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

Working Draft 05, 8 September 2015

Document identifier:

sstc-saml-metadata-errata-2.0-wd-05

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 (errata editor)
Eric Goodman, Individual (errata editor)

Contributors to the Errata:

Rob Philpott, EMC Corporation
Nick Ragouzis, Enosis Group
Thomas Wisniewski, Entrust
Greg Whitehead, HP
Heather Hinton, IBM
Connor P. Cahill, Intel
Scott Cantor, Internet2
Nate Klingenstein, Internet2
RL 'Bob' Morgan, Internet2
John Bradley, Individual
Jeff Hodges, Individual
Joni Brennan, Liberty Alliance
Eric Tiffany, Liberty Alliance
Thomas Hardjono, M.I.T.
Tom Scavo, NCSA
Peter Davis, NeuStar, Inc.
Frederick Hirsch, Nokia Corporation
Paul Madsen, NTT Corporation
Ari Kermaier, Oracle Corporation
Hal Lockhart, Oracle Corporation
Prateek Mishra, Oracle Corporation
Brian Campbell, Ping Identity
Anil Saldhana, Red Hat Inc.
Jim Lien, RSA Security
Jahan Moreh, Sigaba
Kent Spaulding, Skyworth TTG Holdings Limited
Emily Xu, Sun Microsystems
David Staggs, Veteran's Health Administration

SAML V2.0 Contributors:

Conor P. Cahill, AOL
John Hughes, Atos Origin

48   Hal Lockhart, BEA Systems
49   Michael Beach, Boeing
50   Rebekah Metz, Booz Allen Hamilton
51   Rick Randall, Booz Allen Hamilton
52   Thomas Wisniewski, Entrust
53   Irving Reid, Hewlett-Packard
54   Paula Austel, IBM
55   Maryann Hondo, IBM
56   Michael McIntosh, IBM
57   Tony Nadalin, IBM
58   Nick Ragouzis, Individual
59   Scott Cantor, Internet2
60   RL 'Bob' Morgan, Internet2
61   Peter C Davis, Neustar
62   Jeff Hodges, Neustar
63   Frederick Hirsch, Nokia
64   John Kemp, Nokia
65   Paul Madsen, NTT
66   Steve Anderson, OpenNetwork
67   Prateek Mishra, Principal Identity
68   John Linn, RSA Security
69   Rob Philpott, RSA Security
70   Jahan Moreh, Sigaba
71   Anne Anderson, Sun Microsystems
72   Eve Maler, Sun Microsystems
73   Ron Monzillo, Sun Microsystems
74   Greg Whitehead, Trustgenix

Abstract:

75

76 SAML profiles require agreements between system entities regarding identifiers, binding support
77 and endpoints, certificates and keys, and so forth. A metadata specification is useful for
78 describing this information in a standardized way. The SAML V2.0 Metadata document defines an
79 extensible metadata format for SAML system entities, organized by roles that reflect SAML
80 profiles. Such roles include that of Identity Provider, Service Provider, Affiliation, Attribute
81 Authority, Attribute Consumer, and Policy Decision Point. This document, known as an "errata
82 composite", combines corrections to reported errata with the original specification text. By design,
83 the corrections are limited to clarifications of ambiguous or conflicting specification text. This
84 document shows deletions from the original specification as struck-through text, and additions as
85 colored underlined text. The "[E*nn*]" designations embedded in the text refer to particular errata
86 and their dispositions.

Status:

88 This errata composite document is a **working draft** based on the original OASIS Standard
89 document that had been produced by the Security Services Technical Committee and approved
90 by the OASIS membership on 1 March 2005. While the errata corrections appearing here are
91 non-normative, they reflect changes specified by the Approved Errata document (currently at
92 Working Draft revision 02), which is on an OASIS standardization track. In case of any
93 discrepancy between this document and the Approved Errata, the latter has precedence.

94 This document includes corrections for errata E7, E33, E34, E37, E41, E62, E66, E68, E69, E74,
95 E76, E77, E81, E82, E83, E87, E88, E89, E91, and E94.

96 Committee members should submit comments and potential errata to the security-
97 services@lists.oasis-open.org list. Others should submit them by following the instructions at
98 http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security.

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

Table of Contents

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.

[E77]A variety of extension points are also included to allow for the use of SAML metadata in non-SAML specifications, profiles, and deployments, and such use is encouraged.

