.Net SAML2 Service Provider Framework

Contents

1 Introduction 4
2 Release history 4
3 Prerequisites 7
3.1 General prerequisites 7
3.2 Prerequisites for the binary distribution 7
3.3 Prerequisites for the source distribution 7
3.4 The open source project log4net. 7
4 Distribution contents 8
5 Running the sample 9
5.1 Preparation 9
5.2 Running the Unit Test 11
5.3 Demonstrating federation 11
5.4 Demonstrating IdP Discovery using Common Domain Cookie 13
5.4.1 Preparation 13
5.4.2 Demonstration 15
5.5 Demonstration of IDP selection when multiple IDP's are available 15
5.5.1 Using the "default" attribute 15
5.5.2 Using the "idpSelectionUrl" attribute 16
5.5.3 Using the IDPSelectionEvent 16
6 Using the framework 18
6.1 Creating your own service provider web site 18
6.2 Working with real third party Identity Providers 19
6.3 Setting up more than one service provider 20
7 Components 21
7.1 SAML 2.0 framework 21
7.2 Demonstration identity provider 21
8 Certificate management 23
9 Configuration reference 25
9.1 <Federation> 25
9.1.1 <SigningCertificate> 25

9.1.2 <AllowedAudienceUris> 26
9.1.3 <Actions> 27
9.1.4 <SessionTimeout> 27
9.2 <Saml20Federation> 28
9.2.1 <ShowError> 28
9.2.2 <ServiceProvider> 28
9.2.3 <CommonDomain> 30
9.2.4 <RequestedAttributes> 30
9.2.5 <IDPEndPoints> 31
10 API reference 33
10.1 Saml20Identity 33
10.2 HttpHandlers 34
10.3 Attribute queries 35
10.4 Sample identity provider API 35
10.5 ForceAuthn and IsPassive 35
11 Common Domain Cookies 36
12 Virk.dk extension 37
13 Troubleshooting 38
13.1 Enabling debug logging 38
14 Audit & Logging 39
15 Session Providers 40
15.1 Implementing your own session provider 40
16 Setting up with ADFS v2. 40
17 References 43

1 Introduction

This document describes the .Net SAML2 Service Provider Framework. The document is naturally a part of the full package and as such it will be extended along with the framework.

Chapter 5 provides a quick start guide for creating a functional sample application of the framework in your own environment.
Each component in this distribution is described in detail in chapters 4 and 7.
The *API reference* chapter explains how the framework can be used to access user information issued by identity providers.

2 Release history

1.0 RC 1	24th of April 08	Extracted the Saml2 framework from Safewhere's codebase. Created a demo service provider. Moved classes into the dk.nita.saml2 namespace. Created this document.
1.0	30th of May 08	Added demonstration identity provider. Implemented SOAP binding. Implemented ARTIFACT binding. Implemented attribute queries. Implemented persistent pseudonyms. Extended documentation.
1.1	25th of July 08	Implemented review comments. Demo IdP configuration is now persistent.
1.3.0.2	11th of September 2008	.Net Framework requirement changed to 3.0 for binary dist.
1.5 RC 1	27th of November 2009	Incorporated 4 patches from Trifork: 1) ID in signaturevalidation. 2) Wrong AttributeConsumingService block 3) Error regarding empty AttributeConsumingService 5) Logging Implemented Configuration feature for the DemoIdP. Corrected æøå-error in sign-out. Implemented replay-check Extended the documentation regarding 1) setup of Common Domain Cookie. 2) logging 3) Configuration of DemoIdP.

1.5	21th of December 2009	Added description of private key access in 2008. Added paragraph on how to Connect DK.NITA with ADFSv2. Moved Project Reference for WebDemoVirk
1.6	27th of January 2010	Improved support for use of multiple IDPs: 1) Added default-attribute to add-element in IDPEndPoints element 2) Added idpSelectionUrl attribute to IDPEndPoints element 3) Added IDPSelectionUtil class and IDPSelectionEventHandler Added section 5.5 in the documentation and updated the Configuration reference. Corrected section 4 and 5.2. Fixed bug in IdentityProviderDemo: Common Domain Cookie demo described in section 5.4 of the documentation now works in other browsers than IE.
1.7	6th of december 2010	Support for POST-binding with regard to single-logout. The metadata generator will automatically add support for POST-binding SLO if new metadata is generated. The service providers will have to exchange metadata with the IdP's again to ensure that the IdP's will use the post-binding. Note that some IdP's might prefer HTTP-Redirect binding if both are enabled (currently both are enabled, you will have to remove the HTTP-Redirect entry from the metadata in that case)
1.7.3	11th of August 2011	Better validation of the reference URI in ds:signature elements
1.7.4	18th of November 2011	Hide detailed errormessages due to security vulnerability in XML Encryption
1.7.5	28th of March 2012	Fixed HttpRedirect endcoding issue to allow Danish character in certificate CN. Saving IDP NameID in session state. AllowCreate XmlIgnored on NameIDPolicy Test option in SP page for AssuranceLevel NameIdFormat added to metadata generation Added template for web.config to help with nemlogin integration.
1.7.6	May 2012	Implemented missing logging required according to “Logningspolitik for den fællesoffentlige log-in-løsning”
1.7.7	2013-09-11	Changed the serialization og the protocolSupportEnumeration element so that it works with the .Net4.5 runtime
1.7.8	2013-11-07	Removed the dependency on log4net by defining an IAuditLogger

		interface. The framework allows the implementor to define and use their own implementation for audit logging (see section 14 for more on audit logging). Removed the refreshing of idP's on every federation request. This is implemented using FileSystemWatcher on the folder that contains the idP metadata. Fixed bug about Single Logout failing when session had expired. The framework now sends a successfull LogoutResponse even if the session has expired. NOTE: Changed to the public API: • dk.nita.saml20.config.IDPEndpoints.Refresh() method is no longer public • dk.nita.saml20.config.IDPEndpoints.metadataLocation changed to a property dk.nita.saml20.config.IDPEndpoints.MetadataLocation
1.7.9	2014-02-25	Removed the default XmlResolver functionality from XmlDocument and XmlTextReader objects. The framework will now ignore external references when resolving XML documents, hereby mitigating potential XXE attacks. Fixed scalability issue on authentication requests to the Demo IDP: HttpContext.Current.Application["authenticationrequest"] has been changed to HttpContext.Current.Session["authenticationrequest"].
1.7.10	2014-06-25	The OIOSAML.net component now supports verification of XML documents signed with SHA256 signature. However, this functionality is only supported in .NET 4.0 or greater. The OIOSAML.net component now uses a custom session implementation instead of the ASP.NET session. Note that session timeout is now configured in the federation config section. For further information see the session provider section. The OIOSAML.net component now supports SOAP logout. Thus, users can be logged out through a back channel. SP must now, in the beginning of each user request, check using the Saml20Identity.IsInitialized() to verify if the user is still logged in or has been logged out using SOAP logout (if it has been enabled). The IAction interface has been expanded with an extra method called SoapLogoutAction(AbstractEndpointHandler, HttpContext, string) which

		is called when a SOAP logout request is received. It is still necessary at each HTTP request to check whether or not the user has been logged out by a SOAP logout request as this is not possible at the time the implementation of SoapLogoutAction is called.
1.7.11	2014-09-25	Enabled EnableViewStateMac, thereby mitigating potential exploit where an attacker may be able to upload and execute arbitrary code on the web server.

3 Prerequisites

3.1 General prerequisites

- IIS must be running asp.net 2.0 mode.

3.2 Prerequisites for the binary distribution

- .NET 3.5 SP1 runtime or later.
- ASP.NET 2.0 or later.
- NUnit 2.4 or later is required for running the unit tests.

3.3 Prerequisites for the source distribution

Same as the binary distribution plus

- Visual Studio 2010
- Nuget package manager (for visual studio 2010)

3.4 The open source project log4net.

The dk.nita.saml20.ext.audit.log4net project uses Log4net to provide audit and logging facilities. More on log4net can be found on http://logging.apache.org/log4net/index.html , a version of the log4net.dll is supplied with the dk.nita solution. This version, or a version found on the Apache site, must be copied to the bin directory of the service provider if log4net auditlogging is used.

4 Distribution contents

The framework is distributed as two packages

- A binary archive containing compiled versions of the framework and sample applications.
- A zip archive containing the source code for the above mentioned components.

