Sustainsys.Saml2 Documentation

Anders Abel

Sep 19, 2023

Saml2

The Sustainsys.Saml2 library adds SAML2P support to ASP.NET web sites, allowing the web site to act as a SAML2 Service Provider (SP). The library was previously named Kentor.AuthServices. Sustainsys.Saml2 is open sourced and contributions are welcome, please see *contributing guidelines* for info on coding standards etc.

CHAPTER 1

Using Sustainsys.Saml2

Using the Sustainsys.Saml2 library to add SAML2P support into your ASP.NET web applications is a two-step process:

1. Reference the Nuget package
2. Provide the necessary configuration information

The exact nature of these steps depends on the ASP.NET integration you're after. See *Getting Started* for all the details.

CHAPTER 2

Versions

There are two supported major version of the library.

1.X is built for .Net Framework only. It is fully supported with security updates being released if needed, but no new functionality is added. The latest 1.X version is recommended for anyone upgrading from previous versions (including from Kentor.AuthServices).

2.X is built for both .Net Framework and .Net Core and is where new features are added. For new implementations 2.X versions are recommended.

CHAPTER 3

Licensing

The library is licensed under the MIT license.

The library was previously (before version 2.0.0) licensed under the GNU Lesser General Public License (LPGL). Please note that the license change only applies to the new versions, the old versions are still under the LGPL license.

3.1 Getting Started

See the sections below which contain information that will help you get started adding SAML2P support into your flavor of ASP.NET.

If you have gotten the appropriate Nuget package installed and then completed the configuration described below and are having any trouble, make sure to check out the *Troubleshooting* for assistance.

A sample SAML identity provider is available to further assist you in getting started if you don't already have a SAML identity provider that you can test with. You can access it directly at https://stubidp.sustainsys.com, or you can download the solution to run it locally yourself (it's a project within the Sustainsys.Saml2 github repository).

3.1.1 ASP.NET Web Forms

The `Saml2AuthenticationModule` provides Saml2 authentication to IIS web sites. In many cases it should just be *configured* in the `web.config` file and work without any code written in the application at all

Nuget Package to use: Sustainsys.Saml2.HttpModule

See *Configuration* for information about how to configure the `web.config` file.

3.1.2 ASP.NET MVC

The `MVC` package contains an MVC controller that will be accessible in your application just by installing the package in the application. For MVC applications a controller is preferred over using the authentication module as it integrates with MVC's error handling.

Nuget Package to use: Sustainsys.Saml2.Mvc

See *Configuration* for information about how to configure the web.config file.

3.1.3 Owin Middleware

The Owin middleware is modeled after the external authentication modules for social login (such as Google, Facebook, Twitter). This allows easy integration with ASP.NET Identity for keeping application specific user and role information.

Nuget Package to use: Sustainsys.Saml2.Owin

See the *Owin Middleware* page for information on how to set up and use the middleware.

3.1.4 ASP.NET Core 2 Handler

The ASP.NET Core 2 Handler is compatbile with the ASP.NET Core 2.X and 3.X authentication model.

Nuget Package to use: Sustainsys.Saml2.AspNetCore2

3.1.5 IdentityServer4 Integration

If you're using IdentityServer, you may want to configure SAML identity providers like Okta or Ping as external identity providers within your IdentityServer implementation.

The ASP.NET Core2 module enable SAML identity providers to be integrated within IdentityServer4 packages.

Nuget Package for IdentityServer4: Sustainsys.Saml2.AspNetCore2

Note: There is also a Sustainsys.Saml2 Nuget package, but this only contains functionality shared across the packages above and is not meant to be referenced directly in other projects.

Note: The protocol handling classes are available as a public API as well, making it possible to reuse some of the internals for writing your own service provider or identity provider.

3.2 Configuration

To use Sustainsys.Saml2 in an application and configure it in web.config (which is the default for the HttpModule and MVC libraries) it must be **enabled** in the application's web.config. The sample applications contains complete working web.config examples. For ASP.NET MVC applications see this working web.config example.

Note: Applications using the Owin library usually make their configuration in code and in that case no web.config changes are needed. If an Owin library is set up to use web.config (by passing true to the Saml2AuthenticationOptions constructor) the information here applies.

