ComponentSpace
SAML for ASP.NET Core
Metadata Guide

Copyright © ComponentSpace Pty Ltd 2017-2025. All rights reserved.
www.componentspace.com

Contents

Introduction 1
Identity Provider Metadata 1
Service Provider Metadata 3
Creating Metadata 6
  Creating Identity Provider Metadata 6
  Creating Service Provider Metadata 7
Exporting Metadata 7
Importing Metadata 8
Signing Metadata 9
Verifying Metadata 9
Monitoring Metadata Updates 9

Introduction

SAML metadata is a standard XML format for exchanging configuration information between identity providers and service providers.

Its use is optional but recommended.

SAML metadata exchange occurs as part of initializing an identity provider's or service provider's internal configuration and before any SAML single sign-on attempts.

Often SAML metadata is provided as an XML file or through a download link.

If not used, SAML configuration may be supplied in an ad hoc format (e.g. in an email or document).

Identity Provider Metadata

The following is an example of SAML metadata describing an identity provider.

A partner service provider would use this information to update its internal configuration to enable single sign-on between the identity provider and itself.

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
entityID="https://ExampleIdentityProvider">
  <md:IDPSSODescriptor WantAuthnRequestsSigned="true"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <md:KeyDescriptor>
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>
MIIDATCCAemgAwIBAgIQdPDr/iI1jbhDMTj5VYya+TANBgkqhkiG9w0BAQsFADAW
MRQwEgYDVQQDEwt3d3cuaWRwLmNvbTAeFw0xMzExMjIwODIwNTJaFw00OTEyMzEx
NDAwMDBaMBYxFDASBgNVBAMTC3d3dy5pZHAuY29tMIIBIjANBgkqhkiG9w0BAQEF
AAOCAQ8AMIIBCgKCAQEAi0XJRLDrcbSyqUd8XG4BgxObQMYLAkENlmJOsAEpl1xM
abUiq1X4v0Fc8ZaCpUE3fFGENMEWgBjnQUUE0WtVUh5JPMsukolf9qljbJkCkvHX
H3O4Uen7vA2oNQWt4bK96SpXADpZKFvpk4D7btKOgU/NamjiqwHI4fI8kFJKwKBJ
chRPUQdC4ljRRmGIrSnpY+t25/d3KGXwbe9Z2MGGy2hyA0tgOWuchIK+1vAKKBUh
9nDEXfr80+xW680w5TqHyDcqbWvQsXXhH0yZLfINKNS6/IojHPsBy7tf36Ck9H5P
w+1PPu6NzBFSz5ZkC8KzrS6vuZXc/ImYrnheMQsqqQIDAQABo0swSTBHBgNVHQEE
QDA+gBD4dY4MCPEmG4sxZrcni8vtoRgwFjEUMBIGA1UEAxMLd3d3LmlkcC5jb22C
EHTw6/4iNY24QzE4+VWMmvkwDQYJKoZIhvcNAQELBQADggEBABhak2aR84MCdyXO
4AKOQvZybsCMdhRq2i1i0WhD4/xe7Ry5haC6TeXIp8Q4cC3MzsrDal74xHI714BW
0loafpHAsXfd9EvkKTVaJ+1Zpe16+SsTL4upS1cGydigqwUzsdpGck4wI1moJ947
7O+46If2gF27u9Cdk7Onxe/5dwLIxWmkVRdbQIH5GsKUeAjOdRQmy+X1MX6KyRoa
CwWGYwxi5Sa+r+3AtDvD4BX0EJGKFZeeM3J/yMpYh/75aN0cFQfDEdJ7C5NE0von
idE0QtIFvsoWtZUtur2fiW7yBxse38TPQsi2r6A6c/TZsZ5bq31yh3gr3kSN62H8
iVKLQLA=
          </ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect" Location="https://localhost:44313/SAML/SingleLogoutService"/>
```

