

# SAML Metadata Management for *eduID.cz*

Milan Sova and Jan Tomášek

CESNET a. l. e,  
Zikova 4, 160 00 Praha 6, Czech Republic  
<http://www.cesnet.cz/>

**Abstract.** Metadata management constitutes a task crucial for operating SAML-based identity federations. Providing the federation participants with the description of its peers including information about their identifiers, locations, and, in particular, their public keys, the metadata must be reliable, available, and accurate. In this paper, we discuss several possible strategies for creating a metadata management system. The final section of the paper describes the system developed for metadata management for *eduID.cz*, the Czech academic identity federation.

**Key words:** SAML, metadata, identity federations

## 1 Introduction

SAML metadata [1] constitutes a basis for describing the infrastructure of a SAML-based identity federation. It describes each system entity (a system producing or consuming a SAML message), identifying in particular its role, supported protocols, endpoints for individual services and public keys used for signature verification and encryption.

Every entity (a producer or a consumer of a SAML message) must know its peer metadata description at the time of the communication. Sometimes, for example when the message should contain data encrypted for the peer, the metadata must be available even before the message is produced.

Since the metadata contains cryptographic keys bound to individual entities, its timely, secure, and reliable distribution is essential for the security of the whole infrastructure.

## 2 Metadata Format

The basic element of metadata, the `EntityDescriptor` describes a particular system entity. A metadata message can contain either a `EntityDescriptor` element describing a particular system entity or a list of `EntityDescriptors` enclosed in the `EntitiesDescriptor` element. The content of the root element should be cryptographically signed (using XML Signature [2]) by its issuer. The signature validation is used by the consumer as the base of trust for the metadata content.

The above-mentioned root elements can carry two optional attributes useful for metadata management:

- `validUntil` indicates the absolute time point at which the metadata expires
- `cacheDuration` indicating the maximum period of time for which a consumer should use the metadata from its local cache

Both parameters may be used to control the size of a time window within which the metadata is supposed to be considered valid by its consumer. Since no metadata revocation mechanism exists, the size of the window should be kept reasonably small.

However, the SAML standard does not define any means for conveying the information about metadata age (e. g. a time of signing or a sequential number). This means that a consumer presented with two different versions of a metadata cannot decide which of them should replace and invalidate the other.

### 3 Metadata Publishing Architectures

Within currently deployed identity federations, a central system is usually responsible for collecting metadata that describes all participating entities. The current version of the collection is published as a `EntitiesDescriptor` at some well-known URL. Individual participants are then downloading the aggregated metadata in regular intervals.

This *aggregated* architecture allows for simple metadata update at the consumer side and for simple trust model - the metadata is considered valid as long as it is downloaded from the pre-configured URL and signed by the pre-configured publisher's key.

The *distributed* architecture envisioned by the SAML standard [1] should allow for more flexibility in establishing relations between individual entities. In this model, every entity is responsible for managing and publishing its own metadata (in the form of an `EntityDescriptor` document). The location of the publishing point can be derived from the `entityID`. SAML proposes two methods for the derivation<sup>1</sup>:

- using publishing point URLs as the `entityID`, or
- resolving the publishing point URL from the host part of the `entityID` using the DDDS [3].

The distributed architecture offers significant advantages compared to the aggregated one. The metadata of a particular peer can be obtained on-demand without the need to periodically download a large-size collection of descriptions of entities most of which would probably never be contacted. The dynamic peer discovery could open the tight binding of entities to a particular federation and allow for easier building of services crossing the federations boundaries.

On the other hand, deploying a trust fabric within the distributing model is more complicated. As said before, the validity period of metadata must be reasonably short to achieve timely update which is required especially for cases such as key change. However, whenever a metadata is being re-published it has to be re-signed. Storing the signing key at the metadata publisher, i. e. the entity itself, is not feasible as it is at the same risk when the entity host is compromised as the data it should sign. For this reason, the metadata signing key should be stored on a different system dedicated for this task.