Note: Applications on Asp.Net Core do not support web.config. Use the Saml2Options class directly.

3.2.1 Config Sections

Three new config sections are required. Add these under `configuration/configSections`. Each of the sections will be a child element of the main `configuration` section and each is described below.

```
<configSections>
    <!-- Add these sections below any existing. -->
    <section name="system.identityModel" type="System.IdentityModel.Configuration.
↪SystemIdentityModelSection, System.IdentityModel, Version=4.0.0.0, Culture=neutral,␣
↪PublicKeyToken=B77A5C561934E089" />
    <section name="system.identityModel.services" type="System.IdentityModel.Services.
↪Configuration.SystemIdentityModelServicesSection, System.IdentityModel.Services,␣
↪Version=4.0.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089" />
    <section name="sustainsys.saml2" type="Sustainsys.Saml2.Configuration.
↪SustainsysSaml2Section, Sustainsys.Saml2"/>
</configSections>
```

3.2.2 Loading Modules

When using the `HttpModule` and the `MVC` controller, the `SessionAuthenticationModule` needs to be loaded and if using the http module that needs to be loaded as well. The `Owin` package does not need any http modules, please see the separate info on the *Owin Middleware*:.

```
<system.webServer>
    <modules>
        <!-- Add these modules below any existing. The SessionAuthenticatioModule
            must be loaded before the Saml2AuthenticationModule -->
        <add name="SessionAuthenticationModule" type="System.IdentityModel.Services.
↪SessionAuthenticationModule, System.IdentityModel.Services, Version=4.0.0.0,␣
↪Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
        <!-- Only add the Saml2AuthenticationModule if you're using the Sustainsys.
↪Saml2.HttpModule
            library. If you are using Sustainsys.Saml2.Mvc you SHOULD NOT load this␣
↪module.-->
        <add name="Saml2AuthenticationModule" type="Sustainsys.Saml2.HttpModule.
↪Saml2AuthenticationModule, Sustainsys.Saml2.HttpModule"/>
    </modules>
</system.webServer>
```

3.2.3 Sustainsys.Saml2 Section

The `sustainsys.saml2` section contains the configuration of the Sustainsys.Saml2 library. It is required for the http module and the mvc controller. The Owin middleware can read web.config, but can also be configured from code (see *Owin middleware*).

A sample section is shown below. For full details and all avaialble options, see *sustainsys.saml2*.

```
<sustainsys.saml2 entityId="http://localhost:17009"
                  returnUrl="http://localhost:17009/SamplePath/"
                  discoveryServiceUrl="http://localhost:52071/DiscoveryService"
                  authenticateRequestSigningBehavior="Always">
    <nameIdPolicy allowCreate="true" format="Persistent"/>
    <metadata cacheDuration="PT42S" validDuration="7.12:00:00" wantAssertionsSigned=
↪"true">
```

(continues on next page)

(continued from previous page)

```
        <organization name="Sustainsys AB" displayName="Sustainsys" url="https://www.Sustainsys.com" language="sv" />
        <contactPerson type="Other" email="info@Sustainsys.se" />
        <requestedAttributes>
        <add friendlyName ="Some Name" name="urn:someName" nameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true" />
        <add name="Minimal" />
        </requestedAttributes>
    </metadata>
    <identityProviders>
        <add entityId="https://stubidp.sustainsys.com/Metadata"
            signOnUrl="https://stubidp.sustainsys.com"
            allowUnsolicitedAuthnResponse="true"
            binding="HttpRedirect"
            wantAuthnRequestsSigned="true">
        <signingCertificate storeName="AddressBook" storeLocation="CurrentUser"
                            findValue="Sustainsys.Saml2.StubIdp" x509FindType="FindBySubjectName" />
        </add>
        <add entityId="example-idp"
            metadataLocation="https://idp.example.com/Metadata"
            allowUnsolicitedAuthnResponse="true"
            loadMetadata = "true" />
    </identityProviders>
    <!-- Optional configuration for signed requests. Required for Single Logout. -->
    <serviceCertificates>
        <add fileName="~/App_Data/Sustainsys.Saml2.Tests.pfx" />
    </serviceCertificates>
    <!-- Optional configuration for fetching IDP list from a federation -->
    <federations>
        <add metadataLocation="https://federation.example.com/metadata.xml" allowUnsolicitedAuthnResponse = "false" />
    </federations>
</sustainsys.saml2>
```

3.2.4 System.IdentityModel Section

There must be a `<system.identityModel>` section in the config file or there will be a runtime error. The section can be empty (use `<system.identityModel />`).

```
<system.identityModel />
```

3.2.5 System.IdentityModel.Services Section

The `<system.identityModel.services>` element configures the built in services. For testing on non ssl sites, the requirement for ssl for the session authentication cookie must be disabled.

