must federate with DEV or DEVTEST identity provider.
2. In the “Administrative Tools” menu select “AD FS 2.0 Management”.
3. In the “Actions” pane to the right, choose “Add Relying Party Trust…”.
4. Select “Enter data about the relying party manually”.



5. Enter a display name, this value is purely informational, so any value will do.

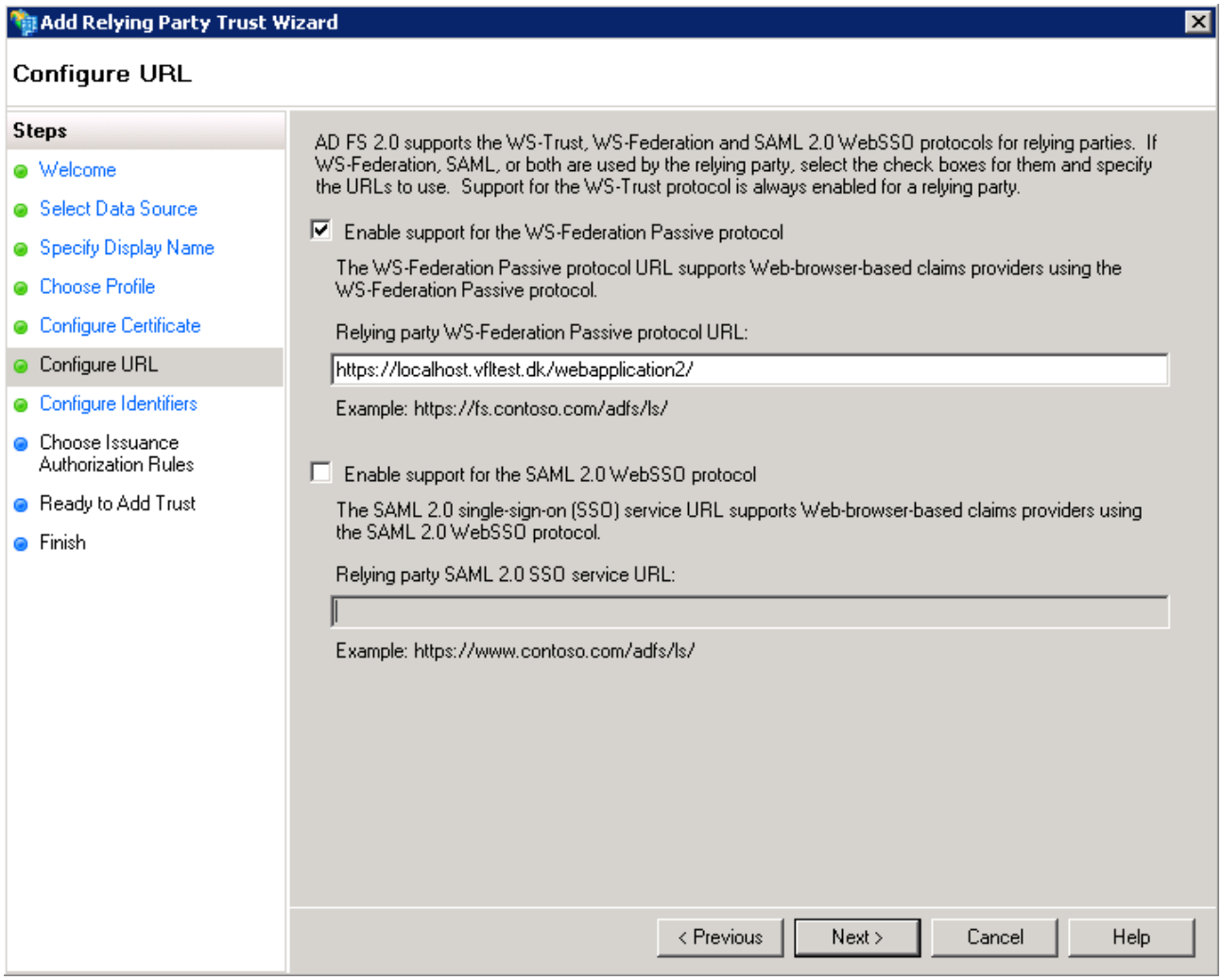


6. Select AD FS 2.0 profile



