Identity & Authorization Management (I.AM)
Federation Metadata

Table of contents

1. Document management

1.1 Document history

Version	Status	Date	Author	Description
1.0	Final	25/03/13	F. Libert	Initial version
1.01	Final	06/07/15	M. Steeman	Adapt INT URL

2. Introduction

All partners working together for Identity & Authorization Management at eHealth are members of the eHealth I.AM Federation.

For Single Sign On between those partners to work well, they need to know certain things about each other: locations used to accept messages, supported profiles and protocols, certificates used to sign messages and so on.

All this information is gathered and defined in SAML 2.0 Metadata XML structures.

Basically, in these structures we describe which party (= Entity) plays which Role(s) in the federation, what services they provide on which location, what ID they use to identify themselves, etc.

For instance, the eHealth IDP is a central party in the federation that plays 2 Roles, “Identity Provider” and “Attribute Authority” (for Web SSO Profiles).

All partners who are hosting an application for which eHealth IDP provides identity and attribute information, are known as SPs and play the role of “Service Provider”.

Every party has its own metadata section which can change over time (especially the information about certificates as these come with expiration dates) and therefore updates of metadata will occur.

Every party is responsible to notify the direct partners in the federation of changes in their system that will be reflected in their metadata part.

The following sections describe updates that change the content of the SAML Metadata and show how partners can prepare themselves for updates.

For detailed explanation of all SAML 2.0 Metadata elements, please consult the official documentation[1]

2.1 Goal of the document

This document describes SAML Metadata in the context of the eHealth I.AM Federation.

Metadata is a heavily overloaded term, but with regard to SAML, it refers to configuration data used to provision SAML entities (such as an SP or IdP) to communicate with each other. It exists in XML form for publishing and interchange.

[1] ***http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf***

3. EntityDescriptor

Each party is described in one or more SAML 2.0 Metadata `<EntityDescriptor>` elements, one for each SAML Entity that plays one or more roles which provides one more services.

Each `<EntityDescriptor>` has a unique `entityID` used to identify the entity in the Federation.

All information that relying parties need to know about one SAML Entity to setup reliable communication with it, can be found in that `<EntityDescriptor>` using SAML 2.0 Metadata elements.

This section will describe only the contents of the `<EntityDescriptor>` elements that are used in the eHealth I.AM Federation and which are subject to change.

Following section will list the different SAML Entities that play a Role in the eHealth I.AM Federation in communication between partners and eHealth.

3.1 RoleDescriptor

Each SAML entity plays one or more roles. For each role played, an extension on the base SAML 2.0 Metadata element `<RoleDescriptor>` is configured inside the `<EntityDescriptor>`. The roles used in eHealth I.AM Federation are **IDPSSO**, **AttributeAuthority** and **SPSSO**.

3.1.1 KeyDescriptor

Updated every time the SAML Entity uses a new Key for message/transport signing/encryption.

A common use of Metadata across SAML Entities is to associate one or more public keys with the system being defined. Every `<RoleDescriptor>` can contain `<KeyDescriptor>` elements for this purpose.

The `<KeyDescriptor>` element is a wrapper around the XML Signature-defined[2] `<ds:KeyInfo>` element, an extensible container for describing keys.

The eHealth I.AM Federation always uses inline `<ds:X509Certificate>` elements as KeyInfo to directly express an explicitly trusted public key. Any number of keys can be included to manage key rollover, and individual keys can be labeled for authentication (= signing), encryption, or both.

This approach has been standardized as part of a Metadata Interoperability profile at OASIS[3].

When a SAML Entity needs to use a new X509Certificate, a rollover process is started: a `<KeyDescriptor>` with the new X509Certificate is aded to the `<EntityDescriptor>` of the entity **BEFORE** the new certificate is actually used. Once relying parties have processed the updated metadata (taking into account a maximum allowed `cacheDuration` as defined in SAML 2.0 Metadata), the new certificate can safely be activated in the system for signing. To end the rollover process, the expired certificate will be removed from the Metadata.

3.1.1.1 Signing

The SAML specification supports signing of SAML Messages. When a SAML entity signs messages, it publishes the X509Certificate in its `<Entitydescriptor>`. Specifying the `use="signing"` is optional and restricts the use of this certificate to signing only.