Danger: It is a severe security risk to leave the `requireSsl` setting as false in a production environment.

```
<system.identityModel.services>
    <federationConfiguration>
```

(continues on next page)

(continued from previous page)

```
        <cookieHandler requireSsl ="false"/>
    </federationConfiguration>
</system.identityModel.services>
```

3.3 Owin Middleware

The Sustainsys Saml2 Owin middleware is designed to be used with an Owin authentication pipeline and is compatible with ASP.NET Identity. Sustainsys Saml2 provides external login in the same way as the built-in Google, Facebook and Twitter providers.

To use the Sustainsys Saml2 middleware, it needs to be configured in `Startup.Auth.Cs`.

```
app.UseSaml2Authentication(new Saml2AuthenticationOptions());
```

The `Saml2AuthenticationOptions` class only contains the Owin-specific configuration (such as the name used to identify the login provider). The rest of the configuration is read from the web.config/app.config and *configured in the same way* as when using the http module or the MVC controller.

If you would like to provide the Saml2-related configuration in code, specify `false` for the `loadConfiguration` constructor parameter and then build the options based on your own logic. For example:

```
var mySaml2Options = new Saml2AuthenticationOptions(false)
// more logic to set SPOptions, etc.
app.UseSaml2Authentication(mySaml2Options);
```

You can see a full example of this in the SampleOwinApplication project included in the source code. See the `Startup.Auth.cs` file.

3.3.1 Selecting Idp

An Owin-based application issues an `AuthenticationResponseChallenge` to ask the middleware to begin the authentication procedure. In that challenge, there is a properties dictionary. To use a specified idp, the entity id of the idp should be entered in that dictionary under the key "idp".

In a typical MVC application that requires some changes to the generated code to enable passing a property to the `AuthenticationProperties` dictionary.

Another, more simple way to pass a value is to put it directly in the Owin environment dictionary under the key "saml2.idp".

Here's an example of how to set the Owin environment value through ASP.NET MVC:

```
var context = HttpContext.GetOwinContext();
context.Environment.Add("saml2.idp", new EntityId(YOUR_IDP_ENTITY_ID));
```

3.3.2 Module Path and Metadata

By default the module path is */Saml2* but you can specify a different modulepath in your SPOptions object mentioned above.

The metadata URL is the root of this module path.

3.4 Troubleshooting

If you're having trouble - don't give up! :)

The items below may point you in the right direction.

- Check the issues archive.
- Check the SAML2 specification, starting with the core section, or the newer OASIS Saml Wiki.
- Log your actual SAML2 conversation with SAML Chrome Panel or SAML Tracer for Firefox.
- Connect an `ILoggerAdapter` to your `SPOptions.Logger`. If you are using the `OWIN` middleware this is done for you automatically and you can see the output in the OWIN/Katana logging.
- Last but not least, download the Saml2 source and check out what's really happening.

3.5 Contributing

Sustainsys.Saml2 is maintained by and have mostly been developed by Sustainsys in Stockholm, Sweden. The library's source code is hosted on github. When doing work on protocol features, it is recommended to consult the official SAML specifications.

3.5.1 Issue tracking

Github issues are used to keep track of issues and releases. For requests of functionality or to report bugs, please open an issue in the github repo. It is advised to open an issue describing the plans before starting any major coding work. Discussing before writing code significantly reduces the risk of getting a pull request denied.

3.5.2 Versioning

Sustainsys uses semantic versioning as defined on http://semver.org/. Given a version number MAJOR.MINOR.PATCH, increment the:

- MAJOR version when you make incompatible API changes,
- MINOR version when you add functionality in a backwards-compatible manner, and
- PATCH version when you make backwards-compatible bug fixes.