Net SAML2 Service Provider Framework.rtf	This documentation.
src\	The framework source code. Distributed as a Visual Studio 2008 solution (dk.nita.saml20.sln) .
src\Sample_Identity_Provider.pfx, src\Sample_Service_Provider.pfx	Demo certificates that can be used to run the IdP/SP demo. Also required to run the unit tests.
src\dk.nita.saml20	The source code of the core framework, i.e. implementation of SAML Assertions and Protocols.
src\dk.nita.saml20.ext.brs	The source code of the extension that provides programmatic access to the virk.dk Authorisations attribute.
src\dk.nita.saml20.ext.audit.log4net	The source code for the log4net implementation of the Audit logger functionality
src\dk.nita.test.saml20	The framework's unit tests and test data. The unit tests must be executed using NUnit 2.4 or compatible unit testing software.
src\dk.nita.test.saml20.ext.brs	The virk.dk Authorisations attribute unit tests and test data. The unit tests must be executed using NUnit 2.4 or compatible unit testing software.
src\WebsiteDemo	An ASP.NET web site that acts as a simple service provider using the framework for federation.
src\IdentityProviderDemo	An ASP.NET web site that acts as a limited identity provider.
src\WebsiteDemoVirk	An ASP.NET web site that acts as a simple service provider for Virk using the framework for federation.
src\dk.nita.saml20\Documentation	Contains this document
bin\	The framework compiled.
bin\Sample_Identity_Provider.pfx, bin\Sample_Service_Provider.pfx	Demo certificates that can be used to run the IdP/SP demo. Also required to run the unit tests.
bin\IdentityProviderDemo\	A compiled version of the demonstration identity provider, which can be installed as a website in IIS.
bin\WebsiteDemo\	A compiled version of the demonstration service provider, which can be installed as a website in IIS.

bin\WebsiteDemoVirk\	A compiled version of the demonstration service provider for Virk, which can be installed as a website in IIS.

5 Running the sample

This chapter describes the steps required to quickly get the included sample up and running to try out the features of the framework.

5.1 Preparation

1. If working with the source distribution, open and compile in Visual Studio. The `Saml2Test` project can be disabled, if you are not interested in executing the included automated tests.

2. Install the certificates (contained in the distribution) to support both the service provider and identity provider. The certificates are located in `dk.nita.saml20\bin` (or `…\src`).
   The password for the certificates is "`test1234`".

   a. Start the Microsoft Management Console (MMC) and add the `Certificates` snap-in. Choose "Computer Account" and then "Local Computer".

   b. Import the certificate in the file `Sample_Identity_Provider.pfx` into "`Local Computer\Personal\Certificates`".

   c. Import the certificates in the file `Sample_Service_Provider.pfx` into "`Local Computer\Personal\Certificates`".

   d. Drag and drop the root certificate "`DemoProviderRoot`" from "`Local Computer\Personal\Certificates`" into "`Local Computer\Trusted Root Certification Authorities\Certificates`"

3. Open up IIS Manager and add two new *Virtual Directories* (*Applications* if on IIS 7):

   a. One named `demo` that points to the service provider, the `WebsiteDemo` directory. The resulting URL should be `http://<machinename>/demo`
   b. One named `IdP` that points to the identity provider, the `IdentityProviderDemo` directory. The resulting URL should be `http://<machinename>/IdP`

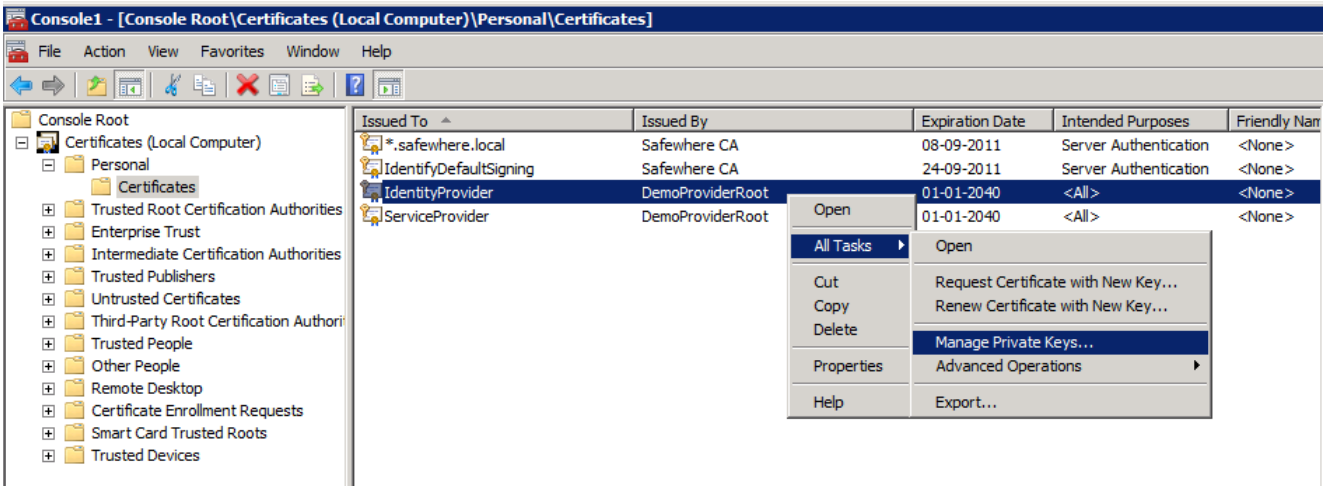
4. Set the correct access control entries, ACLs, on the certificates just installed in step 2:

   Note: If running on IIS 6.0 or above you will most likely have applied the default setup of the websites,

which means they will be run by the built-in account, NETWORK SERVICE. This account must be granted read access to the certificates – this can either be done directly from the MMC with the certificate snap-in, or as described below (Will not work on Windows 7 or Windows Server 2008 R2):

a. From a Command Prompt run FindPrivateKey.exe located in dk.nita.saml20\bin with the options LocalMachine and My, i.e. “FindPrivateKey.exe LocalMachine My”.
b. Select one of the priviously imported certificates and click OK.
c. On the Security tab add “NETWORK SERVICE“ to “Group and user names”.
d. Repeat the above steps with the second certificate.

On Windows server 2008 and Windows 7, the private key properties can be accessed directly from the certificate snap-in in the MMC, as shown in the screenshot below:



5. Modify the Web.config of the identity provider, the `IdentityProviderDemo` application. One app setting must be changed:

    a. The <`IDPDataDirectory`> element holds the path to the directory containing metadata of service providers. Make sure to specify a directory that is accessible by the web server (typically the Network Service account).

6. Modify the `Web.config` of the service provider, the `WebsiteDemo` application. Two elements must be changed in the `<SAML20Federation>` section:

    a. The `server` attribute must contain the base URL without sub-directories.(This setting will be used as the base for all URLs that are exposed to federation partners through metadata.) Change this to `http://<machinename>/`
    b. The <IDPEndPoints> element holds the path to the directory containing metadata of identity providers. Make sure to specify a directory that is accessible by the web server

(typically the Network Service account).

c. Change the entity id to your demo IdP's actual entity id:

```
<IDPEndPoints metadata="C:\metadata\">
      <add id="DemoIdPEntityId">
            <CertificateValidation>
                  <add type="…"/>
            </CertificateValidation>
      </add>
</IDPEndPoints>
```

You need to change the value "DemoIdPEntityId" to the actual entity id of your demo IdP. You can see the correct entity id in the IdP's control panel (Step 8 shows you how to get to the control panel).

The next step is to exchange metadata between the identity and service providers.

7. Go to the service provider website (http://<machinename>/demo/default.aspx), and download the metadata to a temporary location of your choice
8. Go to the identity provider control panel (http://<machinename>/IdP/Control.aspx) and configure the running IdP
    a. Chose the certificate to identify this identity provider. Chose the IdentityProvider certificate (CN=IdentityProvider, O=NITA, C=DK) in *LocalMachine* and *My* store.
    b. Enter the URL for this identity provider. Avoid localhost or 127.0.0.1
    c. Download the metadata and place it into the directory specified in step 6.b
    d. Add the sample service provider by uploading the metadata just downloaded to the temporary location used in step 7

9Copy log4net.dll (found in the distribution of OIOSAML.net 1.5) to the bin directory of the demo service provider.

