ComponentSpace

SAML v1.1 for.NET Developer Guide

www.componentspace.com

Contents

1 Introduction 1
1.1 Features 1
1.2 Benefits 1
1.3 Prerequisites 1
2 Getting Started 1
2.1 Adding a Reference 2
2.2 Distribution 2
3 Single Sign-On Applications 2
3.1 Identity Provider Applications 3
3.2 Service Provider Applications 3
3.3 XML Signatures and Key Management 3
3.4 Troubleshooting XML Signatures 4
3.5 Creating Test Keys and Certificates 4
3.5.1 Microsoft Certificate Server 4
3.5.2 Keytool.exe 5
4 Examples 5
4.1 SAML Assertion Examples 5
4.2 SAML Protocol Examples 5
4.3 SAML Single Sign-on Examples 6
4.4 Signature Examples 7
4.5 OpenSAML Signature Examples 7
5 Troubleshooting 8
5.1 Tracing 8
5.1.1 Tracing in Web Applications 8
5.1.2 Tracing in Non-Web Applications 8
6 Class Library Reference 9
7 Frequently Asked Questions 9
8 Support 13

1 Introduction

The ComponentSpace SAML v1.1 component is a Microsoft .NET class library that allows you to work with SAML v1.1 assertions, protocol messages, bindings and profiles.

The class library is a toolkit for adding support for SAML v1.1 single sign-on (SSO) to ASP.NET applications, using SAML assertion security tokens in web service applications, and for any custom use of SAML assertions.

1.1 Features

- Supports SAML v1.1 Assertions, Protocol, Bindings and Profiles as defined by the OASIS standard (www.oasis-open.org)
- Easily create, modify and access SAML assertions and protocol messages using the encapsulating classes
- Generate and verify XML signatures on SAML assertions and protocol messages
- Add SAML single sign-on support to your ASP.NET applications

1.2 Benefits

- Easy to use class library enabling SAML v1.1 functionality to be quickly added to your .NET applications
- Developer based licensing with no runtime royalties meaning you don’t pay per deployment or end user
- Developed in C# with full source code available for purchase
- Includes class library reference and example applications with full source code

1.3 Prerequisites

The class library requires the .NET v2.0 framework or above and is for use with Visual Studio 2005 or above.

It has been tested with .NET framework versions 2.0, 3.0, 3.5, 4.0, 4.5 and 4.6, using Visual Studio 2005, 2008, 2010, 2012, 2013 and 2015, on 32-bit and 64-bit Windows Server 2003, 2008, and 2012, as well as Windows 7, 8 and 10.

2 Getting Started

1. If you haven't already done so, install this product by double clicking the Microsoft Installer (MSI) file and following the installation steps.

   A free evaluation copy is available from our web site.
2. Compile and run the example applications (see section 4).
3. Refer to the class library reference for help using the classes.
4. Add a reference to the class library in your application (see section 2.1).

5. Once completed, install or distribute the class library DLL with your application (see section 2.2).

Please feel free to contact us if you need any assistance (see section 8).

2.1 Adding a Reference

To use the class library in Visual Studio, you need to add a reference to the class library DLL from within your project.

With your project open, in the Solution Explorer right click the project and click *Add Reference...*. Click the *Browse* button and browse to the ComponentSpace.SAML2 DLL. You will find the DLL in the bin directory under the installation directory (e.g. C:\Program Files (x86)\ComponentSpace SAML v2.0 for .NET\Bin).

With your project open, in the Solution Explorer right click the project and click *Add*

Once the reference has been added you can refer to the various classes from within your project.

2.2 Distribution

The class library's runtime is royalty free which means it may be freely distributed with your application. The only file that should be distributed is the class library DLL.

3 Single Sign-On Applications

Single sign-on refers to the user logging in once, at the identity provider, and being able to access applications at service providers with no further logins being required.

An identity provider collects user identity information when the user logs in at the identity provider. The identity provider supplies service providers with a SAML assertion representing the identity of the user logged in at the identity provider. Service providers use the credentials in the SAML assertion to perform a local login at the service provider.

There are two profiles that describe how the SAML assertion is passed from an identity provider to a service provider: browser/post and browser/artifact.

In the browser/post profile, the SAML assertion is contained within a SAML protocol response message that’s included as a hidden field within a form that’s submitted to the service provider. Submission of the form can either be automatic through JavaScript or require the user to click a submit button.

In the browser/artifact profile, the SAML assertion is created but not immediately sent to the service provider. Instead it is saved at the service provider and keyed off an artifact. This artifact is sent to the service provider via an HTTP redirect. The service provider sends a SAML protocol request message containing the artifact to the identity provider. The identity provider looks up the SAML assertion using the artifact and returns it in a SAML protocol response message.