```
    <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://localhost:44313/SAML/SingleLogoutService"/>
    <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect" Location="https://localhost:44313/SAML/SingleSignOnService"/>
    <md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://localhost:44313/SAML/SingleSignOnService"/>
  </md:IDPSSODescriptor>
</md:EntityDescriptor>
```

The most important and commonly used SAML metadata settings for an identity provider are listed below.

entityID [required]

The entity ID is the unique name for the identity provider.

It should either be a URI (i.e. URN or URL).

For maximum interoperability, it's recommended that a URL is used.

The URL doesn't necessarily have to reference a web resource.

It's recommended the URL is based off the organization's domain name or the applicable web application's address.

WantAuthnRequestsSigned [optional]

The identity provider expects that SAML authn request sent by service providers as part of SP-initiated SSO to be signed.

The default is false.

X509Certificate [required]

SAML responses or assertions sent by the identity provider should be signed.

The included base-64 encoded X.509 certificate should be used by service providers to verify these signatures.

If more than one certificate is included, the signing certificate is identified either by a KeyDescriptor use attribute of "signing" or by not having a KeyDescriptor use attribute.

SingleLogoutService [optional]

The single logout service specifies the endpoint that receives SAML logout messages.

The Binding identifies the transport mechanism and should be one of the following:

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

The Location is the URL of the single logout service endpoint.

More than one SingleLogoutService may be specified. Typically, this is done when multiple bindings are supported.

If SAML logout is not supported, a SingleLogoutService is not required.

NameIDFormat [optional]

The NameIDFormat specifies the name identifier format supported by the identity provider.

More than one NameIDFormat may be specified.

The default is effectively "urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified". This means that the interpretation of the format of name identifiers is left to the application.

SingleSignOnService [required]

The single sign-on service specifies the endpoint that receives SAML authn requests.

The Binding identifies the transport mechanism and should be one of the following:

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

The Location is the URL of the single sign-on service endpoint.

More than one SingleSignOnService may be specified. Typically, this is done when multiple bindings are supported.

If SP-initiated SSO is not supported, a SingleSignOnService is not required but must still be specified to satisfy the SAML metadata schema.

Service Provider Metadata

The following is an example of SAML metadata describing a service provider.

A partner identity provider would use this information to update its internal configuration to enable single sign-on between the service provider and itself.

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
entityID="https://ExampleServiceProvider">
  <md:SPSSODescriptor
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <md:KeyDescriptor>
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>
MIIC/jCCAeagAwIBAgIQCGehfcnv6r5My/fnrbfDejANBgkqhkiG9w0BAQsFADAV
MRMwEQYDVQQDEwp3d3cuc3AuY29tMB4XDTEzMTEyMjA4MjMyMVoXDTQ5MTIzMTE0
MDAwMFowFTETMBEGA1UEAxMKd3d3LnNwLmNvbTCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBAMPm/ew9jaGWpQS1C7KtpvgzV4nSOIFPgRt/nlRYR+pUWdDE
fSKmyjK28nkQ1KKujRJTnvnmZydmUrmEFpVv+giBiUkvCJY3PxZ/EDSsF3R/OzWh
kUv5nfAXPnqkX9x22b6+vUof6WiLGyAW6lOYMCVADjTSl9pSaUtIaANdx9maERcT
9eQbGSnjim0WurFRYs9ZE8ttErrMH9+Su4246YDqOPAkz6La4cHHMPQdcFQT5p/c
uXBfU1vl1tWdBEgAY3xHYZE8u5TTJ/vp9UxyU1MwfeO2g9VDRcokLQHrj6wFxtvu
fA+WtUKYJGUu2p/qSuaw7eS6UFjUn49aVqg9OacCAwEAAaNKMEgwRgYDVR0BBD8w
PYAQ1/S0ibdvfdFkJ9T9oIPluKEXMBUxEzARBgNVBAMTCnd3dy5zcC5jb22CEAhn
oX3J7+q+TMv35623w3owDQYJKoZIhvcNAQELBQADggEBAAHlmVoAZUt6paeFvtQb
c/iaJe/Fhd+JG1U0jyjlFDcCn8erLihEbhb3mFBBMF25oO67gfA1JJXZrmHry3Nl
```

```
OZuovqRqm8v7wg8n0nQa1HUWkUC2TBgfg1HE8/2rmSF2PngiEi18VOxRDxx0WXMN
ZX6JebJ1kCOCpT/x7aupS7T1GrIPmDLxjnC9Bet7pRynfomjP/6iU21/xOIF6xB9
Yf1a/kQbYdAVt2haYKIfvaF3xsq1X5tCXc9ijhBMgyaoqA+bQJD/l3S8+yCmMxEY
ZjAVLEkyGlU4Uwo01cKEYbXIG/YVq+4CaIRxIfMvV+j8gzTLHTXI+pHEMfMhyYa0
pzM=
            </ds:X509Certificate>
          </ds:X509Data>
        </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect" Location="https://localhost:44360/SAML/SingleLogoutService"/>
    <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://localhost:44360/SAML/SingleLogoutService"/>
    <md:AssertionConsumerService index="0" isDefault="true"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://localhost:44360/SAML/AssertionConsumerService"/>
  </md:SPSSODescriptor>
</md:EntityDescriptor>
```

