Metadata for SAML 2.0 Web Browser SSO Profiles

Working Draft 00, 15 September 2003

Document identifier:
sstc-saml-metadata-2.0-draft-00

Location:

http://www.oasis-open.org/apps/org/workgroup/security/download.php

Editor:
Jahan Moreh, Sigaba <jmoreh@sigaba.com>

Contributors:
Prateek Mishra, Netegrity
Jeff Hodges, Sun Microsystems
Charles Knouse, Oblix
Rob Philpott, RSA
Frederick Hirsch, Nokia
Tim Moses, Entrust

Abstract:
The SAML Web Browser SSO Profiles require agreements between source and destination sites about information such as URLs, source and destination IDs, certificates and keys, and so forth. Metadata definitions are useful for describing this information in a standardized way. This document defines metadata that describe the elements and attributes required to use the SAML Web Browser SSO Profiles. Since the Liberty Alliance Web SSO Profiles are directly based on the SAML Web SSO Profiles, the metadata defined in this document borrows extensively from the metadata definitions in the draft Liberty Alliance 1.2 specifications.

Status:
Interim draft. Send comments to the editor.

Committee members should send comments on this specification to the security-services@lists.oasis-open.org list. Others should subscribe to and send comments to the security-services-comment@lists.oasis-open.org list. To subscribe, send an email message to security-services-comment-request@lists.oasis-open.org with the word “subscribe” as the body of the message.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Security Services TC web page (http://www.oasis-open.org/committees/security/).

Table of Contents

1 Introduction

The SAML Web Browser SSO Profiles [SAMLBind] require agreement between a source and destination site about supported protocols, URL of assertion producers and consumers, source and destination IDs, certificates and keys, and so forth. Sources and destinations of SAML Web Browser SSO Profiles can exchange this information via a set of metadata as defined in this document.

The Liberty Alliance 1.2 draft specifications include metadata description and discovery protocols [libMD]. This document specifies elements and attributes from the Liberty 1.2 draft specification that are used to describe the required metadata for entities using the SAML Web Browser SSO Profiles.

1.1 Notation

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this specification are to be interpreted as described in IETF RFC 2119 [RFC2119].

```
Listings of productions or other normative code appear like this.
```

```
Example code listings appear like this.
```

Note: Non-normative notes and explanations appear like this.

Conventional XML namespace prefixes are used throughout this specification to stand for their respective namespaces as follows, whether or not a namespace declaration is present in the example:

The prefix `xsd`: stands for the W3C XML Schema namespace

The prefix `saml:` stands for the SAML assertion namespace [SAMLCore].

The prefix `samlp:` stands for the SAML request-response protocol namespace [SAMLCore].

The prefix `ds:` stands for the W3C XML Signature namespace, `http://www.w3.org/2000/09/xmldsig#` [XMLSig].

1.2 Trust Model

This document specifies a single trust model between assertion producers and assertion consumers. This trust model is based on exchanging public keys in the metadata. These public keys, which may be contained in a standard X.509 certificate, are used by assertion consumers to verify assertions requests, or responses that may be signed by assertion producers. Additional trust models and corresponding credentials are beyond the scope of this specification.

2 Metadata for SAML Web Browser SSO Profiles

The Liberty metadata discovery protocol provides for real-time exchanges of metadata between *providers.* SAML 2.0 does not use Liberty's publishing protocols for real-time exchange of metadata. Rather, SAML source and destination sites must *a priori* have obtained metadata regarding each other. That is, the protocol for exchanging metadata is outside the scope of this specification.

The SAML 2.0 Metadata specification adopts Liberty's terminology for *providers* as follows:

- Identity Provider (IDP) means the *source* of an authentication context. The source performs initial user authentication.
- Service Provider (SP) means the *destination* that provides services and requires an authentication context.

The metadata schema described in this specification allows for a single method of representation. This representation consists of a single document expressing all of the metadata for a single provider identified by one or more `providerID`s. Note that both the IDP and the SP are usually *producers* and *consumers* of metadata.