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 [Schema1]. 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 [E77](and potentially non-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 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/2002/REC-xmldsig-core-
20020212/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="saml-schema-assertion-2.0.xsd"/>
    <import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="http://www.w3.org/2001/xml.xsd"/>
```

```
242        <annotation>
243            <documentation>
244                Document identifier: saml-schema-metadata-2.0
245                Location: http://docs.oasis-open.org/security/saml/v2.0/
246                Revision history:
247                  V2.0 (March, 2005):
248                     Schema for SAML metadata, first published in SAML 2.0.
249            </documentation>
250        </annotation>
251    …
252    </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 [E77]protocol binding endpoint at which an [E77] 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 [E77] 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. [E41]If the `ResponseLocation` attribute is omitted, any response messages associated with a protocol or profile may be assumed to be handled at the URI indicated by the `Location` attribute.

288 In most contexts, elements of this type appear in unbounded sequences in the schema. This is to permit a
289 protocol or profile to be offered by an entity at multiple endpoints, usually with different protocol bindings,
290 allowing the metadata consumer to choose an appropriate endpoint for its needs. Multiple endpoints
291 might also offer "client-side" load-balancing or failover, particular in the case of a synchronous protocol
292 binding.

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

295 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

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

310 `index` [Required]

311   A required attribute that assigns a unique integer value to the endpoint so that it can be
312   referenced in a protocol message. The index value need only be unique within a collection of like
313   elements contained within the same parent element (i.e., they need not be unique across the
314   entire instance).

315 `isDefault` [Optional]

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

318 In any such sequence of [E37]indexed endpoints that share a common element name and namespace
319 (i.e. all instances of `<md:AssertionConsumerService>` within a role), the default endpoint is the first
320 such endpoint with the `isDefault` attribute set to `true`. If no such endpoints exist, the default endpoint
321 is the first such endpoint without the `isDefault` attribute set to `false`. If no such endpoints exist, the
322 default endpoint is the first element in the sequence.

323 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

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

335
```
<complexType name="localizedNameType">
```

```
336        <simpleContent>
337            <extension base="string">
338                <attribute ref="xml:lang" use="required"/>
339            </extension>
340        </simpleContent>
341    </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:

```
346    <complexType name="localizedURIType">
347        <simpleContent>
348            <extension base="anyURI">
349                <attribute ref="xml:lang" use="required"/>
350            </extension>
351        </simpleContent>
352    </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[E77] 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 [E94] before attempting to refresh it.

`Name` [Optional]

> A string name that identifies a group of[E77] 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.

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

379   Contains the metadata for one or more[E77] entities, or a nested group of additional metadata.

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

383 [E76]When not used as the root element of a metadata instance, a `validUntil` or `cacheDuration`
384 attribute MAY be used to impose a shorter expiration or cache duration than that of the parent or root
385 element, but never a longer one; the smaller value takes precedence.

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

```
388  <element name="EntitiesDescriptor" type="md:EntitiesDescriptorType"/>
389  <complexType name="EntitiesDescriptorType">
390     <sequence>
391        <element ref="ds:Signature" minOccurs="0"/>
392        <element ref="md:Extensions" minOccurs="0"/>
393        <choice minOccurs="1" maxOccurs="unbounded">
394           <element ref="md:EntityDescriptor"/>
395           <element ref="md:EntitiesDescriptor"/>
396        </choice>
397     </sequence>
398     <attribute name="validUntil" type="dateTime" use="optional"/>
399     <attribute name="cacheDuration" type="duration" use="optional"/>
400     <attribute name="ID" type="ID" use="optional"/>
401     <attribute name="Name" type="string" use="optional"/>
402  </complexType>
403  <element name="Extensions" type="md:ExtensionsType"/>
404  <complexType final="#all" name="ExtensionsType">
405     <sequence>
406        <any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
407     </sequence>
408  </complexType>
```

409 ## 2.3.2 Element <EntityDescriptor>

410 The `<EntityDescriptor>` element specifies metadata for a single[E77] entity. A single entity may act
411 in many different roles in the support of multiple profiles. This specification directly supports the following
412 concrete roles as well as the abstract `<RoleDescriptor>` element for extensibility (see subsequent
413 sections for more details):

414   • SSO Identity Provider

415   • SSO Service Provider

416   • Authentication Authority

417   • Attribute Authority

418   • Policy Decision Point

419   • Affiliation

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

421 `entityID` [Required]

422   Specifies the unique identifier of the[E77] entity whose metadata is described by the element's
423   contents.

424 `ID` [Optional]

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

426 `validUntil` [Optional]

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

429 `cacheDuration` [Optional]

430   Optional attribute indicates the maximum length of time a consumer should cache the metadata
431   contained in the element and any contained elements [E94] before attempting to refresh it.

