ComponentSpace
SAML for ASP.NET
Configuration Guide

www.componentspace.com

Contents

Introduction

The ComponentSpace SAML for ASP.NET is a .NET framework class library that provides SAML v2.0 assertions, protocol messages, bindings and profiles functionality.

The primary SAML APIs, defined by the ISAMLIdentityProvider and ISAMLServiceProvider interfaces, make use of SAML configuration for various settings such as SAML provider names, X.509 certificates, URLs and various flags affecting processing. For more information about these APIs, refer to the SAML for ASP.NET Developer Guide.

SAML configuration may be specified either as XML or programmatically.

SAML Configuration Options

There are a number of options for specifying SAML configuration.

1. XML in a saml.config file
2. Programmatically through the SAML configuration API
3. Programmatically by implementing ISAMLConfigurationResolver

Using an XML configuration file is the simplest approach and requires no additional coding.

If SAML configuration information is stored in a database, it must be set programmatically.

If the SAML configuration changes infrequently, it may be set using the SAML configuration API, typically at application start-up.

If the SAML configuration changes frequently, it's better to implement the ISAMLConfigurationResolver interface for the on-demand retrieval of SAML configuration information.

SAML Configuration XML

SAML configuration may be specified as XML in an XML file (e.g. saml.config).

If not otherwise specified, the SAML configuration is expected to be in a saml.config file in the application's root folder.

A different file may be specified through the SAMLConfigFile app setting in web.config.

```
<add key="SAMLConfigFile" value="c:\config\saml.config"/>
```

Identity Provider Example Configuration

The following is an example of setting the identity provider configuration through XML.

```
<SAMLConfiguration xmlns="urn:componentspace:SAML:2.0:configuration">
 <IdentityProvider
  Name="https://ExampleIdentityProvider"
  Description="Example Identity Provider">
  <LocalCertificates>
   <Certificate FileName="Certificates\idp.pfx" Password="password"/>
  </LocalCertificates>
 </IdentityProvider>
```

```
  <PartnerServiceProviders>
    <!-- Web forms example -->
    <PartnerServiceProvider
      Name="https://ExampleServiceProvider"
      Description="Example Service Provider"
      WantAuthnRequestSigned="true"
      SignSAMLResponse="true"
      SignAssertion="false"
      EncryptAssertion="false"

AssertionConsumerServiceUrl="https://localhost:44338/SAML/AssertionConsumerService.aspx"
      SingleLogoutServiceUrl="https://localhost:44338/SAML/SLOService.aspx">
      <PartnerCertificates>
        <Certificate FileName="Certificates\sp.cer"/>
      </PartnerCertificates>
    </PartnerServiceProvider>
  </PartnerServiceProviders>
</SAMLConfiguration>
```

Service Provider Example Configuration

The following is an example of setting the service provider configuration through XML.

```
<SAMLConfiguration xmlns="urn:componentspace:SAML:2.0:configuration">
  <ServiceProvider
    Name="https://ExampleServiceProvider"
    Description="Example Service Provider"
    AssertionConsumerServiceUrl="~/SAML/AssertionConsumerService.aspx">
    <LocalCertificates>
      <Certificate FileName="Certificates\sp.pfx" Password="password"/>
    </LocalCertificates>
  </ServiceProvider>

  <PartnerIdentityProviders>
    <!-- Web forms example -->
    <PartnerIdentityProvider
      Name="https://ExampleIdentityProvider"
      Description="Example Identity Provider"
      SignAuthnRequest="true"
      SingleSignOnServiceUrl="https://localhost:44390/SAML/SSOService.aspx"
      SingleLogoutServiceUrl="https://localhost:44390/SAML/SLOService.aspx">
      <PartnerCertificates>
        <Certificate FileName="Certificates\idp.cer"/>
      </PartnerCertificates>
    </PartnerIdentityProvider>
  </PartnerIdentityProviders>
</SAMLConfiguration>
```

XML Schema

An XML schema file, saml-config-schema-v<version-number>.xsd, is included in the documentation folder (eg. saml-config-schema-v1.0.xsd).

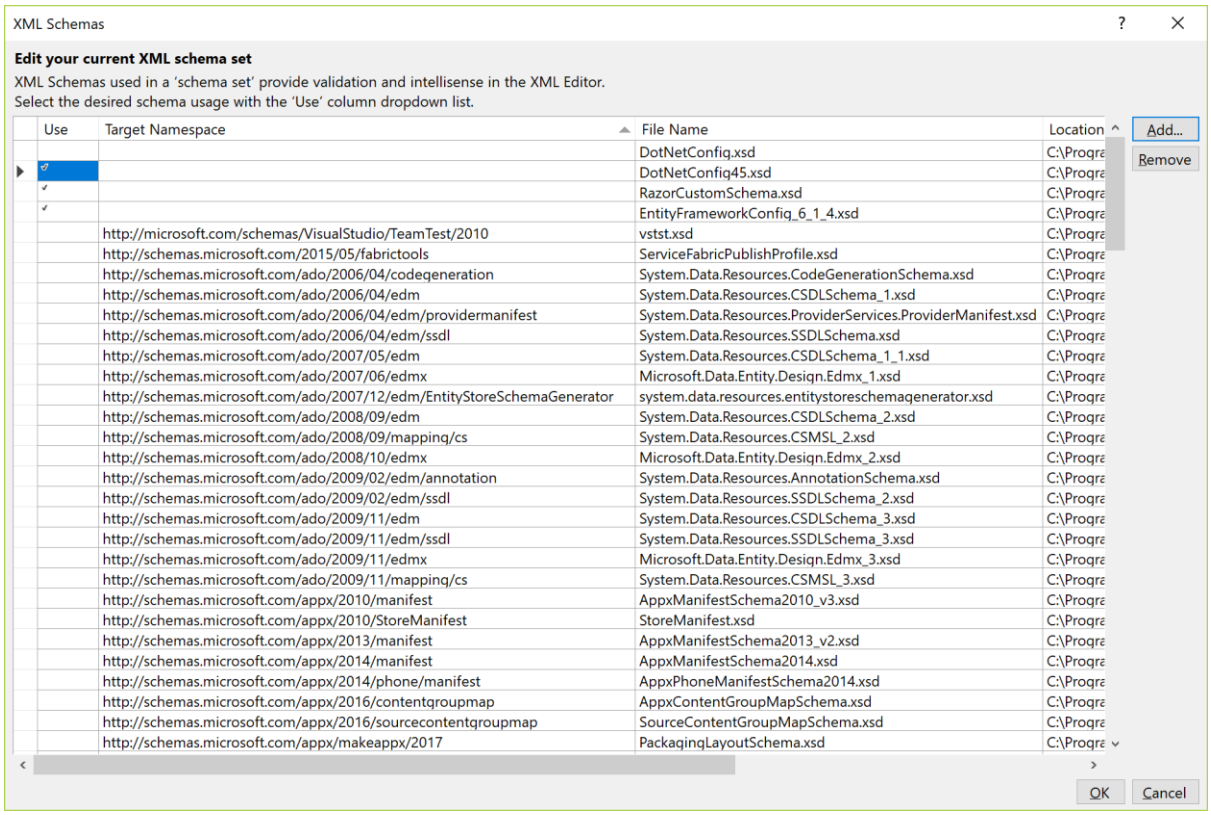
The corresponding file is also available under https://www.componentspace.com/schemas (e.g. https://www.componentspace.com/schemas/saml-config-schema-v1.0.xsd).