The most important and commonly used SAML metadata settings for a service provider are listed below.

entityID [required]

The entity ID is the unique name for the service provider.

It should either be a URI (i.e. URN or URL).

For maximum interoperability, it's recommended that a URL is used.

The URL doesn't necessarily have to reference a web resource.

It's recommended the URL be based off the organization's domain name or the applicable web application's address.

AuthnRequestsSigned [optional]

The service provider will sign SAML authn requests sent as part of SP-initiated SSO.

The default is false.

WantAssertionsSigned [optional]

The service provider expects that SAML assertions sent by identity providers to be signed.

The default is false.

X509Certificate (signing) [optional]

SAML authn requests sent by the service provider may be signed.

The included base-64 encoded X.509 certificate should be used by identity providers to verify these signatures.

If more than one certificate is included, the signing certificate is identified either by a KeyDescriptor use attribute of “signing” or by not having a KeyDescriptor use attribute.

If SAML authn requests are not signed, a signing certificate is not required.

X509Certificate (encryption) [optional]

SAML assertions sent by the identity provider may be encrypted.

The included base-64 encoded X.509 certificate should be used by identity providers to encrypt SAML assertions.

If more than one certificate is included, the encryption certificate is identified either by a KeyDescriptor use attribute of “encryption” or by not having a KeyDescriptor use attribute.

If SAML assertions are not to be encrypted, an encryption certificate is not required.

SingleLogoutService [optional]

The single logout service specifies the endpoint that receives SAML logout messages.

The Binding identifies the transport mechanism and should be one of the following:

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

The Location is the URL of the single logout service endpoint.

More than one SingleLogoutService may be specified. Typically, this is done when multiple bindings are supported.

If SAML logout is not supported, a SingleLogoutService is not required.

NameIDFormat [optional]

The NameIDFormat specifies the name identifier format supported by the service provider.

More than one NameIDFormat may be specified.

The default is effectively “urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified”. This means that the interpretation of the format of name identifiers is left to the application.

AssertionConsumerService [required]

The assertion consumer service specifies the endpoint that receives SAML responses.

The Binding identifies the transport mechanism and should be one of the following:

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

The Location is the URL of the assertion consumer service endpoint.

An optional index identifies an AssertionConsumerService by a unique number.

An optional isDefault flag identifies the default AssertionConsumerService.

More than one AssertionConsumerService may be specified. Typically, this is done when multiple bindings are supported.

Creating Metadata

The CreateMetadata console application project may be used to generate SAML metadata.

It may be used to generate SAML metadata for the local identity provider or service provider for distribution to partner providers.

Less commonly, it may also be used to generate SAML metadata on behalf of partner providers who have supplied ad-hoc configuration information and where partner metadata is required for internal consumption.

CreateMetadata may be run as follows.

```
dotnet CreateMetadata.dll
```

It will prompt for various input required to generate the metadata.

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

Creating Identity Provider Metadata

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

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

Entity ID:

Specify a URL that uniquely identifies the identity provider.

This name must match with the LocalIdentityProviderConfiguration Name property in the SAML configuration.

X.509 signature certificate file [None]:

Specify the path to the X.509 certificate file (e.g. .cer file) whose certificate should be used for verifying signatures.

Single Sign-On Service URL:

Specify the single sign-on service URL.

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

Single Logout Service URL [None]:

Specify the single logout service URL.

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

Name ID Format [None]:

Specify the name identifier format supported by the identity provider.