For further details refer to the OASIS standard (www.oasis-open.org).

3.1 Identity Provider Applications

The example identity provider application (page 6) demonstrates the basic operation of an identity provider. It supports no single sign-on, in which case the user has to log in at both the identity provider and service provider, and single sign-on using either the browser/post or browser/artifact profile.

Once logged in at the identity provider, any access to the service provider is made through the identity provider's inter-site transfer page (InterSiteTransfer.aspx) which handles both the browser/post and browser/artifact profile for the identity provider.

If using the browser/artifact profile, the identity provider's SAML responder page (SAMLResponder.aspx) handles SAML protocol requests from service providers. It uses the received artifact to look up the previously generated SAML assertion, creates a SAML protocol response containing this SAML assertion, and returns it to the service provider.

3.2 Service Provider Applications

The example service provider application (page 6) demonstrates the basic operation of a service provider. It supports no single sign-on, in which case the user has to log in at the service provider, and single sign-on using either the browser/post or browser/artifact profile.

If using browser/post, the assertion consumer page (AssertionConsumer.aspx) receives the form posted by the identity provider, reconstructs the SAML protocol response, retrieves the SAML assertion from the response, and uses the subject contained within the SAML assertion to perform an automatic login at the service provider. It then redirects to the target service provider page.

If using browser/artifact, the artifact receiver page (ArtifactReceiver.aspx) receives the artifact from the identity provider. It then sends the identity provider a SAML protocol request containing the artifact, receives the SAML protocol response, retrieves the SAML assertion from the response, and uses the subject contained within the SAML assertion to perform an automatic login at the service provider. It then redirects to the target service provider page.

3.3 XML Signatures and Key Management

The SAML specification supports the signing of SAML assertions and SAML protocol request and response messages. The example identity provider and service provider demonstrate signing the SAML protocol response.

The identity provider loads an X.509 certificate and corresponding private key from file during application startup (Global.asax).

It is also possible to load certificates and keys from a user or machine certificate store. Depending on the operating environment best practice is to retrieve certificates and keys from the local machine certificate store rather than the current user certificate store. Please refer to the Microsoft knowledge base article Q322371 for more information.

The service provider optionally loads an X.509 certificate from file. If this file is supplied then signatures are verified using the loaded certificate. If this file is not supplied then a

certificate is assumed to be included in the XML signature and is used to verify the signature.

3.4 *Troubleshooting XML Signatures*

When verifying an XML signature over a SAML assertion or protocol message either true is returned, indicating verification succeeded, or false is returned, indicating verification failed. Typically verification fails because either the signed XML has been altered in some manner or the wrong certificate is used to perform the verification.

The first step is to ensure the correct certificate is being used. The certificate is either contained in the XML signature or is loaded from a certificate file or store. If you are using a separately loaded certificate rather than a certificate contained within the XML signature, ensure the certificate is the correct one.

If you believe the correct certificate is being used you can run a utility to see whether the signature can be verified. The signature examples (page 7) include a signature verification application, VerifySaml.exe. For example, to verify the signature on a SAML protocol response you would run:

> VerifySaml -response [–c <certificateFileName>] <filename>
>
> The certificateFileName is a CER file containing the certificate to use to verify the signature. Only specify this parameter if the certificate is being loaded from a certificate file or store. If the certificate is included in the XML signature, then do not specify this parameter.
>
> The filename is the file containing the SAML protocol response as XML.

If you are familiar with Java you can also run an equivalent Java utility (page 7) which uses OpenSAML. For example, to verify the signature on a SAML protocol response you would run VerifySaml.bat with the same parameters as above.

3.5 Creating Test Keys and Certificates

When creating an XML signature over a SAML assertion or protocol message you need to supply a private key and certificate. The private key and certificate may either be loaded from a key store or a PFX (PKCS12) file.

3.5.1 Microsoft Certificate Server

There are a number of ways to generate a private key and certificate. One approach is to make a request to the Windows 2003 certificate server and have it issue a key and certificate. You can then install it in a certificate store and, if required, export the certificate to a CER file or the key and certificate to a PFX file. The steps are outlined below. For more detailed information on using the Microsoft certificate Server refer to the MSDN.

1. If not already done, install the Certificate Services Windows component. This installs a certification authority (CA) to issue certificates.
2. Navigate to the certificate service (e.g. http://localhost/certsrv) and request a certificate. Select the "advanced certificate request" and then "Create and submit a

request to this CA". Fill in the certificate request details, specifying the certificate type as server authentication certificate and make sure "Mark keys as exportable" is checked.