### 4 *eduID.cz* Metadata Management

The main requirements for the metadata management system for *eduID.cz*, the Czech academic identity federation, were to provide a robust service publishing

<sup>1</sup> Note that both methods require the `entityIDs` to conform to the URL syntax.

aggregated metadata based on the descriptions of entities supplied by their authoritative administrators. The system should allow easy delegation of modification rights to individual `EntityDescriptor`s without any intervention of the central system administrator. The signing of the metadata should be performed on a dedicated secured system. The metadata signing and publication procedures must be fully automated so that the validity period of any version of metadata may be limited to a short time, possibly several hours. The system must allow for an upgrade to support the distributed publication architecture in the future while still fulfilling all previously mentioned requirements.

We decided to build a centralized system driven by messages originating from individual metadata administrators. The messages contain either the metadata for a particular entity in the form of an `EntityDescriptor` document or a command delegating rights to modify some entities' metadata to another person. To allow for flexibility, the messages can come through different channels. In the current deployment, the system accepts email messages as well as HTTP POST data. All messages must be signed using personal X.509 certificates issued by CESNET CA. The system support XML Signatures signatures on the `EntityDescriptor` element level and S/MIME and the PKCS#7 HTML form signatures supported by Mozilla-based browsers for email and HTTP POST channels, respectively.

The system archives all delivered messages for auditing purposes. Current versions of `EntityDescriptor`s are being used to periodically produce and publish aggregated metadata for the federation. Thus, the entity administrator is involved only when changing its entity description while the metadata can be published with arbitrarily short validity period.

The following section describes the central part of the system in more details.

## 5 Central System

The central system is responsible for storing, archiving and verification of the metadata elements describing the entities (Service providers, SPs, and Identity providers, IdPs) within the federation. An aggregated federation metadata is being periodically assembled from the stored elements and published as one digitally signed XML file.

The central system provides interfaces for collecting metadata information about entities as well as for issuing commands to delegate modification permissions to specific metadata. The delegation model is hierarchical.

In addition, the central system also provides messaging subsystem which notifies responsible metadata owners about problems with their particular metadata elements. For example, it sends an email notification when the X.509 certificate used included in the entity description element is about to expire.

### 5.1 Collected Data

**Entity Descriptions** The interface of the central system allows storing the metadata elements describing the SPs and IdPs. Each stored metadata element must contain exactly one `EntityDescriptor`. We have extended the SAML metadata schema for management purposes.

The following three attributes within the `http://eduID.cz/metadata` (the `emd` prefix is used for the namespace throughout this document) were added to the `EntityDescriptor` element: `emd:serial` is used for storing serial number (a “version”) of the `EntityDescriptor`, `emd:created` is used for storing the date when the particular element was received by the central system, and `emd:deleted` is used for marking the metadata element as deleted.

Our further enhancement to the SAML standard is the digital signing of the metadata elements by an X509 certificate. User can choose from three signing methods: XML Signature, S/MIME or PKCS#7. The central system stores metadata elements exactly in the same form as they were received, so it is always possible to verify if the stored metadata elements are intact.

**Delegation** The delegation command message contains a `emd:Delegation` element. It defines the scope of the delegation and the persons who should be allowed to modify metadata within that scope. The scope is based on DNS names of the hosts hosting the particular entity instance. When the `scope` attribute of the `emd:Condition` element equals to the string “host”, the the person(s) identified by the `ds:KeyName` element are granted permissions to sign metadata of the entities hosted at the server named within the content of the element. When the value of the `scope` attribute is “zone”, the named person(s) can modify the metadata for entities hosted at servers within the DNS zone specified in the content of the `emd:Condition` element.

The following example describes the delegation to modify metadata for entities hosted at servers in the DNS zone `cesnet.cz` to the authors of this paper.

```
<emd:Delegation xmlns:emd="http://eduID.cz/metadata"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig" emd:serial="2">

  <emd:Condition scope="zone">cesnet.cz</emd:Condition>

  <ds:KeyInfo>
    <ds:KeyName>cn=Milan Sova,o=CESNET,...</ds:KeyName>
    <ds:KeyName>cn=Jan Tomasek,o=CESNET,...</ds:KeyName>
  </ds:KeyInfo>
</emd:Delegation>
```

The designated persons are identified by the subjects of their X.509 certificates; the person issuing the rights is identified by the subject of the certificate used for validating the signature on the message.

Everyone can delegate its own rights to any other administrator by delivering the appropriate message to the central system.