This may be used to enable Visual Studio Intellisense when editing SAML configuration XML or when using an XML schema validator.

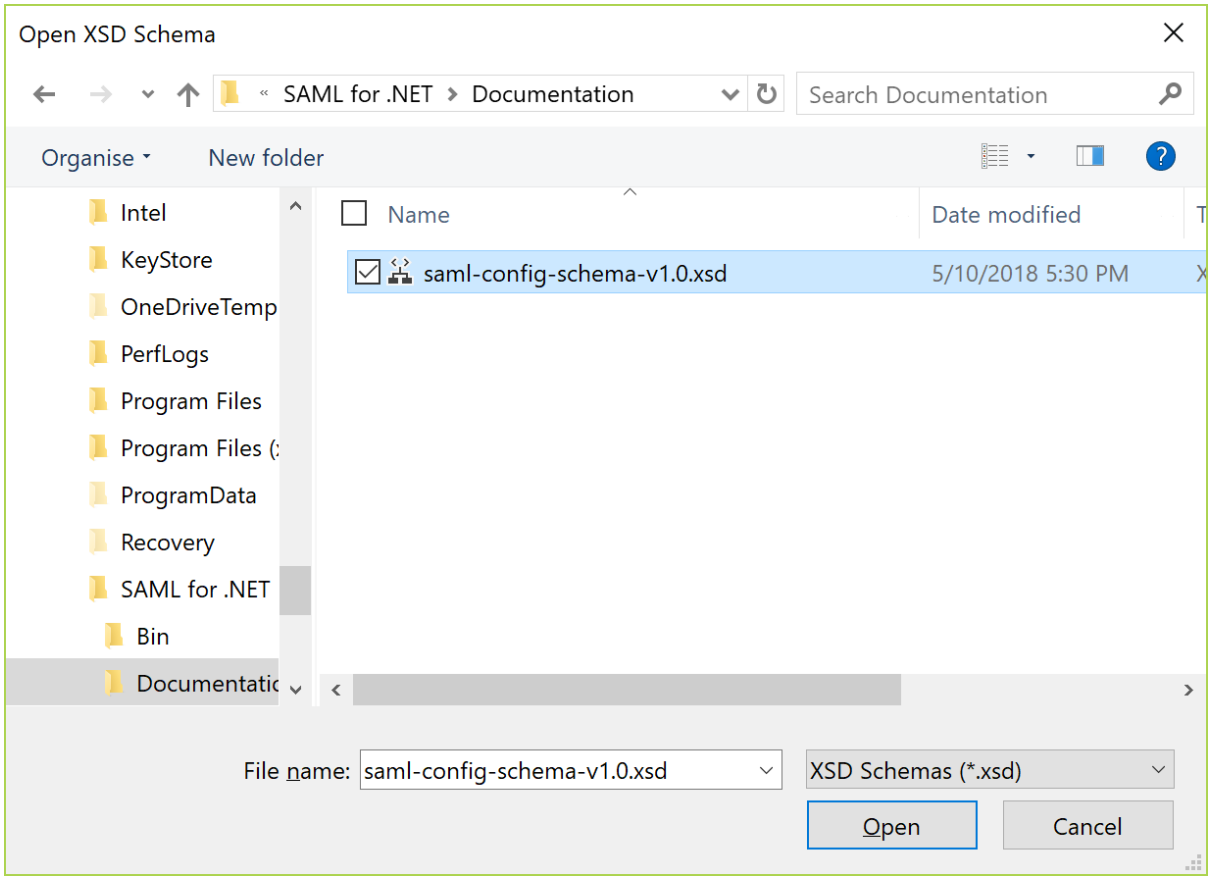
Enabling Visual Studio Intellisense

In Visual Studio, open a saml.config file in the XML editor. This should be the default editor for this file type.

In the properties window, edit the Schemas property.



Click the Add button and select the XML schema file.