[2] ***http://www.w3.org/TR/xmldsig-core/***

[3] ***https://wiki.oasis-open.org/security/SAML2MetadataIOP***

Example KeyDescriptor for signing (authentication)

```
<KeyDescriptor use="signing">
    <ds:KeyInfo>
        <ds:X509Data>
            <ds:X509Certificate>
MIIE0DCCA7igAwIBAgILAQAAAAABMu4tWh8wDQYJKoZIhvcNAQEFBQAwNDELMAkG
A1UEBhMCQkUxFjAUBgNVBAMTDUdvdmVybm1lbnQgQ0ExDTALBgNVBAUTBDIwMTAw
HhcNMTExMDEwMTUwMTMyWhcNMTMwMTEwMTUwMTMyWjCBrTEcMBoGA1UEAxMTQ0JF
PTA4MDkzOTQ0MjcsIElBTTELMAkGA1UEBhMCQkUxDDAKBgNVBAsTA0lBTTEXMBUG
…
YzEo/D24vSYKI7Clw3SkKPUcqv3u68IPs8wFL/Nowmxy6HGAvDlt1fQBpwePVKif
GOygUcz0KWHMqNV7IJzyXrF2nbvg3TUJKaDR0zV4CjzLpaCIIY1wY6e2/08mxf/Q
5D7YO3sTxmjixkjRqCKXBCJa0CjXxT3/8Pfg5lHGNr7onIL84SMCZREur5I03u64
HiqHBtSZaDWrw7d4CcjY/NoPfHO8hmAXEBMTm4zEhG4Nw0+2
            </ds:X509Certificate>
        </ds:X509Data>
    </ds:KeyInfo>
</KeyDescriptor>
```

Note

The `<ds:X509Certificate>` element contains an X.509Certificate, encoded as a base64 string.

3.1.1.2 Encryption

The SAML 2.0 specification supports encryption of SAML Messages. SAML Entities supporting encryption must publish their public key certificate in their `<EntityDescriptor>` so relying parties know what public key to use for encrypting messages to them. Specifying the `use="encryption"` is optional and restricts the use of this certificate to encryption only.

Example KeyDescriptor for encryption (confidentiality)

```
<KeyDescriptor use="encryption">
    <ds:KeyInfo>
        <ds:X509Data>
            <ds:X509Certificate>
MIIE0DCCA7igAwIBAgILAQAAAAABMu4tWh8wDQYJKoZIhvcNAQEFBQAwNDELMAkG
A1UEBhMCQkUxFjAUBgNVBAMTDUdvdmVybm1lbnQgQ0ExDTALBgNVBAUTBDIwMTAw
HhcNMTExMDEwMTUwMTMyWhcNMTMwMTEwMTUwMTMyWjCBrTEcMBoGA1UEAxMTQ0JF
PTA4MDkzOTQ0MjcsIElBTTELMAkGA1UEBhMCQkUxDDAKBgNVBAsTA0lBTTEXMBUG
…
YzEo/D24vSYKI7Clw3SkKPUcqv3u68IPs8wFL/Nowmxy6HGAvDlt1fQBpwePVKif
GOygUcz0KWHMqNV7IJzyXrF2nbvg3TUJKaDR0zV4CjzLpaCIIY1wY6e2/08mxf/Q
5D7YO3sTxmjixkjRqCKXBCJa0CjXxT3/8Pfg5lHGNr7onIL84SMCZREur5I03u64
HiqHBtSZaDWrw7d4CcjY/NoPfHO8hmAXEBMTm4zEhG4Nw0+2
            </ds:X509Certificate>
        </ds:X509Data>
    </ds:KeyInfo>
</KeyDescriptor>
```

Note

The `<ds:X509Certificate>` element contains an X.509Certificate, encoded as a base64 string.

3.1.1.3 SSL

When a client connects to a server over an SSL connection, the server authenticates itself with a certificate. Using an SSL handshake, the client MUST verify that it is actually connecting to the server it wants to. SAML entities providing service endpoints over an SSL connection publish their SSL certificate in an additional <KeyDescriptor> (unless the endpoint is for use in Web Browsers which have built-in support for SSL verification using domain validation and certificate chain trust).