3.5.3 Coding Conventions

The coding conventions follow the classic .NET style of coding, with the following styles:

- Always use `{}` for if statements, even when there is only one line.
- Code analysis is enabled and all code should compile without compiler warnings or code analysis errors. Code analysis warnings that are not relevant are supressed in the source. Rules should only be disabled on a global level if it really is appropriate to disable the rule for the entire code base. Unknown words are added to CustomDictionary.xml instead of suppressing individual warnings.
- Private members in classes are named with camelCasing, no underscores or similar.
- Member variables are not prefixed with `this.` unless required to resolve ambiguity (such as in a constructor having parameters with the same name as the members).

- Any single method is short enough to fit on one screen (on a typical laptop monitor, not a 30-inch development monster-monitor in vertical orientation).
- The code is formatted to (mostly) fit in 80 columns.

3.5.4 Unit Tests

The Sustainsys.Saml2 library has been developed using TDD (Test Driven Development). All functionality is covered by tests, and it will remain that way. Pull requests will only be merged if they contain tests covering the added functionality. Parts of the code that aren't practically possible to test because of tight integration with the web server (see e.g. `CommandResult.ApplyPrincipal`) are excluded from this rule and should be marked with an `[ExcludeFromCodeCoverage]` attribute. The code coverage report is at 100.00% coverage and should remain so.

3.5.5 Continuous Integration / Build Server

All pull requests are built on AppVeyor and code coverage is checked.

3.5.6 Branching

To make a clean pull request, it is important to follow some git best practices. Nancy has an excellent guide that outlines the steps required.

3.5.7 Licensing

The library is licensed under MIT (for the master branch) and by submitting code it is accepted that the submitted code will be released under the same license. Third party code may only be added to the library if the author of the pull request holds the copyright to the code, or the code is previously licensed under a license compatible with MIT.

3.6 `<sustainsys.saml2>` Element

The `<sustainsys.saml2>` element is a child node of the `<configuration>` element. Its attributes are listed and described below, and its child elements are listed as well and are linked to full explanations of each.

3.6.1 Attributes

`returnUrl` The Url that you want users to be redirected to once the authentication is complete. This is typically the start page of the application, or a special signed in start page.

`entityId` The name that this service provider will use for itself when sending messages. The name will end up in the `Issuer` field in outcoing authnRequests.

The SAML standard requires the entityId to be an absolute URI. Typically it should be the URL where the metadata is presented. E.g. http://sp.example.com/Saml2/.

`discoveryService` (Optional) Specifies an idp discovery service to use if no idp is specified when calling sign in. Without this attribute, the first idp known will be used if none is specified.

`modulePath` (Optional) Indicates the base path of the Saml2 endpoints. Defaults to /Saml2 if not specified. This can usually be left as the default, but if several instances of Saml2 are loaded into the same process they must each get a separate base path.

`authenticateRequestSigningBehavior` (Optional) Sets the signing behavior for generated AuthnRequests. Three values are supported:

- `Never`: Saml2 will never sign any created AuthnRequests.
- `Always`: Saml2 will always sign all AuthnRequests.
- `IfIdpWantAuthnRequestsSigned` (default if the attribute is missing): Saml2 will sign AuthnRequests if the idp is configured for it (through config or listed in idp metadata).

`validateCertificates` (Optional) Normally certificates for the IDPs signing use is communicated through metadata and in case of a breach, the metadata is updated with new data. If you want extra security, you can enable certificate validation (the default value for this attribute is `false`). Please note that the SAML metadata specification explicitly places no requirements on certificate validation, so don't be surprised if an Idp certificate doesn't pass validation.

`publicOrigin` (Optional) Indicates the base url of the Saml2 endpoints. It should be the root path of the application. E.g. The SignIn url is built up as `PublicOrigin + / + modulePath + /SignIn`. Defaults to Url of the current http request if not specified. This can usually be left as the default, but if your internal address of the application is different than the external address the generated URLs (such as `AssertionConsumerServiceURL` in the `saml2p:AuthnRequest`) then this will be incorrect. The use case for this is typically with load balancers or reverse proxies. It can also be used if the application can be accessed by several external URLs to make sure that the registered in metadata is used in communication with the Idp.