432 `<ds:Signature>` [Optional]

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

435 `<Extensions>` [Optional]

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

439 `<RoleDescriptor>`, `<IDPSSODescriptor>`, `<SPSSODescriptor>`,
440 `<AuthnAuthorityDescriptor>`, `<AttributeAuthorityDescriptor>`, `<PDPDescriptor>` [One
441 or More]

442 **OR**

443 `<AffiliationDescriptor>` [Required]

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

446 `<Organization>` [Optional]

447   Optional element identifying the organization responsible for the[E77] entity described by the
448   element.

449 `<ContactPerson>` [Zero or More]

450   Optional sequence of elements identifying various kinds of contact personnel.

451 `<AdditionalMetadataLocation>` [Zero or More]

452   Optional sequence of namespace-qualified locations where additional metadata exists for
453   the[E77] entity. This may include metadata in alternate formats or describing adherence to other
454   non-SAML specifications.

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

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

459 [E76]When not used as the root element of a metadata instance, a `validUntil` or `cacheDuration`
460 attribute MAY be used to impose a shorter expiration or cache duration than that of the parent or root
461 element, but never a longer one; the smaller value takes precedence.

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

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

```
469  <element name="EntityDescriptor" type="md:EntityDescriptorType"/>
470  <complexType name="EntityDescriptorType">
471      <sequence>
472          <element ref="ds:Signature" minOccurs="0"/>
473          <element ref="md:Extensions" minOccurs="0"/>
474          <choice>
475              <choice maxOccurs="unbounded">
476                  <element ref="md:RoleDescriptor"/>
477                  <element ref="md:IDPSSODescriptor"/>
478                  <element ref="md:SPSSODescriptor"/>
479                  <element ref="md:AuthnAuthorityDescriptor"/>
480                  <element ref="md:AttributeAuthorityDescriptor"/>
481                  <element ref="md:PDPDescriptor"/>
482              </choice>
483              <element ref="md:AffiliationDescriptor"/>
484          </choice>
485          <element ref="md:Organization" minOccurs="0"/>
486          <element ref="md:ContactPerson" minOccurs="0" maxOccurs="unbounded"/>
487          <element ref="md:AdditionalMetadataLocation" minOccurs="0"
488  maxOccurs="unbounded"/>
489      </sequence>
490      <attribute name="entityID" type="md:entityIDType" use="required"/>
491      <attribute name="validUntil" type="dateTime" use="optional"/>
492      <attribute name="cacheDuration" type="duration" use="optional"/>
493      <attribute name="ID" type="ID" use="optional"/>
494      <anyAttribute namespace="##other" processContents="lax"/>
495  </complexType>
```

2.3.2.1 Element <Organization>

497 The `<Organization>` element specifies basic information about an organization responsible for an[E77]
498 entity or role. The use of this element is always optional. Its content is informative in nature and does not
499 directly map to any core SAML elements or attributes. Its **OrganizationType** complex type consists of the
500 following elements:

501 `<Extensions>` [Optional]

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

505 `<OrganizationName>` [One or More]

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

507 `<OrganizationDisplayName>` [One or More]

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

509 `<OrganizationURL>` [One or More]

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

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

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

```
516  <element name="Organization" type="md:OrganizationType"/>
```

```
517    <complexType name="OrganizationType">
518       <sequence>
519          <element ref="md:Extensions" minOccurs="0"/>
520          <element ref="md:OrganizationName" maxOccurs="unbounded"/>
521          <element ref="md:OrganizationDisplayName" maxOccurs="unbounded"/>
522          <element ref="md:OrganizationURL" maxOccurs="unbounded"/>
523       </sequence>
524       <anyAttribute namespace="##other" processContents="lax"/>
525    </complexType>
526    <element name="OrganizationName" type="md:localizedNameType"/>
527    <element name="OrganizationDisplayName" type="md:localizedNameType"/>
528    <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 an[E77] 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.

[E82] At least one child element SHOULD be present in a <ContactPerson> element..

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

```
556    <element name="ContactPerson" type="md:ContactType"/>
557    <complexType name="ContactType">
558       <sequence>
559          <element ref="md:Extensions" minOccurs="0"/>
560          <element ref="md:Company" minOccurs="0"/>
561          <element ref="md:GivenName" minOccurs="0"/>
```