Intellisense now should be enabled.

```
1  <?xml version="1.0"?>
2  <SAMLConfiguration xmlns="urn:componentspace:SAML:2.0:configuration">
3    <ServiceProvider
4      Name="https://ExampleServiceProvider"
5      Description="Example Service Provider"
6      AssertionConsumerServiceUrl="~/SAML/AssertionConsumerService.aspx">
7      <LocalCertificates>
8        <Certificate FileName="Certificates\sp.pfx" Password="password"/>
9        <Certificate />
10 FileName                                    ates>
11 Key                                         >
12 Password
13 SerialNumber                                roviders>
14 StoreLocation                               example -->
15 StoreName                                   yProvider
16 String                                      /ExampleIdentityProvider"
17 Subject                                     Example Identity Provider"
18 Thumbprint                                  est="true"
19                                             erviceUrl="https://localhost:44390/SAML/SSOService.aspx"
20                                             erviceUrl="https://localhost:44390/SAML/SLOService.aspx">
21       <PartnerCertificates>
22         <Certificate FileName="Certificates\idp.cer"/>
23       </PartnerCertificates>
24     </PartnerIdentityProvider>
```

Programmatically Specifying Configuration

In many scenarios, storing the SAML configuration as XML is the simplest and preferred approach.

However, there may be instances where this isn't the case. For example, the configuration may be stored in a custom database. Once retrieved, it would be set using the SAML configuration API.

In the application's start-up class, the SAML configuration is specified.

```
SAMLController.Configuration = samlConfiguration;
```

Identity Provider Example Configuration

The following is an example of setting the identity provider configuration programmatically.

Typically, rather than setting hard-coded values, these would be read from a custom database.

```
SAMLConfiguration samlConfiguration = new SAMLConfiguration()
{
   LocalIdentityProviderConfiguration = new LocalIdentityProviderConfiguration()
   {
      Name = "http://ExampleIdentityProvider",
      LocalCertificates = new List<CertificateConfiguration>()
      {
         new CertificateConfiguration()
         {
            FileName = @"certificates\idp.pfx",
            Password = "password"
         }
      }
   }
};

samlConfiguration.AddPartnerServiceProvider(
   new PartnerServiceProviderConfiguration()
   {
      Name = "http://ExampleServiceProvider",
      WantAuthnRequestSigned = true,
      SignSAMLResponse = true,
      AssertionConsumerServiceUrl =
"http://localhost:51901/SAML/AssertionConsumerService.aspx",
      SingleLogoutServiceUrl = "http://localhost:51901/SAML/SLOService.aspx",
      PartnerCertificates = new List<CertificateConfiguration>()
      {
         new CertificateConfiguration()
         {
            FileName = @"certificates\sp.cer"
         }
      }
   });

SAMLController.Configuration = samlConfiguration;
```

Service Provider Example Configuration

The following is an example of setting the service provider configuration programmatically.

Typically, rather than setting hard-coded values, these would be read from a custom database.

```
SAMLConfiguration samlConfiguration = new SAMLConfiguration()
{
   LocalServiceProviderConfiguration = new LocalServiceProviderConfiguration()
   {
      Name = "http://ExampleServiceProvider",
      AssertionConsumerServiceUrl = "~/SAML/AssertionConsumerService.aspx",
      LocalCertificates = new List<CertificateConfiguration>()
      {
         new CertificateConfiguration()
         {
            FileName = @"certificates\sp.pfx",
            Password = "password"
         }
      }
   }
};

samlConfiguration.AddPartnerIdentityProvider(
   new PartnerIdentityProviderConfiguration()
   {
      Name = "http://ExampleIdentityProvider",
      SignAuthnRequest = true,
      SingleSignOnServiceUrl = "http://localhost:51801/SAML/SSOService.aspx",
      SingleLogoutServiceUrl = "http://localhost:51801/SAML/SLOService.aspx",
      PartnerCertificates = new List<CertificateConfiguration>()
      {
         new CertificateConfiguration()
         {
            FileName = @"certificates\idp.cer",
         }
      }
   });

SAMLController.Configuration = samlConfiguration;
```

Updating Configuration

The following is an example of accessing and updating the SAML configuration.

```
SAMLController.Configuration.AddPartnerIdentityProvider(
   new PartnerIdentityProviderConfiguration()
   {
      Name = "http://ExampleIdentityProvider2",
      SignAuthnRequest = true,
      SingleSignOnServiceUrl = "http://localhost:51802/SAML/SSOService.aspx",
```

```
    SingleLogoutServiceUrl = "http://localhost:51802/SAML/SLOService.aspx",
    PartnerCertificates = new List<CertificateConfiguration>()
    {
      new CertificateConfiguration()
      {
        FileName = @"certificates\idp.2cer",
      }
    }
  });
```

Implementing ISAMLConfigurationResolver

The ISAMLConfigurationResolver interface provides an alternative mechanism for specifying SAML configuration. Rather than calling the SAML configuration API to specify configuration, the ISAMLConfigurationResolver interface is implemented to return SAML configuration as requested. This approach might be preferred to support very dynamic SAML configurations.

As a convenience, when not all interface methods are to be implemented, the AbstractSAMLConfigurationResolver class may be extended.

In the application's start-up class, the configuration resolver is registered.

```
SAMLController.ConfigurationResolver = new CustomConfigurationResolver();
```

Identity Provider Example Configuration

The following is an example of implementing ISAMLConfigurationResolver as the identity provider.

Typically, rather than setting hard-coded values, these would be read from a custom database.

```
using ComponentSpace.SAML2.Configuration.Resolver;

public class ExampleIdentityProviderConfigurationResolver : AbstractSAMLConfigurationResolver
{
  public override LocalIdentityProviderConfiguration
GetLocalIdentityProviderConfiguration(string configurationID)
  {
    return new LocalIdentityProviderConfiguration()
    {
      Name = "http://ExampleIdentityProvider",
      LocalCertificates = new List<CertificateConfiguration>()
      {
        new CertificateConfiguration()
        {
          FileName = @"certificates\idp.pfx",
          Password = "password"
        }
      }
    };
  }
```

```
   public override PartnerServiceProviderConfiguration
GetPartnerServiceProviderConfiguration(string configurationID, string partnerName)
   {
      return new PartnerServiceProviderConfiguration()
      {
         Name = "http://ExampleServiceProvider",
         WantAuthnRequestSigned = true,
         SignSAMLResponse = true,
         AssertionConsumerServiceUrl =
"http://localhost:51901/SAML/AssertionConsumerService.aspx",
         SingleLogoutServiceUrl = "http://localhost:51901/SAML/SLOService.aspx",
         PartnerCertificates = new List<CertificateConfiguration>()
         {
            new CertificateConfiguration()
            {
               FileName = @"certificates\sp.cer"
            }
         }
      };
   }
}
```

Service Provider Example Configuration

The following is an example of implementing ISAMLConfigurationResolver as the service provider.