If you need to set this value on a per-request basis, provide a `GetPublicOrigin` Notification function instead.

`outboundSignAlgorithm` (Optional) By default Saml2 uses SHA256 signatures if running on .NET 4.6.2 or later or when you have called `GlobalEnableSha256XmlSignatures()`. Otherwise, it uses SHA1 signatures. Use this attribute to set the default signing algorithm for any messages (including metadata) that Saml2 generates. Possible values:

- `SHA1` (or http://www.w3.org/2000/09/xmldsig#rsa-sha1)
- `SHA256`
- `SHA384`
- `SHA512`

The full url identifying the algorithm can also be provided. The algorithm can be overridden for each IdentityProvider too.

`minIncomingSigningAlgorithm` (Optional) The minimum strength required on signatures on incoming messages. Messages with a too weak signing algorithm will be rejected. By default Saml2 requires SHA256 signatures if running on .NET 4.6.2 or later or when you have called `GlobalEnableSha256XmlSignatures()`. Otherwise, it uses SHA1 signatures.

Possible values:

- `SHA1` (or http://www.w3.org/2000/09/xmldsig#rsa-sha1)
- `SHA256`
- `SHA384`
- `SHA512`

The full url identifying the algorithm can also be provided.

3.6.2 Elements

The following are the possible children elements of the `<sustainsys.saml2>` element. Each are provided as a link below with full explanations of each.

- *nameIdPolicy*
- *requestedAuthnContext*
- *metadata*
- *identityProviders*
- *federations*
- *serviceCertificates*
- *compatibility*

3.7 `<nameIdPolicy>` Element

This is an **optional** child element of the *sustainsys.saml2* element.

This element controls the generation of `NameIDPolicy` element in AuthnRequests. The element is only created if either `allowCreate` or `format` are set to a non-default value.

3.7.1 Attributes

`allowCreate` **(Optional)** Default value is empty, which means that the attribute is not included in generated AuthnRequests. Supported values are `true` or `false`.

`format` **(Optional)** Sets the requested format of `NameIDPolicy` for generated authnRequests.

Supported values (see section 8.3 in the SAML2 Core specification for explanations of the values).

- `Unspecified`
- `EmailAddress`
- `X509SubjectName`
- `WindowsDomainQualifiedName`
- `KerberosPrincipalName`
- `EntityIdentifier`
- `Persistent`
- `Transient`

If no value is specified, no format is specified in the generated AuthnRequests. If `Transient` is specified, it is not permitted to specify `allowCreate` (see 3.4.1.1 in the SAML2 Core spec).

3.8 `<requestedAuthnContext>` Element

This is an **optional** child element of the *sustainsys.saml2* element.

3.8.1 Attributes

classRef (Optional) Class reference for authentication context. Either specify a full URI to identify an authentication context class, or a single word if using one of the predefined classes in the SAML2 Authentication context specification:

- InternetProtocol
- InternetProtocolPassword
- Kerberos
- MobileOneFactorUnregistered
- MobileTwoFactorUnregistered
- MobileOneFactorContract
- MobileTwoFactorContract
- Password
- PasswordProtectedTransport
- PreviousSession
- X509
- PGP
- SPKI
- XMLDSig
- Smartcard
- SmartcardPKI
- SoftwarePKI
- Telephony
- NomadTelephony
- PersonalTelephony
- AuthenticatedTelephony
- SecureRemotePassword
- TLSClient
- TimeSyncToken
- unspecified

comparison (Optional) Comparison method for authentication context as signalled in AuthnRequests. Valid values are:

- Exact (default)
- Minimum
- Maximum
- Better

Minimum is an inclusive comparison, meaning the specified classRef or anything better is accepted. Better is exclusive, meaning that the specified classRef is not accepted.

3.9 `<metadata>` Element

This is an **optional** child element of the *sustainsys.saml2* element.

The metadata part of the configuration can be used to tweak the generated metadata. These configuration options only affect how the metadata is generated, no other behavior of the code is changed.

3.9.1 Attributes

`cacheDuration` (Optional) Describes for how long in anyone should cache the metadata presented by the service provider before trying to fetch a new copy. Defaults to one hour.

