Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0

Committee Draft 02e, 11 November 2004

Document identifier:

sstc-saml-metadata-2.0-cd-02e

Location:

http://www.oasis-open.org/committees/documents.php?wg_abbrev=security

Editors:

Scott Cantor, Internet2
Jahan Moreh, Sigaba
Rob Philpott, RSA Security
Eve Maler, Sun Microsystems

SAML V2.0 Contributors:

Conor P. Cahill, AOL
Hal Lockhart, BEA Systems
Michael Beach, Boeing
Rick Randall, Booze, Allen, Hamilton
Tim Alsop, CyberSafe Limited
Nick Ragouzis, Enosis
John Hughes, Atos Origin
Paul Madsen, Entrust
Irving Reid, Hewlett-Packard
Paula Austel, IBM
Maryann Hondo, IBM
Michael McIntosh, IBM
Tony Nadalin, IBM
Scott Cantor, Internet2
RL 'Bob' Morgan, Internet2
Rebekah Metz, NASA
Prateek Mishra, Netegrity
Peter C Davis, Neustar
Frederick Hirsch, Nokia
John Kemp, Nokia
Charles Knouse, Oblix
Steve Anderson, OpenNetwork
John Linn, RSA Security
Rob Philpott, RSA Security
Jahan Moreh, Sigaba
Anne Anderson, Sun Microsystems
Jeff Hodges, Sun Microsystems
Eve Maler, Sun Microsystems

Ron Monzillo, Sun Microsystems
Greg Whitehead, Trustgenix

Abstract:

SAML profiles require agreements between system entities regarding identifiers, binding support and endpoints, certificates and keys, and so forth. A metadata specification is useful for describing this information in a standardized way. This document defines an extensible metadata format for SAML system entities, organized by roles that reflect SAML profiles. Such roles include that of Identity Provider, Service Provider, Affiliation, Attribute Authority, Attribute Consumer, and Policy Decision Point.

Status:

This is a **second Committee Draft** approved by the Security Services Technical Committee on 21 September 2004.

Committee members should submit comments and potential errata to the security-services@lists.oasis-open.org list. Others should submit them by filling out the web form located at http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security. The committee will publish on its web page (http://www.oasis-open.org/committees/security) a catalog of any changes made to this document.

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 web page for the Security Services TC (http://www.oasis-open.org/committees/security/ipr.php).

Table of Contents

1 Introduction 5
1.1 Notation 5
2 Metadata for SAML V2.0 6
2.1 Namespaces 6
2.2 Common Types 7
2.2.1 Simple Type entityIDType 7
2.2.2 Complex Type EndpointType 7
2.2.3 Complex Type IndexedEndpointType 8
2.2.4 Complex Type localizedNameType 8
2.2.5 Complex Type localizedURIType 9
2.3 Root Elements 9
2.3.1 Element <EntitiesDescriptor> 9
2.3.2 Element <EntityDescriptor> 10
2.3.2.1 Element <Organization> 12
2.3.2.2 Element <ContactPerson> 12
2.3.2.3 Element <AdditionalMetadataLocation> 14
2.4 Role Descriptor Elements 14
2.4.1 Element <RoleDescriptor> 14
2.4.1.1 Element <KeyDescriptor> 15
2.4.2 Complex Type SSODescriptorType 16
2.4.3 Element <IDPSSODescriptor> 17
2.4.4 Element <SPSSODescriptor> 18
2.4.4.1 Element <AttributeConsumingService> 19
2.4.4.2 Element <RequestedAttribute> 19
2.4.5 Element <AuthnAuthorityDescriptor> 20
2.4.6 Element <PDPDescriptor> 21
2.4.7 Element <AttributeAuthorityDescriptor> 21
2.5 Element <AffiliationDescriptor> 22
2.6 Examples 23
3 Signature Processing 27
3.1 XML Signature Profile 27
3.1.1 Signing Formats and Algorithms 27
3.1.2 References 27
3.1.3 Canonicalization Method 27
3.1.4 Transforms 28
3.1.5 KeyInfo 28
4 Metadata Publication and Resolution 29
4.1 Publication and Resolution via Well-Known Location 29
4.1.1 Publication 29
4.1.2 Resolution 29
4.2 Publishing and Resolution via DNS 29
4.2.1 Publication 30
4.2.1.1 First Well Known Rule 30
4.2.1.2 The Order Field 30
4.2.1.3 The Preference Field 30
4.2.1.4 The Flag Field 31
4.2.1.5 The Service Field 31

113 4.2.1.6 The Regex and Replacement Fields....31
114 4.2.2 NAPTR Examples....32
115 4.2.2.1 Entity Metadata NAPTR Examples....32
116 4.2.2.2 Name Identifier Examples....32
117 4.2.3 Resolution....32
118 4.2.3.1 Parsing the Unique Identifier....32
119 4.2.3.2 Obtaining Metadata via the DNS....33
120 4.2.4 Metadata Location Caching....33
121 4.3 Post-Processing of Metadata....33
122 4.3.1 Metadata Instance Caching....33
123 4.3.2 Handling of HTTPS Redirects....33
124 4.3.3 Processing of XML Signatures and General Trust Processing....33
125 4.3.3.1 Processing Signed DNS Zones....34
126 4.3.3.2 Processing Signed Documents and Fragments....34
127 4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL....34
128 5 References....35
129 6 Registration of MIME media type application/samlmetadata+xml....36
130 Appendix A. Acknowledgments....40
131 Appendix B. Notices....42

1 Introduction

SAML profiles require agreements between system entities regarding identifiers, binding support and endpoints, certificates and keys, and so forth. A metadata specification is useful for describing this information in a standardized way. This specification defines an extensible metadata format for SAML system entities, organized by roles that reflect SAML profiles. Such roles include that of SSO Identity Provider, SSO Service Provider, Affiliation, Attribute Authority, Attribute Requester, and Policy Decision Point.

This specification further defines profiles for the dynamic exchange of metadata among system entities, which may be useful in some deployments.

The SAML conformance document [SAMLConform] lists all of the specifications that comprise SAML V2.0.

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: Notes like this are sometimes used to highlight non-normative commentary.

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:

Prefix	XML Namespace	Comments
`saml:`	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace [SAMLCore]. The prefix is generally elided in mentions of SAML assertion-related elements in text.
`samlp:`	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace [SAMLCore]. The prefix is generally elided in mentions of XML protocol-related elements in text.
`md:`	urn:oasis:names:tc:SAML:2.0:metadata	This is the SAML V2.0 metadata namespace, defined in a schema [SAMLMeta-xsd].
`ds:`	http://www.w3.org/2000/09/xmldsig#	This is the XML Signature namespace [XMLSig].
`xenc:`	http://www.w3.org/2001/04/xmlenc#	This is the XML Encryption namespace [XMLEnc].
`xs:`	http://www.w3.org/2001/XMLSchema	This namespace is defined in the W3C XML Schema specification . In schema listings, this is the default namespace and no prefix is shown. For clarity, the prefix is generally shown in specification text when XML Schema-related constructs are mentioned.

2 Metadata for SAML V2.0

SAML metadata is organized around an extensible collection of roles representing common combinations of SAML protocols and profiles supported by system entities. Each role is described by an element derived from the extensible base type of `RoleDescriptor`. Such descriptors are in turn collected into the `<EntityDescriptor>` container element, the primary unit of SAML metadata. An entity might alternatively represent an affiliation of other entities, such as an affiliation of service providers. The `<AffiliationDescriptor>` is provided for this purpose.

Such descriptors may in turn be aggregated into nested groups using the `<EntitiesDescriptor>` element.

A variety of security mechanisms for establishing the trustworthiness of metadata can be supported, particularly with the ability to individually sign most of the elements defined in this specification.

Note that when elements with a parent/child relationship contain common attributes, such as caching or expiration information, the parent element takes precedence (see also Section 4.3.1).

> **Note**: As a general matter, SAML metadata is not to be taken as an authoritative statement about the capabilities or options of a given system entity. That is, while it should be accurate, it need not be exhaustive. The omission of a particular option does not imply that it is or is not unsupported, merely that it is not claimed. As an example, a SAML attribute authority or identity provider might support any number of attributes not named in an `<AttributeAuthorityDescriptor>`. Omissions might reflect privacy or any number of other considerations. Conversely, indicating support for a given attribute does not imply that a given requester can or will receive it.

2.1 Namespaces

SAML Metadata uses the following namespace (defined in a schema [SAMLMeta-xsd]):

```
urn:oasis:names:tc:SAML:2.0:metadata
```

This specification uses the namespace prefix `md:` to refer to the namespace above.

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

```
<schema
    targetNamespace="urn:oasis:names:tc:SAML:2.0:metadata"
    xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
schema.xsd"/>
    <import namespace="http://www.w3.org/2001/04/xmlenc#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-
20021210/xenc-schema.xsd"/>
    <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
        schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
    <import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="http://www.w3.org/2001/xml.xsd"/>
    <annotation>
```

```
        <documentation>
            Document identifier: sstc-saml-schema-metadata-2.0
            Location: http://www.oasis-
open.org/committees/documents.php?wg_abbrev=security
            Revision history:
            V2.0 (August, 2004):
              Schema for SAML metadata, first published in SAML 2.0.
        </documentation>
    </annotation>
…
</schema>
```

2.2 Common Types

The SAML V2.0 Metadata specification defines several types as described in the following subsections. These types are used in defining SAML V2.0 Metadata elements and attributes.

2.2.1 Simple Type entityIDType

The simple type **entityIDType** restricts the XML schema data type **anyURI** to a maximum length of 1024 characters. **entityIDType** is used as a unique identifier for SAML entities. See also Section 8.3.6 of [SAMLCore]. An identifier of this type MUST be unique across all entities that interact within a given deployment. The use of a URI and holding to the rule that a single URI MUST NOT refer to different entities satisfies this requirement.