3. Using the Certification Authority MMC snap-in, view the pending requests and issue a certificate.
4. Back at the certificate service click, view the status of the pending certificate request and click the link to install the certificate.
5. Using the Certificates MMC snap-in, view the certificate to confirm that it has been installed. If required you can export the certificate and private key to a PFX file but make sure to check "Include all certificates in the certification path if possible". You can also export the certificate only if required.

3.5.2 Keytool.exe

Another approach is to use the Java 1.5 JDK's keytool.exe. For example:

keytool.exe -genkey -dname "cn=test" -alias test -keypass password -keyalg RSA -validity 3650 -keystore test.pfx -storepass password -storetype pkcs12

To generate a certificate file:

keytool.exe -export -alias test -keystore test.jks -storepass password -storetype pkcs12 -file test.cer

4 Examples

Example projects, with full source code and developed using Microsoft Visual Studio, are included with the product. You will find these projects in sub-directories of the installation directory or you may navigate to them from the Start menu.

The example applications must be built and published prior to their use.

4.1 SAML Assertion Examples

AssertionExample

AssertionExample is a VB.NET project that demonstrates the creation, manipulation and conversion, to and from XML, of SAML assertions.

AssertionSignatureExample

AssertionSignatureExample is a VB.NET project that demonstrates generating and verifying XML signatures contained in SAML assertions.

4.2 SAML Protocol Examples

ProtocolExample

ProtocolExample is a VB.NET project that demonstrates the creation, manipulation and conversion, to and from XML, of SAML protocol messages.

ProtocolSignatureExample

ProtocolSignatureExample is a VB.NET project that demonstrates generating and verifying XML signatures contained in SAML protocol messages.

4.3 SAML Single Sign-on Examples

Example identity provider and service providers are included with the product.

To run these examples:

1. Build the example projects including IdentityProvider and ServiceProvider.
2. Install both applications under IIS.

   For example, under the IIS Default Web Site, add an application named IdentityProvider with a physical path of C:\Program Files\ComponentSpace\SAML v1.1 for .NET 2.0\Examples\PrecompiledWebIdentityProvider.

   Similarly, add an application named ServiceProvider with a physical path of C:\Program Files\ComponentSpace\SAML v1.1 for .NET 2.0\Examples\PrecompiledWeb\ServiceProvider.
3. Using your browser, navigate to:

   http://localhost/IdentityProvider
4. Log into the identity provider using the credentials idp-user and password.
5. At the identity provider site you have the option of using SAML single sign-on to access the service provider site. You can use either the Browser/Post or Browser/Artifact profile.

   Using either of these options will automatically sign you in at the service provider as idp-user and redirect you to the service provider site.

   You also have the option of not performing a single sign-on in which case you must manually log into the service provider using the credentials sp-user and password.
6. Please note that these example applications are hard-coded to expect the above URLs. You must modify them if you wish to run them at different locations.

IdentityProvider

IdentityProvider is a VB.NET project that demonstrates browser/artifact and browser/post support in an ASP.NET application acting as the identity provider.

ServiceProvider

ServiceProvider is a VB.NET project that demonstrates browser/artifact and browser/post support in an ASP.NET application acting as the service provider.

4.4 Signature Examples

Sign

Sign is a C# project that demonstrates signing SAML assertions and protocol messages. It is also a useful utility for debugging signature verification errors.

Usage:

SignSaml [-assertion | -request | -response] –k <keystore> -p <password> <filename>

where the assertion, request or response parameter indicates the type of object to sign, the keystore is a PFX file containing a key, the password is the password to the keystore, and the file contains the XML to be signed.

Verify

Verify is a C# project that demonstrates verifying signatures on SAML assertions and protocol messages. It is also a useful utility for debugging signature verification errors.

Usage:

VerifySaml [-assertion | -request | -response] [–c <certificateFileName>] <filename>

where the assertion, request or response parameter indicates the type of object whose signature is to be verified, the certificateFileName is a CER file containing the certificate to use to verify the signature, and the file contains the signed XML to be verified. If no certificateFileName is specified then the certificate contained in the signed XML is used.

4.5 OpenSAML Signature Examples

Included are some simple example Java classes that use OpenSAML to generate and verify XML digital signatures for SAML assertions, requests and responses.

You can use these classes to confirm that signatures generated by the ComponentSpace SAML component can be verified by OpenSAML and vice versa.

To build and run the examples you will need to download OpenSAML from www.opensaml.org and place the jars in the lib directory.