Examples of valid format strings:

- 1 day, 2 hours: `1.2:00:00`
- 42 seconds: `0:00:42`

`validDuration` (Optional) Sets the maximum time that anyone may cache the generated metadata. If `cacheDuration` is specified, the remote party should try to reload metadata after that time. If that refresh fails, `validDuration` determines for how long the old metadata may be used before it must be discarded.

In the metadata, the time is exposed as an absolute `validUntil` date and time. That absolute time is calculated on metadata generation by adding the configured `validDuration` to the current time.

Examples of valid format strings:

- 1 day, 2 hours: `1.2:00:00`
- 42 seconds: `0:00:42`

`wantAssertionSigned` (Optional) Signal to IDPs that we want the Assertions themselves signed and not only the SAML response. Saml2 supports both, so for normal usage this shouldn't matter. If set to `false` the entire `wantAssertionsSigned` attribute is dropped from the metadata as the default values is `false`.

3.9.2 Elements

The following are the possible children elements of the `<metadata>` element. Each are provided as a link below with full explanations of each.

- *organization*
- *contactPerson*
- *requestedAttributes*

3.10 `<organization>` Element

Optional child element of the *<metadata> Element* element.

Provides information about the organization supplying the SAML2 entity (in plain English that means the organization that supplies the application that Saml2 is used in).

3.10.1 Attributes

`name` The name of the organization.

`displayName` The display name of the organization.

`url` URL to the organization's web site.

`language` In the generated metadata, the `name`, `displayName` and `url` attributes have a language specification. If none is specified, the `xml:lang` attribute will be generated with an empty value.

3.11 `<contactPerson>` Element

Optional child element of the *<metadata> Element* element.

3.11.1 Attributes

`type` The type attribute indicates the type of the contact and is picked from the `ContactType` enum. Valid values are:

- `Administrative`
- `Billing`
- `Other`
- `Support`
- `Technical`

`company` (Optional) Name of the person's company.

`givenName` (Optional) Given name of the contact person.

`surname` (Optional) Surname of the contact person.

`phoneNumber` (Optional) Phone number of the contact person. The SAML standard allows multiple phone number to be specified. Saml2 supports that, but not through the configuration file.

`email` (Optional) Email address of the person. The SAML standard allows multiple email addresses to be specified. Saml2 supports that, but not through the configuration file.

3.12 `<requestedAttributes>` Element

Optional child element of the *<metadata> Element* element.

List of attributes that the SP requests to be included in the assertions generated by an identity provider. Each attribute is added to the list with an `<add>` element.

The element should look something like this:

```
<requestedAttributes>
    <add name="" friendlyName="" nameFormat="" isRequired=""/>
    <add name="" friendlyName="" nameFormat="" isRequired=""/>
    ...
</requestedAttributes>
```

3.12.1 Attributes

name The name of the attribute. This is usually in the form of an urn/oid, e.g. `urn:oid:1.2.3`. The format of the name should be specified in the `nameFormat` attribute.

friendlyName (Optional) An optional friendly (i.e. human readable) friendly name of the attribute that will be included in the metadata. Please note that the SAML2 standard specifically forbids the friendlyName to be used for anything other than information to a human. All matching of attributes must use the `name`.

nameFormat (Optional) Format of the name attribute. Valid values are:

- `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`
- `urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified`
- `urn:oasis:names:tc:SAML:2.0:attrname-format:basic`

isRequired (Optional) `true` or `false` value indicating whether the attribute is required by the service provider or just a request that it would be nice if the Idp includes it.

3.13 `<identityProviders>` Element

This is an **optional** child element of the *sustainsys.saml2* element.

It indicates a list of identity providers known to the service provider.