```
562        <element ref="md:SurName" minOccurs="0"/>
563        <element ref="md:EmailAddress" minOccurs="0" maxOccurs="unbounded"/>
564        <element ref="md:TelephoneNumber" minOccurs="0" maxOccurs="unbounded"/>
565     </sequence>
566     <attribute name="contactType" type="md:ContactTypeType" use="required"/>
567     <anyAttribute namespace="##other" processContents="lax"/>
568  </complexType>
569  <element name="Company" type="string"/>
570  <element name="GivenName" type="string"/>
571  <element name="SurName" type="string"/>
572  <element name="EmailAddress" type="anyURI"/>
573  <element name="TelephoneNumber" type="string"/>
574  <simpleType name="ContactTypeType">
575     <restriction base="string">
576        <enumeration value="technical"/>
577        <enumeration value="support"/>
578        <enumeration value="administrative"/>
579        <enumeration value="billing"/>
580        <enumeration value="other"/>
581     </restriction>
582  </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 an[E77] 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:

```
591  <element name="AdditionalMetadataLocation"
592  type="md:AdditionalMetadataLocationType"/>
593  <complexType name="AdditionalMetadataLocationType">
594     <simpleContent>
595        <extension base="anyURI">
596           <attribute name="namespace" type="anyURI" use="required"/>
597        </extension>
598     </simpleContent>
599  </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) defines a specific collection of operational behaviors 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.

611 `validUntil` [Optional]

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

614 `cacheDuration` [Optional]

615   Optional attribute indicates the maximum length of time a consumer should cache the metadata
616   contained in the element and any contained elements [E94] before attempting to refresh it

617 `protocolSupportEnumeration` [Required]

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

623 `errorURL` [Optional]

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

626 `<ds:Signature>` [Optional]

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

629 `<Extensions>` [Optional]

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

633 `<KeyDescriptor>` [Zero or More]

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

636 `<Organization>` [Optional]

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

639 `<ContactPerson>` [Zero or More]

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

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

643 [E76]A `validUntil` or `cacheDuration` attribute MAY be used to impose a shorter expiration or cache
644 duration than that of the parent or root element, but never a longer one; the smaller value takes
645 precedence.

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

```
648     <element name="RoleDescriptor" type="md:RoleDescriptorType"/>
649     <complexType name="RoleDescriptorType" abstract="true">
650       <sequence>
651         <element ref="ds:Signature" minOccurs="0"/>
652         <element ref="md:Extensions" minOccurs="0"/>
653         <element ref="md:KeyDescriptor" minOccurs="0" maxOccurs="unbounded"/>
654         <element ref="md:Organization" minOccurs="0"/>
```

```
655            <element ref="md:ContactPerson" minOccurs="0" maxOccurs="unbounded"/>
656        </sequence>
657        <attribute name="ID" type="ID" use="optional"/>
658        <attribute name="validUntil" type="dateTime" use="optional"/>
659        <attribute name="cacheDuration" type="duration" use="optional"/>
660        <attribute name="protocolSupportEnumeration" type="md:anyURIListType"
661    use="required"/>
662        <attribute name="errorURL" type="anyURI" use="optional"/>
663        <anyAttribute namespace="##other" processContents="lax"/>
664    </complexType>
665    <simpleType name="anyURIListType">
666        <list itemType="anyURI"/>
667    </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.

[E62]A use value of "signing" means that the contained key information is applicable to both signing and TLS/SSL operations performed by the entity when acting in the enclosing role.

A use value of "encryption" means that the contained key information is suitable for use in wrapping encryption keys for use by the entity when acting in the enclosing role.

If the use attribute is omitted, then the contained key information is applicable to both of the above uses.

[E68]The inclusion of multiple <KeyDescriptor> elements with the same use attribute (or no such attribute) indicates that any of the included keys may be used by the containing role or affiliation. A relying party SHOULD allow for the use of any of the included keys. When possible the signing or encrypting party SHOULD indicate as specifically as possible which key it used to enable more efficient processing.

[E69]The <ds:KeyInfo> element is a highly generic and extensible means of communicating key material. This specification takes no position on the allowable or suggested content of this element, nor on its meaning to a relying party. As a concrete example, no implications of including an X.509 certificate by value or reference are to be assumed. Its validity period, extensions, revocation status, and other relevant content may or may not be enforced, at the discretion of the relying party. The details of such processing, and their security implications, are out of scope; they may, however, be addressed by other SAML profiles.

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

```
700    <element name="KeyDescriptor" type="md:KeyDescriptorType"/>
701    <complexType name="KeyDescriptorType">
702        <sequence>
```

```
703        <element ref="ds:KeyInfo"/>
704        <element ref="md:EncryptionMethod" minOccurs="0"
705    maxOccurs="unbounded"/>
706      </sequence>
707      <attribute name="use" type="md:KeyTypes" use="optional"/>
708    </complexType>
709    <simpleType name="KeyTypes">
710      <restriction base="string">
711        <enumeration value="encryption"/>
712        <enumeration value="signing"/>
713      </restriction>
714    </simpleType>
715    <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:

```
736    <complexType name="SSODescriptorType" abstract="true">
737      <complexContent>
738        <extension base="md:RoleDescriptorType">
739          <sequence>
740            <element ref="md:ArtifactResolutionService" minOccurs="0"
741    maxOccurs="unbounded"/>
742            <element ref="md:SingleLogoutService" minOccurs="0"
743    maxOccurs="unbounded"/>
744            <element ref="md:ManageNameIDService" minOccurs="0"
745    maxOccurs="unbounded"/>
746            <element ref="md:NameIDFormat" minOccurs="0"
747    maxOccurs="unbounded"/>
748          </sequence>
749        </extension>
750      </complexContent>
751    </complexType>
752    <element name="ArtifactResolutionService" type="md:IndexedEndpointType"/>
753    <element name="SingleLogoutService" type="md:EndpointType"/>
754    <element name="ManageNameIDService" type="md:EndpointType"/>
```

```
755        <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 [E33]Query/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.