Typically, rather than setting hard-coded values, these would be read from a custom database.

```
using ComponentSpace.SAML2.Configuration.Resolver;

public class ExampleServiceProviderConfigurationResolver : AbstractSAMLConfigurationResolver
{
   public override LocalServiceProviderConfiguration GetLocalServiceProviderConfiguration(string
configurationID)
   {
      return new LocalServiceProviderConfiguration()
      {
         Name = "http://ExampleServiceProvider",
         AssertionConsumerServiceUrl = "~/SAML/AssertionConsumerService.aspx",
         LocalCertificates = new List<CertificateConfiguration>()
         {
            new CertificateConfiguration()
            {
               FileName = @"certificates\sp.pfx",
               Password = "password"
            }
         }
      };
   }
```

```
   public override PartnerIdentityProviderConfiguration
GetPartnerIdentityProviderConfiguration(string configurationID, string partnerName)
   {
      return new PartnerIdentityProviderConfiguration()
      {
         Name = "http://ExampleIdentityProvider",
         SignAuthnRequest = true,
         SingleSignOnServiceUrl = "http://localhost:51801/SAML/SSOService.aspx",
         SingleLogoutServiceUrl = "http://localhost:51801/SAML/SLOService.aspx",
         PartnerCertificates = new List<CertificateConfiguration>()
         {
            new CertificateConfiguration()
            {
               FileName = @"certificates\idp.cer",
            }
         }
      };
   }
}
```

Multi-Tenancy Support

Multi-tenancy support refers to a single application accommodating multiple customers or tenants each of whom has their own separate SAML configuration.

For the majority of use cases, a single SAML configuration will suffice and multi-tenancy support is not required.

As with a single SAML configuration, multiple SAML configurations may be specified through XML, programmatically or via the ISAMLConfigurationResolver interface.

The following is an example outline of multiple SAML configurations.

```
<SAMLConfigurations xmlns="urn:componentspace:SAML:2.0:configuration">
 <SAMLConfiguration ID="tenant1">
  <ServiceProvider Name="SP1"/>
  <PartnerIdentityProviders>
   <PartnerIdentityProvider Name="IdP1"/>
   <PartnerIdentityProvider Name="IdP2"/>
  </PartnerIdentityProviders>
 </SAMLConfiguration>

 <SAMLConfiguration ID="tenant2">
  <ServiceProvider Name="SP2"/>
  <PartnerIdentityProviders>
   <PartnerIdentityProvider Name="IdP3"/>
   <PartnerIdentityProvider Name="IdP4"/>
  </PartnerIdentityProviders>
 </SAMLConfiguration>
</SAMLConfigurations>
```

The ID property uniquely identifies each of the SAML configurations.

Configuration Selection

Prior to processing SSO and SLO requests, a SAML configuration must be selected. This is done by setting the SAMLController.ConfigurationID property. Refer to the SAML for ASP.NET Developer Guide for more information.

The following example specifies the SAML configuration to use when processing the SAML response.

```
// Identify the tenant (application specific, details not shown).
var tenantID = GetTenantID();

// Specify the SAML configuration.
SAMLController.ConfigurationID = tenantID;

// Receive and process the SAML assertion contained in the SAML response.
SAMLServiceProvider.ReceiveSSO(…);
```

Identifying the Tenant

The application is responsible for identifying the tenant and therefore the ID to specify when calling SAMLController.ConfigurationID.

Possible methods include:

- Separate subdomain names for each tenant
- Query string parameter
- Special HTTP headers or cookies
- IP address ranges

SAMLConfigurations

The SAMLConfigurations class is the top-level class specifying the SAML configurations.

Configurations [required]

The Configurations is a list of one or more SamlConfiguration items. Each SamlConfiguration item corresponds to a tenant in a multi-tenancy application. In the more common single tenancy application, a single SamlConfiguration is defined.

ReloadOnConfigurationChange [optional]

The flag indicates whether the application should be reloaded if the configuration changes.

If the saml.config file changes, the application is reloaded to pick-up the new configuration.

The default is true.

ValidateMessagesAgainstSchema [optional]

The flag indicates whether SAML messages should be validated against the XML schema.

Validating SAML messages against the XML schema is good practice but does result in a small performance hit.

The default is false.

SAMLConfiguration

The SAMLConfiguration class specifies a single SAML configuration for a local identity provider or service provider.

ID [optional]

Each SAMLConfiguration is identified by a unique ID. This ID is internal to the configuration and is not exposed to partner providers.

An ID is only required if there are multiple SAML configurations.

LocalIdentityProviderConfiguration [optional]

The LocalIdentityProviderConfiguration specifies the local identity provider's configuration.

LocalServiceProviderConfiguration [optional]

The LocalServiceProviderConfiguration specifies the local service provider's configuration.

PartnerIdentityProviderConfigurations [optional]

The PartnerIdentityProviderConfigurations is the list of one or more PartnerIdentityProviderConfiguration items. Each PartnerIdentityProviderConfiguration specifies the configuration to participate in SSO with a partner identity provider.

PartnerServiceProviderConfigurations [optional]

The PartnerServiceProviderConfigurations is the list of one or more PartnerServiceProviderConfiguration items. Each PartnerServiceProviderConfiguration specifies the configuration to participate in SSO with a partner service provider.