The following schema fragment defines the **entityIDType** simple type:

```
<simpleType name="entityIDType">
   <restriction base="anyURI">
      <maxLength value="1024"/>
   </restriction>
</simpleType>
```

2.2.2 Complex Type EndpointType

The complex type **EndpointType** describes a SAML protocol binding endpoint at which a SAML entity can be sent protocol messages. Various protocol or profile-specific metadata elements are bound to this type. It consists of the following attributes:

`Binding` [Required]

A required attribute that specifies the SAML binding supported by the endpoint. Each binding is assigned a URI to identify it.

`Location` [Required]

A required URI attribute that specifies the location of the endpoint. The allowable syntax of this URI depends on the protocol binding.

`ResponseLocation` [Optional]

Optionally specifies a different location to which response messages sent as part of the protocol or profile should be sent. The allowable syntax of this URI depends on the protocol binding.

The `ResponseLocation` attribute is used to enable different endpoints to be specified for receiving request and response messages associated with a protocol or profile, not as a means of load-balancing or redundancy (multiple elements of this type can be included for this purpose). When a role contains an element of this type pertaining to a protocol or profile for which only a single type of message (request or response) is applicable, then the `ResponseLocation` attribute is unused.

In most contexts, elements of this type appear in unbounded sequences in the schema. This is to permit a

protocol or profile to be offered by an entity at multiple endpoints, usually with different protocol bindings, allowing the metadata consumer to choose an appropriate endpoint for its needs. Multiple endpoints might also offer "client-side" load-balancing or failover, particular in the case of a synchronous protocol binding.

This element also permits the use of arbitrary elements and attributes defined in a non-SAML namespace. Any such content MUST be namespace-qualified.

The following schema fragment defines the **EndpointType** complex type:

```
<complexType name="EndpointType">
   <sequence>
      <any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
   </sequence>
   <attribute name="Binding" type="anyURI" use="required"/>
   <attribute name="Location" type="anyURI" use="required"/>
   <attribute name="ResponseLocation" type="anyURI" use="optional"/>
   <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
```

2.2.3 Complex Type IndexedEndpointType

The complex type **IndexedEndpointType** extends **EndpointType** with a pair of attributes to permit the indexing of otherwise identical endpoints so that they can be referenced by protocol messages. It consists of the following additional attributes:

`index` [Required]

A required attribute that assigns a unique integer value to the endpoint so that it can be referenced in a protocol message.

`isDefault` [Optional]

An optional boolean attribute used to designate the default endpoint among an indexed set. If omitted, the value is assumed to be `false`.

In any such sequence of like endpoints based on this type, the default endpoint is the first such endpoint with the `isDefault` attribute set to `true`. If no such endpoints exist, the default endpoint is the first such endpoint without the `isDefault` attribute set to `false`. If no such endpoints exist, the default endpoint is the first element in the sequence.

The following schema fragment defines the **IndexedEndpointType** complex type:

```
<complexType name="IndexedEndpointType">
   <complexContent>
      <extension base="md:EndpointType">
         <attribute name="index" type="unsignedShort" use="required"/>
         <attribute name="isDefault" type="boolean" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.2.4 Complex Type localizedNameType

The **localizedNameType** complex type extends a string-valued element with a standard XML language attribute. The following schema fragment defines the **localizedNameType** complex type:

```
<complexType name="localizedNameType">
   <simpleContent>
      <extension base="string">
         <attribute ref="xml:lang" use="required"/>
      </extension>
   </simpleContent>
</complexType>
```

2.2.5 Complex Type localizedURIType

The **localizedURIType** complex type extends a URI-valued element with a standard XML language attribute.

The following schema fragment defines the **localizedURIType** complex type:

```
<complexType name="localizedURIType">
   <simpleContent>
      <extension base="anyURI">
         <attribute ref="xml:lang" use="required"/>
      </extension>
   </simpleContent>
</complexType>
```

2.3 Root Elements

A SAML metadata instance describes either a single entity or multiple entities. In the former case, the root element MUST be `<EntityDescriptor>`. In the latter case, the root element MUST be `<EntitiesDescriptor>`.

2.3.1 Element <EntitiesDescriptor>

The `<EntitiesDescriptor>` element contains the metadata for an optionally named group of SAML entities. Its **EntitiesDescriptorType** complex type contains a sequence of `<EntityDescriptor>` elements, `<EntitiesDescriptor>` elements, or both:

`ID` [Optional]

A document-unique identifier for the element, typically used as a reference point when signing.

`validUntil` [Optional]

Optional attribute indicates the expiration time of the metadata contained in the element and any contained elements.

`cacheDuration` [Optional]

Optional attribute indicates the maximum length of time a consumer should cache the metadata contained in the element and any contained elements.

`Name` [Optional]

A string name that identifies a group of SAML entities in the context of some deployment.

`<ds:Signature>` [Optional]

An XML signature that authenticates the containing element and its contents, as described in Section 3.

`<Extensions>` [Optional]

This contains optional metadata extensions that are agreed upon between a metadata publisher and consumer. Extension elements MUST be namespace-qualified by a non-SAML-defined namespace.

`<EntitiesDescriptor>` or `<EntityDescriptor>` [One or More]

Contains the metadata for one or more SAML entities, or a nested group of additional metadata.

When used as the root element of a metadata instance, this element MUST contain either a `validUntil` or `cacheDuration` attribute. It is RECOMMENDED that only the root element of a metadata instance contain either attribute.

The following schema fragment defines the <EntitiesDescriptor> element and its **EntitiesDescriptorType** complex type:

```
<element name="EntitiesDescriptor" type="md:EntitiesDescriptorType"/>
<complexType name="EntitiesDescriptorType">
   <sequence>
      <element ref="ds:Signature" minOccurs="0"/>
      <element ref="md:Extensions" minOccurs="0"/>
      <choice minOccurs="1" maxOccurs="unbounded">
         <element ref="md:EntityDescriptor"/>
         <element ref="md:EntitiesDescriptor"/>
      </choice>
   </sequence>
   <attribute name="validUntil" type="dateTime" use="optional"/>
   <attribute name="cacheDuration" type="duration" use="optional"/>
   <attribute name="ID" type="ID" use="optional"/>
   <attribute name="Name" type="string" use="optional"/>
</complexType>
<element name="Extensions" type="md:ExtensionsType"/>
<complexType final="#all" name="ExtensionsType">
   <sequence>
      <any namespace="##other" processContents="lax"
maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.3.2 Element <EntityDescriptor>

The <EntityDescriptor> element specifies metadata for a single SAML entity. A single entity may act in many different roles in the support of multiple profiles. This specification directly supports the following concrete roles as well as the abstract <RoleDescriptor> element for extensibility (see subsequent sections for more details):

- SSO Identity Provider
- SSO Service Provider
- Authentication Authority
- Attribute Authority
- Policy Decision Point
- Affiliation

Its **EntityDescriptorType** complex type consists of the following elements and attributes:

entityID [Required]

Specifies the unique identifier of the SAML entity whose metadata is described by the element's contents.

ID [Optional]

A document-unique identifier for the element, typically used as a reference point when signing.

validUntil [Optional]

Optional attribute indicates the expiration time of the metadata contained in the element and any contained elements.

cacheDuration [Optional]

Optional attribute indicates the maximum length of time a consumer should cache the metadata contained in the element and any contained elements.

<ds:Signature> [Optional]

An XML signature that authenticates the containing element and its contents, as described in Section 3.

<Extensions> [Optional]

This contains optional metadata extensions that are agreed upon between a metadata publisher and consumer. Extension elements MUST be namespace-qualified by a non-SAML-defined namespace.

<RoleDesciptor>, <IDPSSODescriptor>, <SPSSODescriptor>, <AuthnAuthorityDescriptor>, <AttributeAuthorityDescriptor>, <PDPDescriptor> [One or More]

OR

<AffiliationDescriptor> [Required]

The primary content of the element is either a sequence of one or more role descriptor elements, or a specialized descriptor that defines an affiliation.

<Organization> [Optional]

Optional element identifying the organization responsible for the SAML entity described by the element.

<ContactPerson> [Zero or More]

Optional sequence of elements identifying various kinds of contact personnel.

<AdditionalMetadataLocation> [Zero or More]

Optional sequence of namespace-qualified locations where additional metadata exists for the SAML entity. This may include metadata in alternate formats or describing adherence to other non-SAML specifications.

Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

When used as the root element of a metadata instance, this element MUST contain either a validUntil or cacheDuration attribute. It is RECOMMENDED that only the root element of a metadata instance contain either attribute.

It is RECOMMENDED that if multiple role descriptor elements of the same type appear, that they do not share overlapping protocolSupportEnumeration values. Selecting from among multiple role descriptor elements of the same type that do share a protocolSupportEnumeration value is undefined within this specification, but MAY be defined by metadata profiles, possibly through the use of other distinguishing extension attributes.

The following schema fragment defines the <EntityDescriptor> element and its **EntityDescriptorType** complex type:

```
<element name="EntityDescriptor" type="md:EntityDescriptorType"/>
<complexType name="EntityDescriptorType">
   <sequence>
      <element ref="ds:Signature" minOccurs="0"/>
      <element ref="md:Extensions" minOccurs="0"/>
      <choice>
         <choice maxOccurs="unbounded">
            <element ref="md:RoleDescriptor"/>
            <element ref="md:IDPSSODescriptor"/>
            <element ref="md:SPSSODescriptor"/>
            <element ref="md:AuthnAuthorityDescriptor"/>
            <element ref="md:AttributeAuthorityDescriptor"/>
            <element ref="md:PDPDescriptor"/>
         </choice>
         <element ref="md:AffiliationDescriptor"/>
```

```
      </choice>
      <element ref="md:Organization" minOccurs="0"/>
      <element ref="md:ContactPerson" minOccurs="0"
maxOccurs="unbounded"/>
      <element ref="md:AdditionalMetadataLocation" minOccurs="0"
maxOccurs="unbounded"/>
   </sequence>
   <attribute name="entityID" type="md:entityIDType" use="required"/>
   <attribute name="validUntil" type="dateTime" use="optional"/>
   <attribute name="cacheDuration" type="duration" use="optional"/>
   <attribute name="ID" type="ID" use="optional"/>
   <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
```

2.3.2.1 Element <Organization>

The `<Organization>` element specifies basic information about an organization responsible for a SAML entity or role. The use of this element is always optional. Its content is informative in nature and does not directly map to any core SAML elements or attributes. Its **OrganizationType** complex type consists of the following elements:

`<Extensions>` [Optional]

This contains optional metadata extensions that are agreed upon between a metadata publisher and consumer. Extensions MUST NOT include global (non-namespace-qualified) elements or elements qualified by a SAML-defined namespace within this element.