[E7]The `WantAuthnRequestsSigned` attribute is intended to indicate to service providers whether or not they can expect an unsigned `<AuthnRequest>` message to be accepted by the identity provider. The identity provider is not obligated to reject unsigned requests nor is a service provider obligated to sign its requests, although it might reasonably expect an unsigned request will be rejected. In some cases, a service provider may not even know which identity provider will ultimately receive and respond to its requests, so the use of this attribute in such a case cannot be strictly defined.

Furthermore, note that the specific method of signing that would be expected is binding dependent. The HTTP Redirect binding (see [SAMLBind]) requires that the signature be applied to the URL-encoded value rather than placed within the XML message, while other bindings generally permit the signature to be within the message in the usual fashion.

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

```
795        <element name="IDPSSODescriptor" type="md:IDPSSODescriptorType"/>
796        <complexType name="IDPSSODescriptorType">
797          <complexContent>
798            <extension base="md:SSODescriptorType">
799              <sequence>
```

```
800                 <element ref="md:SingleSignOnService" maxOccurs="unbounded"/>
801                 <element ref="md:NameIDMappingService" minOccurs="0"
802     maxOccurs="unbounded"/>
803                 <element ref="md:AssertionIDRequestService" minOccurs="0"
804     maxOccurs="unbounded"/>
805                 <element ref="md:AttributeProfile" minOccurs="0"
806     maxOccurs="unbounded"/>
807                 <element ref="saml:Attribute" minOccurs="0"
808     maxOccurs="unbounded"/>
809              </sequence>
810              <attribute name="WantAuthnRequestsSigned" type="boolean"
811     use="optional"/>
812           </extension>
813        </complexContent>
814     </complexType>
815     <element name="SingleSignOnService" type="md:EndpointType"/>
816     <element name="NameIDMappingService" type="md:EndpointType"/>
817     <element name="AssertionIDRequestService" type="md:EndpointType"/>
818     <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`. [E7]A value of `false` (or omission of this attribute) does not imply that the service provider will never sign its requests or that a signed request should be considered an error. However, an identity provider that receives an unsigned `<samlp:AuthnRequest>` message from a service provider whose metadata contains this attribute with a value of true MUST return a SAML error response and MUST NOT fulfill the request.

`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. [E7]Note that an enclosing signature at the SAML binding or protocol layer does not suffice to meet this requirement, for example signing a `<samlp:Response>` containing the assertion(s) or a TLS connection.

`<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`. [E87] The default element is the first element with the `isDefault` attribute set to true. If no such elements exist, the default element is the first element without the `isDefault` attribute set to false. If no such elements exist, the default element is the first element in the sequence.

The following schema fragment defines the `<SPSSODescriptor>` element and its

850 **SPSSODescriptorType** complex type:

```
851  <element name="SPSSODescriptor" type="md:SPSSODescriptorType"/>
852  <complexType name="SPSSODescriptorType">
853     <complexContent>
854        <extension base="md:SSODescriptorType">
855           <sequence>
856              <element ref="md:AssertionConsumerService"
857  maxOccurs="unbounded"/>
858              <element ref="md:AttributeConsumingService" minOccurs="0"
859  maxOccurs="unbounded"/>
860           </sequence>
861           <attribute name="AuthnRequestsSigned" type="boolean"
862  use="optional"/>
863           <attribute name="WantAssertionsSigned" type="boolean"
864  use="optional"/>
865        </extension>
866     </complexContent>
867  </complexType>
868  <element name="AssertionConsumerService" type="md:IndexedEndpointType"/>
```

2.4.4.1 Element <AttributeConsumingService>

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

873 index [Required]

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

876 isDefault [Optional]

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

879 <ServiceName> [One or More]

880   One or more language-qualified names for the service [E88] that are suitable for human
881   consumption.

882 <ServiceDescription> [Zero or More]

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

884 <RequestedAttribute> [One or More]

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

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