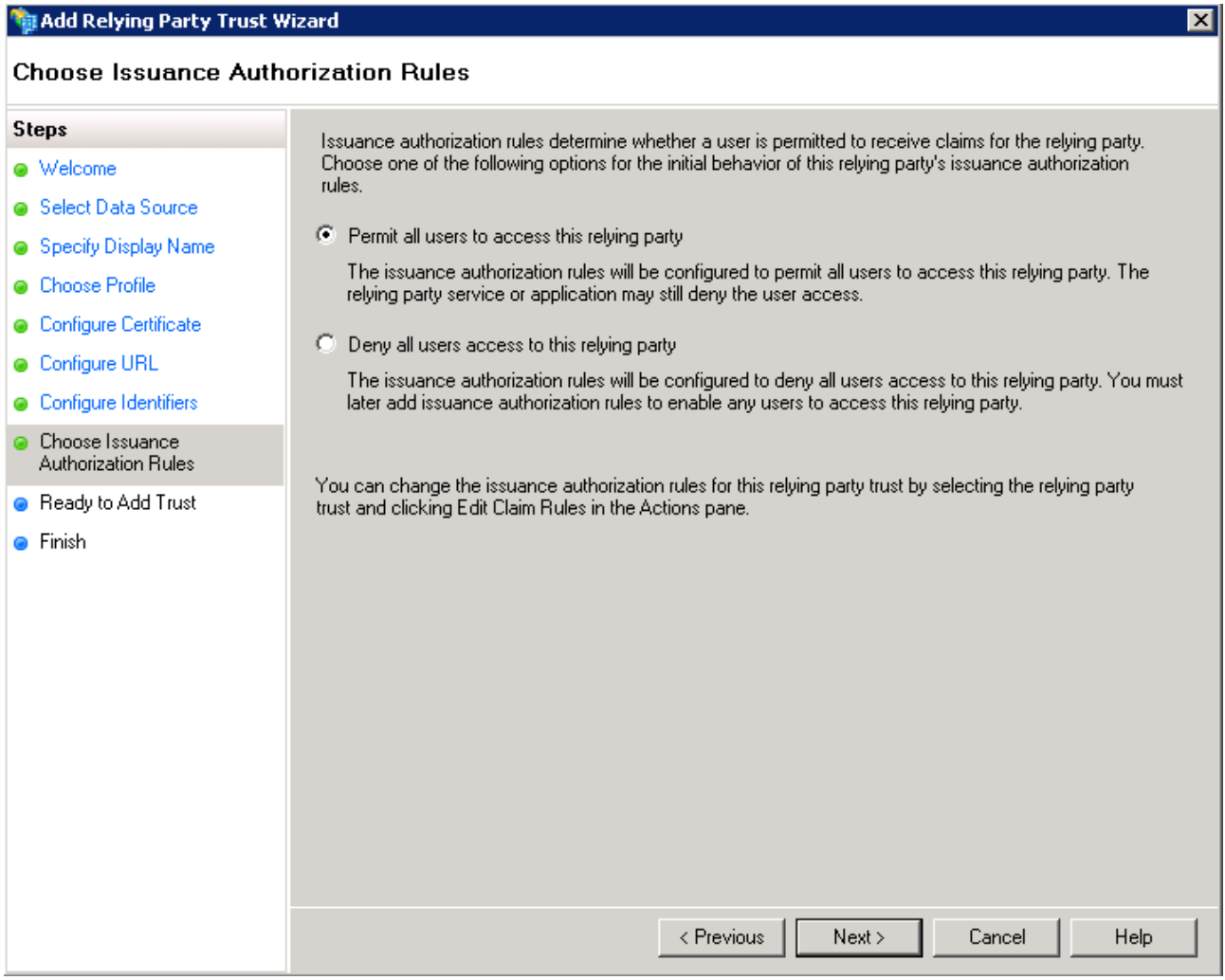

7. Leave “Configure Certificate”, security tokens will not be encrypted.

8. Select “Enable support for the WS-Federation Passive protocol” and enter the url (including a trailing slash) for the web application which is the relying party. Note that support for WS-Federation or SAML 2.0 should only be enabled for web applications, NOT for web services.