The required jars are: opensaml-1.1.jar; xmlsec-20050514.jar; log4j-1.2.5.jar; commons-logging-1.03.jar and commons-codec-1.3.jar.

The example Java classes were built using Java 1.5. The included build.bat may be used to rebuild the samltests.jar.

The following batch files may be used to run the example classes:

- signsaml.bat
- verifysaml.bat.

Sign

Sign is a Java class that signs SAML assertions and protocol messages. It is also a useful utility for debugging signature verification errors.

Verify

Verify is a Java class that verifies signatures on SAML assertions and protocol messages. It is also a useful utility for debugging signature verification errors.

5 Troubleshooting

5.1 Tracing

To help resolve problems, tracing internal to the product may be enabled. If you are experiencing a problem you may be asked to enable tracing.

5.1.1 Tracing in Web Applications

To enable diagnostic tracing, update your application's web.config to include a `<system.diagnostics>` section as shown in the example configuration below.

```
<system.diagnostics>
  <switches>
    <add name="ComponentSpace.SAML" value="1"/>
  </switches>
  <trace autoflush="true" indentsize="4">
    <listeners>
      <add name="TextWriter"
           type="System.Diagnostics.TextWriterTraceListener"
           initializeData="c:\temp\SAML.log"/>
    </listeners>
  </trace>
</system.diagnostics>
```

You must ensure the directory where the log file will be written exists. In this example, the directory c:\temp must exist in order for the SAML.log file to be created.

The user account running the web application must have write permission to this directory. For example, give the IIS_USERS group write permission to the directory.

You may set the ComponentSpace.SAML switch value to zero and one to disable and enable tracing respectively. Normally this can be done without restarting the web site.

The example identity provider and service provider applications include a diagnostic tracing section in their web.config files.

5.1.2 Tracing in Non-Web Applications

Create a standard .NET configuration file for your application, if one doesn't already exist. The configuration file name consists of your application file name and *.config*.

For example, if your application is called *myapplication.exe* then its configuration file should be named *myapplication.exe.config*. The configuration file must be located in the same directory as the application executable.