LocalIdentityProviderConfiguration

The LocalIdentityProviderConfiguration specifies the configuration for the local identity provider.

Its base class is LocalProviderConfiguration.

SingleSignOnServiceUrl [optional]

The single sign-on service URL is the location of the local identity provider's SSO service where SAML authn requests are received as part of SP-initiated SSO.

If specified, it may be used to perform certain security checks as part of the SAML protocol.

Its use is optional but recommended.

LocalServiceProviderConfiguration

The LocalServiceProviderConfiguration specifies the configuration for the local service provider.

Its base class is LocalProviderConfiguration.

AssertionConsumerServiceUrl [optional]

The assertion consumer service URL is the location of the local service provider's ACS where SAML responses are received as part of SSO.

If specified, it may be used to perform certain security checks as part of the SAML protocol.

Its use is optional but recommended.

PartnerIdentityProviderConfiguration

The PartnerIdentityProviderConfiguration specifies the configuration for a partner identity provider.

Its base class is PartnerProviderConfiguration.

SingleSignOnServiceUrl [optional]

The single sign-on service URL is the location of the partner identity provider's SSO service where SAML authn requests are sent as part of SP-initiated SSO. If only IdP-initated SSO is supported, this URL may be omitted.

SingleSignOnServiceBinding [optional]

The single sign-on service binding specifies the transport mechanism (i.e. SAML binding) to use when sending SAML authn requests to the partner identity provider.

The binding options are:

- urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect
- urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST

The default is urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect.

SignAuthnRequest [optional]

The flag specifies whether SAML authn requests sent to the partner identity provider should be signed. Signing authn requests is recommended but optional.

The default is false.

ForceAuthn [optional]

The flag specifies whether the force authentication attribute in SAML authn requests sent to the partner identity provider should be set.

The default is false.

WantAssertionOrResponseSigned [optional]

The flag specifies whether either SAML responses or assertions received from the partner identity provider should be signed. If the flag is set and neither the SAML response nor SAML assertion is signed or the signature cannot be verified, this is considered an error.

Signing ensures the identity of the sender and the integrity of the content. Signatures will be generated by the partner provider with its private key and verified by the local provider with the partner provider's public key.

Generally, it doesn't matter whether the SAML response or assertion is signed. The payload of the SAML response is the SAML assertion so signing the SAML response includes the SAML assertion.

It's recommended that WantAssertionOrResponseSigned is set to true.

The default is true.

WantSAMLResponseSigned [optional]

The flag specifies whether SAML responses received from the partner identity provider should be signed. If the flag is set and either the SAML response isn't signed or the signature cannot be verified, this is considered an error.

Signing ensures the identity of the sender and the integrity of the content. Signatures will be generated by the partner provider with its private key and verified by the local provider with the partner provider's public key.

It's recommended that WantAssertionOrResponseSigned, WantSAMLResponseSigned or WantAssertionSigned is set to true.

The default is false.

WantAssertionSigned [optional]

The flag specifies whether SAML assertions received from the partner identity provider should be signed. If the flag is set and either the SAML assertion isn't signed or the signature cannot be verified, this is considered an error.

Signing ensures the identity of the sender and the integrity of the content. Signatures will be generated by the partner provider with its private key and verified by the local provider with the partner provider's public key.

It's recommended that WantAssertionOrResponseSigned, WantSAMLResponseSigned or WantAssertionSigned is set to true.

The default is false.

WantAssertionEncrypted [optional]

The flag specifies whether SAML assertions received from the partner identity provider should be encrypted. If the flag is set and either the SAML assertion isn't encrypted or cannot be decrypted, this is considered an error.

Encrypting ensures the privacy of the content. Assertions will be encrypted by the partner provider with the local provider's public key and decrypted by the local provider with its private key.

If the SAML assertion includes sensitive information it's recommended that it's encrypted. This SAML assertion encryption is in addition to the privacy provided at the transport layer when using the recommended HTTPS protocol. In many scenarios, encryption of the assertion is not required.

The default is false.

ProviderName [optional]

The provider name is included in the SAML authn requests sent to the partner identity provider.

AuthnContextComparison [optional]

The comparison method is included in the SAML authn requests sent to the partner identity provider.

The comparison method is used to evaluate the requested contexts.

The comparison methods are:

- exact
- minimum
- maximum
- better

The default is to not include a comparison which is equivalent to specifying exact.

OverridePendingAuthnRequest [optional]

The flag specifies whether a pending authentication request sent as part of SP-initiated SSO may be overridden and an IdP-initiated SAML response received.

If a local service provider sends an authentication request it expects the SAML response it receives to come from the partner identity provider it sent the authentication request to and that the SAML response is in response to this authentication request.

If a different identity provider sends a SAML response or the expected identity provider sends a SAML response but it is not in response to this authentication request, this is considered an error.

Setting the flag to true overrides this security check and permits SP-initiated SSO to be interrupted by IdP-initiated SSO.

The default is false.

DisableIdPInitiatedSso [optional]

The flag specifies whether IdP-initiated SSO is supported.

Both IdP-initiated and SP-initiated SSO are supported.

Setting the flag to true disables IdP-initiated SSO.

The default is false.

DisableAssertionReplayCheck [optional]

The flag specifies whether checks for SAML assertion replay attacks are disabled.

Each SAML assertion includes a unique ID. A cache of received SAML assertion IDs is maintained and if an ID matches a previously received ID this is considered an error.

Setting the flag to true disables this check.

The default is false.

DisableRecipientCheck [optional]

A SAML assertion may include a subject confirmation recipient URI. This identifies the intended recipient of the SAML assertion. If included it should match the service provider's assertion consumer service URL specified by the AssertionConsumerServiceUrl configuration property.

Setting the flag to true disables this check.

The default is false.