9. Leave “Configure Identifiers”, the identifier has been transferred from the previous step in the wizard.

10. Select “Permit all users to access the relying party”.

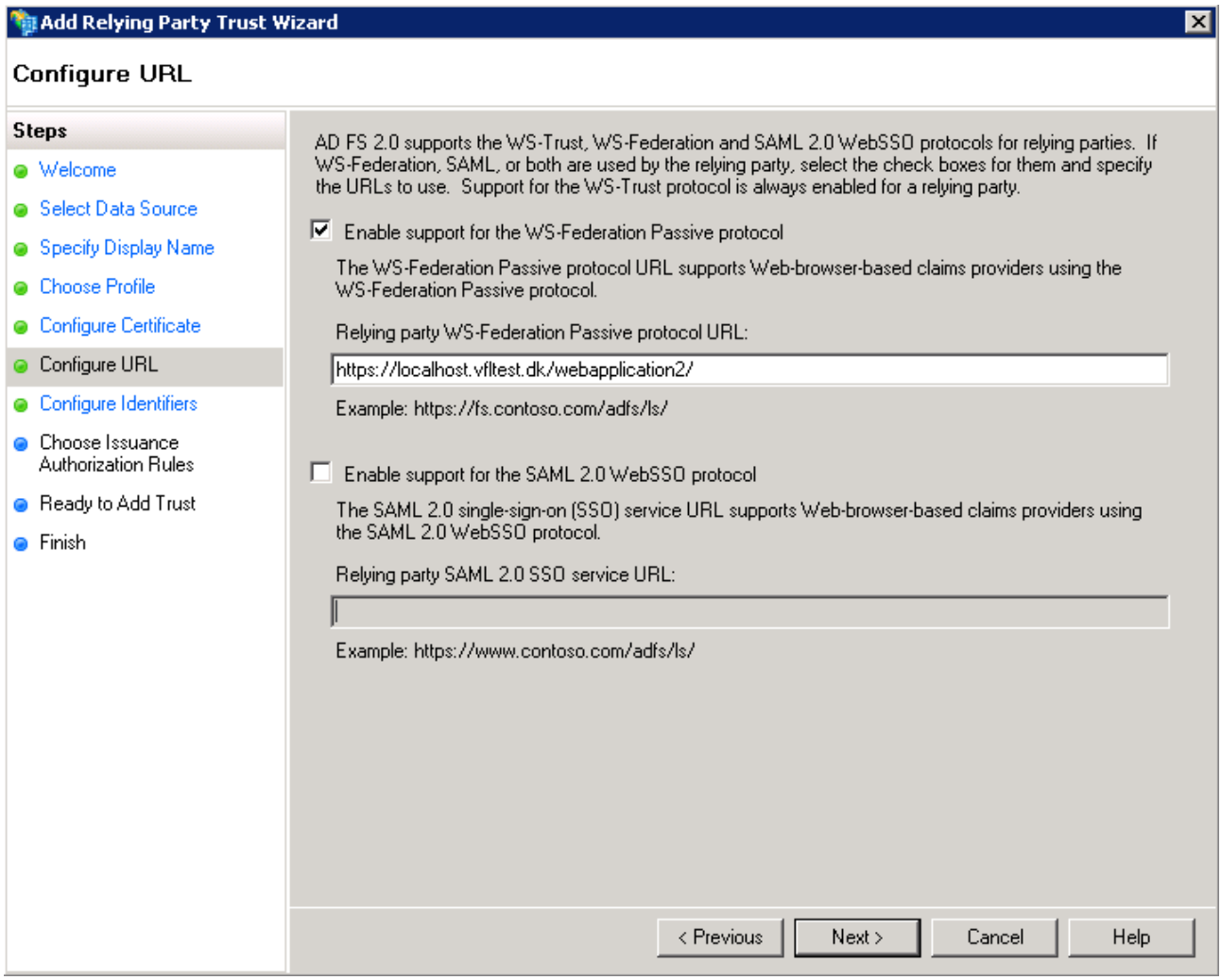



11. Wizard step “Ready to Add Trust” is informational so just continue.

12. In the last step click “Close” button. If “Open the Edit Claim Rules dialog for this relying party…..” is checked, the “Edit Claim Rules” dialog opens when finishing the wizard.

    A note about relying party web application hosted on Windows Server 2003. When the security token containing claims, that is issued by the Security Token Service (STS) upon a success full authentication of the user, is issued, it is digitally signed by the STS signing certificate. By default this signing is based on the SHA-256 hash algorithm. For the relying party to read the security token, the SHA-256 hashing algorithm must be installed on the server hosting the web application, for the federation process between the STS and the relying party to function.