```
888  <element name="AttributeConsumingService"
889  type="md:AttributeConsumingServiceType"/>
890  <complexType name="AttributeConsumingServiceType">
891     <sequence>
892        <element ref="md:ServiceName" maxOccurs="unbounded"/>
893        <element ref="md:ServiceDescription" minOccurs="0"
894  maxOccurs="unbounded"/>
895        <element ref="md:RequestedAttribute" maxOccurs="unbounded"/>
896     </sequence>
897     <attribute name="index" type="unsignedShort" use="required"/>
898     <attribute name="isDefault" type="boolean" use="optional"/>
899  </complexType>
```

```
900        <element name="ServiceName" type="md:localizedNameType"/>
901        <element name="ServiceDescription" type="md:localizedNameType"/>
```

2.4.4.1.1 [E34]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).

[E89] If no `NameFormat` value is provided, the identifier `urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified` (see Section 8.2.1 of [SAMLv2Core]) is in effect

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:

```
916        <element name="RequestedAttribute" type="md:RequestedAttributeType"/>
917        <complexType name="RequestedAttributeType">
918           <complexContent>
919              <extension base="saml:AttributeType">
920                 <attribute name="isRequired" type="boolean" use="optional"/>
921              </extension>
922           </complexContent>
923        </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 [E33]Query/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:

```
941        <element name="AuthnAuthorityDescriptor"
942        type="md:AuthnAuthorityDescriptorType"/>
943        <complexType name="AuthnAuthorityDescriptorType">
944           <complexContent>
```

```
945          <extension base="md:RoleDescriptorType">
946             <sequence>
947                <element ref="md:AuthnQueryService" maxOccurs="unbounded"/>
948                <element ref="md:AssertionIDRequestService" minOccurs="0"
949    maxOccurs="unbounded"/>
950                <element ref="md:NameIDFormat" minOccurs="0"
951    maxOccurs="unbounded"/>
952             </sequence>
953          </extension>
954       </complexContent>
955    </complexType>
956    <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 [E33]Query/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:

```
974    <element name="PDPDescriptor" type="md:PDPDescriptorType"/>
975    <complexType name="PDPDescriptorType">
976       <complexContent>
977          <extension base="md:RoleDescriptorType">
978             <sequence>
979                <element ref="md:AuthzService" maxOccurs="unbounded"/>
980                <element ref="md:AssertionIDRequestService" minOccurs="0"
981    maxOccurs="unbounded"/>
982                <element ref="md:NameIDFormat" minOccurs="0"
983    maxOccurs="unbounded"/>
984             </sequence>
985          </extension>
986       </complexContent>
987    </complexType>
988    <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:

994 `<AttributeService>` [One or More]

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

998 `<AssertionIDRequestService>` [Zero or More]

999   Zero or more elements of type **EndpointType** that describe endpoints that support the profile of
1000   the Assertion [E33]Query/Request protocol defined in [SAMLProf] or the special URI binding for
1001   assertion requests defined in [SAMLBind].

1002 `<NameIDFormat>` [Zero or More]

1003   Zero or more elements of type **anyURI** that enumerate the name identifier formats supported by
1004   this authority. See Section 8.3 of [SAMLCore] for some possible values for this element.

1005 `<AttributeProfile>` [Zero or More]

1006   Zero or more elements of type **anyURI** that enumerate the attribute profiles supported by this
1007   authority. See [SAMLProf] for some possible values for this element.

1008 `<saml:Attribute>` [Zero or More]

1009   Zero or more elements that identify the SAML attributes supported by the authority. Specific
1010   values MAY optionally be included, indicating that only certain values permitted by the attribute's
1011   definition are supported.

1012 The following schema fragment defines the `<AttributeAuthorityDescriptor>` element and its
1013 **AttributeAuthorityDescriptorType** complex type:

```
1014  <element name="AttributeAuthorityDescriptor"
1015  type="md:AttributeAuthorityDescriptorType"/>
1016  <complexType name="AttributeAuthorityDescriptorType">
1017      <complexContent>
1018          <extension base="md:RoleDescriptorType">
1019              <sequence>
1020                  <element ref="md:AttributeService" maxOccurs="unbounded"/>
1021                  <element ref="md:AssertionIDRequestService" minOccurs="0"
1022  maxOccurs="unbounded"/>
1023                  <element ref="md:NameIDFormat" minOccurs="0"
1024  maxOccurs="unbounded"/>
1025                  <element ref="md:AttributeProfile" minOccurs="0"
1026  maxOccurs="unbounded"/>
1027                  <element ref="saml:Attribute" minOccurs="0"
1028  maxOccurs="unbounded"/>
1029              </sequence>
1030          </extension>
1031      </complexContent>
1032  </complexType>
1033  <element name="AttributeService" type="md:EndpointType"/>
```

1034   2.5 Element <AffiliationDescriptor>

1035 The `<AffiliationDescriptor>` element is an alternative to the sequence of role descriptors
1036 described in Section 2.4 that is used when an `<EntityDescriptor>` describes an affiliation of[E77]
1037 entities (typically service providers) rather than a single entity. The `<AffiliationDescriptor>`
1038 element provides a summary of the individual entities that make up the affiliation along with general
1039 information about the affiliation itself. Its **AffiliationDescriptorType** complex type contains the following
1040 elements and attributes:

1041 `affiliationOwnerID` [Required]

1042   Specifies the unique identifier of the entity responsible for the affiliation. The owner is NOT

1043   presumed to be a member of the affiliation; if it is a member, its identifier MUST also appear in an
1044   `<AffiliateMember>` element.