The Unit test and sample are now ready to run.

5.2 Running the Unit Test

The framework contains a number of automated tests that can be executed with NUnit 2.4 (or compatible software), and can be executed by loading the dk.nita.test.saml20.dll and/or dk.nita.test.saml20.ext.brs.dll into NUnit and executing the tests. Note that the tests are only included in the source distribution of OIOSAML.NET.

5.3 Demonstrating federation

1. Access the sample service provider website (http://<machinename>/demo/default.aspx)

and click the link to “Go to My page”

2. If this is the first time in this session you access an area requiring login, you will need to authenticate with an appropriate identity provider

3. In the standard sample configuration you will be sent directly to authentication with the identity provider (see section 7.2 for a set of valid usernames and passwords). If more than one known identity provider is available, you will be asked to choose which identity provider to authenticate with.

4. After authentication you will be redirected to “My page” of the service provider, which shows a list of the attributes that were issued about the user by the identity provider

   Note: This authentication is communicated back to the service provider by a freshly issued *SAML Assertion* which carries information about the user, as well as being signed and therefore impossible to tamper with.

5. Clicking “Logout” will terminate the user’s session and initiate the logout conversation between the service provider(s) and the identity provider.

To illustrate the dependence on correctly configured and always valid and not revoked certificates try to change the certificate of the service provider or the identity provider after exchanging metadata. This will cause messages between the federation partners to be rejected due to invalid signatures.

5.4 Demonstrating IdP Discovery using Common Domain Cookie

This section illustrates how to perform IdP Discovery as describe in the SAML Identity Provider Discovery Profile by enabling the Common Domain Cookie approach in the Federation. Note that IdP Discovery is only relevant when multiple IdPs are present in the same federation.

The setup explained in this section corresponds to the configuration illustrated in the first figure in chapter 11, which consists of two Service Providers and two Identity Providers. It is assumed that the certificates are already installed as explained in section 5.1.

5.4.1 Preparation

1. Create the directory "`C:\inetpub\cdctest`".
2. Copy the "`dk.nita.saml20\WebsiteDemo`" and "`dk.nita.saml20\IdentityProviderDemo`" directories to the "`C:\inetpub\cdctest`" directory.
3. Rename the "`WebsiteDemo`" folder to "`sp1`" and the "`IdentityProviderDemo`" folder to "`idp`".
4. Make a copy of the "`sp1`" folder and name the new copy "`sp2`".
5. Open the IIS Manager and add three new *Sites*:
    a. One named `demosp1` that points to Service Provider 1, the `sp1` directory.
    b. One named `demosp2` that points to Service Provider 2, the `sp2` directory.
    c. And finally one named `demoidp` pointing to the Identity Provider, the `idp` directory.
6. In the IIS Manager edit the bindings for the three sites:
    a. For Service Provider 1:
    Add the entries: demosp1.commondomain.local
    demosp1.local
    b. For Service Provider 2:
    Add the entries: demosp2.commondomain.local
    demosp2.local
    c. Identity Provider:
    Add the entry: commondomain.local
    (Note that the Identity Provider must be located on the domain corresponding to the common domain. This is not a requirement for Identity Providers in general, but is the only way to make it work with the Demo IdP.
7. Add the following entries to the hosts file (C:\Windows\System32\drivers\etc\hosts):

```
127.0.0.1  commondomain.local
127.0.0.1  demosp1.commondomain.local
127.0.0.1  demosp2.commondomain.local
127.0.0.1  demosp1.local
127.0.0.1  demosp2.local
```

8. Modify the Web.config of the identity provider, the `idp` application. One app setting must be changed:

a. The <IDPDataDirectory> element holds the path to the directory containing metadata of service providers. Make sure to specify a directory that is accessible by the web server (typically the Network Service account).

2. Add a new file, cdcreader.ashx, to each of the two Service Providers, sp1 and sp2
   a. Put the following into each cdcreader.ashx file:
   <%@ WebHandler Class="dk.nita.saml20.protocol.Saml20CDCReader" %>
3. Modify the Web.config files of the two Service Providers, the sp1 and sp2 applications:
   a. Two elements in the <SAML20Federation> section must be changed:
      i. The id attribute of the <ServiceProvider> element must be changed to something unique for each service provider, eg. http://sp1.example.net and http://sp2.example.net
      ii. The server attribute of the <ServiceProvider> element should be changed to http://demospX.local – where X is either 1 or 2 depending on the Service Provider.
      iii. The <IDPEndPoints> element holds the path to the directory containing metadata of Identity Providers. Make sure to specify a directory that is accessible by the web server (typically the Network Service account).
      iv. In order to demonstrate the Common Domain Cookie, add a dummy Identity Provider like so:

```
<IDPEndPoints metadata="C:\inetpub\cdctest\sp1\metadata\">
      <add id="DemoIdPEntityId">
        <CertificateValidation>
          <add
type="dk.nita.saml20.Specification.SelfIssuedCertificateSpecification,
dk.nita.saml20"/>
        </CertificateValidation>
      </add>
         <add id="dummy"></add>
    </IDPEndPoints>
```

      You need to change the "DemoIdPEntityId" to the actual entity id of your demo IdP.
      You can see the correct entity id in the IdP's control panel.
   b. Change the six places where it says "/demo/…" to ust "/"
   c. Reading of the Common Domain Cookie can be enabled by adding the <CommonDomain> element to the <SAML20Federation> element:

```
<CommonDomain enabled="true"
localReaderEndpoint="http://demospX.commondomain.local/cdcr
eader.ashx" />
```

   (where **X** is either 1 or 2)

   d. Change the <Federation> section to included to <Audience> tags containing the id's of the two service providers (the id attributes from the <ServiceProvider> elements mentioned above.)
11. The next step is to exchange metadata between the Identity and Service Providers:
   a. Go to the Service Provider 1's website (http://demosp1.local/default.aspx), and download the metadata to a temporary location of your choice.
   b. Repeat the previous step for Service Provider 2.

c. Go to the Identity Provider's control panel (http://demoidp/Control.aspx) and configure the running IdP:
    i. Chose the certificate to identify this identity provider. Chose the IdentityProvider certificate (CN=IdentityProvider, O=NITA, C=DK) in *LocalMachine* and *My* store.

    ii. Enter the URL for this identity provider. Normally the default will be OK, but avoid localhost or 127.0.0.1

    iii. Download the metadata and place it into the directory (specified in 9.a.iii – "`C:\inetpub\cdctest\sp1\metadata\`".

d. Add the two Service Providers by uploading their metadata, which was download to the temporary locations used in steps 10.a and 10.b.

5.4.2 Demonstration

1. Access Service Provider 1's website (http://demosp1.local/Default.aspx) and click "Go to My page"
2. Select which Identity Provider to authenticate the user from the list of Identity Providers. Notice the dummy Identity Provider which cannot be selected, since it is not needed in order to demonstrate the Common Domain Cookie proof of concept.
3. Login as the user "Lene1" and password "Test1234".
4. Access Service Provider 2's website (http://demosp2.local/Default.aspx) and click "Go to My page"
5. Notice that the user is not asked to login again, since he has already been authenticated.
6. Clicking "Logout" will terminate the user's session.

5.5 Demonstration of IDP selection when multiple IDP's are available

In case more than one IDP has been configured in the IDPEndPoints collection, and no Common Domain Cookie has yet been set, it is possible to select authorization IDP in 3 ways:

1. By setting a default IDP in the SAML20Federation configuration section (see section 5.5.1)
2. In the SAML20Federation configuration section, by specifying a URL on the website that lets the user select IDP (see section 5.5.2)
3. By implementing a .NET event handler that returns the IDPEndpoint to use (see section 5.5.3)

The list above is ordered, that is, in case a default is configured (method no. 1), use of the methods no. 2 and 3 are ignored by OIOSAML.NET. In the samples below it is therefore important to remove changes done to the configuration file in the previous example.

5.5.1 Using the "default" attribute

1. In the example above, modify the Web.config by adding a `default="true"` attribute to the

<add…> element for the Idp: `<add id="http://commondomain.local/" default="true">`

2. Restart the browser to make sure the common domain cookie is removed
3. Access Service Provider 1's website (http://demosp1.local/Default.aspx) and click "Go to My page"
4. Notice that the user is not prompted, which IDP to use, since he is immediately redirected to the default IDP.