    On a Windows Server 2003 the SHA-256 hashing algorithm is not installed by default. Source code is available that enable the SHA-256 hashing algorithm. In Visual Studio create a .NET 4.0 console project and add the file from TFS: $/DLBRLogin/DLBRLogin/trunk/Tools/SHA-256. How-to instructions are available in the source code.




13. Note that only 2 claims is issued by default when a user is authenticated by the IdP. These claims (claimtype) are http://schemas.microsoft.com/ws/2008/06/identity/claims/authenticationmethod and http://schemas.microsoft.com/ws/2008/06/identity/claims/authenticationinstant, describing how and when the user was authenticated. Examples of values are

http://schemas.microsoft.com/ws/2008/06/identity/authenticationmethod/password (if the user was authenticated by a password) and “2012-12-18T13:09:54.814Z” accordingly.

A Claims Rule Language example of how to issue other claim types, such as Active Directory groups membership and Windows logon user id, is shown in Appendix 2.

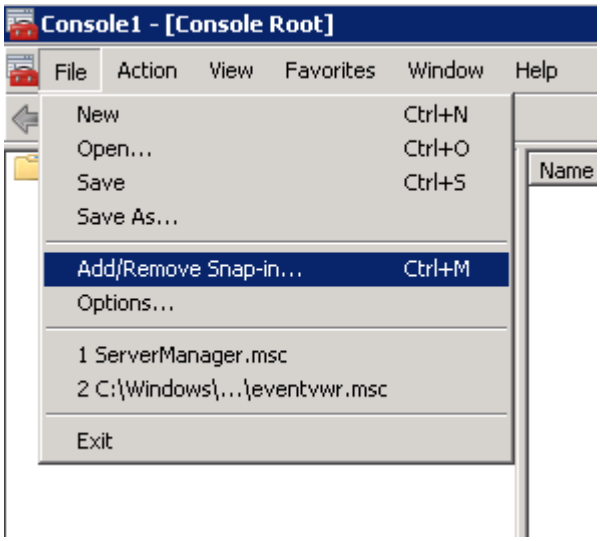
# 3 Appendix 1

## 3.1 Manually configuring IIS

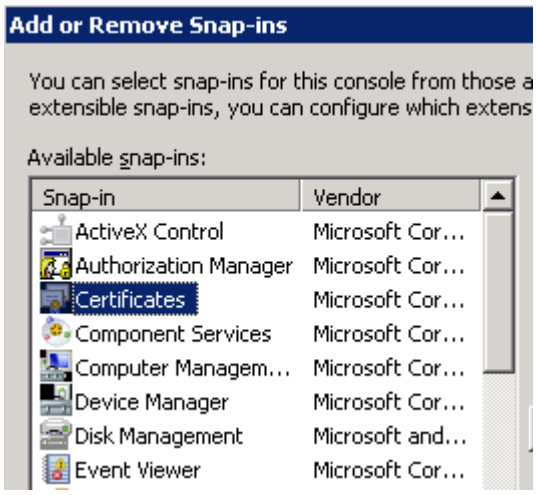
Instead of using the PowerShell script “CreateWebsite.ps1” to facilitate creation of IIS application pool and web site, a new web site can manually be configured using “Internet Information Services Manager”. Be aware that SSL certificate with subject “*.vfltest.dk” must be installed in store “LocalMachine\My” prior to creating the web site.

Installation of the SSL certificate can be done using “Microsoft Management Console”:

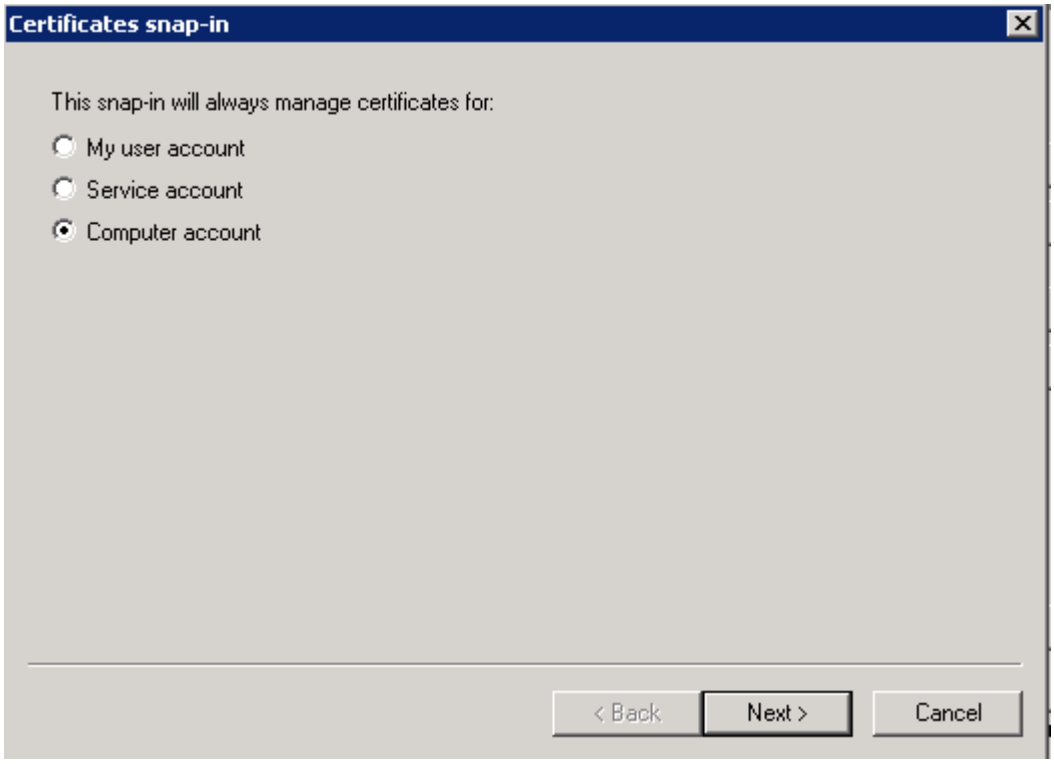
1. In the Windows start menu select “Run…”, type mmc and click OK.
2. In MMC, choose “File – Add/Remove Snap-in…”



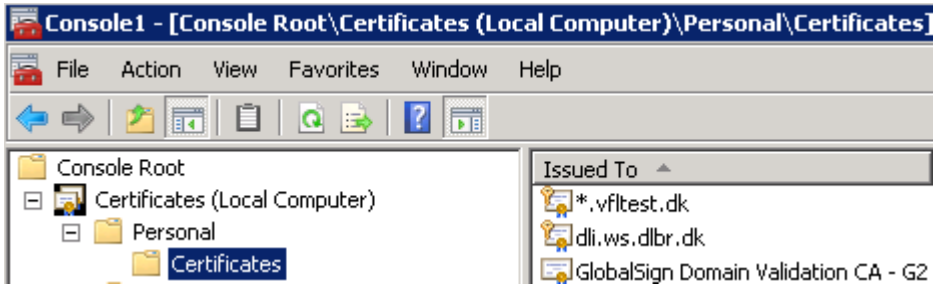

3. Double-click on “Certificates”



4. Select “Computer account”, click “Next”, “Finish” and “OK”

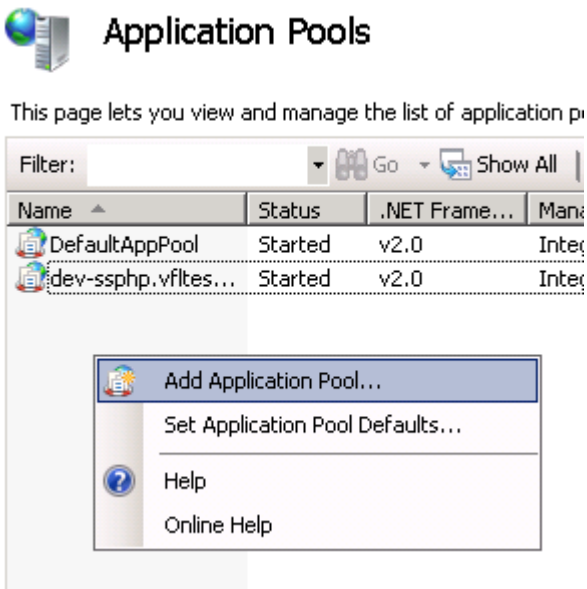



5. Expand “Certificates – Personal – Certificates”. Right-click the “Certificates” folder and choose “All tasks – Import…”. In the wizard, select file type “Personal Information Type (*.pfx)” and select the file to be imported. When prompted, type the password for the private key.