Want authn requests signed? [False]:

Specify whether SAML authn requests are expected to be signed.

SAML metadata file [metadata.xml]:

Specify the file where the generated metadata will be saved.

Creating Service Provider Metadata

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

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

Entity ID:

Specify a URL that uniquely identifies the service provider.

This name must match with the LocalServiceProviderConfiguration Name property in the SAML configuration.

X.509 signature certificate file [None]:

Specify the path to the X.509 certificate file (e.g. .cer file) whose certificate should be used for verifying signatures.

X.509 encryption certificate file [None]:

Specify the path to the X.509 certificate file (e.g. .cer file) whose certificate should be used for encrypting SAML assertions.

Assertion Consumer Service URL:

Specify the assertion consumer service URL.

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

Single Logout Service URL [None]:

Specify the single logout service URL.

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

Name ID Format [None]:

Specify the name identifier format supported by the service provider.

Authn requests signed? [False]:

Specify whether SAML authn requests will be signed.

Want assertions signed? [False]:

Specify whether SAML assertions are expected to be signed.

SAML metadata file [metadata.xml]:

Specify the file where the generated metadata will be saved.

Exporting Metadata

The ExportMetadata console application project may be used to generate SAML metadata from the specified SAML configuration.

ExportMetadata may be run as follows.

```
dotnet ExportMetadata.dll
```

It will prompt for various input required to generate the metadata.

SAML configuration file to export [appsettings.json]:

Specify the path to the SAML configuration file to be exported as SAML metadata.

Configuration path [SAML]:

Specify the JSON path to the SAML configuration within the file.

X.509 signature certificate file [None]:

Specify the path to the X.509 certificate file (e.g. .cer file) whose certificate should be used for verifying signatures.

X.509 encryption certificate file [None]:

Specify the path to the X.509 certificate file (e.g. .cer file) whose certificate should be used for encrypting SAML assertions.

SAML metadata file [metadata.xml]:

Specify the file where the generated metadata will be saved.

Importing Metadata

The ImportMetadata console application project may be used to import SAML metadata as SAML configuration.

ImportMetadata may be run as follows.

```
dotnet ImportMetadata.dll
```

It will prompt for various input required to import the metadata.

SAML metadata file to import:

Specify the path to the SAML metadata file to import.

SAML configuration file [saml.json]:

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

A file is created containing one or more PartnerIdentityProvider or PartnerServiceProvider configurations.

These should be manually merged into an existing SAML configuration file (e.g. appsettings.json).

Signing Metadata

The GenerateSignature console application project may be used to sign SAML metadata.

GenerateSignature may be run as follows.

```
dotnet GenerateSignature.dll <fileName> --certificate <certificateFileName> --password <password>
```

For example:

```
dotnet GenerateSignature.dll metadata.xml --certificatec sp.pfx --password password
```

Verifying Metadata

The VerifySignature console application project may be used to verify SAML metadata signatures.

VerifySignature may be run as follows.

```
dotnet VerifySignature.dll <fileName> [--certificate <certificateFileName>]
```

For example:

```
dotnet VerifySignature.dll metadata.xml --certificate sp.cer
```

Monitoring Metadata Updates

The ManageMetadataUpdates console application project may be used to monitor changes to partners' SAML metadata.

ManageMetadataUpdates may be run as follows.

```
dotnet ManageMetadataUpdates.dll <command>
```

The command is one of:

- list
- read
- update
- delete

A small database is used to keep track of metadata changes.

To list all entries in the database:

```
dotnet ManageMetadataUpdates.dll list
```

To see the details for a specific entry in the database, identified by the SAML metadata download URL:

```
dotnet ManageMetadataUpdates.dll read <url>
```

To update the metadata entry by downloading the metadata if it has changed and optionally verifying any signature:

```
dotnet ManageMetadataUpdates.dll update <url> [--certificate <certificateFileName>]
```

The optional certificate file is used to verify the signature, if the SAML metadata is signed.

Any metadata changes are listed.

The most common metadata change is the replacement of an expired X.509 certificate with a new certificate. The new certificate is saved to the file system.

To delete an entry from the database, identified by the SAML metadata download URL:

```
dotnet ManageMetadataUpdates.dll delete <url>
```