5.5.2 Using the "idpSelectionUrl" attribute

With this method, the developer can specify a page that the user should be redirected to, in case multiple IDP's are available:

1. In case you modified the Web.config as explained in 5.5.1, remove the `default="true"`
2. In the <IDPEndPoints> element, add the idpSelectionUrl and value, so the element looks like this: `<IDPEndPoints metadata="C:\inetpub\cdctest\sp1\metadata\" idpSelectionUrl="/IDPSelectionDemo.aspx">`
3. Restart the browser to make sure the common domain cookie is removed
4. Access Service Provider 1's website (http://demosp1.local/Default.aspx) and click "Go to My page"
5. Notice that the browser opens a web-page from the WebsiteDemo project (IDPSelectionDemo.aspx), and not the default IDP selection list from the OIOSAML.NET framework. IDPSelectionDemo.aspx is just a sample of, how a page could look like. The styling and contents is completely defined by the web-developer. The only thing to remember is, that the link, that the user clicks, points to the URL returned by the `GetIDPLoginUrl()` method on the `dk.nita.saml20.config.IDPEndPoint` class. See IDPSelectionDemo.aspx.cs for an example.

5.5.3 Using the IDPSelectionEvent

The third method for selecting an IDP end point is done programmatically by using a .NET event:

1. In case you modified the Web.config as explained in 5.5.1, remove the attribute `default="true"`
2. In case you modified the Web.config as explained in 5.5.2, remove the attribute `idpSelectionUrl="/IDPSelectionDemo.aspx"`
3. See an example of a IDPSelectionEvent handler in Global.asax.cs, named `_idpSelectionEventHandler`. To use this handler add this line to `Application_Start` (in Global.asax.cs):

   ```
   IDPSelectionUtil.IDPSelectionEvent += _idpSelectionEventHandler;
   ```

4. Since we are professional developers, always remembering to clean up, dispose etc. in our code (!) , add this line to Application_End (in Global.asax.cs):

   ```
   IDPSelectionUtil.IDPSelectionEvent -= _idpSelectionEventHandler;
   ```

5. Restart the browser to make sure the common domain cookie is removed
6. Access Service Provider 1's website (http://demosp1.local/Default.aspx) and click "Go to My page"

7. Notice that the user is not prompted, which IDP to use, since he is immediately redirected to the IDP chosen by the event handler.

6 Using the framework

To develop a new web site using the framework you may follow the procedure below for getting up and running. This procedure serves only as a guide as your environment may differ from what is described below.

Also described are the steps to work with real identity providers and how to set up more than one service provider.

6.1 Creating your own service provider web site

1. Create a new web project in Visual Studio.
2. In the solution, include the Saml2 project containing the framework (may be left out if you keep a compiled version of the framework for reference)
3. Add a reference to the Saml2 project (or, if working with a compiled version, add a reference to the dk.nita.saml20 assembly, dk.nita.saml20.dll)
4. Create three ASP.NET handlers (ashx files) each with the content as illustrated below:
    - login.ashx (Your choice of name, but note it down as you will reference it later)

      ```
      <%@ WebHandler Class="dk.nita.saml20.protocol.Saml20SignonHandler" %>
      ```

    - `logout.ashx` (Your choice of name, but note it down as you will reference it later)

      ```
      <%@ WebHandler Class="dk.nita.saml20.protocol.Saml20LogoutHandler" %>
      ```

    - `metadata.ashx` (Your choice of name, but note it down as you will reference it later)

      ```
      <%@ WebHandler Class="dk.nita.saml20.protocol.Saml20MetadataHandler"
      %>
      ```

5. Modify Web.config (add a web.config file if not already there):
    - In the configSections section, add two additional sections:

      ```
      <section name="Federation"
               type="dk.nita.saml20.config.ConfigurationReader,
               dk.nita.saml20" />
      <section name="SAML20Federation"
               type="dk.nita.saml20.config.ConfigurationReader,
               dk.nita.saml20"/>
      ```

    - In the system.web section, add/change the authentication element:

      ```
      <authentication mode="Forms">
      ```

```
        <forms cookieless="UseCookies"
           loginUrl="<YOUR LOGIN HANDLER AS NAMED ABOVE>"
           name="<YOUR COOKIE NAME>"  />
    </authentication>
```

- To describe which parts of the application are protected by forms authentication, you must use the ASP.NET location tag (see the sample for an example of this). The reason for this is that you need unauthenticated access to not only the login.ashx handler, but also the metadata.ashx handler.

- For fastest results, copy from the sample service provider, WebSiteDemo, the `Federation` section and modify to your application. Refer to the reference in section 10.1 for details on each element.
  - *Note that to configure this element for a real world application, you will need your own certificate configured correctly for access by the web server.*
  - *Note that the* AllowedAudiencesUris *element must be set to match the* Id of the service provider in the `SAML20Federation` as described below

- For fastest results, copy from the sample service provider, WebSiteDemo, the `SAML20Federation` section and modify to your application. Refer to the reference in section 10.2 for details on each element.
  - *Note that the* Id attribute must be unique and match the audience requirements of the `Federation` section (see previous point)
  - *Note that for the signon service endpoint you do not have to set a redirect URL. If you want the user to be able to directly access any part of your application, and authentication on the way if necessary, remove this redirectUrl attribute.*
  - *Note the IdPEndPoints element which must point to a valid directory*

6. Download the metadata of your application from the endpoint specified in the SAML20Federation element as described above (e.g. https://hostname/MyApp/metadata.ashx). These metadata must be given to the identity provider(s) of your choice.

7. Install the metadata of your identity provider(s) in the directory specified in the `Saml20Federation` section

8. That's it. Now try it out with the sample identity provider and finally integrate with a third party identity provider

6.2 Working with real third party Identity Providers

9. Download the service provider metadata as exported by the metadata endpoint supported by the framework (see section 10.2.1)
10. Send the metadata to the relevant identity provider(s)
11. Get the corresponding metadata from the identity provider(s). These metadata must be stored in the designated directory as specified by the IDPEndpoints configuration element (see section 10.2.3).

6.3 Setting up more than one service provider

1. The simplest way is to just copy the sample service provider, WebsiteDemo, or to go ahead and build a custom web site and subsequently enable federation through this framework.

2. No matter how this is done, setting up more than one service provider, you must be sure to set authentication cookie names to different values across machine and domain:

   a. **Important note regarding cookies**

   b. It is important to specify different names for the browser cookies used by each website if more than one of the service provider is installed on the same host.

   c. Cookie names are configured in `web.config` using the `name` attribute of the `<forms>` element, and the `cookieName` attribute of the `<sessionState>` element. Both configuration options are elaborated in the ASP.NET reference documentation.

3. For each service provider added to the federation they must register their metadata with the identity provider

Once more than one service provider is available, you should notice that the user will not be re-authenticated when logging in at the second service provider.

7 Components

This chapter provides a description of each component in the distribution.

7.1 SAML 2.0 framework

The SAML 2.0 framework is the main component of this distribution. Chapter 9 is a reference of the framework's configuration parameters, while chapter 11 is an API reference detailing the classes that are required for an application to use the framework.

7.2 Demonstration identity provider

The demonstration identity provider is included so that it is not required for a development environment to have a fully configured identity provider to test out basic federation.

The demonstration identity provider has the following features

- Accepts authentication requests using HTTP-REDIRECT.
- Issues authentication responses using HTTP-POST.
- Logout requests and responses using HTTP-REDIRECT.
- Issues and consumes metadata.
- Handles authentication and logout of user sessions across several service providers.
- Validates signatures.

The demonstration identity provider uses the SAML 2.0 Framework and is primarily meant as a companion to the demonstration service provider in order to deliver an easily created environment in which to get acquainted with the framework. It should not be used as a permanent substitute for at real identity provider in a development environment.