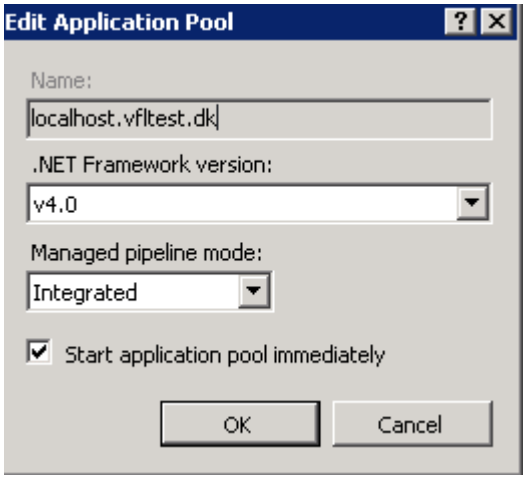


Configuring IIS application pool and web site can be done using “Internet Information Services (IIS) Manager”:

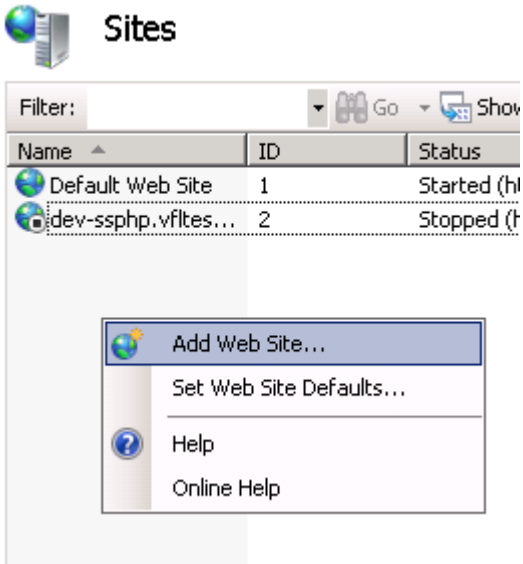
1. After installation of the SSL certificate, it’s time to create the web site, which is accomplished in “Internet Information Services (IIS) Manager (found in Start – Administrative Tools). Start with creating a new application pool by clicking on “Application Pools” in the left pane:



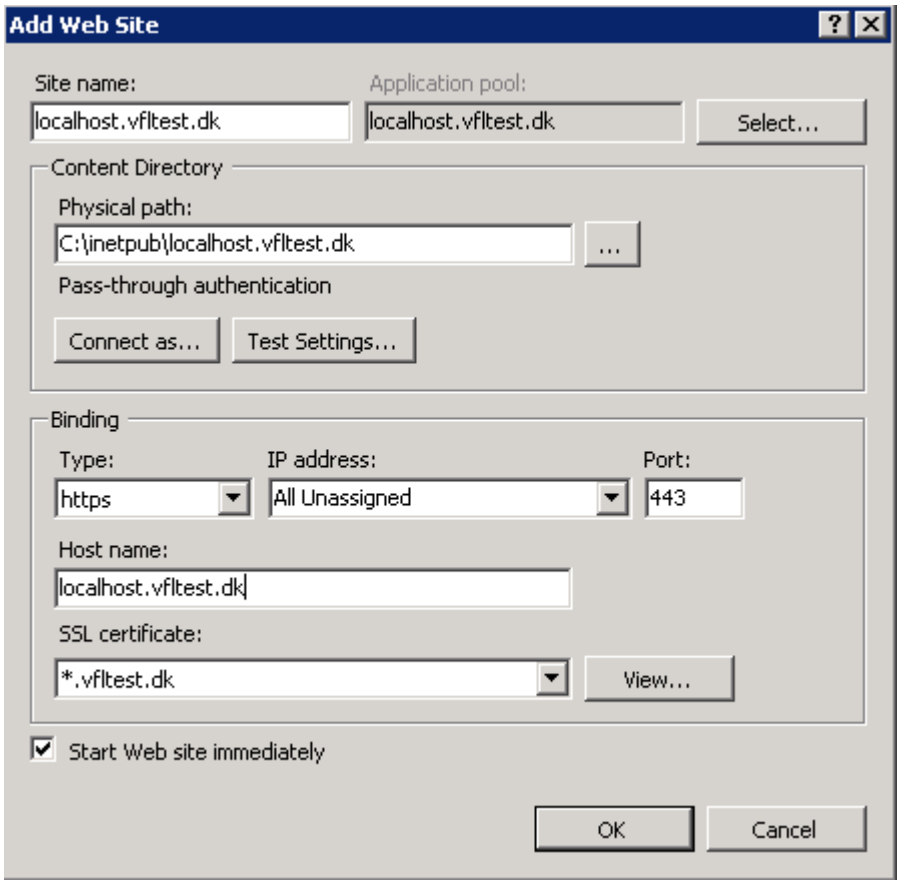

2. Configure it to use .Net Framework version “4.0” and set Managed pipeline mode to “Integrated”.



3. Create a new web site by clicking on “Sites” in the left pane




4. Configure it to use the previously created application pool. Choose “https” binding and select “*.vfltest.dk” as SSL certificate.