`<OrganizationName>` [One or More]

One or more language-qualified names that may or may not be suitable for human consumption.

`<OrganizationDisplayName>` [One or More]

One or more language-qualified names that are suitable for human consumption.

`<OrganizationURL>` [One or More]

One or more language-qualified URIs that specify a location to which to direct a user for additional information. Note that the language qualifier refers to the content of the material at the specified location.

Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

The following schema fragment defines the `<Organization>` element and its **OrganizationType** complex type:

```
<element name="Organization" type="md:OrganizationType"/>
<complexType name="OrganizationType">
   <sequence>
      <element ref="md:Extensions" minOccurs="0"/>
      <element ref="md:OrganizationName" maxOccurs="unbounded"/>
      <element ref="md:OrganizationDisplayName" maxOccurs="unbounded"/>
      <element ref="md:OrganizationURL" maxOccurs="unbounded"/>
   </sequence>
   <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
<element name="OrganizationName" type="md:localizedNameType"/>
<element name="OrganizationDisplayName" type="md:localizedNameType"/>
<element name="OrganizationURL" type="md:localizedURIType"/>
```

2.3.2.2 Element <ContactPerson>

The `<ContactPerson>` element specifies basic contact information about a person responsible in some capacity for a SAML entity or role. The use of this element is always optional. Its content is informative in nature and does not directly map to any core SAML elements or attributes. Its **ContactType** complex type

consists of the following elements and attributes:

`contactType` [Required]

Specifies the type of contact using the **ContactTypeType** enumeration. The possible values are `technical`, `support`, `administrative`, `billing`, and `other`.

`<Extensions>` [Optional]

This contains optional metadata extensions that are agreed upon between a metadata publisher and consumer. Extension elements MUST be namespace-qualified by a non-SAML-defined namespace.

`<Company>` [Optional]

Optional string element that specifies the name of the company for the contact person.

`<GivenName>` [Optional]

Optional string element that specifies the given (first) name of the contact person.

`<SurName>` [Optional]

Optional string element that specifies the surname of the contact person.

`<EmailAddress>` [Zero or More]

Zero or more elements containing mailto: URIs representing e-mail addresses belonging to the contact person.

`<TelephoneNumber>` [Zero or More]

Zero or more string elements specifying a telephone number of the contact person.

Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

The following schema fragment defines the `<ContactPerson>` element and its **ContactType** complex type:

```
<element name="ContactPerson" type="md:ContactType"/>
<complexType name="ContactType">
   <sequence>
      <element ref="md:Extensions" minOccurs="0"/>
      <element ref="md:Company" minOccurs="0"/>
      <element ref="md:GivenName" minOccurs="0"/>
      <element ref="md:SurName" minOccurs="0"/>
      <element ref="md:EmailAddress" minOccurs="0"
maxOccurs="unbounded"/>
      <element ref="md:TelephoneNumber" minOccurs="0"
maxOccurs="unbounded"/>
   </sequence>
   <attribute name="contactType" type="md:ContactTypeType"
use="required"/>
   <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
<element name="Company" type="string"/>
<element name="GivenName" type="string"/>
<element name="SurName" type="string"/>
<element name="EmailAddress" type="anyURI"/>
<element name="TelephoneNumber" type="string"/>
<simpleType name="ContactTypeType">
   <restriction base="string">
      <enumeration value="technical"/>
      <enumeration value="support"/>
      <enumeration value="administrative"/>
      <enumeration value="billing"/>
      <enumeration value="other"/>
   </restriction>
```

```
</simpleType>
```

2.3.2.3 Element <AdditionalMetadataLocation>

The `<AdditionalMetadataLocation>` element is a namespace-qualified URI that specifies where additional XML-based metadata may exist for a SAML entity. Its **AdditionalMetadataLocationType** complex type extends the **anyURI** type with a `namespace` attribute (also of type **anyURI**). This required attribute MUST contain the XML namespace of the root element of the instance document found at the specified location.

The following schema fragment defines the `<AdditionalMetadataLocation>` element and its **AdditionalMetadataLocationType** complex type:

```
<element name="AdditionalMetadataLocation"
type="md:AdditionalMetadataLocationType"/>
<complexType name="AdditionalMetadataLocationType">
   <simpleContent>
      <extension base="anyURI">
         <attribute name="namespace" type="anyURI" use="required"/>
      </extension>
   </simpleContent>
</complexType>
```

2.4 Role Descriptor Elements

The elements in this section make up the bulk of the operational support component of the metadata. Each element (save for the abstract one) define a specific collection of operational behavior in support of SAML profiles defined in [SAMLProf].

2.4.1 Element <RoleDescriptor>

The `<RoleDescriptor>` element is an abstract extension point that contains common descriptive information intended to provide processing commonality across different roles. New roles can be defined by extending its abstract **RoleDescriptorType** complex type, which contains the following elements and attributes:

`ID` [Optional]

A document-unique identifier for the element, typically used as a reference point when signing.

`validUntil` [Optional]

Optional attribute indicates the expiration time of the metadata contained in the element and any contained elements.

`cacheDuration` [Optional]

Optional attribute indicates the maximum length of time a consumer should cache the metadata contained in the element and any contained elements.

`protocolSupportEnumeration` [Required]

A whitespace-delimited set of URIs that identify the set of protocol specifications supported by the role element. For SAML V2.0 entities, this set MUST include the SAML protocol namespace URI, `urn:oasis:names:tc:SAML:2.0:protocol`. Note that future SAML specifications might share the same namespace URI, but SHOULD provide alternate "protocol support" identifiers to ensure discrimination when necessary.

`errorURL` [Optional]

Optional URI attribute that specifies a location to direct a user for problem resolution and additional support related to this role.

<ds:Signature> [Optional]

An XML signature that authenticates the containing element and its contents, as described in Section 3.

<Extensions> [Optional]

This contains optional metadata extensions that are agreed upon between a metadata publisher and consumer. Extension elements MUST be namespace-qualified by a non-SAML-defined namespace.

<KeyDescriptor> [Zero or More]

Optional sequence of elements that provides information about the cryptographic keys that the entity uses when acting in this role.

<Organization> [Optional]

Optional element specifies the organization associated with this role. Identical to the element used within the <EntityDescriptor> element.

<ContactPerson> [Zero or More]

Optional sequence of elements specifying contacts associated with this role. Identical to the element used within the <EntityDescriptor> element.

Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

The following schema fragment defines the <RoleDescriptor> element and its **RoleDescriptorType** complex type:

```
<element name="RoleDescriptor" type="md:RoleDescriptorType"/>
<complexType name="RoleDescriptorType" abstract="true">
   <sequence>
      <element ref="ds:Signature" minOccurs="0"/>
      <element ref="md:Extensions" minOccurs="0"/>
      <element ref="md:KeyDescriptor" minOccurs="0"
maxOccurs="unbounded"/>
      <element ref="md:Organization" minOccurs="0"/>
      <element ref="md:ContactPerson" minOccurs="0"
maxOccurs="unbounded"/>
   </sequence>
   <attribute name="ID" type="ID" use="optional"/>
   <attribute name="validUntil" type="dateTime" use="optional"/>
   <attribute name="cacheDuration" type="duration" use="optional"/>
   <attribute name="protocolSupportEnumeration" type="md:anyURIListType"
use="required"/>
   <attribute name="errorURL" type="anyURI" use="optional"/>
   <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
<simpleType name="anyURIListType">
   <list itemType="anyURI"/>
</simpleType>
```

2.4.1.1 Element <KeyDescriptor>

The <KeyDescriptor> element provides information about the cryptographic key(s) that an entity uses to sign data or receive encrypted keys, along with additional cryptographic details. Its **KeyDescriptorType** complex type consists of the following elements and attributes:

use [Optional]

Optional attribute specifying the purpose of the key being described. Values are drawn from the **KeyTypes** enumeration, and consist of the values encryption and signing.

<ds:KeyInfo> [Required]

Optional element that directly or indirectly identifies a key. See[XMLSig] for additional details on the use of this element.

<EncryptionMethod> [Zero or More]

Optional element specifying an algorithm and algorithm-specific settings supported by the entity. The exact content varies based on the algorithm supported. See [XMLEnc] for the definition of this element's **xenc:EncryptionMethodType** complex type.

The following schema fragment defines the <KeyDescriptor> element and its **KeyDescriptorType** complex type:

```
<element name="KeyDescriptor" type="md:KeyDescriptorType"/>
<complexType name="KeyDescriptorType">
   <sequence>
      <element ref="ds:KeyInfo"/>
      <element ref="md:EncryptionMethod minOccurs="0"
maxOccurs="unbounded"/>
   </sequence>
   <attribute name="use" type="md:KeyTypes" use="optional"/>
</complexType>
<simpleType name="KeyTypes">
   <restriction base="string">
      <enumeration value="encryption"/>
      <enumeration value="signing"/>
   </restriction>
</simpleType>
<element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>
```

2.4.2 Complex Type SSODescriptorType

The **SSODescriptorType** abstract type is a common base type for the concrete types **SPSSODescriptorType** and **IDPSSODescriptorType**, described in subsequent sections. It extends **RoleDescriptorType** with elements reflecting profiles common to both identity providers and service providers that support SSO, and contains the following additional elements:

<ArtifactResolutionService> [Zero or More]

Zero or more elements of type **IndexedEndpointType** that describe indexed endpoints that support the Artifact Resolution profile defined in [SAMLProf]. The ResponseLocation attribute MUST be omitted.

<SingleLogoutService> [Zero or More]

Zero or more elements of type **EndpointType** that describe endpoints that support the Single Logout profiles defined in [SAMLProf].

<ManageNameIDService> [Zero or More]

Zero or more elements of type **EndpointType** that describe endpoints that support the Name Identifier Management profiles defined in [SAMLProf].