The users, passwords and issued attributes can be set-up in the web.config of the DemoIdP as shown in the example below:

```
<configSections>
    <section  name="demoIdp"
type="IdentityProviderDemo.Logic.DemoIdPConfigurationSection"/>
  </configSections>
  <demoIdp>
    <users>
      <add userName="Lene" password ="Test1234"
ppid="PPID-FDFFE8F1-D92C-4838-B46D-B3DD558E700E">
        <attributes>
          <add name="urn:FirstName" value="Lene"/>
          <add name="urn:LastName" value="Hansen"/>
          <add name="urn:Age" value="32"/>
          <add name="urn:oid:0.9.2342.19200300.100.1.3"  value="lene@company.dk"/>
          <add name="urn:dk:company:attribute:Role"   value="Medarbejder"/>
          <add name="urn:dk:company:attribute:Role"   value="Udvikler"/>
        </attributes>
      </add>
      <add userName="Åge" password ="Test1234"
ppid="PPID-7CDE9A20-8A40-429a-A390-FFAB7DF84DF3">
        <attributes>
          <add name="urn:FirstName" value="Åge"/>
          <add name="urn:LastName" value="Børgesen"/>
          <add name="urn:Age" value="23"/>
          <add name="urn:oid:0.9.2342.19200300.100.1.3"  value="Åge@company.dk"/>
          <add name="urn:dk:company:attribute:Role"   value="Øverste Chef"/>
        </attributes>
      </add>
    </users>
  </demoIdp>
```

8 Certificate management

The certificates supplied and installed with the sample included in this framework, work nicely when being used by the person who installed the certificates, which is the case when e.g. running the NUnit tests supplied.

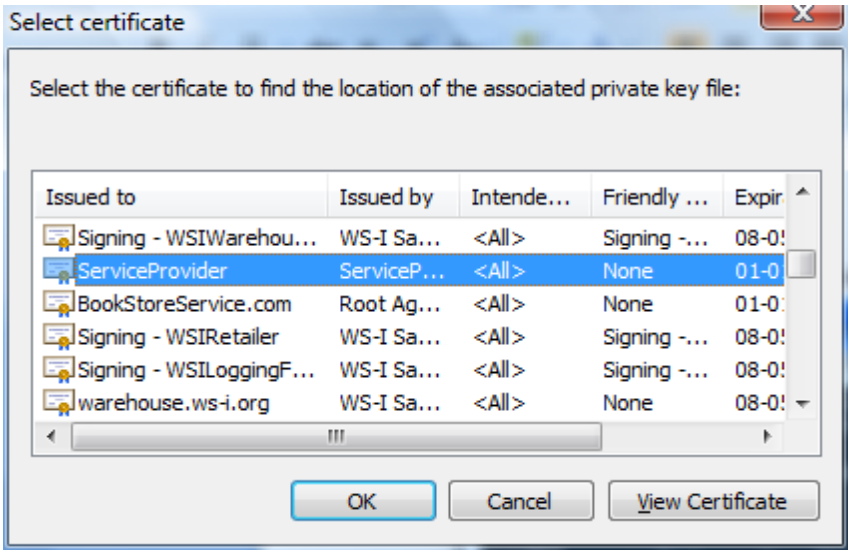
When using the certificates of the sample, the account running the web site (NETWORK SERVICE by default) must be granted read access to the private keys. This will ensure that communication between the service provider and the identity provider may be signed correctly.

The following procedure uses a small utility supplied in the distribution. With the tool and this procedure you may grant access to the private keys of the certificates as required by the web site setup.

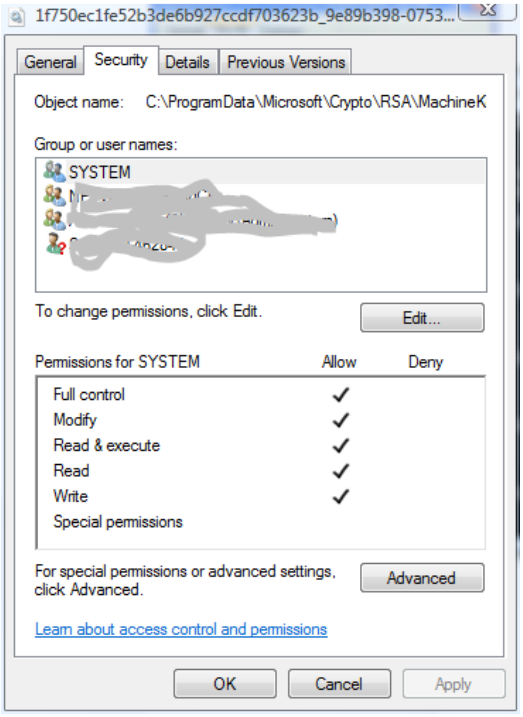
1. Open a command prompt in the directory where the framework is installed (..\bin or ..\src).
2. Enter the following command:

   ```
   FindPrivateKey.exe LocalMachine My
   ```

   to open the following dialog



3. Expand the 'Issued To' column, select the certificate you are going to use for the service provider (the supplied certificate read "ServiceProvider") and click the OK button.
4. Next the dialog below appears. Go to the Security tab:



5. If the relevant account (most often NETWORK SERVICE) is not already present click the “Add…” button, and grant the account read access.

6. Before closing the above dialog, verify that the account is now present on the list.

9 Configuration reference

All XML elements of this reference belong in the namespace '`urn:dk.nita.saml20.configuration`', unless otherwise noted.

9.1 <Federation>

This element contains settings that are general to federation: i.e. the certificate and identifier of the machine in the federation.

The elements attributes are listed below:

`auditLoggingType`	The fully qualified name of the class that implements the IAuditLogger interface. This interface allows implementors to define and use their own audit logging functionality. If the attribute is not provided the framework will use System.Diagnostics tracing as default for the audit logging.
`sessionType`	The fully qualified name of the class that implements the ISessions interface. This interface allows service providers to define and use their own session implementation. If the attribute is not provided the framework will use the default implementation dk.nita.saml20.session.inproc.InProcSessions as default for handling the session state.

9.1.1 <SigningCertificate>

This element specifies the service provider's certificate, which is used to verify the identity of the service provider to its service partners. The element's attributes are listed in the following

`x509FindType`	Specifies which certificate attribute that will be used to identify the service provider's certificate. The documentation of the .net framework enumeration `X509FindType` lists the possible values for this attribute. A common way to locate a certificate is to search for its subject's distinguished name or its thumbprint. The service provider will use the first certificate that matches the specified search criteria.
`findValue`	The value of the attribute that is used to identify the certificate, e.g. its subject or thumbprint.
`storeLocation`	The location of the certificate store to use. The documentation of the .net framework enumeration `StoreLocation` lists the possible values for this attribute.
`storeName`	Specifies which certificate store the certificate can be found in. The documentation of the .net framework enumeration `StoreName` lists the possible values for this

	attribute.
`validOnly`	Search only the valid certificates. An invalid or expired certificate may cause federation partners to reject communication, so enabling this option may give an early warning that a certificate should be replaced. Value should be either `true` or `false`.

Most of the above values for a given certificate can be found using the 'Certificates' management application included with windows.

9.1.2 <AllowedAudienceUris>

Assertions are issued to specific audiences. This ensures that an assertion cannot be used at a different service provider than the one that was intended by the identity provider. This configuration setting is a list of audiences that are allowed for assertions sent to the service provider. The list must at least contain the identifier of the service provider (See 10.2.1).

9.1.3 <Actions>

The <Actions> element defines a list of actions that take place on the service provider side when an assertion is received from the IdP. The element is optional, and when not present , a default set of actions are performed. Actions are performed in the sequence in which they are added. The default set of actions would look like this in the configuration file:

```
<Actions>
        <add name="SetSamlPrincipal" type="dk.nita.saml20.Actions.SamlPrincipalAction, dk.nita.saml20 " />
        <add name="Redirect" type="dk.nita.saml20.Actions.RedirectAction, dk.nita.saml20" />
</Actions>
```

A `<clear/>` tag can be used to clear the list of actions, and a `<remove>` tag can be used to remove a single action by name, eg.: `<remove name="SetSamlPrincipal"/>.`

It is possible to write your own custom actions to perform any task needed during login and logout. Your action must implement the `dk.nita.saml20.Actions.IAction interface, and if you plan to make an action that performs a redirect you should note the following:`

1) A redirect must be the last action in the list, since redirecting ends the action pipeline.
2) Redirecting during logout should only be performed when the logout is NOT initiated by the IdP. If it is initiated by the IdP, a redirect action should do nothing. You will know whether or not the logout is IdP initiated by checking the Boolean parameter `IdPInitiated` of the `LogoutAction` function of the `IAction` interface .