1045 `ID` [Optional]

1046   A document-unique identifier for the element, typically used as a reference point when signing.

1047 `validUntil` [Optional]

1048   Optional attribute indicates the expiration time of the metadata contained in the element and any
1049   contained elements.

1050 `cacheDuration` [Optional]

1051   Optional attribute indicates the maximum length of time a consumer should cache the metadata
1052   contained in the element and any contained elements [E94] before attempting to refresh it.

1053 `<ds:Signature>` [Optional]

1054   An XML signature that authenticates the containing element and its contents, as described in
1055   Section 3.

1056 `<Extensions>` [Optional]

1057   This contains optional metadata extensions that are agreed upon between a metadata publisher
1058   and consumer. Extension elements MUST be namespace-qualified by a non-SAML-defined
1059   namespace.

1060 `<AffiliateMember>` [One or More]

1061   One or more elements enumerating the members of the affiliation by specifying each member's
1062   unique identifier. See also Section 8.3.6 of [SAMLCore].

1063 `<KeyDescriptor>` [Zero or More]

1064   Optional sequence of elements that provides information about the cryptographic keys that the
1065   affiliation uses as a whole, as distinct from keys used by individual members of the affiliation,
1066   which are published in the metadata for those entities.

1067 Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

1068 [E76]A `validUntil` or `cacheDuration` attribute MAY be used to impose a shorter expiration or cache
1069 duration than that of the parent or root element, but never a longer one; the smaller value takes
1070 precedence.

1071 The following schema fragment defines the `<AffiliationDescriptor>` element and its
1072 **AffiliationDescriptorType** complex type:

```
1073 <element name="AffiliationDescriptor" type="md:AffiliationDescriptorType"/>
1074 <complexType name="AffiliationDescriptorType">
1075    <sequence>
1076       <element ref="ds:Signature" minOccurs="0"/>
1077       <element ref="md:Extensions" minOccurs="0"/>
1078       <element ref="md:AffiliateMember" maxOccurs="unbounded"/>
1079       <element ref="md:KeyDescriptor" minOccurs="0" maxOccurs="unbounded"/>
1080    </sequence>
1081    <attribute name="affiliationOwnerID" type="md:entityIDType"
1082 use="required"/>
1083    <attribute name="validUntil" type="dateTime" use="optional"/>
1084    <attribute name="cacheDuration" type="duration" use="optional"/>
1085    <attribute name="ID" type="ID" use="optional"/>
1086    <anyAttribute namespace="##other" processContents="lax"/>
1087 </complexType>
1088 <element name="AffiliateMember" type="md:entityIDType"/>
```