<NameIDFormat> [Zero or More]

Zero or more elements of type **anyURI** that enumerate the name identifier formats supported by this system entity acting in this role. See Section 8.3 of [SAMLCore] for some possible values for this element.

The following schema fragment defines the **SSODescriptorType** complex type:

```
<complexType name="SSODescriptorType" abstract="true">
   <complexContent>
      <extension base="md:RoleDescriptorType">
         <sequence>
```

```
                <element ref="md:ArtifactResolutionService" minOccurs="0"
maxOccurs="unbounded"/>
                <element ref="md:SingleLogoutService" minOccurs="0"
maxOccurs="unbounded"/>
                <element ref="md:ManageNameIDService" minOccurs="0"
maxOccurs="unbounded"/>
                <element ref="md:NameIDFormat" minOccurs="0"
maxOccurs="unbounded"/>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
    <element name="ArtifactResolutionService" type="md:IndexedEndpointType"/>
    <element name="SingleLogoutService" type="md:EndpointType"/>
    <element name="ManageNameIDService" type="md:EndpointType"/>
    <element name="NameIDFormat" type="anyURI"/>
```

2.4.3 Element <IDPSSODescriptor>

The `<IDPSSODescriptor>` element extends **SSODescriptorType** with content reflecting profiles specific to identity providers supporting SSO. Its **IDPSSODescriptorType** complex type contains the following additional elements and attributes:

`WantAuthnRequestsSigned` [Optional]

Optional attribute that indicates a requirement for the `<samlp:AuthnRequest>` messages received by this identity provider to be signed. If omitted, the value is assumed to be `false`.

`<SingleSignOnService>` [One or More]

One or more elements of type **EndpointType** that describe endpoints that support the profiles of the Authentication Request protocol defined in [SAMLProf]. All identity providers support at least one such endpoint, by definition. The `ResponseLocation` attribute MUST be omitted.

`<NameIDMappingService>` [Zero or More]

Zero or more elements of type **EndpointType** that describe endpoints that support the Name Identifier Mapping profile defined in [SAMLProf]. The `ResponseLocation` attribute MUST be omitted.

`<AssertionIDRequestService>` [Zero or More]

Zero or more elements of type **EndpointType** that describe endpoints that support the profile of the Assertion Request protocol defined in [SAMLProf] or the special URI binding for assertion requests defined in [SAMLBind].

`<AttributeProfile>` [Zero or More]

Zero or more elements of type **anyURI** that enumerate the attribute profiles supported by this identity provider. See [SAMLProf] for some possible values for this element.

`<saml:Attribute>` [Zero or More]

Zero or more elements that identify the SAML attributes supported by the identity provider. Specific values MAY optionally be included, indicating that only certain values permitted by the attribute's definition are supported. In this context, "support" for an attribute means that the identity provider has the capability to include it when delivering assertions during single sign-on.

The following schema fragment defines the `<IDPSSODescriptor>` element and its **IDPSSODescriptorType** complex type:

```
<element name="IDPSSODescriptor" type="md:IDPSSODescriptorType"/>
<complexType name="IDPSSODescriptorType">
   <complexContent>
      <extension base="md:SSODescriptorType">
```

```
          <sequence>
             <element ref="md:SingleSignOnService"
maxOccurs="unbounded"/>
             <element ref="md:NameIDMappingService" minOccurs="0"
maxOccurs="unbounded"/>
             <element ref="md:AssertionIDRequestService" minOccurs="0"
maxOccurs="unbounded"/>
             <element ref="md:AttributeProfile" minOccurs="0"
maxOccurs="unbounded"/>
             <element ref="saml:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
          </sequence>
          <attribute name="WantAuthnRequestsSigned" type="boolean"
use="optional"/>
       </extension>
   </complexContent>
</complexType>
<element name="SingleSignOnService" type="md:EndpointType"/>
<element name="NameIDMappingService" type="md:EndpointType"/>
<element name="AssertionIDRequestService" type="md:EndpointType"/>
<element name="AttributeProfile" type="anyURI"/>
```

2.4.4 Element <SPSSODescriptor>

The `<SPSSODescriptor>` element extends **SSODescriptorType** with content reflecting profiles specific to service providers. Its **SPSSODescriptorType** complex type contains the following additional elements and attributes:

`AuthnRequestsSigned` [Optional]

Optional attribute that indicates whether the `<samlp:AuthnRequest>` messages sent by this service provider will be signed. If omitted, the value is assumed to be `false`.

`WantAssertionsSigned` [Optional]

Optional attribute that indicates a requirement for the `<saml:Assertion>` elements received by this service provider to be signed. If omitted, the value is assumed to be `false`. This requirement is in addition to any requirement for signing derived from the use of a particular profile/binding combination.

`<AssertionConsumerService>` [One or More]

One or more elements that describe indexed endpoints that support the profiles of the Authentication Request protocol defined in [SAMLProf]. All service providers support at least one such endpoint, by definition.

`<AttributeConsumingService>` [Zero or More]

Zero or more elements that describe an application or service provided by the service provider that requires or desires the use of SAML attributes.

At most one `<AttributeConsumingService>` element can have the attribute `isDefault` set to `true`. When multiple elements are specified and none has the attribute `isDefault` set to `true`, then the first element whose `isDefault` attribute is not set to `false` is to be used as the default. If all elements have their `isDefault` attribute set to `false`, then the first element is considered the default.

The following schema fragment defines the `<SPSSODescriptor>` element and its **SPSSODescriptorType** complex type:

```
<element name="SPSSODescriptor" type="md:SPSSODescriptorType"/>
<complexType name="SPSSODescriptorType">
   <complexContent>
      <extension base="md:SSODescriptorType">
         <sequence>
```

```
            <element ref="md:AssertionConsumerService"
maxOccurs="unbounded"/>
            <element ref="md:AttributeConsumingService" minOccurs="0"
maxOccurs="unbounded"/>
         </sequence>
         <attribute name="AuthnRequestsSigned" type="boolean"
use="optional"/>
         <attribute name="WantAssertionsSigned" type="boolean"
use="optional"/>
      </extension>
   </complexContent>
</complexType>
<element name="AssertionConsumerService"
type="md:IndexedEndpointType"/>
```

2.4.4.1 Element <AttributeConsumingService>

The `<AttributeConsumingService>` element defines a particular service offered by the service provider in terms of the attributes the service requires or desires. Its **AttributeConsumingServiceType** complex type contains the following elements and attributes:

`index` [Required]

A required attribute that assigns a unique integer value to the element so that it can be referenced in a protocol message.

`isDefault` [Optional]

Identifies the default service supported by the service provider. Useful if the specific service is not otherwise indicated by application context. If omitted, the value is assumed to be `false`.

`<ServiceName>` [One or More]

One or more language-qualified names for the service.

`<ServiceDescription>` [Zero or More]

Zero or more language-qualified strings that describe the service.

`<RequestedAttribute>` [One or More]

One or more elements specifying attributes required or desired by this service.

The following schema fragment defines the `<AttributeRequestingService>` element and its **AttributeRequestingServiceType** complex type:

```
<element name="AttributeConsumingService"
type="md:AttributeConsumingServiceType"/>
<complexType name="AttributeConsumingServiceType">
   <sequence>
      <element ref="md:ServiceName" maxOccurs="unbounded"/>
      <element ref="md:ServiceDescription" minOccurs="0"
maxOccurs="unbounded"/>
      <element ref="md:RequestedAttribute" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="index" type="unsignedShort" use="required"/>
   <attribute name="isDefault" type="boolean" use="optional"/>
</complexType>
<element name="ServiceName" type="md:localizedNameType"/>
<element name="ServiceDescription" type="md:localizedNameType"/>
```

2.4.4.2 Element <RequestedAttribute>

The `<RequestedAttribute>` element specifies a service provider's interest in a specific SAML attribute, optionally including specific values. Its **RequestedAttributeType** complex type extends the

saml:AttributeType with the following attribute:

`isRequired` [Optional]

Optional XML attribute indicates if the service requires the corresponding SAML attribute in order to function at all (as opposed to merely finding an attribute useful or desirable).

If specific `<saml:AttributeValue>` elements are included, then only matching values are relevant to the service. See [SAMLCore] for more information on attribute value matching.

The following schema fragment defines the `<RequestedAttribute>` element and its **RequestedAttributeType** complex type:

```
<element name="RequestedAttribute" type="md:RequestedAttributeType"/>
<complexType name="RequestedAttributeType">
   <complexContent>
      <extension base="saml:AttributeType">
         <attribute name="isRequired" type="boolean" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.4.5 Element <AuthnAuthorityDescriptor>

The `<AuthnAuthorityDescriptor>` element extends **RoleDescriptorType** with content reflecting profiles specific to authentication authorities, SAML authorities that respond to `<samlp:AuthnQuery>` messages. Its **AuthnAuthorityDescriptorType** complex type contains the following additional element:

`<AuthnQueryService>` [One or More]

One or more elements of type **EndpointType** that describe endpoints that support the profile of the Authentication Query protocol defined in [SAMLProf]. All authentication authorities support at least one such endpoint, by definition.

`<AssertionIDRequestService>` [Zero or More]

Zero or more elements of type **EndpointType** that describe endpoints that support the profile of the Assertion Request protocol defined in [SAMLProf] or the special URI binding for assertion requests defined in [SAMLBind].

`<NameIDFormat>` [Zero or More]

Zero or more elements of type **anyURI** that enumerate the name identifier formats supported by this authority. See Section 8.3 of [SAMLCore] for some possible values for this element.

The following schema fragment defines the `<AuthnAuthorityDescriptor>` element and its **AuthnAuthorityDescriptorType** complex type:

```
<element name="AuthnAuthorityDescriptor"
type="md:AuthnAuthorityDescriptorType"/>
<complexType name="AuthnAuthorityDescriptorType">
   <complexContent>
      <extension base="md:RoleDescriptorType">
         <sequence>
            <element ref="md:AuthnQueryService" maxOccurs="unbounded"/>
            <element ref="md:AssertionIDRequestService" minOccurs="0"
maxOccurs="unbounded"/>
            <element ref="md:NameIDFormat" minOccurs="0"
maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
<element name="AuthnQueryService" type="md:EndpointType"/>
```

2.4.6 Element <PDPDescriptor>

The <PDPDescriptor> element extends **RoleDescriptorType** with content reflecting profiles specific to policy decision points, SAML authorities that respond to <samlp:AuthzDecisionQuery> messages. Its **PDPDescriptorType** complex type contains the following additional element:

<AuthzService> [One or More]

One or more elements of type **EndpointType** that describe endpoints that support the profile of the Authorization Decision Query protocol defined in [SAMLProf]. All policy decision points support at least one such endpoint, by definition.

<AssertionIDRequestService> [Zero or More]

Zero or more elements of type **EndpointType** that describe endpoints that support the profile of the Assertion Request protocol defined in [SAMLProf] or the special URI binding for assertion requests defined in [SAMLBind].

<NameIDFormat> [Zero or More]

Zero or more elements of type **anyURI** that enumerate the name identifier formats supported by this authority. See Section 8.3 of [SAMLCore] for some possible values for this element.

The following schema fragment defines the <PDPDescriptor> element and its **PDPDescriptorType** complex type:

```
<element name="PDPDescriptor" type="md:PDPDescriptorType"/>
<complexType name="PDPDescriptorType">
   <complexContent>
      <extension base="md:RoleDescriptorType">
         <sequence>
            <element ref="md:AuthzService" maxOccurs="unbounded"/>
            <element ref="md:AssertionIDRequestService" minOccurs="0"
maxOccurs="unbounded"/>
            <element ref="md:NameIDFormat" minOccurs="0"
maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
<element name="AuthzService" type="md:EndpointType"/>
```

2.4.7 Element <AttributeAuthorityDescriptor>

The <AttributeAuthorityDescriptor> element extends **RoleDescriptorType** with content reflecting profiles specific to attribute authorities, SAML authorities that respond to <samlp:AttributeQuery> messages. Its **AttributeAuthorityDescriptorType** complex type contains the following additional elements:

<AttributeService> [One or More]

One or more elements of type **EndpointType** that describe endpoints that support the profile of the Attribute Query protocol defined in [SAMLProf]. All attribute authorities support at least one such endpoint, by definition.

<AssertionIDRequestService> [Zero or More]

Zero or more elements of type **EndpointType** that describe endpoints that support the profile of the Assertion Request protocol defined in [SAMLProf] or the special URI binding for assertion requests defined in [SAMLBind].

<NameIDFormat> [Zero or More]

Zero or more elements of type **anyURI** that enumerate the name identifier formats supported by

this authority. See Section 8.3 of [SAMLCore] for some possible values for this element.

`<AttributeProfile>` [Zero or More]

Zero or more elements of type **anyURI** that enumerate the attribute profiles supported by this authority. See [SAMLProf] for some possible values for this element.

`<saml:Attribute>` [Zero or More]

Zero or more elements that identify the SAML attributes supported by the authority. Specific values MAY optionally be included, indicating that only certain values permitted by the attribute's definition are supported.

The following schema fragment defines the `<AttributeAuthorityDescriptor>` element and its **AttributeAuthorityDescriptorType** complex type:

```
<element name="AttributeAuthorityDescriptor"
type="md:AttributeAuthorityDescriptorType"/>
<complexType name="AttributeAuthorityDescriptorType">
   <complexContent>
      <extension base="md:RoleDescriptorType">
         <sequence>
            <element ref="md:AttributeService" maxOccurs="unbounded"/>
            <element ref="md:AssertionIDRequestService" minOccurs="0"
maxOccurs="unbounded"/>
            <element ref="md:NameIDFormat" minOccurs="0"
maxOccurs="unbounded"/>
            <element ref="md:AttributeProfile" minOccurs="0"
maxOccurs="unbounded"/>
            <element ref="saml:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
<element name="AttributeService" type="md:EndpointType"/>
```

2.5 Element <AffiliationDescriptor>

The `<AffiliationDescriptor>` element is an alternative to the sequence of role descriptors described in Section 2.4 that is used when an `<EntityDescriptor>` describes an affiliation of SAML entities (typically service providers) rather than a single entity. The `<AffiliationDescriptor>` element provides a summary of the individual entities that make up the affiliation along with general information about the affiliation itself. Its **AffiliationDescriptorType** complex type contains the following elements and attributes:

`affiliationOwnerID` [Required]

Specifies the unique identifier of the entity responsible for the affiliation. The owner is NOT presumed to be a member of the affiliation; if it is a member, its identifier MUST also appear in an `<AffiliateMember>` element.

`ID` [Optional]

A document-unique identifier for the element, typically used as a reference point when signing.

`validUntil` [Optional]

Optional attribute indicates the expiration time of the metadata contained in the element and any contained elements.

`cacheDuration` [Optional]

Optional attribute indicates the maximum length of time a consumer should cache the metadata contained in the element and any contained elements.

968 <ds:Signature> [Optional]

969 An XML signature that authenticates the containing element and its contents, as described in
970 Section 3.

971 <Extensions> [Optional]

972 This contains optional metadata extensions that are agreed upon between a metadata publisher
973 and consumer. Extension elements MUST be namespace-qualified by a non-SAML-defined
974 namespace.

975 <AffiliateMember> [One or More]

976 One or more elements enumerating the members of the affiliation by specifying each member's
977 unique identifier. See also Section 8.3.6 of [SAMLCore].

978 <KeyDescriptor> [Zero or More]

979 Optional sequence of elements that provides information about the cryptographic keys that the
980 affiliation uses as a whole, as distinct from keys used by individual members of the affiliation,
981 which are published in the metadata for those entities.

982 Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

983 The following schema fragment defines the <AffiliationDescriptor> element and its
984 **AffiliationDescriptorType** complex type:

```
985 <element name="AffiliationDescriptor"
986 type="md:AffiliationDescriptorType"/>
987 <complexType name="AffiliationDescriptorType">
988    <sequence>
989       <element ref="ds:Signature" minOccurs="0"/>
990       <element ref="md:Extensions" minOccurs="0"/>
991       <element ref="md:AffiliateMember" maxOccurs="unbounded"/>
992       <element ref="md:KeyDescriptor" minOccurs="0"
993 maxOccurs="unbounded"/>
994    </sequence>
995    <attribute name="affiliationOwnerID" type="md:entityIDType"
996 use="required"/>
997    <attribute name="validUntil" type="dateTime" use="optional"/>
998    <attribute name="cacheDuration" type="duration" use="optional"/>
999    <attribute name="ID" type="ID" use="optional"/>
1000   <anyAttribute namespace="##other" processContents="lax"/>
1001 </complexType>
1002 <element name="AffiliateMember" type="md:entityIDType"/>
```

1003 2.6 Examples

1004 The following is an example of metadata for a SAML system entity acting as an identity provider and an
1005 attribute authority. A signature is shown as a placeholder, without the actual content.
1006

```
1007 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1008    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1009    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1010    entityID="https://IdentityProvider.com/SAML">
1011    <ds:Signature>...</ds:Signature>
1012     <IDPSSODescriptor WantAuthnRequestsSigned="true"
1013       protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1014         <KeyDescriptor use="signing">
1015             <ds:KeyInfo>
1016                 <ds:KeyName>IdentityProvider.com SSO Key</ds:KeyName>
1017             </ds:KeyInfo>
1018         </KeyDescriptor>
1019         <ArtifactResolutionService isDefault="true" index="0"
1020           Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1021           Location="https://IdentityProvider.com/SAML/Artifact"/>
```

```
        <SingleLogoutService
          Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
          Location="https://IdentityProvider.com/SAML/SLO/SOAP"/>
        <SingleLogoutService
          Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
          Location="https://IdentityProvider.com/SAML/SLO/Browser"
          ResponseLocation="https://IdentityProvider.com/SAML/SLO/Response"/>
        <NameIDFormat>
          urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
        </NameIDFormat>
        <NameIDFormat>
          urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
        </NameIDFormat>
        <NameIDFormat>
          urn:oasis:names:tc:SAML:2.0:nameid-format:transient
        </NameIDFormat>
        <SingleSignOnService
          Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
          Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
        <SingleSignOnService
          Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
          Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
        <saml:Attribute
          NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
          Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
          FriendlyName="eduPersonPrincipalName">
        </saml:Attribute>
        <saml:Attribute
          NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
          Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
          FriendlyName="eduPersonAffiliation">
            <saml:AttributeValue>member</saml:AttributeValue>
            <saml:AttributeValue>student</saml:AttributeValue>
            <saml:AttributeValue>faculty</saml:AttributeValue>
            <saml:AttributeValue>employee</saml:AttributeValue>
            <saml:AttributeValue>staff</saml:AttributeValue>
        </saml:Attribute>
    </IDPSSODescriptor>
    <AttributeAuthorityDescriptor
       protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <KeyDescriptor use="signing">
            <ds:KeyInfo>
                <ds:KeyName>IdentityProvider.com AA Key</ds:KeyName>
            </ds:KeyInfo>
        </KeyDescriptor>
        <AttributeService
          Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
          Location="https://IdentityProvider.com/SAML/AA/SOAP"/>
        <AssertionIDRequestService
          Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
          Location="https://IdentityProvider.com/SAML/AA/URI"/>
        <NameIDFormat>
          urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
        </NameIDFormat>
        <NameIDFormat>
          urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
        </NameIDFormat>
        <NameIDFormat>
          urn:oasis:names:tc:SAML:2.0:nameid-format:transient
        </NameIDFormat>
        <saml:Attribute
          NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
          Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
          FriendlyName="eduPersonPrincipalName">
        </saml:Attribute>
        <saml:Attribute
          NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
```

```
          Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
          FriendlyName="eduPersonAffiliation">
            <saml:AttributeValue>member</saml:AttributeValue>
            <saml:AttributeValue>student</saml:AttributeValue>
            <saml:AttributeValue>faculty</saml:AttributeValue>
            <saml:AttributeValue>employee</saml:AttributeValue>
            <saml:AttributeValue>staff</saml:AttributeValue>
        </saml:Attribute>
    </AttributeAuthorityDescriptor>
    <Organization>
        <OrganizationName xml:lang="en">Identity Providers R US</OrganizationName>
        <OrganizationDisplayName xml:lang="en">
          Identity Providers R US, a Division of Lerxst Corp.
        </OrganizationDisplayName>
        <OrganizationURL xml:lang="en">https://IdentityProvider.com</OrganizationURL>
    </Organization>
</EntityDescriptor>
```

The following is an example of metadata for a SAML system entity acting as a service provider. A signature is shown as a placeholder, without the actual content. For illustrative purposes, the service is one that does not require users to uniquely identify themselves, but rather authorizes access on the basis of a role-like attribute.