Each identity provider is added as an `<add>` element to the `<identityProviders>` element and the element will end up looking something like what is shown below. Note the possible child element of the `<signingCertifcate>` which is shown in the second added identity provider element below.

```
<identityProviders>
    <add entityId="" signOnUrl="" logoutUrl="" allowUnsolicitedAuthnResponse=""␣
↪binding=""
        wantAuthnRequestsSigned="" loadMetadata="" metadataLocation=""␣
↪disableOutboundLogoutRequests="" outboundSigningAlgorithm=""/>
    <add entityId="" signOnUrl="" logoutUrl="" allowUnsolicitedAuthnResponse=""␣
↪binding=""
        wantAuthnRequestsSigned="" loadMetadata="" metadataLocation=""␣
↪disableOutboundLogoutRequests="" outboundSigningAlgorithm="">
        <signingCertificate storeName="" storeLocation="" findValue="" x509FindType="
↪" />
    </add>
    ...
</identityProviders>
```

3.13.1 Attributes

entityId The issuer name that the idp will be using when sending responses. When `<loadMetadata>` is enabled, the entityId is treated as a URL to for downloading the metadata.

signOnUrl (Optional) The url where the identity provider listens for incoming sign on requests. The url has to be written in a way that the client understands, since it is the client web browser that will be redirected to the url. Specifically, this means that using a host name only url or a host name that only resolves on the network of the server won't work.

`logoutUrl` **(Optional)** The url where the identity provider listens for incoming logout requests and responses. To enable single logout behaviour there must also be a service certificate configured in Saml2 as all logout messages must be signed.

`allowUnsolicitedAuthnResponse` Allow unsolicited responses. That is, Idp initiated sign on where there was no prior AuthnRequest. If true `InResponseTo` is not required and the IDP can initiate the authentication process. If false `InResponseTo` is required and the authentication process must be initiated by an AuthnRequest from this SP. Note that if the authentication was SP-intiatied, `RelayState` and `InResponseTo` must be present and valid.

`binding` **(Optional)** The binding that the services provider should use when sending requests to the identity provider. One of the supported values of the `Saml2BindingType` enum.

- `HttpRedirect`
- `HttpPost`
- `Artifact`

`wantAuthnRequestsSigned` **(Optional)** Specifies whether the Identity provider wants the AuthnRequests signed. Defaults to `false`.

`loadMetadata` **(Optional)** Load metadata from the idp and use that information instead of the configuration. It is possible to use a specific certificate even though the metadata is loaded, in that case the configured certificate will take precedence over any contents in the metadata.

`metadataLocation` **(Optional)** The SAML2 metadata standard strongly suggests that the `Entity Id` of a SAML2 entity is a URL where the metadata of the entity can be found. When loading metadata for an idp, Saml2 normally interprets the EntityId as a url to the metadata. If the metadata is located somewhere else it can be specified with this configuration parameter. The location can be a URL, an absolute path to a local file or an app relative path (e.g. `~/App_Data/IdpMetadata.xml`)

`disableOutboundLogoutRequests` **(Optional)** Disable outbound logout requests to this idp, even though Saml2 is configured for single logout and the idp supports it. This setting might be usable when adding SLO to an existing setup, to ensure that everyone is ready for SLO before activating.

`outboundSigningAlgorithm` **(Optional)** By default Saml2 uses SHA256 signatures if running on .NET 4.6.2 or later and otherwise SHA1 signatures. Set this to set the signing algorithm for any outbound messages for this identity provider. Possible values:

- `SHA1`
- `SHA256`
- `SHA384`
- `SHA512`

3.13.2 Elements

The following are the possible children elements of the `<identityProviders>` element. Each are provided as a link below with full explanations of each.

- *signingCertificate*

3.14 `<signingCertificate>` Element

Optional element of the *identityProvider* element.

The certificate that the identity provider uses to sign its messages. The certificate can either be loaded from file if the `fileName` attribute is specified or from a certificate store if the other attributes are specified. If a `fileName` is specified that will take precedence and the other attributes will be ignored.

> **Warning:** File-based certificates are only recommended for testing and during development. In production environments it is better to use the certificate store.

3.14.1 Attributes

`fileName` A file name to load the certificate from. The path is relative to the execution path of the application. Make sure to heed the warning above – *best to use store-based certificates for non-development environments.*

`storeName` Name of the certificate store to search for the certificate. It is recommended to keep the certificate of the identity provider in the "Other People" store which is specified by the `AddressBook` enum value. Valid values are those from the `System.Security.Cryptography.X509Certificates.StoreName` enumeration.

`storeLocation` The location of the store to search for the certificate. On production services it is recommended to use the LocalMachine value, while it makes more sense to use CurrentUser in development setups. Valid values are those from the `System.Security.Cryptography.X509Certificates.StoreLocation` enumeration.