9.1.4 <SessionTimeout>

The <SessionTimeout> element defines when the OIOSAML.net session state must expire. The OIOSAML.net component uses sliding expiration, which means the session timeout is reset on each request to the session. The value must be specified in minutes and the default value is 30 minutes if no SessionTimeout element has been specified. This value should be equal to or higher than the authentication session (e.g. if forms authentication is used). Otherwise, strange behaviour can occur because the system thinks that the user is logged id but no principal exists in the OIOSAML.net session.

9.2 <Saml20Federation>

The `<Saml20Federation>` element contains configuration options that are specific to the SAML 2.0 protocol.

9.2.1 <ShowError>

This setting is used for development purposes alone – it default to false if omitted. When set to true, any errors, both exceptions and validation errors are shown in the browser. Due to a security issue with XML Encryption, this setting must be set to false (or omitted) when used in production, otherwise an attacker might be able to decrypt any messages send to the serviceprovider.

Example

```
<ShowError>false</ShowError>
```

9.2.2 <ServiceProvider>

This element contains the following attributes

Attributes	
`Id`	The service provider's identifier. This is often an URI signaling the domain of the service provider.
`server`	The base URL of the host where the service provider resides. No sub-directories.

The element must have the following child elements

Element name	`ServiceEndpoint`
Description	Configures the HTTP endpoints used by the service provider to communicate with its federation partners. Each endpoint must correspond to a handler in the service provider website
Attributes	
`Type`	Determines the function of the endpoint. Must be one of the following • `signon` • `logout` • `metadata` The service provider must have one of each type of endpoint to function fully.
`localpath`	The address at which the endpoint is.
`redirectUrl`	The URL to which the user will be sent after the handler is done executing.

Element name	`NameIdFormats`
Description	Lets you specify which NameIdFormats are supported by the service provider.

Attributes	
All	true/false When true all known SAML NameIdFormats are supported, and when false only those explicitly added are supported. See example below.
Example	
	<NameIdFormats all="false"> <add nameIdFormat="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName"/> </NameIdFormats>

You should also include the following two elements, which are used when generating metadata, and are required by some IdP's:

```
<md:ContactPerson contactType="administrative" xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">
        <md:Company>Firma</md:Company>
        <md:GivenName>Fornavn</md:GivenName>
        <md:SurName>Efternavn</md:SurName>
        <md:EmailAddress>email@firma.com</md:EmailAddress>
        <md:TelephoneNumber>12345678</md:TelephoneNumber>
</md:ContactPerson>

<md:Organization xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">
        <md:OrganizationName>Firma</md:OrganizationName>
        <md:OrganizationDisplayName>Firmanavn</md:OrganizationDisplayName>
        <md:OrganizationURL>Error! Hyperlink reference not valid.>
</md:Organization>
```

9.2.3 <CommonDomain>

The <CommonDomain> element contains configuration options for common domain cookie reading, and has the following attributes:

Attributes	
`Enabled`	A Boolean value indicating whether or not common domain cookie reading is turned on.
`localReaderEndpoint`	The fully qualified url to a local cookie reader endpoint. The host part of this endpoint should be a sub domain to the common domain, e.g. sp.commondomain.com

The following is an example of a CommonDomain section:

```
<CommonDomain enabled="true" localReaderEndpoint="http://mysp.commondomain.local/cdcreader.ashx" />
```

Read more about how to configure common domain cookie reading in chapter 11.

9.2.4 <RequestedAttributes>

This element names the SAML attributes that the service provider requires from its federation partners.

Element name	`RequestedAttributes`
Description	The list of attributes that the service provider wants assertions to contain when it receives them.
Child elements	
Element name	`att`
Description	Describes the required attribute.
Attributes	
`Name`	The SAML attribute's identifier. Note that this attribute refers to the `name` property of a SAML attribute, not the optional `friendlyName` property that may be found in the identity provider's metadata.
`isRequired`	An optional attribute that specifies whether the SAML attribute should be listed as

	'required' in the service provider's metadata. Further explanation of this attribute can be found in [Metadata] section 2.4.4.2.

9.2.5 <IDPEndPoints>

The configuration in the `<IDPEndPoints>` element determines how the service provider communicates with its federation partners. The minimal configuration specifies the directory in which metadata of the federation partners can be found, and uses the default SAML bindings for communication.

Attributes	
`metadata`	The path to the directory where the metadata of the federation partners can be found. Make sure that the directory is readable by the user running the web server.
`idpSelectionUrl`	URL for custom webpage that lets the user select from a list of available IDP's. The attribute is used only when no common domain cookie is set, and no add-element (see below) has the default-property set to `true`. In case no idpSelectionUrl is specified, a default IDPSelectionPage is displayed. See example of using this attribute in section 5.5.2.

The `<IDPEndPoints>` section can furthermore override the settings that are found in the metadata of a federation partner. This is useful for situations where the federation partner allows several ways of communicating and the default way is not desirable.

Element name	`add`
Description	Enables configuration of which transport binding to use for communicating with a federation partner. Also makes it possible to override endpoint addresses and provide a user-readable name for the federation partner.
Attributes	
`Id`	The id of the federation partner to which this configuration pertains. This id must have a match in one of the metadata files that are known by the service provider.
`Name`	An optional setting that contains a human-readable name for the federation partner. The name will replace the federation partner's id, in cases where it is necessary to present the user with a choice of federation partners.
`omitAssertionSignatureCheck`	Set this value to true if for some reason you do not wish the signature of the assertions from this IdP to be checked (for example if assertions are not signed).
`forceAuthn`	Force authentication on each authnrequest
`isPassive`	AuthnRequests are passive
`default`	When set to true, this IdP will be used for authentication in case no Common Domain Cookie is set. If no IdP has `default` set to `true`, and more than one IdP is present in the IDPEndPoints collection, the user will be prompted to

	select from the list of IdP's.
Child elements	
Element name	`SSO`
Description	Configuration of the Single sign on endpoint of the federation partner.
Attributes	
`url`	An optional parameter that contains the SSO endpoint's URL. If this parameter is left out, the URL found in the federation partner's metadata is used. Override this with care.
`binding`	Specify which binding to use when sending an authentication request to the federation partner. If this attribute is left unspecified, and the federation partner allows it, the HTTP-REDIRECT binding will be used. Allowed values are: "`POST`", "`REDIRECT`" and "`ARTIFACT`".
Element name	`SLO`
Description	Configuration of the Single logout endpoint of the federation partner.
Attributes	
`url`	An optional parameter that contains the SLO endpoint's URL. If this parameter is left out, the URL found in the federation partner's metadata is used. Override this with care.
`binding`	Specify which binding to use when sending a logout request to the federation partner. If this attribute is left unspecified, and the federation partner allows it, the HTTP-REDIRECT binding will be used. Allowed values are: "`POST`", "`REDIRECT`" and "`ARTIFACT`".
Element name	AttributeQuery
Description	Lets you enable httpBaiscAuth with username and password for attribute queries
Example	<AttributeQuery enableHttpBasicAuth="true" username="username" password="p@assw0rd" />
Element name	ArtifactResolution
Description	Lets you enable httpBasicAuth with username and password for artifact resolution.
Example	< ArtifactResolution enableHttpBasicAuth="true" username="username" password="p@assw0rd" />

10 API reference

10.1 Saml20Identity

The `Saml20Identity` class is the primary class for interacting with the information received from the identity provider after the user has signed in.

It extends the `System.Security.Principal.IIdentity` interface with methods that provide access to the received attributes and the principal's persistent pseudonym.

The SAML extended principal is obtained through the `Saml20Identity.Current` property. This property is `null` if the user has not been authenticated with a SAML assertion.

Class	`Saml20Identity`
Properties	
`static Saml20Identity Current`	Retrieves the user's identity and the attributes that were extracted from the SAML assertion. Returns `null` if the user has not been authenticated using SAML. If `null` is returned the system must ensure that the user is logged out of the system.
`string PersistentPseudonym`	This property holds the persistent pseudonym, if one was used when authenticating the user.
Methods	
`static bool IsInitialized`	
Returns `true` if the user has been authenticated using SAML. Can be used to check that the `Saml20Identity` is available. If false is returned the SP must ensure that the user is logged out. This check should be made in the beginning of each user request.	
`bool HasAttribute(string)`	
Checks whether the attribute with the given name was issued with the assertion.	
`List<SamlAttribute> this [string]`	
Retrieves a list of attributes with the given name. Example of usage: `List<SamlAttribute> name = Saml20Identity.Current["urn:oid:2.5.4.5"];` This method will throw a `KeyNotFoundException` if the attribute is not found. Use `HasAttribute` to verify the presence of an attribute before calling this method.	