The metadata schema in this specification **does not allow** for single document describing multiple, distinct providers identified by multiple `providerID`s, or multiple documents which individually describe portions of a single provider's metadata.

Multiple `providerIDs` may be used to identify a single provider. This allows multiple unique identifiers for the provider. This is supported in the metadata schema by using multiple `SPDescriptors` (or multiple `IDPDescriptors`) in a metadata description, each associated with a single providerID.

2.1 Schema declaration

The primary container for a published metadata document is `EntityDescriptor` (see Section 2.2.1 for more details). The immediate child node of `EntityDescriptor` expects one or more of:

`IDPDescriptor` (see Section 2.1.6 for more details)

`SPDescriptor` (see Section 2.1.7 for more details)

2.1.1 Namespaces in metadata

The SAML metadata structures are modeled after the metadata schema specified by the Liberty Alliance Project (see [libMD] for additional details). The namespaces referenced in [libMD] that are used in SAML metadata structures are:

`ds`: is described by the W3C XML signature schema [XMLSig]

`saml`: is described by SAML assertion namespace [SAMLCore]

The namespaces referenced in [libMD] that are not used in SAML metadata structures are:

`isf`: the Liberty Services Framework schema

`pip`: the Liberty Personal Information Profile schema

The following schema fragment illustrates the use of namespaces in SAML metadata documents:

```
<xsd:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:metadata"
xmlns="urn:oasis:names:tc:SAML:2.0:metadata "
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
```

2.1.2 Data type `entityIDType`

The data type `entityIDType` is adopted without modification from **[libMD]**. Basically, `entityIDType` restricts the XML schema datatype `anyURI` to a length of 1024 bytes. Within this specification `entityIDType` is used to describe a unique ID attribute for providers. The schema fragment for `entityIDType` is:

```
<xsd:simpleType name="entityIDType">
  <xsd:restriction base="xsd:anyURI">
    <xsd:maxLength id="maxlengid" value="1024" />
    <xsd:pattern value="" />
  </xsd:restriction>
</xsd:simpleType>
```

2.1.3 Common attributes

SAML metadata specification adopts two common attributes from **[libMD]**:

`providerID` of type `entityIDType` indicates the providerID of the provider, described by the children of the node.

`validUntil` of type `dateTime` indicates expiration date and time of the node and its descendants. If `dateTime` expressions evaluate to nonequivalent values, parsers MUST adhere to the most restrictive value (i.e., the earliest dateTime).

The schema fragment for `providerID` and `validUntil` is:

```
  <xsd:attribute name="providerID" type="entityIDType" />
  <xsd:attribute name="validUntil" type="xsd:dateTime" />
```

2.1.4 Common elements

SAML 2.0 metadata specification adopts common elements from **[libMD]**as described in the following subsections.

2.1.4.1 Element `Extension`

The `Extension` element allows for arbitrary content extensions from other namespaces. Providers can use this element to exchange any metadata that is not defined by this specification. The schema fragment for `Extension` is:

```
<xsd:element name="Extension" type="extensionType" />
```

```
<xsd:complexType name="extensionType">
 <xsd:sequence >
      <xsd:any namespace="##other" processContents="lax" maxOccurs="unbounded"
/>
  </xsd:sequence>
</xsd:complexType>
```

2.1.4.2 Element Organization

The Organization element provides some basic information about the provider. This optional element is informatory in nature and does not directly map to any SAML elements or attributes. The Organization element consists of the following:

Element OrganizationName of type string specifies the organizational name of the provider. Exactly one OrganizationName MUST be specified.

Element OrganizationDisplayName of type string specifies the organizational name of the provider suitable for display. Exactly one OrganizationDisplayName MUST be specified.

Element OrganizationURL of type anyURI specifies a URL for the organization and may be used to direct a principal to the provider for additional information. Exactly one OrganizationURL MUST be specified.

Element Extension of type extensionType allows for inclusion of additional arbitrary information about the organization. This is an optional element.