DisableTimePeriodCheck [optional]

A SAML assertion may include attributes identifying a time period in which the SAML assertion is valid. If included the time at which the SAML assertion is received should be within this time period.

Setting the flag to true disables this check.

The default is false.

DisableAudienceRestrictionCheck [optional]

A SAML assertion may include an audience restriction URI. This identifies the intended recipient of the SAML assertion. If included it should match the service provider's name.

Setting the flag to true disables this check.

The default is false.

DisableAuthnContextCheck [optional]

A SAML assertion may include an authentication context. This identifies the mechanism by which the user was authenticated at the identity provider. For example, if the user was authenticated by password, the authentication context would be "urn:oasis:names:tc:SAML:2.0:ac:classes:Password". If included it should match the authentication context class specified by the optional ExpectedAuthnContext configuration property.

Setting the flag to true disables this check.

The default is false.

PartnerServiceProviderConfiguration

The PartnerServiceProviderConfiguration specifies the configuration for a partner service provider.

Its base class is PartnerProviderConfiguration.

AssertionConsumerServiceUrl [optional]

The assertion consumer service URL is the location of the partner service provider's ACS where SAML responses are sent as part of SSO.

WantAuthnRequestSigned [optional]

The flag specifies whether SAML authn requests received from the partner service provider should be signed. Receiving signed authn requests is recommended but optional.

The default is false.

SignSAMLResponse [optional]

The flag specifies whether SAML responses sent to the partner service provider should be signed.

Signing ensures the identity of the sender and the integrity of the content. Signatures will be generated by the local provider with its private key and verified by the partner provider with the local provider's public key.

It's recommended that either SignSAMLResponse or SignAssertion is set to true.

The default is false.

SignAssertion [optional]

The flag specifies whether SAML assertions sent to the partner service provider should be signed.

Signing ensures the identity of the sender and the integrity of the content. Signatures will be generated by the local provider with its private key and verified by the partner provider with the local provider's public key.

It's recommended that either SignSAMLResponse or SignAssertion is set to true.

The default is false.

EncryptAssertion [optional]

The flag specifies whether SAML assertions sent to the partner service provider should be encrypted.

Encrypting ensures the privacy of the content. Assertions will be encrypted by the local provider with the partner provider's public key and decrypted by the partner provider with its private key.

If the SAML assertion includes sensitive information, it's recommended that it's encrypted. This SAML assertion encryption is in addition to the privacy provided at the transport layer when using the recommended HTTPS protocol. In many scenarios, encryption of the assertion is not required.

The default is false.

AssertionLifeTime [optional]

The assertion lifetime specifies the time span for which the SAML assertion is valid. It is the current UTC time plus or minus the assertion lifetime time span.

The time span should be kept short but not so short as to cause issues when server clocks are not synchronized exactly.

The default is 3 minutes.

RelayState [optional]

The relay state is sent as part of IdP-initiated SSO and specifies the URL the service provider should redirect to once SSO completes.

LocalProviderConfiguration

The LocalProviderConfiguration is an abstract base class.

Its base class is ProviderConfiguration.

SingleLogoutServiceUrl [optional]

The single logout service URL is the location of the local provider's SLO service where SAML logout messages are received. If SLO is not supported, this URL may be omitted.

If specified, it may be used to perform certain security checks as part of the SAML protocol.

Its use is optional but recommended.

ResolveToHttps [optional]

The flag specifies whether local URLs should be resolved to HTTPS.

This is useful when using an SSL terminating device such as a load balancer.

For example, if true, a local URL of http://www.sp.com/SAML/AssertionConsumerService.aspx would be resolved to https://www.sp.com/SAML/AssertionConsumerService.aspx when included in the SAML authn request sent to the identity provider.

The default is true.

PartnerProviderConfiguration

The PartnerProviderConfiguration is an abstract base class.

Its base class is ProviderConfiguration.

SingleLogoutServiceUrl [optional]

The single logout service URL is the location of the partner provider's SLO service where SAML logout messages are sent. If SLO is not supported, this URL may be omitted.

SingleLogoutServiceResponseUrl [optional]

The single logout service response URL is the location of the partner provider's SLO service where SAML logout responses are sent. If SLO is not supported or the same partner provider endpoint receives logout requests and responses, this URL may be omitted.

SingleLogoutServiceBinding [optional]

The single logout service binding specifies the transport mechanism (i.e. SAML binding) to use when sending SAML logout messages to the partner provider.

The binding options are:

- urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect
- urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST

The default is urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect.

LogoutRequestLifeTime [optional]

The assertion lifetime specifies the time span for which the SAML logout request is valid. It is the current UTC time plus or minus the logout request lifetime time span.

The time span should be kept short but not so short as to cause issues when server clocks are not synchronized exactly.

The default is 3 minutes.

SignLogoutRequest [optional]

The flag specifies whether SAML logout requests sent to the partner provider should be signed.

Signing ensures the identity of the sender and the integrity of the content. Signatures will be generated by the local provider with its private key and verified by the partner provider with the local provider's public key.

The default is false.

SignLogoutResponse [optional]

The flag specifies whether SAML logout responses sent to the partner provider should be signed.

Signing ensures the identity of the sender and the integrity of the content. Signatures will be generated by the local provider with its private key and verified by the partner provider with the local provider's public key.

The default is false.

WantLogoutRequestSigned [optional]

The flag specifies whether SAML logout requests received from the partner provider should be signed. If the flag is set and either the logout request isn't signed or the signature cannot be verified, this is considered an error.

Signing ensures the identity of the sender and the integrity of the content. Signatures will be generated by the partner provider with its private key and verified by the local provider with the partner provider's public key.

The default is false.

WantLogoutResponseSigned [optional]

The flag specifies whether SAML logout responses received from the partner provider should be signed. If the flag is set and either the logout response isn't signed or the signature cannot be verified, this is considered an error.

Signing ensures the identity of the sender and the integrity of the content. Signatures will be generated by the partner provider with its private key and verified by the local provider with the partner provider's public key.