10.2 HttpHandlers

The endpoints handling protocol messages are implemented using ASP.NET Http Handlers.

There are 3 endpoint types that must be installed to get a functioning service provider.

Endpoint type	HTTP Handler class
Sign on	`dk.nita.saml20.protocol.Saml20SignonHandler`
Logout	`dk.nita.saml20.protocol.Saml20LogoutHandler`
Metadata	`dk.nita.saml20.protocol.Saml20MetadataHandler`

A service provider installation must have all 3 handlers installed. The metadata endpoint can be removed once the service provider's configuration is finalized and its metadata file has been downloaded.

Furthermore there are two extra endpoints, of which at least one is necessary to install if common domain cookie reading is enabled (more details about this in chapter 11):

Endpoint type	HTTP Handler class
Local cookie reading	`dk.nita.saml20.protocol.Saml20CDCReader`
IdP cookie writer return point	`dk.nita.saml20.protocol.SAML20CDCIdPReturnPoint`

ASP.NET provides (at least) two ways to add HTTP handlers to a web application

Web.config

HTTP handlers can be added to a web application by adding them in the `<httpHandlers>` section of `web.config`. Consult the MSDN documentation for a reference on the `<httpHandlers>` element. The `verb` attribute must be set to "`*`" for the handlers.

Website files

A handler can be added to a website as an `.ashx` file.

Example of how to add sign on handler in a file called `signon.ashx`:

```
<%@ WebHandler Class="dk.nita.saml20.protocol.Saml20SignonHandler" %>
```

More examples of `.ashx` files can be found in the website demo in the distribution.

10.3 Attribute queries

An attribute query enables a service provider to request more attributes on an authenticated subject from the federation partner that authenticated him. More information on attribute queries can be found in section 3.3.2.3 of [SAML].

The queries are executed with the currently authenticated user as a subject. Returned attributes are available through the `Saml20Identity` class after the query finishes.

The following is an example of how to query for all available attributes about the currently authenticated user

```
Saml20AttributeQuery query = Saml20AttributeQuery.GetDefault();
query.PerformQuery(Context);
```

Specific attributes are requested by adding the name of the attributes to the request.

```
Saml20AttributeQuery query = Saml20AttributeQuery.GetDefault();
query.AddAttribute("urn:oid:2.5.4.11");
query.PerformQuery(Context);
```

10.4 Sample identity provider API

The user data issued by the demonstration identity provider are hard coded, so if you wish to modify these it is necessary to recompile from the source distribution.

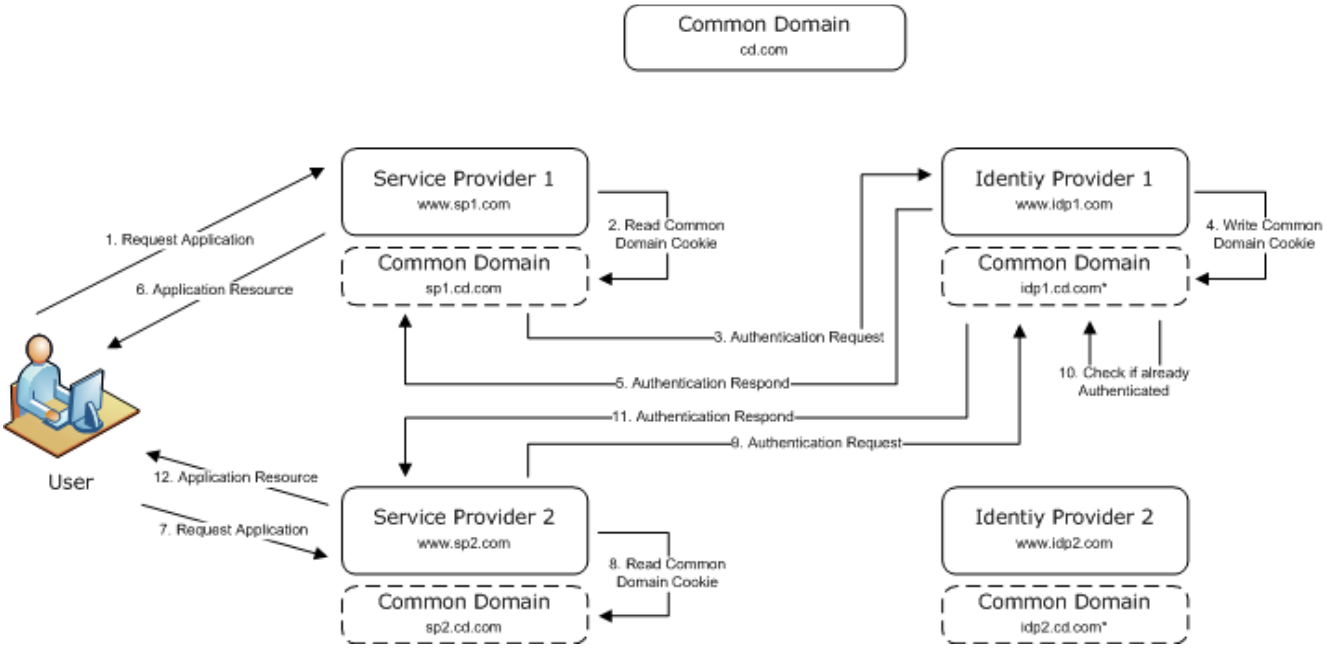
The list of available attributes can be found in the `IdentityProviderDemo.Logic.IDPConfig` class. The available users and their attributes are defined in the class `IdentityProviderDemo.Logic.UserData`.

10.5 ForceAuthn and IsPassive

If you wish to set the ForceAuthn and IsPassive flags for a given authentication request, this can be done by setting two query string variables when hitting the login handler. The names of the query string variables in question are defined as constants in the class Saml20AbstractEndpointHandler. The constants are: `IDPForceAuthn` and `IDPIsPassive`. The value of the query string variables must be of type `bool (True/true` or `False/false)`. The default value for each query string variable is false. Example of a login reguest:
https://XXX/demo/login.ashx?cidp=https%3a%2f%2fsaml.test-nemlog-in.dk%2f&forceAuthn=False&isPassive=False.

11 Common Domain Cookies

This chapter will describe how to enable the reading of common domain cookies in the federation. The setup used in this demonstration of the common domain cookie is illustrated in the figure below.



As illustrated in the figure above, there reside two Service Providers and two Identity Providers in the basic setup. The following briefly explains the scenario illustrated in the figure.

1. The user requests a web application resource from Service Provider 1.
2. Service Provider1 determines that the resource is protected and that no common domain cookie exists, since the user has no SSO session with an Identity Provider. The user is prompted to select one of the two possible Identity Providers.
3. The user selects Identity Provider 1, which causes Service Provider 1 to redirect the user to Identity Provider1 with a request as parameter.
4. Identity Provider 1 establishes that the user has no current (IdP) session, and therefore initiates the authentication of the user after which the Identity Provider stores the identifier of itself in the common domain cookie.
5. Identity Provider 1 sends the user back to Service Provider 1 with a response containing a SAML assertion. The Service Provider validates the assertion and performs an authorization check on the resource originally requested by the user.
6. The resource is finally returned to the user.
7. The user requests a web application resource from Service Provider 2.
8. Service Provider2 determines that the resource is protected and that a common domain cookie exists.
9. The user is redirected to the Identity Provider specified in the cookie, namely Identity Provider 1.
10. Identity Provider 1 establishes that the user is already authenticated.
11. The user is sent back to Service Provider 2 with a response containing a SAML assertion. Service Provider 2 validates the assertion and performs an authorization check on the resource originally

requested by the user.

12. The resource is finally returned to the user.

12 Virk.dk extension

The assembly dk.nita.saml20.ext.brs.dll is a framework that allows programmatic access to the Authorisations attribute from Virk.dk.

It is important to note that the Authorisations attribute is only supplied through attribute query when using POST binding (The only binding currently supported by this toolkit when connecting to virk.dk).