The schema fragment for Organization is:

```
<xsd:element name="Organization" type="organizationType" />
<xsd:complexType name="organizationType">
<xsd:sequence>
    <xsd:element name="OrganizationName" type="xsd:string" />
    <xsd:element name="OrganizationDisplayName" type="xsd:string" />
    <xsd:element name="OrganizationURL" type="xsd:anyURI" />
    <xsd:element ref="Extension" minOccurs="0" />
  </xsd:sequence>
</xsd:complexType>
```

2.1.5 Base type providerDescriptorType

The providerDescriptorType generically describes metadata for SAML identity and service providers. The providerDescriptorType is extended by IDPDescriptorType and SPDescriptorType, which further describe specific metadata about identity and service providers respectively (See Sections 2.1.6 and 2.1.7 for more details).

The type providerDescriptorType consists of the following attributes and elements:

providerID [required]

validUntil [required]

protocolSupportEnumeration [required]

id [optional]

ds:keyInfo [optional]

Organization [optional]

Extension [optional]

ds:Signature [optional]

The schema fragment for providerDescriptorType is:

```
<xsd:complexType name="providerDescriptorType">
 <xsd:sequence>
  <xsd:element ref="ds:KeyInfo" minOccurs="0" />
  <xsd:element ref="Organization" minOccurs="0" />
  <xsd:element ref="Extension" minOccurs="0" />
  <xsd:element ref="ds:Signature" minOccurs="0" />
 </xsd:sequence>
 <xsd:attribute ref="providerID" use="required" />
 <xsd:attribute ref="validUntil" use="required" />
 <xsd:attribute name="protocolSupportEnumeration" type="xsd:NMTOKENS"
use="required" />
  <xsd:attribute name="id" type="xsd:ID" use="optional" />
</xsd:complexType>
```

2.1.5.1 Attribute providerID

The attribute providerID specifies a unique ID for the provider. The providerID MUST be unambiguous to the entities that process the metadata (since the providerID is a URI, the designated value should be unambiguous regardless of context).

The following additional rule applies to Identity Providers whose saml:issuer attribute is a URI. The providerID attribute MUST be the same as the saml:issuer attribute specified in Section 2.3 of **[SAMLCore]**. Note that the preceding rule is consistent with the specification in Section 2.3 of **[SAMLCore]** wherein the normative specification for saml:issuer is: *The issuer name* SHOULD be unambiguous to the intended relying parties. SAML authorities may use an identifier such as a URI reference that is designed to be unambiguous regardless of context.

2.1.5.2 Attribute protocolSupportEnumeration

The attribute protocolSupportEnumeration describes the protocol release type of the provider described by providerID. This attribute is of the type NMTOKEN, which allows for the enumeration of a set of SAML protocol releases which the provider MUST support. The data type of the tokens MUST be URNs and the value must be one of the following:

urn:oasis:names:tc:SAML:1.0:protocol

urn:oasis:names:tc:SAML:1.1:protocol

urn:oasis:names:tc:SAML:2.0:protocol

2.1.5.3 Attribute validUntil

The attribute validUntil indicates the date and time beyond which the metadata can not be guaranteed as being valid. Entities that process a provider's metadata MUST reject the metadata as stale if the value of this attribute evaluates to a date and time before the current date and time.

2.1.5.4 Attribute `id`

The attribute `id` specifies the fragment identifier for the instance node. This attribute is optional, unless the node is being signed. For signed nodes an id attribute MUST be specified.

2.1.5.5 Element `ds:keyInfo`

The element `ds:keyInfo` contains the provider's public key in a format allowed by **[XMLSig]**. The provider's public key is used for verifying signed assertions, requests, and responses.

2.1.5.6 Element `Organization`

This element is used to provide information about the provider's organization. See Section 2.1.4.2 for details.

2.1.5.7 Element `Extension`

This element may be used to specify additional metadata about the provider. An entity that does not understand the value of this element SHOULD reject the metadata as unacceptable.

2.1.5.8 Element `ds:Signature`

A provider may digitally sign its metadata. The element `ds:Signature` specifies a signature that is conformant with the specification in **[XMLSig]**. Note that the public key that verifies the signature of the metadata document must be exchanged out of band. That is, the public key in the metadata document, as described in Section 2.1.5.5 SHOULD only be used for verifying assertions, requests, and responses.