`findValue` A search term to use to find the certificate. The value will be searched for in the field specified by the `x509FindType` attribute.

`x509FindType` The field that will be seach for a match to the value in findValue. For security, it is recommended to use `FindBySerialNumber`.

Valid values are those from the `System.Security.Cryptography.X509Certificates.X509FindType` enumeration.

> **Warning:** There is a nasty bug when copying a serial number from the certificate info displayed by certificate manager and the browser. There is a hidden character before the first hex digit that will mess up the matching. Once pasted into the config, use the arrow keys to make sure that there is not an additional invisible character at the start of the serial number string.

3.15 `<federations>` Element

This is an **optional** child element of the *sustainsys.saml2* element.

This element contains a list of federations that the service provider knows and trusts.

As with some other elements, individual items are added via an `<add>` element inside this element, so you'll end up with XML that looks like the following:

```
<federations>
    <add metadataLocation="" allowUnsolicitedAuthnResponse="" />
    <add metadataLocation="" allowUnsolicitedAuthnResponse="" />
    ...
</federations>
```

3.15.1 Attributes

`metadataLocation` URL to the full metadata of the federation. Saml2 will download the metadata and add all identity providers found to the list of known and trusted identity providers. The location can be a URL, an absolute path to a local file or an app relative path (e.g. `~/App_Data/IdpMetadata.xml`)

`allowUnsolicitedAuthnResponse` `true` or `false` value indicating whether unsolicited authn responses should be allowed from the identity providers in the federation.

3.16 `<serviceCertificates>` Element

This is an **optional** child element of the *sustainsys.saml2* element.

Specifies the certificate(s) that the service provider uses for encrypted assertions (and for signed requests, once that feature is added). If neither of those features are used, this element can be ommitted.

The public key(s) will be exposed in the metadata and the private key(s) will be used during decryption/signing.

Individual certificates are added via an `<add>` element, so the resulting XML will be similar to the following:

```
<serviceCertificates>
    <add use="" status="" metadataPublishOverride="" />
    <add use="" status="" metadataPublishOverride="" />
    ...
</serviceCertificates>
```

3.16.1 Attributes

`use` Indicates how the certificate will be used. Options are:

- `Signing`
- `Encryption`
- `Both` (default)

`status` Indicates whether the certificate is a current or future certificate – used in key rollover scenarios. Options are:

- `Current` (default)
- `Future`

`metadataPublishOverride` By default the certificate will be used and published by the rules shown in the table below. To override this behavior choose one of the following options for this attribute:

- `None` (Default) - published according to the rules in the table below.
- `PublishUnspecified`
- `PublishEncryption`
- `PublishSigning`
- `DoNotPublish`

Use	Status	Published in Metadata	Used by Saml2
Both	Current	Unspecified *unless Future key exists*, then Signing	Yes
Both	Future	Unspecified	For decryption only
Signing	Current	Signing	Yes
Signing	Future	Signing	No
Encryption	Current	Encryption *unless Future key exists* then not published	Yes
Encryption	Future	Encryption	Yes

`filepath` Filename and path to local SP certificate, only to be used in testing.

`storeName` Store name when to be used with certificate store, ex. My

`storeLocation` Store name when to be used with certificate store, ex. LocalMachine

`x509FindType` Type to find the X509 certificate in store ex. FindByThumbprint.

Allows for System.Security.Cryptography.X509Certificates.X509FindTypes enum.

`findValue` The value to be find according to the x509FindType.

3.17 `<compatibility>` Element

This is an **optional** child element of the *sustainsys.saml2* element.

Enables overrides of default behaviour to increase compatibility with identity providers.

3.17.1 Attributes

`unpackEntitiesDescriptorInIdentityProviderMetadata` **(Optional)** If an `EntitiesDescriptor` element is found when loading metadata for an IdentityProvider, automatically check inside it if there is a single `EntityDescriptor` and in that case use it.

`IgnoreAuthenticationContextInResponse` Do not read the `AuthnContext` element in Saml2Response. If you do not need these values to be present as claims in the generated identity, using this option can prevent XML format errors (`System.Xml.XmlException: ID0013: The value must be an absolute URI`), when value cannot parse as absolute URI.