After the user is authenticated and the Authorisations attribute has been obtained through an attribute query, you can use code like following to determine if a user X has a given privilege Y:

```
BRSUtil util = new BRSUtil();

if(util.HasAuthorisationAttribute())
{

	string cvrNumber = "12365454";

	string privilege = "aPrivilege";

	bool b = util.HasPrivilegeForCvrNumber(cvrNumber, privilege);

}
```

The function `HasPrivilegeforCvrNumber` throws an exception if the Authorisations attribute has not been obtained for the current user, which is why the `HasAuthorisationsAttribute` should be called. The `BRSUtil` class also has a function called `HasPrivilegeForProductionUnit` which has the same method signature as `HasPrivilegeforCvrNumber`, but the first parameter should be a production unit id instead of a cvr number.

Please refer to the WebsiteDemoVirk folder for a more complete example on how to use this extension.

13 Troubleshooting

13.1 Enabling debug logging

Debug logging can be enabled for any website using the framework by adding the following section to web.config:

```
	<system.diagnostics>
		<trace autoflush="true" ></trace>
		<sources>
			<source name="dk.nita.saml20" switchValue="Information">
				<listeners>
					<add name="trace"/>
				</listeners>
			</source>
		</sources>
		<sharedListeners>
			<add name="trace" type="System.Diagnostics.XmlWriterTraceListener"
initializeData="C:\logs\saml2.tracelog"/>
		</sharedListeners>
	</system.diagnostics>
```

Please note that the debug information will be written to the file specified in the `initializeData` attribute and that the directory (in this case `c:\logs`) must exist.
If you need further information to be traced you can change switchValue from “Error” to “Information”.

14 Audit & Logging

The dk.nita.saml20.Logging.AuditLogging class is responsible for audit logging events in the framework. The class uses the configured implementation of the IAuditLogger interface and thereby provides a plug-in design allowing other different implementations for audit logger.

The AuditLogging class uses a System.Diagnostics Trace as default audit logger implementation if no configuration has been applied.

The dk.nita.saml20.ext.audit.log4net project supplies a log4net implementation of the IAuditLogger interface.

The web.config file for the demo service provider shows an example of setting up log4net logging to a file, but log4net can also log to relational databases etc.

The example is shown here (remember to configure the auditLoggingType attribute of the Federation element to use to log4net implementation):

```
<configuration>
   <configSections>
    …
      <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler,Log4net"/>
   </configSections>
   <log4net>
      <root>
         <level value="All" />
            <appender-ref ref="LogFileAppender" />
      </root>
      <appender name="LogFileAppender" type="log4net.Appender.RollingFileAppender" >
                  <param name="File" value="C:\temp\log.txt" />
                  <param name="AppendToFile" value="true" />
                  <rollingStyle value="Size" />
                  <maxSizeRollBackups value="10" />
                  <maximumFileSize value="10MB" />
                  <staticLogFileName value="true" />
                  <layout type="log4net.Layout.PatternLayout">
                        <param name="ConversionPattern" value="%-5p%d{yyyy-MM-dd hh:mm:ss}-%m%n" />
                  </layout>
      </appender>
   </log4net>
…
   <Federation xmlns="urn:dk.nita.saml20.configuration"
   auditLoggingType="dk.nita.saml20.Logging.Log4NetAuditLogger, dk.nita.saml20.ext.audit.log4net">
```

15 Session Providers

A custom made session handling has been made in order to support random access to all active sessions. This is not supported by the ASP.NET session state. Random access is needed when receiving SOAP Logout requests which through a back channel must be able to log arbitrary user out.

A session state provider can be obtained through the dk.nita.saml20.session.SessionFactory.SessionContext. It returns an implementation of the ISessons interface. Note that the session provider is only for internal use to the OIOSAML.net component and is therefore not accessible outside the OIOSAML.net component.

15.1 Implementing your own session provider

Two interfaces must be implemented in order to implement a new session provider. Namely, dk.nita.saml20.session.ISessions and dk.nita.saml20.session.ISession. An alternative to implementing the ISessions interface is to inherit from dk.nita.saml20.session.AbstractSessions. AbstractSessions comes with a default implementation on how to generate session id's and on how to persist session id's between user requests using cookies.

See configuration section on how to configure the OIOSAML.net component to use your own session provider.

15.2 Default implementations

Two default implementations has been provided out of the box. An InProcSessions and an AppFabricSessions implementation.

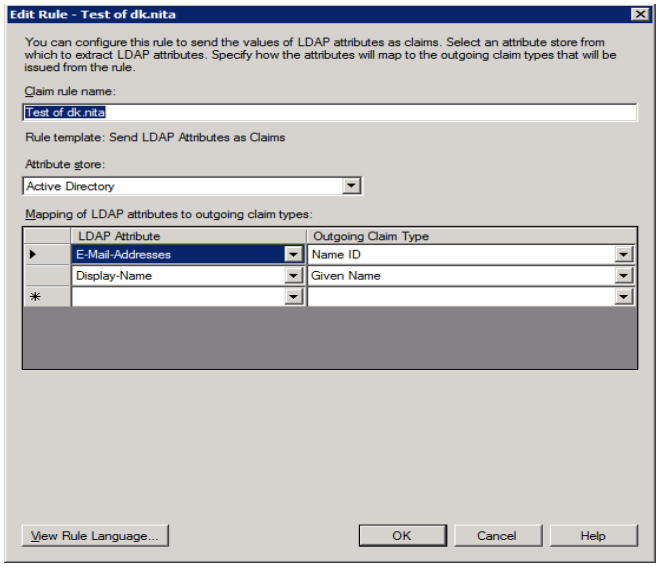
The InProcSessions implementation is an in-memory implementation that are suitable for service providers running on a single server. If nothing has been configured the InProcSessions is used as default.

The AppFabricSessions implementation makes use of AppFabric Cache and is suitable for service providers that will be running in a load balanced setup. The implementation can be retrieved from NuGet which also automatically makes the proper configurations in the web.config file. However, the AppFabric configuration section must be updated with settings matching your own environment.

16 Setting up with ADFS v2.

The following shows how to setup dk.nita with ADFS v2 as Identify Provider.

1. Install Windows Identify Framework: http://www.microsoft.com/downloads/details.aspx?familyid=EB9C345F-E830-40B8-A5FE-AE7A864C4D76&displaylang=en#filelist
2. Enable SSL on IIS (Preferably use "real" certificate, i.e. not self issued)
3. Download ADFS 2 RC: http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=118c3588-9070-426a-b655-6cec0a92c10b
4. Configure ADFS 2.0
5. Setup the relying party, that is the Demo Service Provider
    a. The Demo Service Provider must be available through SSL
        i. Remember to change the "Server" attribute of the ServiceProvider in the SP's web.config. It should be on HTTPS if ADFS is to accept the endpoints
    b. If you want to have automatic metadata exchange and monitoring the the SSL certificate on the SP must be valid and trusted (I.e. no need to accept a "broken" certificate)
    c. If you don't want to go through the trust exercise, just download the SP metadata into a file and load the file into ADFS
6. Add "Claim rules" (use "Edit Claim Rules…" on the popup menu of the relying party)
    a. MUST: Add two LDAP claims:



Add any other claims you like

7. Go to the service provider machine and download the ADFS metadata and put them in the appropriate directory, e.g. “c:\metadata”

8. Open the SP in a browser and try to log in.

9. If you get an error like “ErrorCode: …status: Responder. Message: .” it may be a problem with the expected hashing algorithm on ADFS:

    a. Check the event log of ADFS and look for trouble with the signature of the SAML Request

    b. If a problem with SHA-1 vs. SHA-256 is indicated, go to ADF and bring up the properties of the relying party (Demo SP).

    c. Go to the “Advanced” tab and change the signature algorithm to SHA-1. (For some reason ADFS is not able to read this from metadata)

2. The provided demo certificates cannot be used in this scenario. Use real issued certificates from a trusted party (e.g. and internal CA).

17 References

[SAML] Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) v2.0
saml-core-2.0-os
http://docs.oasis-open.org/security/saml/v2.0/

[Metadata] Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0
saml-metadata-2.0-os
http://docs.oasis-open.org/security/saml/v2.0/

[Bindings] Bindings for the OASIS Security Assertion Markup Language (SAML) v2.0
saml-bindings-2.0-os
http://docs.oasis-open.org/security/saml/v2.0/