The default is false.

EncryptLogoutNameID [optional]

The flag specifies whether Name IDs in logout requests sent to the partner provider should be encrypted.

Encrypting ensures the privacy of the content. Name IDs will be encrypted by the local provider with the partner provider's public key and decrypted by the partner provider with its private key.

If the Name ID is sensitive, it's recommended that it's encrypted. This Name ID encryption is in addition to the privacy provided at the transport layer when using the recommended HTTPS protocol. In many scenarios, encryption of the Name ID is not required.

The default is false.

IssuerFormat [optional]

The issuer format specifies the format of the issuer field included in SAML messages.

NameIDFormat [optional]

The name ID format specifies the format of the name identifier. For a local identity provider, the format is included with the SAML assertion name identifier. For a local service provider, the format is included with the SAML authentication request name identifier policy.

DigestMethod [optional]

The digest method specifies how to generate the digest for XML signatures.

The supported digest methods are:

- http://www.w3.org/2000/09/xmldsig#sha1
- http://www.w3.org/2001/04/xmlenc#sha256
- http://www.w3.org/2001/04/xmldsig-more#sha384
- http://www.w3.org/2001/04/xmlenc#sha512

The default is http://www.w3.org/2001/04/xmlenc#sha256.

SignatureMethod [optional]

The signature method specifies how to generate XML signatures.

The supported signature methods are:

- http://www.w3.org/2000/09/xmldsig#rsa-sha1
- http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
- http://www.w3.org/2001/04/xmldsig-more#rsa-sha384
- http://www.w3.org/2001/04/xmldsig-more#rsa-sha512

- http://www.w3.org/2007/05/xmldsig-more#sha1-rsa-MGF1
- http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1
- http://www.w3.org/2007/05/xmldsig-more#sha384-rsa-MGF1
- http://www.w3.org/2007/05/xmldsig-more#sha512-rsa-MGF1

- http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha1
- http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256
- http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha384
- http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512

The default is http://www.w3.org/2001/04/xmldsig-more#rsa-sha256.

WantDigestMethod [optional]

The digest method specifies the required digest method of received XML signatures.

Refer to DigestMethod for valid values.

If unspecified, any digest method is permitted.

WantSignatureMethod [optional]

The signature method specifies the required signature method of received XML signatures.

Refer to SignatureMethod for valid values.

If unspecified, any signature method is permitted.

KeyEncryptionMethod [optional]

The key encryption method specifies how to encrypt the symmetric key used in XML encryption.

The supported key encryption methods are:

- http://www.w3.org/2001/04/xmlenc#rsa-1_5
- http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p

The default is http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p.

DataEncryptionMethod [optional]

The data encryption method specifies how to encrypt the data in XML encryption.

The supported data encryption methods are:

- http://www.w3.org/2001/04/xmlenc#tripledes-cbc
- http://www.w3.org/2001/04/xmlenc#aes128-cbc
- http://www.w3.org/2001/04/xmlenc#aes192-cbc
- http://www.w3.org/2001/04/xmlenc#aes256-cbc

The default is http://www.w3.org/2001/04/xmlenc#aes256-cbc.

ClockSkew [optional]

The clock skew specifies the time span to allow for differences between local and partner computer clocks when checking time intervals.

The time span should be kept short but not so short as to cause issues when server clocks are not synchronized exactly.

The default is 3 minutes.

AuthnContext [optional]

The authentication context specifies the mechanism by which the user was authenticated at the identity provider.

For example, if the user was authenticated by password, the authentication context would be “urn:oasis:names:tc:SAML:2.0:ac:classes:Password”.

The authentication context is included with the SAML assertion authentication statement.

UseEmbeddedCertificate [optional]

The flag specifies whether to use the X.509 certificate embedded in the XML signature when verifying the signature.

If the embedded certificate is used, no assumptions can be made about the identity of the sender.

Embedded certificates should not be used in production.

The default is false.

DisableDestinationCheck [optional]

A SAML message may include a destination URI identifying the address to which the message has been sent. If included it should match the provider's URL where the message was received.

For example, for a SAML response the destination should be the local service provider's assertion consumer service URL as specified by the AssertionConsumerServiceUrl configuration property.

Setting the flag to true disables this check.

The default is false.

DisableInboundLogout [optional]

Setting the flag to true disables inbound SAML logout requests.

The default is false.

DisableOutboundLogout [optional]

Setting the flag to true disables outbound SAML logout requests.

The default is false.

DisableInResponseToCheck [optional]

All SAML messages includes a unique ID. SAML responses that are in response to a particular SAML request include an in-response-to attribute identifying the SAML request.

Setting the flag to true disables checking to ensure the in-response-to attribute is present and correct.

The default is false.

DisablePendingLogoutCheck [optional]

If a SAML logout response is received without having previously sent a logout request, this is considered an error.

Setting the flag to true disables this check.

The default is false.

DisableLogoutResponseStatusCheck [optional]

If a SAML logout response is received with an error status, this is considered an error.

Setting the flag to true disables this check.

The default is false.

PartnerCertificates [optional]

Zero or more **Certificate** configuration entries identifying partner X.509 certificates that are issued to the partner provider and used by the local provider. Partner certificates do not include private keys. Typically, only a single certificate is specified. If more than one certificate is specified and a security operation using the certificates fails, the operation is retried using the next certificate in the list until either successful or all certificates have been tried.

As an example, if the SAML assertion received by the local service provider is signed each partner certificate is used in an attempt to verify the signature.

Multiple partner certificates support scenarios including the phased rollover of expired certificates.

ProviderConfiguration

The ProviderConfiguration specifies properties common to all local and partner providers.

Name [required]

All local and partner providers must have a unique name. Partner providers will supply their names. Local names should be universally unique and, for maximum interoperability, be in the form of a URL. The URL doesn't have to locate a resource but it's common for it to point to the home page of the web application or the download link to the local provider's SAML metadata.