Include in the `system.diagnostics` section a switch and listeners as shown in the example configuration below.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.diagnostics>
    <switches>
      <add name="ComponentSpace.SAML" value="1"/>
    </switches>
    <trace autoflush="true" indentsize="4">
      <listeners>
        <add name="TextWriter"
             type="System.Diagnostics.TextWriterTraceListener"
             initializeData="SAML.log"/>
      </listeners>
    </trace>
  </system.diagnostics>
</configuration>
```

Alternatively you may add this configuration information to *machine.config* to trace all applications using the product.

6 Class Library Reference

The reference section is contained within a separate help file. You can find the reference help file in the installation directory or navigate to it using the Start menu.

7 Frequently Asked Questions

1. *What is SAML?*

   The Security Assertion Markup Language (SAML) consists of a set of standards published by the OASIS organization (www.oasis-open.org). Refer to their web site for more details.

2. *What version of SAML is supported?*

   This component supports SAML v1.1. We have a separate component that supports SAML v2.0.

3. *I can't build the examples. Why can't the ComponentSpace namespace be found?*

The assembly containing the namespace or the project referencing it has been moved. The simplest way to fix this is to re-add the reference to the project by browsing to where the DLL is located (typically the installation directory).

4. *I keep receiving an error message saying the trial period has expired. What does this mean?*

You are using an evaluation version of the component and the trial period has ended. If you need to extend the evaluation period, please contact us.

5. *How do I tell if I'm using an evaluation version?*

Navigate to the component DLL and right click it to bring up the file properties. Under the Version tab, the description will specify whether or not it's an evaluation version.

6. *I'm not sure how to use the product. What can I do?*

Whether you're evaluating the product or are an existing customer, please feel free to contact us with any questions you might have.

7. *The product is missing a feature I really need. What can I do?*

Please contact us if there's additional functionality you would like to see. Your feedback is most welcome and will be given careful consideration.

8. *Does the product support SAML single sign-on?*

Yes. The product includes the necessary functionality for enabling SAML single sign-on at either the identity provider or service provider web site. You need to integrate this functionality with your existing web applications. Refer to page 2 for more information.

9. *Can I generate and verify XML digital signatures on SAML assertions and protocol messages?*

Yes. Refer to the Reference section and the Examples for more information.

10. *Is the product compatible with OpenSAML?*

    Yes. The product has been tested against OpenSAML.

11. *Is the product compatible with Product X?*

It should be but if you have any questions please contact us. A simple test is to integrate the example identity provider or service provider application with the product in question.

12. *How do I create a SAML assertion?*

    Use the ComponentSpace.SAML.Assertions.Assertion class to create a SAML assertion.

    The AssertionExample VB.NET project demonstrates creating SAML assertions.

13. *How do I convert a SAML assertion to and from XML?*

    The ComponentSpace.SAML.Assertions.Assertion class includes a constructor that creates a SAML assertion from an XML element. This class also includes a *ToXml* method that converts the SAML assertion object to XML.

    The AssertionExample VB.NET project demonstrates serializing and deserializing SAML assertions to and from XML.

14. *How do I sign a SAML assertion?*

    The ComponentSpace.SAML.Assertions.AssertionSignature class has methods for generating and verifying signatures on SAML assertions that are serialized to XML.

    Typically you create a SAML assertion using the ComponentSpace.SAML.Assertions.Assertion class. Once complete, you convert it to XML using the ToXml method of this class. Then you pass this XML to the *Generate* method of the AssertionSignature class.

    The AssertionSignatureExample VB.NET project demonstrates signing and verifying SAML assertions.

15. *How do I verify a signed SAML assertion?*

    The ComponentSpace.SAML.Assertions.AssertionSignature class has methods for generating and verifying signatures on SAML assertions that are serialized to XML.

    Typically you verify the signature by passing the XML to the Verify method of the AssertionSignature class. Once verified, you create a SAML assertion from the XML using the ComponentSpace.SAML.Assertions.Assertion class.

    The AssertionSignatureExample VB.NET project demonstrates signing and

verifying SAML assertions.

16. *How do I create a SAML protocol message?*

    Use the ComponentSpace.SAML.Protocol.Request and ComponentSpace.SAML.Protocol.Response classes to create SAML protocol request and response messages.

    The ProtocolExample VB.NET project demonstrates creating SAML protocol messages.

17. *How do I convert a SAML protocol message to and from XML?*

    The ComponentSpace.SAML.Protocol.Request and ComponentSpace.SAML.Protocol.Response classes include a constructor that creates a SAML protocol request or response from an XML element. These classes also include a ToXml method that converts the SAML protocol request or response object to XML.

    The ProtocolExample VB.NET project demonstrates serializing and deserializing SAML protocol requests and responses to and from XML.

18. *How do I sign a SAML response?*

    The ComponentSpace.SAML.Protocol.ResponseSignature class has methods for generating and verifying signatures on SAML protocol responses that are serialized to XML.

    Typically you verify the signature by passing the XML to the Verify method of the ResponseSignature class. Once verified, you create a SAML protocol response from the XML using the ComponentSpace.SAML.Protocol.Response class.

    The ProtocolSignatureExample VB.NET project demonstrates signing and verifying SAML protocol messages.

19. *How do I verify a signed SAML response?*

    The ComponentSpace.SAML.Protocol.ResponseSignature class has methods for generating and verifying signatures on SAML protocol responses that are serialized to XML.

    Typically you verify the signature by passing the XML to the Verify method of the ResponseSignature class. Once verified, you create a SAML protocol response from the XML using the ComponentSpace.SAML.Protocol.Response class.

The ProtocolSignatureExample VB.NET project demonstrates signing and verifying SAML protocol messages.

20. *How do I add single sign-on support to my web application?*

    Refer to page 2 for more information on single sign-on applications.

21. *How do I implement a single sign-on identity provider?*

    Refer to page 2 for more information on creating an identity provider.

22. *How do I implement a single sign-on service provider?*

    Refer to page 2 for more information on creating an identity provider.

23. *How do I implement browser/post single sign-on?*

    Refer to page 2 for more information on creating an identity provider.

24. *How do I implement browser/artifact single sign-on?*

    Refer to page 2 for more information on creating an identity provider.
    |

25. *I cannot verify the signature being sent by a 3rd party. What do I do?*

    Refer to page 4 for information on troubleshooting signature verification.

26. *What's the difference between CurrentUser and LocalMachine when referring to certificate store locations?*

    Please refer to the Microsoft knowledge base article Q322371. If running in an ASP.NET application then keys may need to be loaded from LocalMachine key stores.

27. *How do I use HTTPS to secure the connection between identity provider and the service provider?*

    The MSDN has numerous articles on configuring HTTPS within an ASP.NET environment. A good starting point is the article *Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication* (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/SecNetHT16.asp).

8 Support

For further information, visit us at www.componentspace.com or send email to info@componentspace.com.

If you need assistance, have a bug to report, or a product enhancement request, send email to support@componentspace.com.