2.1.6 Element `IDPDescriptor`

The element `IDPDescriptor` is of type `IDPDescriptorType`, which extends `providerDescriptorType` with the following attributes and elements:

SoapEndPoint [required]

SingleSignonServiceURL [required]

SingleSignonProtocolProfile [required]

`IssuerID` [optional]

The schema fragment for `IDPDescriptor` is:

```
<xsd:complexType name="IDPDescriptorType">
 <xsd:complexContent>
   <xsd:extension base="providerDescriptorType">
    <xsd:sequence>
      <xsd:element name="SoapEndpoint" type="xsd:anyURI" />
      <xsd:element name="SingleSignOnServiceURL" type="xsd:anyURI" />
      <xsd:element name="SingleSignOnProtocolProfile"
                    type="xsd:anyURI" maxOccurs="unbounded" />

    </xsd:sequence>
     <xsd:attribute name="IssuerID" type="xsd:string"
use="optional" />
   </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>
```

The element IDPDescriptor does not explicitly contain a SAML Artifact Source ID as specified in Section 4.1.1.8 of [SAMLBind]. In order to construct the Source ID, providers MUST use the following rules:

Each provider selects the identity provider's metadata attribute `providerID`,

The provider constructs the Source ID component of the artifact by taking the SHA-1 hash of the metadata attribute `providerID`, which results in a 20-byte binary value. Note that Source ID value, used to construct the artifact, is not encoded in hexadecimal.

2.1.6.1 Element `SoapEndpoint`

The element `SoapEndpoint` specifies the URL for the SAML SOAP Binding Service at the identity provider's site (as described in [SAMLBind] section 3.1.3).

2.1.6.2 Element `SingleSignOnServiceURL`

The element `SingleSignOnServiceURL` specifies the URL for the identity provider's inter-site transfer service (as described in [SAMLBind], sections 4.1.1.3 and 4.1.2.3)

2.1.6.3 Element `SingleSignonProtocolProfile`

The element `SingleSignonProtocolProfile` specifies the SAML browser profile(s) supported by the identity provider. Each instance of this element MUST have a value correspond to one of the URIs specified in Section 4.1.1.1 or 4.1.2.1 of [SAMLBind].

2.1.6.4 Attribute `IssuerID`

The attribute `IssuerID` specifies the issuer ID of the assertion as described in Section 2.3 of [SAMLBind]. This is an optional attribute. When this attribute is not specified, a service provider MUST use the value of the attribute `providerID` to identify assertions issued by this Identity Provider (See section 2.1.5.1).

In order to maintain compatibility with the Liberty metadata specifications, it is RECOMMENDED that Identity Providers not include this attribute in their metadata specification. Instead, Identity Providers should use a URI for the value of their `saml:issuer` and specify the same URI as the value for the attribute `providerID` (See section 2.1.5.1). Specifying a URI for the value of `saml:issuer` is consistent with the recommendations in Section 2.3 of [SAMLBind].

2.1.7 Element `SPDescriptor`

The element `SPDescriptor` is of type `SPDescriptorType`, which extends `providerDescriptorType` with the following element:

AssertionConsumerServiceURL [required]

SingleSignonProtocolProfile [required]

The schema fragment for `SPDescriptor` is:

```
<xsd:element name="SPDescriptor" type="SPDescriptorType" />
<xsd:complexType name="SPDescriptorType">
 <xsd:complexContent>
  <xsd:extension base="providerDescriptorType">
     <xsd:sequence>

      <xsd:element name="AssertionConsumerServiceURL" type="xsd:anyURI" />
```

```
      <xsd:element name="SingleSignOnProtocolProfile" type="xsd:anyURI"  />
    </xsd:sequence>
  </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>
```

2.1.7.1 Element AssertionConsumerServiceURL

The element AssertionConsumerServiceURL is a URL consisting of a host name and a path. For service providers that use SAML Browser/Artifact profile, the value of AssertionConsumerServiceURL MUST be set to the artifact receiver host name and path (section 4.1.1.5 of **[SAMLBind]**).