The name corresponds to the entity ID, if SAML metadata is used.

Description [optional]

The description is purely for documentation and is not part of SAML SSO.

LocalCertificates [optional]

Zero or more **Certificate** configuration entries identifying local X.509 certificates that are issued to and used by the local provider. Local certificates must include private keys. Typically, only a single certificate is specified. If more than one certificate is specified and a security operation using the certificates fails, the operation is retried using the next certificate in the list until either successful or all certificates have been tried.

As an example, if the SAML assertion received by the local service provider is encrypted each local certificate is used in an attempt to decrypt the SAML assertion.

Multiple local certificates support scenarios including the staggered rollover of expired certificates.

CertificateConfiguration

The CertificateConfiguration identifies an X.509 certificate stored as a string, in a file, in the Windows certificate store, or elsewhere in the configuration.

If the certificate is stored as a string, the certificate base-64 encoded string must be specified and, if the certificate includes a private key, the password.

If the certificate is stored in a file, the file name must be specified and, if the file includes a private key, the password.

If the certificate is stored in the Windows certificate store, the store name and location must be specified as well as the certificate's serial number, thumbprint or subject name.

If the certificate is stored elsewhere in the configuration, the configuration key must be specified. This may be used to retrieve certificates stored in an Azure key vault

Use [optional]
Specifies the certificate's use. It can be:

- Encryption

- Signature
- Any

The default is Any.

String [optional]
The string is the base-64 encoded X.509 certificate.

Key [optional]
The key identifies the X.509 certificate stored in the app settings. For certificates in an Azure key vault, the certificate is specified by its configuration key.

FileName [optional]
The file name is the relative or absolute path to the X.509 certificate file.

Password [optional]
The password protects the private key.

StoreLocation [optional]
For certificates in a Windows certificate store, the store location specifies the location.

The store location may be:

- CurrentUser
- LocalMachine

The default is the local machine.

StoreName [optional]
For certificates in a Windows certificate store, the store name specifies the store.

The store name may be:

- AddressBook
- AuthRoot
- CertificateAuthority
- Disallowed
- My
- Root
- TrustedPeople
- TrustedPublisher

The default is the My (i.e. personal) store.

SerialNumber [optional]
For certificates in a Windows certificate store, the certificate is specified by its serial number.

Thumbprint [optional]
For certificates in a Windows certificate store, the certificate is specified by its thumb print.

SubjectName [optional]
For certificates in a Windows certificate store, the certificate is specified by its subject name.

URL

Local and partner provider URLs may be absolute or relative.

URLs are relative to the host name and port number of the current HTTP request.

For example, an assertion consumer service URL may be specified absolutely.

https://localhost:44360/SAML/AssertionConsumerService

Alternatively, it may be specified as a path.

~/SAML/AssertionConsumerService

This is converted to an absolute URL using the base URL of the current HTTP request.

Although the more common use case it to specify relative local URLs, relative partner URLs may be specified if, for example, the local and partner provider are installed on the same server.

Creating SAML Configuration

The CreateConfiguration console application project may be used to generate SAML configuration.

It may be used to generate SAML configuration for the local identity provider or service provider.

CreateConfiguration may be run as follows.

```
CreateConfiguration.exe
```

It will prompt for various input required to generate the SAML configuration.

The prompts will vary depending on whether identity provider or service provider configuration is to be generated.

Creating Local Identity Provider Configuration

Create Identity Provider or Service Provider configuration (IdP | SP):

Specify identity provider (IdP) configuration is to be generated.

Name:

Specify a name that uniquely identifies the local identity provider.

For maximum compatibility, a URL is recommended.

For example, it could be the URL of the web site or application although it doesn't necessarily have to point to a web resource.

Single Sign-On Service URL [None]:

Specify the single sign-on service URL.

This is the identity provider endpoint that will receive SAML authn requests.

If SP-initiated SSO will not be supported, this input is not required.

Single Logout Service URL [None]:

Specify the single logout service URL.

This is the identity provider endpoint that will receive SAML logout messages.

If SAML logout will not be supported, this input is not required.

X.509 signature certificate PFX file [None]:

Specify the path to the X.509 certificate file (i.e. PFX file) whose private key will be used for generating signatures.

The identity provider should sign either the SAML response or assertion and so a signature certificate PFX normally is required.

X.509 certificate PFX password [None]:

Specify the password that protects the PFX file.

SAML configuration file [saml.config]:

Specify the file where the generated SAML configuration will be saved.

Creating Local Service Provider Configuration

Create Identity Provider or Service Provider configuration (IdP | SP):

Specify service provider (SP) configuration is to be generated.

Name:

Specify a name that uniquely identifies the local service provider.

For maximum compatibility, a URL is recommended.

For example, it could be the URL of the web site or application although it doesn't necessarily have to point to a web resource.

Assertion Consumer Service URL [None]:

Specify the assertion consumer service URL.

This is the service provider endpoint that will receive SAML responses.

Normally this input should be specified.

Single Logout Service URL [None]:

Specify the single logout service URL.

This is the service provider endpoint that will receive SAML logout messages.

If SAML logout will not be supported, this input is not required.

X.509 signature certificate PFX file [None]:

Specify the path to the X.509 certificate file (i.e. PFX file) whose private key will be used for generating signatures.

If SAML messages will be signed a signature certificate PFX is required.

X.509 certificate PFX password [None]:

Specify the password that protects the PFX file.

SAML configuration file [saml.config]:

Specify the file where the generated SAML configuration will be saved.

SAML Metadata

SAML configuration is different from SAML metadata.

SAML metadata is defined by the SAML v2.0 specification as a standard format for exchanging configuration information between SAML providers.

SAML configuration includes enough information to implement SAML SSO at the local provider.

The SAML for ASP.NET Metadata Guide describes how to generate, import and export metadata.