Example KeyDescriptor for SSL

```
<KeyDescriptor>
    <ds:KeyInfo>
        <ds:X509Data>
            <ds:X509Certificate>
MIIE0DCCA7igAwIBAgILAQAAAAABMu4tWh8wDQYJKoZIhvcNAQEFBQAwNDELMAkG
A1UEBhMCQkUxFjAUBgNVBAMTDUdvdmVybm1lbnQgQ0ExDTALBgNVBAUTBDIwMTAw
HhcNMTExMDEwMTUwMTMyWhcNMTMwMTEwMTUwMTMyWjCBrTEcMBoGA1UEAxMTQ0JF
PTA4MDkzOTQ0MjcsIElBTTELMAkGA1UEBhMCQkUxDDAKBgNVBAsTA0lBTTEXMBUG
…
YzEo/D24vSYKI7Clw3SkKPUcqv3u68IPs8wFL/Nowmxy6HGAvDlt1fQBpwePVKif
GOygUcz0KWHMqNV7IJzyXrF2nbvg3TUJKaDR0zV4CjzLpaCIIY1wY6e2/08mxf/Q
5D7YO3sTxmjixkjRqCKXBCJa0CjXxT3/8Pfg5lHGNr7onIL84SMCZREur5I03u64
HiqHBtSZaDWrw7d4CcjY/NoPfHO8hmAXEBMTm4zEhG4Nw0+2
            </ds:X509Certificate>
        </ds:X509Data>
    </ds:KeyInfo>
</KeyDescriptor>
```

Note

The `<ds:X509Certificate>` element contains an X.509Certificate, encoded as a base64 string.

3.2 IDPSSODescriptor

RoleDescriptor extension reflecting profiles specific to identity providers supporting SSO.

Role played by the eHealth IDP in the I.AM Federation.

3.2.1 ArtifactResolutionService

Updated every time the SAML Entity uses a new Binding or Location for ArtifactResolution.

SAML includes the ability to rely on redirects containing small strings called "artifacts" that the consuming site uses to pull the complete message. The eHealth IDP supports this with inbound SOAP endpoints to perform artifact → message resolution. For each SOAP endpoint an `<ArtifactResolutionService>` is defined.

Example

```
<ArtifactResolutionService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://www.ehealth.fgov.be/idp/profile/SAML2/SOAP/ArtifactResolution” />
```

3.2.2 SingleSignOnService

Updated every time the SAML Entity uses a new Binding or Location for SingleSignOn.

IdPs support SSO protocols by including one or more `<md:SingleSignOnService>` endpoint elements in their metadata. These are the locations to which the SP (or some other web site acting on its behalf) will send the user to the IdP with a protocol-specific request of some kind.

Example

```
<SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://www.ehealth.fgov.be/idp/profile/SAML2/POST/SSO"/>
```

3.3 AttributeAuthorityDescriptor

RoleDescriptor extension reflecting profiles specific to attribute authorities, SAML authorities that respond to `<samlp:AttributeQuery>` messages.

Role played by the eHealth IDP, AA and STS in the I.AM Federation.

3.3.1 Attributeservice

Updated every time the SAML Entity uses a new Binding or Location for Attribute queries.

Entities that support attribute queries document this by including the `<md:AttributeAuthorityDescriptor>` role in their metadata containing one or more `<md:AttributeService>` endpoint elements. These are the SOAP endpoints to which SPs or other software may send SAML attribute queries.

Example

```
<AttributeService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://www.ehealth.fgov.be/idp/profile/SAML2/SOAP/AttributeQuery" />
```

3.4 SPSSODescriptor

RoleDescriptor extension reflecting profiles specific to service providers.

Role played by the partners that need to contact one of the services of eHealth SAML entities that play the role of IDPSSO and/or AttributeAuthority in the I.AM Federation.

3.4.1 AssertionConsumerService

Updated every time the SAML Entity uses a new Binding or Location for AssertionConsumer.

SPs support SSO protocols by including one or more `<md:AssertionConsumerService>` endpoint elements in their metadata. These are the locations to which the IdP will eventually send the user at the SP. By enumerating them in the metadata, the IdP can ensure that the user's information is sent only to authorized locations.

Example

```
<AssertionConsumerService index="1" Binding="urn:oasis:names:tc:SAML:1.0:profiles:browser-
post" Location="https://www.partner.be/example/Shibboleth.sso/SAML/POST" />
```

3.5 fed:SecurityTokenServiceType

This RoleDescriptor extension is for services supporting WS-Federation.

Since version 1.2, OASIS Standard WS-Federation[4] supports SAML Metadata to establish a federation and describe service roles for WS-Federation Identity Provider services. It extends the core concept by removing the restriction that it only describes SAML entities.

The service roles are derived from `<RoleDescriptor>` using the `xsi:type` extensibility mechanism.

[4] ***http://docs.oasis-open.org/wsfed/federation/200706***

One role is the fed:SecurityTokenServiceType to describe a role similar to `<IDPSSODescriptor>`.

Role played by the eHealth IDP and STS in the I.AM Federation.