For service providers that use SAML Browser/POST profile, the value of AssertionConsumerServiceURL MUST be set to the assertion consumer host name and path (section 4.1.2.4 of **[SAMLBind]**).

Service providers that support both SAML Browser/Artifact and Browser/POST profiles SHOULD provide separate SPDescriptors, each having a different providerID and a different AssertionConsumerServiceURL. The two SPDescriptors MAY be encapsulated in a single EntityDescriptor.

2.1.7.2 Element SingleSignonProtocolProfile

The element SingleSignonProtocolProfile specifies the SAML browser profile(s) supported by the service provider. Its value MUST be one of the URNs specified in Section 4.1.1.1 or 4.1.2.1 of **[SAMLBind]**.

2.2 Entity Descriptors

An entity descriptor is a container for one or more provider descriptors for a single provider. It is permissible for an entity descriptor to contain more than one descriptor for the same provider. For example, a service provider that supports both Browser/Artifact and Browser/POST profiles may specify its metadata in two service provider descriptors and encapsulate both in a single entity descriptor. However, an entity descriptor MUST NOT contain descriptors from multiple providers.

2.2.1 Element EntityDescriptor

The element EntityDescriptor is a container for one or more provider descriptors. The schema fragment for EntityDescriptor is:

```
<xsd:element name="EntityDescriptor" type="entityDescriptorType" />
<xsd:complexType name="entityDescriptorType" mixed="false">
   <xsd:sequence maxOccurs="unbounded">
    <xsd:element ref="SPDescriptor" minOccurs="0" />
    <xsd:element ref="IDPDescriptor" minOccurs="0" />
   </xsd:sequence>
</xsd:complexType>
```

3 Examples

This section provides examples of metadata for Source (Identity Provider) and Destination (Service Provider).

3.1 Identity Provider

The example below illustrates the metadata for an Identity Provider that supports SAML Browser/Artifact and Browser/POST profiles. This metadata is not signed.

```
<EntityDescriptor>
 <IDPDescriptor
   providerID="http://www.IDP-bpa.net/SSOMechanism"
   validUntil="2003-10-11T00:33:32Z"
   protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol" >
   <ds:KeyInfo>
     <ds:X509Data>
      <ds:X509Certificate>
          MIICYjCCAiECAQAwCQYHKoZIzjgEAzAaMRgwFgYDVQQDEw9
          zaWdhYmEtc3BhLXNpZ24wHhcNMDMwMzE4MjIwMzU5WhcNMDM
          wNDE3MjIwMzU5WjAaMRgwFgYDVQQDEw9zaWdhYmEtc3BhLXNp
          ……
          ……
      </ds:X509Certificate>
     </ds:X509Data>
  </ds:KeyInfo>
   <Organization>
          <OrganizationName>Id Broker Corporation</OrganizationName>
          <OrganizationDisplayName>
                 ID Broker Corp.
          </OrganizationDisplayName>
          <OrganizationURL>http://www.idbcorpusa.net</OrganizationURL>
   </Organization>
   <SoapEndPoint>https://www.IDP-bpa.net/serv/soaper</SoapEndPoint>
   <SingleSignOnServiceURL>
          https://www.IDP-bpa.net/serv/sso
   </SingleSignOnServiceURL>
   <SingleSignOnProtocolProfile>
          urn:oasis:names:tc:SAML:1.0:profiles:artifact
   </SingleSignOnProtocolProfile>
  <SingleSignOnProtocolProfile>
          urn:oasis:names:tc:SAML:1.0:profiles:browser-post
   </SingleSignOnProtocolProfile
 </IDPDescriptor>
</EntityDescriptor>
```