## 5.2 Supported Methods for Submitting Messages

The central system supports two channels for submitting metadata or delegation messages. The first way is to deliver the message in the proper XML format (`EntityDescriptor` or `emd:Delegation`) by email to the central system. The message must be signed either

at the XML level using XML Signature or at the email message level using S/MIME standard. After receiving the email message, its signature is validated and the correctness of the XML message is verified. Correct and properly signed messages are stored in the central system database for further processing. The sender is informed about the state of his/her request by email.

The second option is to send the properly formatted and signed XML message to the HTTP interface of the central server using HTTP POST. The alternative to XML Signature is the `crypto.signText` feature of Mozilla-based browsers. The file is processed in the same way as the email message except that the user is informed about the state of his request via an HTTP response.

### 5.3 Life Cycle of the Metadata Element

The storage of the central system keeps a complete history of each received metadata element. Every revision is stored in the database even when the metadata element is to be deleted. The life cycle of the metadata element goes through 4 possible states.

**State: 0 (prepared)** – The state of any metadata element immediately after it is received. The elements in this state are not published into the federation metadata file.

**State: 1 (active)** – The central system periodically checks all metadata elements in state 0 and if any of them fulfills all conditions for publishing then its state is changed from state 0 to state 1. The conditions for this transition are the following: the metadata element must be the last known revision and has to be signed by an authorized person. Metadata elements in this state are published into the federation metadata file.

**State: 2 (marked)** – In case the signing permissions for a person are revoked, the central system searches for all metadata elements in state 1 signed by that person and changes their state to 2. The person requesting the revocation is informed about all entity descriptors concerned.

The metadata element can be in this transitional state only for a limited amount of time. If, during this period, the metadata element is not replaced by a newer version signed by an authorized person, the state of the metadata element is changed to 3. The metadata elements in state 2 are published into the federation metadata file.

**State: 3 (deleted)** – The metadata element in this state is not published into the federation metadata. Re-activation of the SP or the IdP described by this element is possible only when a new version of the metadata element is submitted to the central system.

Similar procedures apply to the delegation messages.

### 5.4 Federation Metadata Publishing

The procedure of publishing federation metadata is rather simple. Every time when a new metadata element is submitted to the storage of the central system, the storage is marked as dirty. After this the system waits for a short period of time for possible new messages and if none arrives, it starts to work.

First, it builds the complete tree of the delegation permissions, then it scans its storage for all metadata elements in state 0 and checks each found element against the

tree of the delegation permissions if the person who signed this element is authorized to do so. All elements signed by an authorized person are upgraded to state 1.

After that, the system scans its storage for latest versions of metadata elements in state 1 or 2 and pushes them to a ready-to-be published stack. When the scan is completed, the stack is transformed into the federation metadata XML file according to the SAML standard.

Finally, the resulting XML metadata file is digitally signed. The signing key is stored in an Hardware Security Module to protect it against loss even when the machine running the central system is compromised. This contributes to the high level of trust to the federation metadata.

The federation metadata file is published as the XML file on the web site of the federation.

## 6 Future Works

The metadata management system for *eduID.cz* fulfills the initial requirements. It is robust, safe and provides for delegation of responsibility. It can be easily extended to support distributed metadata publication by adding the possibility to export individual `EntityDescriptor` elements so that the entities could install them at their own publishing points. We plan to implement the functionality as soon as the federation software requires it.

## References

1. Cantor, S. et al.: Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, OASIS Open (2005)
2. Eastlake 3rd, D., Reagle, J., Solo, D.: (Extensible Markup Language) XML-Signature Syntax and Processing. RFC 3275, IETF (2002)
3. Mealling, M.: Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS. RFC 3401, IETF (2002)
4. Mealling, M.: Dynamic Delegation Discovery System (DDDS) Part Two: The Algorithm. RFC 3402, IETF (2002)
5. Mealling, M.: Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database. RFC 3403, IETF (2002)
6. Mealling, M.: Dynamic Delegation Discovery System (DDDS) Part Four: The Uniform Resource Identifiers (URI). RFC 3404, IETF (2002)
7. Mealling, M.: Dynamic Delegation Discovery System (DDDS) Part Five: URI.ARPA Assignment Procedures. RFC 3405, IETF (2002)
8. Mealling, M.: Dynamic Delegation Discovery System (DDDS) Part Two: The Algorithm. RFC 3402, IETF (2002)