# 4 Appendix 2

## 4.1 How to issue Windows account name (logon user id) as a claim

1. Login to dev-idp.vfltest.dk or devtest-idp.vfltest.dk server, depending on whether the application must federate with DEV or DEVTEST identity provider.
2. Open “Start – Administrative Tools – AD FS 2.0 Management”.
3. Expand “Trust Relationships – Relying Party Trusts”, right-click the relevant relying party trust registration and choose “Edit Claim Rules…”.
4. Click on “Add Rule…” and select “Pass Through or Filter an Incoming Claim”.
5. Enter a Claim Rule Name. The value is optional.
6. In the “Incoming Claim type” drop-down box select “Windows account name”.
7. Select “Pass through all claim values” and click “Finish” button. Note that the format of Windows account name is “domain\userid”, e.g. PROD\LCMCM.

## 4.2 How to issue group membership as claims

1. Execute steps 1-3 in section 4.1.
2. Click on “Add Rule…” and select “Send LDAP attributes as Claims”.
3. Enter a Claim Rule Name. The value is optional.
4. In the “Attribute store” drop-down box select “Active Directory”.
5. In the “LDAP Attribute” drop-down box select “Token Groups – Unqualified Names”.
6. In the “Outgoing Claim Type” drop-down box select “Role”.

## 4.3 How to issue specific group membership as custom claims with DCF groups as an example

1. Execute steps 1-3 in section 4.1.
2. Click on “Add Rule…” and select “Send Claims Using a Custom Rule”.
3. Enter a Claim Rule Name. The value is optional.
4. Add the following as “Custom Rule”:
   c:[Type ==
   "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname", Issuer ==
   "AD AUTHORITY"] => add(store = "Active Directory", types = ("TokenGroups"), query =
   ";tokenGroups;{0}", param = c.Value);
   This rule, based on the “windowsaccountname” for the user, adds all groups to the type “TokenGroups”.

5. Add the following as a new “Custom Rule”:
   c:[Type == "TokenGroups", Value =~ "^(?i)GTALCDCF"] => issue(Type = "http://dcf.ws.dlbr.dk/ws/2008/04/authorization/claims/serviceauthorizations", Issuer = c.Issuer, OriginalIssuer = c.OriginalIssuer, Value = regexreplace(c.Value, "^GTALC", ""), ValueType = c.ValueType);
   This rule selects groups from “TokenGroups”, based on group name starting with “GTALCDCF”. Finally “GTALC” is stripped from group name making all groups starting with “DCF”.
   Note that the order of adding (executing) these rules is fixed as the rule in this section uses output from the rule in the previous section (4.3.4).
6. To make the role claims available in the web application, add the “roleClaimType” to web.config in section “samlSecurityTokenRequirement”
   ```
   <samlSecurityTokenRequirement….
   …..
   <roleClaimType value= http://dcf.ws.dlbr.dk/ws/2008/04/authorization/claims/serviceauthorizations />
   </samlSecurityTokenRequirement>
   ```
   Note that http://dcf.ws.dlbr.dk/ws/2008/04/authorization/claims/serviceauthorizations is a custom claim type (i.e. not one of the standard claim types issued by AD FS 2.0 out-of-the-box).
7. Add the following C# code to the application to iterate the claims:

```
        protected void Page_Load(object sender, EventArgs e)
        {
            Response.Write(DumpClaims());
        }

        private string DumpClaims()
        {
            var principal = (ClaimsPrincipal)Thread.CurrentPrincipal;
            var identity = (IClaimsIdentity)principal.Identity;

            var result = new StringBuilder();
            var level = "Identity";
            while (identity != null)
            {
                var claimStrings =
                    identity.Claims.Select(
                        claim =>

string.Format("<tr><td>{0}</td><td>{1}</td><td>{2}</td><td>{3}</td><td>{4}</td></tr>",
                                        claim.ClaimType, claim.Issuer,
claim.OriginalIssuer, claim.Subject, claim.Value));

                var formattedClaimsForIdentity = level + "<br /><table
border='1'><tr><td>Claimtype</td><td>Issuer</td><td>OriginalIssuer</td><td>Subject</td><td>Value</td></tr> " + string.Join("\n", claimStrings) + "</table>";
                result.AppendLine(formattedClaimsForIdentity);
                identity = identity.Actor;
                level = level + ".Actor";
            }
            return result.ToString();
        }
```

Note that the code is not required for the application to execute. It has only informational value and can be utilized in debugging scenarios.