3.2 Service Provider

The example below illustrates the metadata for a Service Provider that only supports SAML Browser/Artifact profile. This metadata is not signed.

```
<EntityDescriptor>
 <SPDescriptor
   providerID="http://www.SP-bpa.net/BA-ResourceProviderMechanism"
   validUntil="2003-08-11T00:13:36Z"
   protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
```

```
   <ds:KeyInfo>
     <ds:X509Data>
      <ds:X509Certificate>
          MIICYjCCAiECAaAwCQYHKoZIzjgEAfAaMRgwF9YDVQQDEw9
          zaWdhYmEtc3BhLyNpZ24wH2cNMDMwMzE4MjIwMzU5WhcNMDM
          wlDE3MjIwMzU5WjAaMRgwFgYDVQQDEw9zaWdhYmEtc3BhLXNp
          ……
          ……
      </ds:X509Certificate>
     </ds:X509Data>
   </ds:KeyInfo>
    <Organization>
          <OrganizationName>
                Exemplary Resource Provider
          </OrganizationName>
          <OrganizationDisplayName>
                Exemplart RP, Inc.
          </OrganizationDisplayName>
          <OrganizationURL>http://www.sp-bpa.net</OrganizationURL>
    </Organization>
    <AssertionConsumerServiceURL>
          https://www.SP-bpa.net/serv/BA-SSOconsumer
    </AssertionConsumerServiceURL >
    <SingleSignOnProtocolProfile>
          urn:oasis:names:tc:SAML:1.0:profiles:artifact
    </SingleSignOnProtocolProfile>
  </SPDescriptor>
</EntityDescriptor>
```

The example below illustrates the metadata for a Service Provider that supports SAML Browser/Artifact and Browser/POST profiles. This metadata is not signed.

In this example the Service Provider specifies two `SPDescriptors`, each having a unique `providerID`. This is to allow the Identity Providers to discern between the Browser/Artifact `AssertionConusmerURL` and the Browser/POST `AssertionConsumerURL`.

```
<EntityDescriptor>
 <SPDescriptor
   providerID=”http://www.SP-bpa.net/BA-ResourceProviderMechanism”
   validUntil=”2003-08-11T00:13:36Z”
   protocolSupportEnumeration=”urn:oasis:names:tc:SAML:2.0:protocol”>
   <ds:KeyInfo>
     <ds:X509Data>
      <ds:X509Certificate>
          MIICYjCCAiECAaAwCQYHKoZIzjgEAfAaMRgwF9YDVQQDEw9
          zaWdhYmEtc3BhLyNpZ24wH2cNMDMwMzE4MjIwMzU5WhcNMDM
          wlDE3MjIwMzU5WjAaMRgwFgYDVQQDEw9zaWdhYmEtc3BhLXNp
          ……
          ……
      </ds:X509Certificate>
     </ds:X509Data>
   </ds:KeyInfo>
    <Organization>
          <OrganizationName>
                Exemplary Resource Provider
          </OrganizationName>
          <OrganizationDisplayName>
                Exemplart RP, Inc.
          </OrganizationDisplayName>
```

```
          <OrganizationURL>http://www.sp-bpa.net</OrganizationURL>
    </Organization>
    <AssertionConsumerServiceURL>
          https://www.SP-bpa.net/serv/BA-SSOconsumer
    </AssertionConsumerServiceURL >
    <SingleSignOnProtocolProfile>
          urn:oasis:names:tc:SAML:1.0:profiles:artifact
    </SingleSignOnProtocolProfile>
  </SPDescriptor>
 <SPDescriptor
    providerID="http://www.SP-bpa.net/BPResourceProviderMechanism"
    validUntil="2003-08-11T00:13:36Z"
    protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <ds:KeyInfo>
      <ds:X509Data>
       <ds:X509Certificate>
          MIICYjCCAiECAaAwCQYHKoZIzjgEAfAaMRgwF9YDVQQDEw9
          zaWdhYmEtc3BhLyNpZ24wH2cNMDMwMzE4MjIwMzU5WhcNMDM
          wlDE3MjIwMzU5WjAaMRgwFgYDVQQDEw9zaWdhYmEtc3BhLXNp
          ……
          ……
       </ds:X509Certificate>
      </ds:X509Data>
   </ds:KeyInfo>
    <Organization>
          <OrganizationName>
                Exemplary Resource Provider
          </OrganizationName>
          <OrganizationDisplayName>
                Exemplart RP, Inc.
          </OrganizationDisplayName>
          <OrganizationURL>http://www.sp-bpa.net</OrganizationURL>
    </Organization>
    <AssertionConsumerServiceURL>
          https://www.SP-bpa.net/serv/BP-SSOconsumer
    </AssertionConsumerServiceURL >
    <SingleSignOnProtocolProfile>
          urn:oasis:names:tc:SAML:1.0:profiles:browser-post
    </SingleSignOnProtocolProfile>
  </SPDescriptor>
</EntityDescriptor>
```