```
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   entityID="https://ServiceProvider.com/SAML">
   <ds:Signature>...</ds:Signature>
    <SPSSODescriptor AuthnRequestsSigned="true"
       protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <KeyDescriptor use="signing">
            <ds:KeyInfo>
                <ds:KeyName>ServiceProvider.com SSO Key</ds:KeyName>
            </ds:KeyInfo>
        </KeyDescriptor>
        <KeyDescriptor use="encryption">
            <ds:KeyInfo>
                <ds:KeyName>ServiceProvider.com Encrypt Key</ds:KeyName>
            </ds:KeyInfo>
            <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        </KeyDescriptor>
        <SingleLogoutService
          Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
          Location="https://ServiceProvider.com/SAML/SLO/SOAP"/>
        <SingleLogoutService
          Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
          Location="https://ServiceProvider.com/SAML/SLO/Browser"
          ResponseLocation="https://ServiceProvider.com/SAML/SLO/Response"/>
        <NameIDFormat>
          urn:oasis:names:tc:SAML:2.0:nameid-format:transient
        </NameIDFormat>
        <AssertionConsumerService isDefault="true" index="0"
          Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
          Location="https://ServiceProvider.com/SAML/SSO/Artifact"/>
        <AssertionConsumerService index="1"
          Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
          Location="https://ServiceProvider.com/SAML/SSO/POST"/>
        <AttributeConsumingService index="0">
            <ServiceName xml:lang="en">Academic Journals R US</ServiceName>
            <RequestedAttribute
             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
             Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7"
             FriendlyName="eduPersonEntitlement">
               <saml:AttributeValue>
                https://ServiceProvider.com/entitlements/123456789
               </saml:AttributeValue>
            </RequestedAttribute>
```

```
        </AttributeConsumingService>
    </SPSSODescriptor>
    <Organization>
        <OrganizationName xml:lang="en">Academic Journals R US</OrganizationName>
        <OrganizationDisplayName xml:lang="en">
           Academic Journals R US, a Division of Dirk Corp.
        </OrganizationDisplayName>
        <OrganizationURL xml:lang="en">https://ServiceProvider.com</OrganizationURL>
    </Organization>
</EntityDescriptor>
```

3 Signature Processing

Various elements in a metadata instance can be digitally signed (as indicated by the element's inclusion of a <ds:Signature> element), with the following benefits:

- Metadata integrity
- Authentication of the metadata by a trusted signer

A digital signature is not always required, for example if the relying party obtains the information directly from the publishing entity directly (with no intermediaries) through a secure channel, with the entity having authenticated to the relying party by some means other than a digital signature.

Many different techniques are available for "direct" authentication and secure channel establishment between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition, the applicable security requirements depend on the communicating applications.

Additionally, elements can inherit signatures on enclosing parent elements that are themselves signed.

In the absence of such context, it is RECOMMENDED that at least the root element of a metadata instance be signed.

3.1 XML Signature Profile

The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility and many choices. This section details the constraints on these facilities so that metadata processors do not have to deal with the full generality of XML Signature processing. This usage makes specific use of the **xs:ID**-typed attributes optionally present on the elements to which signatures can apply. These attributes are collectively referred to in this section as the identifier attributes.

3.1.1 Signing Formats and Algorithms

XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and detached.

SAML metadata MUST use enveloped signatures when signing the elements defined in this specification. SAML processors SHOULD support the use of RSA signing and verification for public key operations in accordance with the algorithm identified by http://www.w3.org/2000/09/xmldsig#rsa-sha1.

3.1.2 References

Signed metadata elements MUST supply a value for the identifier attribute on the signed element. The element may or may not be the root element of the actual XML document containing the signed metadata element.
Signatures MUST contain a single <ds:Reference> containing a URI reference to the identifier attribute value of the metadata element being signed. For example, if the identifier attribute value is "foo", then the URI attribute in the <ds:Reference> element MUST be "#foo".
As a consequence, a metadata element's signature MUST apply to the content of the signed element and any child elements it contains.

3.1.3 Canonicalization Method

SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the <ds:CanonicalizationMethod> element of <ds:SignedInfo>, and as a <ds:Transform> algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML metadata

1205 embedded in an XML context can be verified independent of that context.

1206 3.1.4 Transforms

1207 Signatures in SAML metadata SHOULD NOT contain transforms other than the enveloped signature
1208 transform (with the identifier http://www.w3.org/2000/09/xmldsig#enveloped-signature) or the exclusive
1209 canonicalization transforms (with the identifier http://www.w3.org/2001/10/xml-exc-c14n# or
1210 http://www.w3.org/2001/10/xml-exc-c14n#WithComments).

1211 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
1212 not, verifiers MUST ensure that no content of the signed metadata element is excluded from the
1213 signature. This can be accomplished by establishing out-of-band agreement as to what transforms are
1214 acceptable, or by applying the transforms manually to the content and reverifying the result as consisting
1215 of the same SAML metadata.

1216 3.1.5 KeyInfo

1217 XML Signature [XMLSig] defines usage of the `<ds:KeyInfo> element.` SAML does not require the
1218 use of `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY
1219 be absent.

4 Metadata Publication and Resolution

Two mechanisms are provided for an entity to publish (and for a consumer to resolve the location of) metadata documents: via a "well-known-location" by directly dereferencing the entity's unique identifier (a URI variously referred to as an *entityID* or *providerID*), or indirectly by publishing the location of metadata in the DNS. Other out-of-band mechanisms are of course also permitted. A consumer that supports both approaches defined in this document MUST attempt resolution via DNS before using the "well-known-location" mechanism.

When retrieval requires network transport of the document, the transport SHOULD be protected with mechanisms providing server authentication and integrity protection. For example, HTTP-based resolution SHOULD be protected with *TLS/SSL* [RFC2246] as amended by [RFC3546].

Various mechanisms are described in this section to aid in establishing trust in the accuracy and legitimacy of metadata, including use of XML signatures, SSL/TLS server authentication, and DNS signatures. Regardless of the mechanism(s) used, relying parties SHOULD have some means by which to establish trust in metadata information before relying on it.

4.1 Publication and Resolution via Well-Known Location

The following sections describe publication and resolution of metadata by means of a well-known location.

4.1.1 Publication

Entities MAY publish their metadata documents at a well known location by placing the document at the location denoted by its unique identifier, which MUST be in the form of a URL (rather than a URN). See Section 8.3.6 of [SAMLCore] for more information about such identifiers. It is STRONGLY RECOMMENDED that `https` URLs be used for this purpose. An indirection mechanism supported by the URL scheme (such as an HTTP 1.1 302 redirect) MAY be used if the document is not placed directly at the location. If the publishing protocol permits MIME-based identification of content types, the content type of the metadata instance MUST be `application/samlmetadata+xml`.

The XML document provided at the well-known location MUST describe the metadata only for the entity represented by the unique identifier (that is, the root element MUST be an `<EntityDescriptor>` with an `entityID` matching the location). If other entities need to be described, the `<AdditionalMetaLocation>` element MUST be used. Thus the `<EntitiesDescriptor>` element MUST NOT be used in documents published using this mechanism, since a group of entities are not defined by such an identifier.

4.1.2 Resolution

If an entity's unique identifier is a URL, metadata consumers MAY attempt to resolve an entity's unique identifier directly, in a scheme-specific manner, by dereferencing the identifier.

4.2 Publishing and Resolution via DNS

To improve the accessibility of metadata documents and provide additional indirection between an entity's unique identifier and the location of metadata, entities MAY publish their metadata document locations in a zone of their corresponding DNS [RFC1034]. The entity's unique identifier (a URI) is used as the input to the process. Since URIs are flexible identifiers, location publication methods and the resolution process are determined by the URI's scheme and fully-qualified name. URI locations for metadata are

subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in [RFC2915] and [RFC3403].

It is RECOMMENDED that entities publish their resource records in signed zone files using [RFC2535] such that relying parties may establish the validity of the published location and authority of the zone, and integrity of the DNS response. If DNS zone signatures are present, relying parties MUST properly validate the signature.

4.2.1 Publication

This specification makes use of the NAPTR resource record described in [RFC2915] and [RFC3403]. Familiarity with these documents is encouraged.

Dynamic Delegation Discovery System (DDDS) [RFC3401] is a general purpose system for the retrieval of information based on an application-specific input string and the application of well known rules to transform that string until a terminal condition is reached requiring a look-up into an application-specific defined database or resolution of a URL based on the rules defined by the application. DDDS defines a specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS necessary to apply DDDS rules.

Entities MAY publish separate URLs when multiple metadata documents need to be distributed, or when different metadata documents are required due to multiple trust relationships that require separate keying material, or when service interfaces require separate metadata declarations. This may be accomplished through the use of the optional `<AdditionalMetaLocation>` element, or through the regexp facility and multiple service definition fields in the NAPTR resource record itself.

If the publishing protocol permits MIME-based identification of content types, the content type of the metadata instance MUST be `application/samlmetadata+xml`.

If the entity's unique identifier is a URN, publication of the corresponding metadata location proceeds as specified in [RFC3404]. Otherwise, the resolution of the metadata location proceeds as specified below.

The following is the application-specific profile of DDDS for SAML metadata resolution.

4.2.1.1 First Well Known Rule

The "first well-known-rule" for processing SAML metadata resolution is to parse the entity's unique identifier and extract the fully-qualified domain name (subexpression 3) as described in Section “Parsing the providerID”.

4.2.1.2 The Order Field

The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY provide multiple NAPTR resource records which MUST be processed by the resolver application in the order indicated by this field.

4.2.1.3 The Preference Field

For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving application. The resolving application MAY ignore this order, in cases where the service field value does not meet the resolver's requirements (e.g.: the resource record returns a protocol the application does not support).

4.2.1.4 The Flag Field

SAML metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying additional resource records are to be processed). The "U" flag indicates that the output of the rule is a URI.

4.2.1.5 The Service Field

The SAML-specific service field, as described in the following BNF, declares the modes by which instance document(s) shall be made available:

```
servicefield = 1("PID2U" / "NID2U") "+" proto [*(":" class) *(":" servicetype)]
proto = 1("https" / "uddi")
class = 1[ "entity" / "entitygroup" )
servicetype = 1(si / "spsso" / "idpsso" / "authn" / "authnauth" / "pdp" / "attrauth" /
"attrcons" / alphanum )
si = "si" [":" alphanum] [":endpoint"]
alphanum = 1*32(ALPHA / DIGIT)
```

where:

- `servicefield` PID2U resolves an entity's unique identifier to metadata URL.
- `servicefield` NID2U resolves a principal's `<NameIdentifier>` into a metadata URL.
- `proto` describes the retrieval protocol (`https` or `uddi`). In the case of UDDI, the URL will be an http(s) URL referencing a WSDL document.
- `class` identifies whether the referenced metadata document describes a single entity, or multiple. In the latter case, the referenced document MUST contain the entity defined by the original unique identifier as a member of a group of entities within the document itself such as an `<AffiliationDescriptor>` or `<EntitiesDescriptor>`.
- `servicetype` allows an entity to publish metadata for distinct roles and services as separate documents. Resolvers who encounter multiple `servicetype` declarations will dereference the appropriate URI, depending on which service is required for an operation (e.g.: an entity operating both as an identity provider and a service provider can publish metadata for each role at different locations). The `authn` service type represents a `<SingleSignOnService>` endpoint.
- `si` (with optional endpoint component) allows the publisher to either directly publish the metadata for a service instance, or by articulating a SOAP endpoint (using `endpoint`).

For example:

- `PID2U+https:entity` - represents the entity's complete metadata document available via the https protocol
- `PID2U+uddi:entity:si:foo` - represents the WSDL document location that describes a service instance "foo"
- `PID2U+https:entitygroup:idpsso` - represents the metadata for a group of entities acting as SSO identity providers, of which the original entity is a member.
- `NID2U+https:idp` - represents the metadata for the SSO identity provider of a principal

4.2.1.6 The Regex and Replacement Fields

The expected output after processing the input string through the regex MUST be a valid `https` URL or UDDI node (WSDL document) address.

4.2.2 NAPTR Examples

4.2.2.1 Entity Metadata NAPTR Examples

Entities publish metadata URLs in the following manner:

```
$ORIGIN provider.biz

;; order pref f service regexp or replacement

IN NAPTR 100 10 "U" PID2U+https:entity
   "!^.*$!https://host.provider.biz/some/directory/trust.xml!" ""
IN NAPTR 110 10 "U" PID2U+https: entity:trust
   "!^.*!https://foo.provider.biz:1443/mdtrust.xml!" ""
IN NAPTR 125 10 "U" PID2U+https:"
IN NAPTR 110 10 "U" PID2U+uddi:entity
   "!^.*$!https://this.uddi.node.provider.biz/libmd.wsdl" ""
```

4.2.2.2 Name Identifier Examples

A principal's employer `example.int` operates an identity provider which may be used by an office supply company to authenticate authorized buyers. The supplier takes a users' email address `buyer@example.int` as input to the resolution process, and parses the email address to extract the FQDN (example.int). The employer publishes the following NAPTR record in the `example.int` DNS:

```
$ORIGIN example.int

IN NAPTR 100 10 "U" NID2U+https:authn
   "!^([^@]+)@(.*)$!https://serv.example.int:8000/cgi-bin/getmd?\1!" ""
IN NAPTR 100 10 "U" NID2U+https:idp
   "!^([^@]+)@(.*)$!https://auth.example.int/app/auth?\1" ""
```

4.2.3 Resolution

When resolving metadata for an entity via the DNS, the unique identifier of the entity is used as the initial input into the resolution process, rather than as an actual location Proceed as follows:

- If the unique identifier is a URN, proceed with the resolution steps as defined in [RFC3404].
- Otherwise, parse the identifier to obtain the fully-qualified domain name.
- Query the DNS for NAPTR resource records of the domain iteratively until a terminal resource record is returned.
- Identify which resource record to use based on the service fields, then order fields, then preference fields of the result set.
- Obtain the document(s) at the provided location(s) as required by the application.

4.2.3.1 Parsing the Unique Identifier

To initiate the resolution of the location of the metadata information, it will be necessary in some cases to decompose the entity's unique identifier (expressed as a URI) into one or more atomic elements.

The following regular expression should be used when initiating the decomposition process:

```
   ^([^:/?#]+:)?/*([^:/?#]*@)?(([^/?:#]*\.)*(([^/?#:\.]+)\.([^/?#:\.]+)))
(:\d+)?([^?#]*)(\?[^#]*)?(#.*)?$
     1            2           34            56             7
8      9       10        11
```

Subexpression 3 MUST result in a Fully-Qualified Domain Name (FQDN), which will be the basis for retrieving metadata locations from this zone.

4.2.3.2 Obtaining Metadata via the DNS

Upon completion of the parsing of the identifier, the application then performs a DNS query for the resulting domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses. Applications MAY exclude from the result set any service definitions that do not concern the present request operations.

Resolving applications MUST subsequently order the result set according to the order field, and MAY order the result set based on the preference set. Resolvers are NOT REQUIRED to follow the ordering of the preferences field. The resulting NAPTR resource record(s) are operated on iteratively (based on the order flag) until a terminal NAPTR resource record is reached.

The result will be a well-formed, absolute URL, which is then used to retrieve the metadata document.

4.2.4 Metadata Location Caching

Location caching MUST NOT exceed the TTL of the DNS zone from which the location was derived. Resolvers MUST obtain a fresh copy of the metadata location upon reaching the expiration of the TTL of the zone.

Publishers of metadata documents should carefully consider the TTL of the zone when making changes to metadata document locations. Should such a location change occur, a publisher MUST either keep the document at both the old and new location until all conforming resolvers are certain to have the updated location (e.g.: time of zone change + TTL), or provide an HTTP Redirect [RFC2616] response at the old location specifying the new location.

4.3 Post-Processing of Metadata

The following sections describe the post-processing of metadata.

4.3.1 Metadata Instance Caching

Document caching MUST NOT exceed the `validUntil` or `cacheDuration` attribute of the subject element(s). If metadata elements have parent elements which contain caching policies, the parent element takes precedence.

To properly process the `cacheDuration` attribute, consumers MUST retain the date and time when the document was retrieved.

When a document or element has expired, the consumer MUST retrieve a fresh copy, which may require a refresh of the document location(s). Consumers SHOULD process document cache processing according to [RFC2616] Section 13, and MAY request the Last-Modified date and time from the HTTP server. Publishers SHOULD ensure acceptable cache processing as described in [RFC2616] (Section 10.3.5 304 Not Modified).

4.3.2 Handling of HTTPS Redirects

Publishers MAY issue an HTTP Redirect (301 Moved Permanently, 302 or 307 Temporary Redirect) [RFC2616], and user agents MUST follow the specified URL in the Redirect response. Redirects SHOULD be of the same protocol as the initial request.

4.3.3 Processing of XML Signatures and General Trust Processing

Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and for the trust ascribed to the entity described by such metadata:

- Trust derived from the signature of the DNS zone from which the metadata location URL was

resolved, ensuring accuracy of the metadata document location(s)

- Trust derived from signature processing of the metadata document itself, ensuring the integrity of the XML document
- Trust derived from the SSL/TLS server authentication of the metadata location URL, ensuring the identity of the publisher of the metadata

Post-processing of the metadata document MUST include signature processing at the XML-document level and MAY include one of the other two processes. Specifically, the relying party MAY choose to trust any of the cited authorities in the resolution and parsing process. Publishers of metadata MUST employ a document-integrity mechanism and MAY employ any of the other two processing profiles to establish trust in the metadata document, governed by implementation policies.

4.3.3.1 Processing Signed DNS Zones

Verification of DNS zone signature SHOULD be processed, if present, as described in [RFC2535].

4.3.3.2 Processing Signed Documents and Fragments

Published metadata documents SHOULD be signed, as described in Section 3, either by a certificate issued to the subject of the document, or by another trusted party. Publishers MAY consider signatures of other parties as a means of trust conveyance.

Metadata consumers MUST validate signatures, when present, on the metadata document as described by Section 3.

4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL

It is STRONGLY RECOMMENDED that publishers implement TLS/SSL URLs; therefore, consumers SHOULD consider the trust inherited from the issuer of the TLS/SSL certificate. Publication URLs may not always be located in the domain of the subject of the metadata document; therefore, consumers SHOULD NOT presume certificates whose subject is the entity in question, as it may be hosted by another trusted party.

As the basis of this trust may not be available against a cached document, other mechanisms SHOULD be used under such circumstances.

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] S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-bindings-2.0-cd-02. See http://www.oasis-open.org/committees/security/.

[SAMLConform] P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0.* OASIS SSTC, September 2004. Document ID sstc-saml-conformance-2.0-cd-02. http://www.oasis-open.org/committees/security/.

[SAMLCore] S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-core-2.0-cd-02. See http://www.oasis-open.org/committees/security/.

[SAMLMeta-xsd] S. Cantor et al., SAML metadata schema. OASIS SSTC, September 2004. Document ID sstc-saml-schema-metadata-2.0. See http://www.oasis-open.org/committees/security/.

[SAMLProf] S. Cantor et al., *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-profiles-2.0-cd-02. See http://www.oasis-open.org/committees/security/.

[SAMLSec] F. Hirsch et al., *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-sec-consider-2.0-cd-02. See http://www.oasis-open.org/committees/security/.

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

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

6 Registration of MIME media type application/samlmetadata+xml

To: ietf-types@iana.org
Subject: Registration of MIME media type `application/samlmetadata+xml`

Introduction

This document defines a MIME media type -- application/samlmetadata+xml -- for use with the XML serialization of Security Assertion Markup Language metadata.

SAML is a work product of the OASIS Security Services Technical Committee [SSTC]. The SAML specifications define XML-based constructs with which one may make, and convey, security assertions. Using SAML, one can assert that an authentication event pertaining to some subject has occurred and convey said assertion to a relying party, for example.

SAML profiles require agreements between system entities regarding identifiers, binding support, endpoints, certificates, keys, and so forth. Such information is treated as metadata by SAML v2.0. [SAMLv2Meta] specifies this metadata, as well as specifying metadata publication and resolution mechanisms. If the publishing protocol permits MIME-based identification of content types, then use of the `application/samlmetadata+xml` MIME media type is required.

MIME media type name: application

MIME subtype name: samlmetadata+xml

Required parameters: none

Optional parameters: charset

Same as charset parameter of application/xml [RFC3023].

Encoding considerations:

Same as for application/xml [RFC3023].

Security considerations:

Per their specification, samlmetadata+xml typed objects do not contain executable content. However, these objects are XML-based [XML], and thus they have all of the general security considerations presented in section 10 of [RFC3023].

SAML metadata [SAMLv2Meta] contains information whose integrity and authenticity is important – identity provider and service provider public keys and endpoint addresses, for example.

To counter potential issues, the publisher may sign samlmetadata+xml typed objects. Any such signature should be verified by the recipient of the data - both as a valid signature, and as being the signature of the publisher.

Additionally, various of the publication protocols, e.g. HTTP-over-TLS/SSL, offer means for ensuring the authenticity of the publishing party and for protecting the metadata in transit. [SAMLv2Meta] also defines prescriptive metadata caching directives, as well as guidance on handling HTTPS redirects, trust processing, server authentication, and related items.

For a more detailed discussion of SAML v2.0 metadata and its security considerations, please see [SAMLv2Meta]. For a discussion of overall SAML v2.0 security considerations and specific security-related design features, please refer to the SAML v2.0 specifications listed in the below bibliography. The specifications containing security-specific information are explicitly listed.

Interoperability considerations:

SAML v2.0 metadata explicitly supports identifying the protocols and versions supported by the identified entities. For example, an identity provider entity can be denoted as supporting SAML v2.0, SAML v1.1 [SAMLv1.1], Liberty ID-FF 1.2 [LAPFF], or even other protocols if they are unambiguously identifiable via URI [RFC2396]. This protocol support information is conveyed via the `protocolSupportEnumeration` attribute of metadata objects of the `RoleDescriptorType`.

Published specification:

[SAMLv2Meta] explicitly specifies use of the application/samlmetadata+xml MIME media type.

Applications which use this media type:

Potentially any application implementing SAML v2.0, as well as those applications implementing specifications based on SAML, e.g. those available from the Liberty Alliance [LAP].

Additional information:

Magic number(s):

In general, the same as for application/xml [RFC3023]. In particular, the XML root element of the returned object will have a namespace-qualified name with:

- a local name of: `EntityDescriptor`, `AffiliationDescriptor`, or `EntitiesDescriptor`
- a namespace URI of: `urn:oasis:names:tc:SAML:2.0:metadata` (the SAMLv2.0 metadata namespace)

File extension(s): none

Macintosh File Type Code(s): none

Person & email address to contact for further information:

This registration is made on behalf of the OASIS Security Services Technical Committee (SSTC) Please refer to the SSTC website for current information on committee chairperson(s) and their contact addresses: http://www.oasis-open.org/committees/security/. Committee members should submit comments and potential errata to the securityservices@lists.oasis-open.org list. Others should submit them by filling out the web form located at http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security.

Additionally, the SAML developer community email distribution list, saml-dev@lists.oasis-open.org, may be employed to discuss usage of the application/samlmetadata+xml MIME media type. The "saml-dev" mailing list is publicly archived here: http://lists.oasis-open.org/archives/saml-dev/. To post to the "saml-dev" mailing list, one must subscribe to it. To subscribe, send a message with the single word "subscribe" in the message body, to: saml-dev-request@lists.oasis-open.org.

Intended usage: COMMON

Author/Change controller:

The SAML specification sets are a work product of the OASIS Security Services Technical Committee (SSTC). OASIS and the SSTC have change control over the SAML specification sets.

Bibliography

[LAP] "*Liberty Alliance Project*". See http://www.projectliberty.org/

[LAPFF] "Liberty Alliance Project: Federation Framework". See http://www.projectliberty.org/resources/specifications.php#box1

[OASIS] "*Organization for the Advancement of Structured Information Systems*". See http://www.oasis-open.org/

[RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifiers (URI): Generic Syntax*. IETF RFC 2396, August 1998. Available at http://www.ietf.org/rfc/rfc2396.txt

[RFC3023] M. Murata, S. St.Laurent, D. Kohn, "*XML Media Types*", IETF Request for Comments 3023, January 2001. Available as http://www.rfc-editor.org/rfc/rfc3023.txt

[SAMLv1.1] OASIS Security Services Technical Committee, "*Security Assertion Markup Language (SAML) Version 1.1 Specification Set"*. OASIS Standard 200308, August 2003. Available as http://www.oasis-open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip

[SAMLv2.0] OASIS Security Services Technical Committee, "*Security Assertion Markup Language (SAML) Version 2.0 Specification Set"*. Committee Draft. Available at http://www.oasis-open.org/committees/security/

[SAMLv2Bind] S. Cantor et al., "*Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0"*. OASIS SSTC, September 2004. Document ID sstc-saml-bindings-2.0-cd-02. See http://www.oasis-open.org/committees/security/

[SAMLv2Core] S. Cantor et al., "*Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0"*. OASIS SSTC, September 2004. Document ID sstc-saml-core-2.0-cd-02. See http://www.oasis-open.org/committees/security/

[SAMLv2Meta] S. Cantor et al., *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-metadata-2.0-cd-02. See http://www.oasis-open.org/committees/security/

[SAMLv2Prof] S. Cantor et al., "*Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0"*. OASIS SSTC, September 2004. Document ID sstc-saml-profiles-2.0-cd-02. See http://www.oasis-open.org/committees/security/

[SAMLv2Sec] F. Hirsch et al., "*Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0"*. OASIS SSTC, September 2004. Document ID sstc-saml-sec-consider-2.0-cd-02. See http://www.oasis-open.org/committees/security/

[SSTC] "*OASIS Security Services Technical Committee*". See http://www.oasis-open.org/committees/security/

[XML] Bray, T., Paoli, J., Sperberg-McQueen, C.M. and E. Maler, François Yergeau, "*Extensible Markup Language (XML) 1.0 (Third Edition)*", World Wide Web Consortium Recommendation REC-xml, Feb 2004, Available as http://www.w3.org/TR/REC-xml/

Appendix A. Acknowledgments

The editors would like to acknowledge the contributions of the OASIS Security Services Technical Committee, whose voting members at the time of publication were:

- Conor Cahill, AOL
- John Hughes, ATOS Origin
- Hal Lockhart, BEA Systems
- Rick Randall, Booz Allen Hamilton
- Ronald Jacobson, Computer Associates
- Gavenraj Sodhi, Computer Associates
- Tim Alsop, CyberSafe Limited
- Paul Madsen, Entrust
- Carolina Canales-Valenzuela, Ericsson
- Dana Kaufman, Forum Systems
- Irving Reid, Hewlett-Packard
- Paula Austel, IBM
- Maryann Hondo, IBM
- Michael McIntosh, IBM
- Anthony Nadalin, IBM
- Nick Ragouzis, Individual
- Scott Cantor, Internet2
- Bob Morgan, Internet2
- Prateek Mishra, Netegrity
- Forest Yin, Netegrity
- Peter Davis, Neustar
- Frederick Hirsch, Nokia
- John Kemp, Nokia
- Senthil Sengodan, Nokia
- Scott Kiester, Novell
- Cameron Morris, Novell
- Charles Knouse, Oblix
- Steve Anderson, OpenNetwork
- Ari Kermaier, Oracle
- Vamsi Motukuru, Oracle
- Darren Platt, Ping Identity
- Jim Lien, RSA Security
- John Linn, RSA Security
- Rob Philpott, RSA Security
- Dipak Chopra, SAP
- Jahan Moreh, Sigaba
- Bhavna Bhatnagar, Sun Microsystems
- Jeff Hodges, Sun Microsystems
- Eve Maler, Sun Microsystems

- Ronald Monzillo, Sun Microsystems
- Emily Xu, Sun Microsystems
- Mike Beach, Boeing
- Greg Whitehead, Trustgenix

The editors also would like to acknowledge the following people for their contributions to previous versions of the OASIS Security Assertions Markup Language Standard:

- Stephen Farrell, Baltimore Technologies
- David Orchard, BEA Systems
- Krishna Sankar, Cisco Systems
- Zahid Ahmed, CommerceOne
- Carlisle Adams, Entrust
- Tim Moses, Entrust
- Nigel Edwards, Hewlett-Packard
- Joe Pato, Hewlett-Packard
- Bob Blakley, IBM
- Marlena Erdos, IBM
- Marc Chanliau, Netegrity
- Chris McLaren, Netegrity
- Lynne Rosenthal, NIST
- Mark Skall, NIST
- Simon Godik, Overxeer
- Charles Norwood, SAIC
- Evan Prodromou, Securant
- Robert Griffin, RSA Security (former editor)
- Sai Allarvarpu, Sun Microsystems
- Chris Ferris, Sun Microsystems
- Emily Xu, Sun Microsystems
- Mike Myers, Traceroute Security
- Phillip Hallam-Baker, VeriSign (former editor)
- James Vanderbeek, Vodafone
- Mark O'Neill, Vordel
- Tony Palmer, Vordel

Finally, the editors wish to acknowledge the following people for their contributions of material used as input to the OASIS Security Assertions Markup Language specifications:

- Thomas Gross, IBM
- Birgit Pfitzmann, IBM

Appendix B. 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.