3.5.1 fed:SecurityTokenServiceEndpoint

Updated every time the Identity Provider service uses a new Address.

This element specifies the endpoint address of a security token service that supports the WS-Federation and WS-Trust[5] interfaces.

Example

```
<fed:SecurityTokenServiceEndpoint>
    <EndpointReference xmlns="http://www.w3.org/2005/08/addressing">
        <Address>https://www.ehealth.fgov.be/IAM/SecurityTokenService</Address>
    </EndpointReference>
</fed:SecurityTokenServiceEndpoint>
```

3.5.2 fed:PassiveRequestorEndpoint

Updated every time the Identity Provider service uses a new Address.

This element specifies the endpoint address of a service that supports the WS-Federation Web (Passive) Requestor protocol.

Example

```
<fed:PassiveRequestorEndpoint>
    <EndpointReference xmlns="http://www.w3.org/2005/08/addressing">
        <Address>https://wwwint.ehealth.fgov.be/R2/idp/profile/Passive/SSO</Address>
    </EndpointReference>
</fed:PassiveRequestorEndpoint>
```

[5] ***http://docs.oasis-open.org/ws-sx/ws-trust/200512***

4. SAML Entities

4.1 eHealth

The eHealth I.AM Federation Metadata has one SAML 2.0 `<EntityDescriptor>` element for each eHealth Identity Service.

They are registered with a unique identifier as all entities in SAML 2.0 Metadata.

`<EntityDescriptor entityID="…">`

The metadata of these entities is published online and maintained to reflect the actual situation at all times. It contains KeyDescriptors for SSL, signing and encryption certificates and configuration for the different supported services, profiles, bindings and locations as was described in previous section.

The WS-Federation Metadata extension[6] is used to describe the Identity Provider service endpoints that support WS-Federation.

The rest of the services is described using core SAML 2.0 Metadata Specification elements.

Partners that wish to integrate with one of the eHealth Identity Services SHOULD retrieve the metadata from the online location to setup trust for the certificates and to know where and how to contact the services.

For optimization, they SHOULD cache the online metadata into a local resource and schedule an automated update process so the local resource remains synchronised with the online version.

If the `<EntityDescriptor>` contains a `cacheDuration` attribute, partners MUST reload the local cached resource before the cacheDuration is expired.

Example

```
<EntityDescriptor cacheDuration="P0Y0M1DT0H0M0.000S" …>
```

`CacheDuration` is defined in the SAML 2.0 Metadata Specifications as an xml `duration`[7] type and specifies the maximum time that a partner MAY cache a local version of the metadata section on which the `cacheDuration` is defined.

In above example, the partner MUST reload the metadata at least once a day (every 24 hours).

Key roll-over (unavoidable when using X.509 Certificates) is implemented as described in the Metadata Interoperability profile (OASIS Standard)[8]. Partners implementing key trust with this profile as well can rely on the online metadata to contain the correct certificates at all times (if they don't violate against the `cacheDuration`).

4.1.1 IDP

Unique Identifier: http://idp.smals-mvm.be/shibboleth

[6] ***http://docs.oasis-open.org/wsfed/federation/v1.2/ws-federation.pdf, section 3. Federation Metadata.***

[7] ***http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#duration***

[8] ***https://wiki.oasis-open.org/security/SAML2MetadataIOP***

Metadata Location:

- PROD: https://www.ehealth.fgov.be/idp/profile/Metadata/SAML
- ACC: https://wwwacc.ehealth.fgov.be/idp/profile/Metadata/SAML
- INT: https://wwwint.ehealth.fgov.be/idp/profile/Metadata/SAML

Likely changes: SSL Certificate, Signing Key, Encryption Key

Less likely changes: Service Locations and Bindings

Roles & Services[9]:

- IDPSSO
    - ArtifactResolutionService
    - SingleSignOnService
- AttributeAuthority
    - AttributeService
- fed:SecurityTokenServiceType
    - PassiveRequestorEndpoint

4.1.2 AA

Unique Identifier: urn:be:fgov:ehealth:aa

Metadata Location:

- PROD: https://services.ehealth.fgov.be/IAM/Metadata/AA
- ACC: https://services-acc.ehealth.fgov.be/IAM/Metadata/AA
- INT: https://services-int.ehealth.fgov.be/IAM/Metadata/AA

Likely changes: SSL Certificate, Signing Key, Encryption Key

Less likely changes: Service Locations and Bindings

Roles & Services[10]:

- AttributeAuthority
    - AttributeService

4.1.3 STS

Unique Identifier: urn:be:fgov:ehealth:sts:1_0

Metadata Location:

- PROD: https://services.ehealth.fgov.be/IAM/Metadata/STS
- ACC: https://services-acc.ehealth.fgov.be/IAM/Metadata/STS
- INT: https://services-int.ehealth.fgov.be/IAM/Metadata/STS

Likely changes: SSL Certificate, Signing Key, Encryption Key

Less likely changes: Service Locations and Bindings

[9] ***See metadata IDP for supported bindings and locations***
[10] ***See metadata AA for supported bindings and locations***

Roles & Services[11]:

- fed:SecurityTokenServiceType
  - SecurityTokenServiceEndpoint
- AttributeAuthority
  - AttributeService

4.2 Partner

All partners that integrate with one of the eHealth I.AM Identity Services are registered in the I.AM Federation to setup a trust relation, define where and how to reach them, etc.

They are registered with their own `<EntityDescriptor>` containing a unique `entityID`.

For this purpose, partners should fill in the 'eHealth I.AM – Registration'[12] form for every environment in which they want to integrate.

4.2.1 SP

Unique Identifier: to be defined when integrating with eHealth. It must be unique across all partners in the eHealth I.AM Federation.

Metadata Location: not public[13]

Likely changes: SSL Certificate, Signing Key, Encryption Key

Less likely changes: Service Locations and Bindings

Roles & Services:

- SPSSO
  - AssertionConsumerService

[11] ***See metadata STS for supported bindings and locations***

[12] ***See document 'eHealth I.AM – Registration', available on the eHealth Portal.***

[13] ***Shibboleth SPs have the opportunity to make their metadata public using the Metadata Generation Handler (https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPHandler#NativeSPHandler-MetadataGenerationHandler) but the purpose of this handler is*** *NOT* ***to supply other systems with production metadata but rather to assist with testing, and generation of metadata examples useful in understanding how to produce actual metadata.***

5. Processing and Caching Metadata

There are products available that support the use of SAML Metadata to describe keys and supported protocols and profiles. This document's intention is not to give an overview of this. If you are looking for such a product, a good starting point may be the official community gathering place at SAML XML.org[14] which has a products[15] section which lists a few tools that support SAML. Some of them also load and process SAML 2.0 metadata.

Some products supporting WS-Federation 1.2 also provide integration for the SAML Metadata RoleDescriptor extensions, defined in the WS-Federation specification.

Listed below are some tools that eHealth and its partners have used over the years and which have proven themselves to be stable.

You are free to use them or any other tool provided by your software vendor, freely available somewhere else or written by your own organization.

If for any reason you cannot use an implementation of the Metadata Interoperability Profile (OASIS Standard)[16] in your environment, you will need to update manually your configuration when certificates for the eHealth Identity Services are renewed (will surely happen) or locations are updated.

5.1 Shibboleth SP

If you implement Web Browser SSO for your application(s) with a Shibboleth SP[17] and the eHealth IDP, processing and caching online metadata xml is done for you if you configure correctly a `<MetadataProvider type="XML"` …> element in the shibboleth2.xml file.

See the eHealth I.AM cookbook for integration using Shibboleth[18] or the Shibboleth WIKI[19].

5.2 Opensaml

OpenSAML[20] is a set of open source C++ & Java libraries meant to support developers working with the Security Assertion Markup Language (SAML). OpenSAML 2, the current version, supports SAML 1.0, 1.1, and 2.0. Additionally, various development groups have found the framework created to support OpenSAML 2 useful for their own work. They are in the process of integrating their code supporting WS-Addressing, WS-Security, WS-Trust and XACML.

The OpenSAML libraries do not provide a complete SAML identity or service provider. If you are looking for such software you could check out the Shibboleth project instead.

The library does provide functionalities for SAML 2.0 Metadata Fetching and Querying. More information on the Shibboleth WIKI[21].

---

[14] ***http://saml.xml.org/***

[15] ***http://saml.xml.org/products***

[16] ***https://wiki.oasis-open.org/security/SAML2MetadataIOP***

[17] ***http://shibboleth.net/products/service-provider.html***

[18] ***See document 'eHealth I.AM – Shibboleth SP', available on the eHealth Portal***

[19] ***https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPMetadataProvider***

[20] ***https://wiki.shibboleth.net/confluence/display/OpenSAML/Home***

[21] ***https://wiki.shibboleth.net/confluence/display/OpenSAML/OSTwoUserMetadata***