4 Validating Metadata Over An Untrusted Channel

The text below is a copy of an email from Tim Moses. It is intended to generate discussion regarding metadata exchange.

4.1 Introduction

In the case that the metadata is not communicated between the source and destination over an authentic channel, the destination's copy of the metadata is not trustworthy. Yet, the trustworthiness of all subsequent communications between the source and destination depends upon the trustworthiness of the metadata. Therefore, a technique is required to validate the destination's copy of the metadata.

Metadata validation MUST be performed using an alternative authentic channel, such as a telephone call or facsimile transmission, company letterhead, business card or face-to-face exchange. In order to minimize the cost of communicating and validating metadata, it MUST be possible to validate metadata verbally and reliably in a telephone call.

4.2 Procedure

The following procedure is RECOMMENDED.

1. The source SHALL calculate a validation string for the metadata according to the procedure described below.
2. The source SHALL communicate the validation string over an authentic channel to the destination.
3. The source SHALL communicate the metadata over an untrusted channel to the destination.
4. The destination SHALL calculate the validation string for the received metadata according to the procedure described below.
5. If the validation string calculated by the destination is identical to the validation string received from the source, then the destination MAY install and rely upon the metadata, otherwise it MUST NOT rely on the metadata.

4.3 Validation string calculation

The validation string SHALL be calculated from the metadata by operating upon it with the SHA-1 hash algorithm. The calculation shall be performed over the <EntityDescriptor> element, starting with the "<" character of the start tag and finishing with the ">" character of the end tag. The right-most 4 bytes of the resulting digest are discarded. The left-most 3 bits of each of the remaining 16 bytes are discarded. The remaining sixteen 5-bit values are represented as alphanumeric characters according to the following table.

00000 > A

00001 > B

... omitting I

11000 > Z

11001 > 3

11010 > 4

591 ...

592 11111 > 9

593

594 Finally, the alphanumeric string is divided into four sub-strings, each of four characters, and the
595 sub-strings are separated by hyphens. For example: A4HY-8KLN-9T3M-7GK4

596

597 Note that hyphens are inserted to assist in dictation. Note also that the ambiguities between I
598 and 1, O and 0 and Z and 2 are eliminated. Also, note that the ambiguity between upper and
599 lower-case letters is eliminated.

5 References

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[SAMLBind] P. Mishra (Editor), Bindings and Profiles for the OASIS *Security Assertion Markup Language (SAML)*, Committee Specification 01, available from http://www.oasis-open.org/committees/security, OASIS, May 2002.

[SAMLCore] P. Hallam-Baker, P., and E. Maler, (Editors), *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)*, Committee Specification 01, available from http://www.oasis-open.org/committees/security, OASIS, May 2002.

[SAMLReqs] D. Platt et al., SAML Requirements and Use Cases, OASIS, December 2001.

[SAMLSecure] Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML), http://www.oasis-open.org/committees/security/docs/cs-sstc-sec-consider-01.doc.

[XMLSig] D. Eastlake et al., XML-Signature Syntax and Processing, http://www.w3.org/TR/xmldsig-core/, World Wide Web Consortium.

[libMD] P. Davis (Editor), Liberty Alliance Project: Metadata description and discovery protocols, DRAFT version: 1.0-06, 13 April 2003.

[**LAProtSchema**] John D. Beatty, John Kemp, Liberty Protocols and Schemas Specification, Draft Version 1.1-05, November 2002.