## 4.4 How to issue Name id as a claim

1. Execute steps 1-3 in section 4.1.
2. Click on “Add Rule…” and select “Send LDAP attributes as Claims”.
3. Enter a Claim Rule Name. The value is optional.
4. In the “Attribute store” drop-down box select “Active Directory”.
5. In the “LDAP Attribute” drop-down box select “SAM-Account-Name”.
6. In the “Outgoing Claim Type” drop-down box select “Name ID”.
7. To make the name id available in the web application, add the “nameClaimType” to web.config in section “samlSecurityTokenRequirement”

```
<samlSecurityTokenRequirement….
…..
<nameClaimType value=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameid
entifier />
</samlSecurityTokenRequirement>
```

8. The name id can be read from the property “`Thread.CurrentPrincipal.Identity.Name`”.

# 5 Appendix 3

## 5.1 Changes in web.config after executing "FedUtil.exe"

The following is added to the web.config of the relying party web application:

```
  <authorization>
      <deny users="?" /> <-- deny access to unauthenticated users, i.e. the whole
app requires the user to authenticate prior to use
  </authorization>
```

(…)

```
<httpModules>
<add name="WSFederationAuthenticationModule"
type="Microsoft.IdentityModel.Web.WSFederationAuthenticationModule,
Microsoft.IdentityModel, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" />
<-- the WSFederationAuthenticationModule drives the WS-Federation Passive
Profile protocol handshake with the IdP, using browser redirects
<add name="SessionAuthenticationModule"
type="Microsoft.IdentityModel.Web.SessionAuthenticationModule,
Microsoft.IdentityModel, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" />
<-- the SessionAuthenticationModule serializes the claims principal derived
from security token received from the IdP, using a set of cookies by default
</httpModules>
```

(…)

```
<microsoft.identityModel>

<service>

<audienceUris>

<add value="https://localhost:6575/" /> <-- the RP identifier, also known as
"realm" and "entityid". By default, WIF will validate that incoming security
tokens are issued to this identifier

</audienceUris>

<federatedAuthentication>

<wsFederation passiveRedirectEnabled="true"
issuer="https://idp.dlbr.dk/adfs/ls/" realm="https://localhost:6575/"
requireHttps="false" /> <-- WIF intercepts any 401 access denied responses
(generated by the MVC Authorize attribute, the authorization section or
other means), and redirects to the IdP specified in "issuer", asking for a
security token for "realm".

<cookieHandler requireSsl="false" />

</federatedAuthentication>

<applicationService>
```

```
<claimTypeRequired>

<claimType type="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"
optional="true" />

<claimType
type="http://schemas.microsoft.com/ws/2008/06/identity/claims/role"
optional="true" />

</claimTypeRequired> <-- purely informational, and can be deleted. The
claims issued is configured per RP on the ADFS side
</applicationService>

<issuerNameRegistry
type="Microsoft.IdentityModel.Tokens.ConfigurationBasedIssuerNameRegistry,
Microsoft.IdentityModel, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35">

<trustedIssuers>

<add thumbprint="EDD4DBAE906DC0AE7DA1CEF554ACA69DB006F72D"
name="https://idp.dlbr.dk/adfs/services/trust" /> <-- Signing certificate
thumbprint of the IdP. Security tokens not signed with the certificate
private key corresponding to this thumbprint are rejected. In order to allow
WIF to validate the signature, security tokens issued by the IdP contains
the public key of the signing certificate used to sign them.

</trustedIssuers>

</issuerNameRegistry>

<certificateValidation certificateValidationMode="None" /> <-- Validation
mode for the IdP signing certificate. "None" means only the thumbprint is
checked, there is no requirement that the signing certificate is issued by a
trusted CA.

</service>

</microsoft.identityModel>
```