[**SAMLInterOp**] Prateek Mishra, Proposed InterOp Scenario for SAML at Catalyst 2002, April 26, 2002, available at http://lists.oasis-open.org/archives/saml-dev/200206/msg00209.html

[**SAMLMETA-XSD**] sstc-saml-1.1-schema-meta-data-draft-03.xsd.xml

Appendix A. Revision History

Rev	Date	By Whom	What
00	2003-09-15	Jahan Moreh	Initial draft based on Draft 07 of SAML 1.1 Metadata specification.

Appendix B. Issues list and resolution

Issue	Resolution
Section 2.1 states: "The immediate child node of `EntityDescriptor` expects one or more of: `IDPDescriptor` (see Section 2.1.6) `SPDescriptor` (see Section 2.1.7)" Rob P. questions: Doesn't "or more" conflict with the representation definition given previously?	Added text (just before section 2.1) to clarify as follows: Multiple `providerIDs` may be used to identify a single provider. This allows multiple unique identifiers for the provider. This is supported in the metadata schema by using multiple `SPDescriptors` (or multiple `IDPDescriptors`) in a metadata description, each associated with a single providerID.
Section 2.5.1. states that "In case the provider is an Identity Provider, the value of the `providerID` MUST be the same as the value of the `Issuer` attribute specified in Section 2.3 of [SAMLCore]." Rob P. comments that: "The issue here is that SAML "Issuer" is a string; NOT a URI. Thus, some SAML Issuer's can't be described in providerID.	Added an optional attribute called IssuerID to the element <IDPDescriptor>. Added text recommending that the IssuerID should not be specified as it is expected that saml:issuer be a URI (per SAMLCore Section 2.3). In summary, a source (Identity Provider) could specify an issuer ID, but it need not (and it is recommended that it should not).
Section 2.1.5.2.Attribute `protocolSupportEnumeration` The attribute `protocolSupportEnumeration` describes the protocol release type of the provider described by `providerID`. This attribute is of the type NMTOKEN, which allows for the enumeration of a set of SAML protocol releases which the provider MUST support. The data type of the tokens MUST be URNs and the value must be one of the following: urn:oasis:names:tc:SAML:1.0 urn:oasis:names:tc:SAML:1.1 Rob P asks: It's not clear to me whether this should identify the protocol version urn(s) or the assertion urn(s) or both.	It should be the protocol URN. Note that there is another element called SingleSignonProtocolProfile (Section 2.1.6.3), which is (currently) either the Brower/Post URN or Browser/Artifact URN. The value of this element pretty much dictates the acceptable assertions. I.e., there is no need to identify assertion urns.
Section 2.1.5.8 The element `ds:keyInfo` contains the provider's public key in a format allowed by [XMLSig]. The provider's public key is used for	Resolution: Added text to clarify signing request/responses: Re: signing metadata: The public key that verifies the metadata is NOT in the metadata

verifying signed assertions, requests, and responses. Rob P. asks: What about requests and responses which can also be signed? For Requests, the provider is the Service Provider. What about the key used to sign the metadata (in ds:Signature below)?	document; added text to clarify this
Section 2.1.6 The provider constructs the Source ID component of the artifact by taking the SHA-1 hash of the metadata attribute `providerID`, which results in a 20-byte binary value. Note that Source ID value, used to construct the artifact, is not encoded in hexadecimal. Rob P. states: SAML defines the SourceID as the SHA-1 hash of the SOAP Binding Service URL (i.e. the SoapEndPoint). Section 2.1.5.1 above declares that the ProviderID of an IdP must be the Issuer.	Possible Resolution: Based on the RECOMMENDATION in the 1.0 binding document (lines 561-569) the source ID is a sha-1 hash of the source's identification URL. I read this as consistent with the providerID. And, yes, Liberty does use the SHA-1 hash of providerID as the source ID. See liberty-architecture-bindings-profiles-v1.1.pdf line 863-869

632
633
634
635
636

Appendix C. TO Dos

Assign publicly available document location

Appendix D. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.