2.6 Examples

1090 The following is an example of metadata for a SAML system entity acting as an identity provider and an
1091 attribute authority. A signature is shown as a placeholder, without the actual content.
1092

```
1093    <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1094        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1095        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1096        entityID="https://IdentityProvider.com/SAML">
1097        <ds:Signature>...</ds:Signature>
1098        <IDPSSODescriptor WantAuthnRequestsSigned="true"
1099            protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1100            <KeyDescriptor use="signing">
1101                <ds:KeyInfo>
1102                    <ds:KeyName>IdentityProvider.com SSO Key</ds:KeyName>
1103                </ds:KeyInfo>
1104            </KeyDescriptor>
1105            <ArtifactResolutionService isDefault="true" index="0"
1106                Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1107                Location="https://IdentityProvider.com/SAML/Artifact"/>
1108            <SingleLogoutService
1109                Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1110                Location="https://IdentityProvider.com/SAML/SLO/SOAP"/>
1111            <SingleLogoutService
1112                Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1113                Location="https://IdentityProvider.com/SAML/SLO/Browser"
1114                ResponseLocation="https://IdentityProvider.com/SAML/SLO/Response"/>
1115            <NameIDFormat>
1116                urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1117            </NameIDFormat>
1118            <NameIDFormat>
1119                urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1120            </NameIDFormat>
1121            <NameIDFormat>
1122                urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1123            </NameIDFormat>
1124            <SingleSignOnService
1125                Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1126                Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1127            <SingleSignOnService
1128                Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1129                Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1130            <saml:Attribute
1131                NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1132                Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1133                FriendlyName="eduPersonPrincipalName">
1134            </saml:Attribute>
1135            <saml:Attribute
1136                NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1137                Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1138                FriendlyName="eduPersonAffiliation">
1139                    <saml:AttributeValue>member</saml:AttributeValue>
1140                    <saml:AttributeValue>student</saml:AttributeValue>
1141                    <saml:AttributeValue>faculty</saml:AttributeValue>
1142                    <saml:AttributeValue>employee</saml:AttributeValue>
1143                    <saml:AttributeValue>staff</saml:AttributeValue>
1144            </saml:Attribute>
1145        </IDPSSODescriptor>
1146        <AttributeAuthorityDescriptor
1147            protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1148            <KeyDescriptor use="signing">
1149                <ds:KeyInfo>
1150                    <ds:KeyName>IdentityProvider.com AA Key</ds:KeyName>
1151                </ds:KeyInfo>
```

```
1152            </KeyDescriptor>
1153            <AttributeService
1154              Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1155              Location="https://IdentityProvider.com/SAML/AA/SOAP"/>
1156            <AssertionIDRequestService
1157              Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
1158              Location="https://IdentityProvider.com/SAML/AA/URI"/>
1159            <NameIDFormat>
1160              urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1161            </NameIDFormat>
1162            <NameIDFormat>
1163              urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1164            </NameIDFormat>
1165            <NameIDFormat>
1166              urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1167            </NameIDFormat>
1168            <saml:Attribute
1169              NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1170              Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1171              FriendlyName="eduPersonPrincipalName">
1172            </saml:Attribute>
1173            <saml:Attribute
1174              NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1175              Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1176              FriendlyName="eduPersonAffiliation">
1177                <saml:AttributeValue>member</saml:AttributeValue>
1178                <saml:AttributeValue>student</saml:AttributeValue>
1179                <saml:AttributeValue>faculty</saml:AttributeValue>
1180                <saml:AttributeValue>employee</saml:AttributeValue>
1181                <saml:AttributeValue>staff</saml:AttributeValue>
1182            </saml:Attribute>
1183        </AttributeAuthorityDescriptor>
1184        <Organization>
1185            <OrganizationName xml:lang="en">Identity Providers R
1186    US</OrganizationName>
1187            <OrganizationDisplayName xml:lang="en">
1188              Identity Providers R US, a Division of Lerxst Corp.
1189            </OrganizationDisplayName>
1190            <OrganizationURL
1191    xml:lang="en">https://IdentityProvider.com</OrganizationURL>
1192        </Organization>
1193    </EntityDescriptor>
1194
```

1195 The following is an example of metadata for a SAML system entity acting as a service provider. A
1196 signature is shown as a placeholder, without the actual content. For illustrative purposes, the service is
1197 one that does not require users to uniquely identify themselves, but rather authorizes access on the basis
1198 of a role-like attribute.
1199

```
1200    <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1201        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1202        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1203        entityID="https://ServiceProvider.com/SAML">
1204        <ds:Signature>...</ds:Signature>
1205        <SPSSODescriptor AuthnRequestsSigned="true"
1206            protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1207            <KeyDescriptor use="signing">
1208                <ds:KeyInfo>
1209                    <ds:KeyName>ServiceProvider.com SSO Key</ds:KeyName>
1210                </ds:KeyInfo>
1211            </KeyDescriptor>
1212            <KeyDescriptor use="encryption">
1213                <ds:KeyInfo>
1214                    <ds:KeyName>ServiceProvider.com Encrypt Key</ds:KeyName>
1215                </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. [E81] Any algorithm defined for use with the XML Signature specification MAY be used.

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>`

1294 algorithm. [E83] Use of Exclusive Canonicalization facilitates the verification of signatures created over
1295 SAML messages when placed into a different XML context than present during signing.

1296 Note that use of this algorithm alone does not guarantee that a particular signed object can be moved
1297 from one context to another safely, nor is that a requirement of signed SAML objects in general, though it
1298 MAY be required by particular profiles.

3.1.4 Transforms

1299

1300 Signatures in SAML metadata SHOULD NOT contain transforms other than the enveloped signature
1301 transform (with the identifier http://www.w3.org/2000/09/xmldsig#enveloped-signature) or the exclusive
1302 canonicalization transforms (with the identifier http://www.w3.org/2001/10/xml-exc-c14n# or
1303 http://www.w3.org/2001/10/xml-exc-c14n#WithComments).

1304 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
1305 not, verifiers MUST ensure that no content of the signed metadata element is excluded from the
1306 signature. This can be accomplished by establishing out-of-band agreement as to what transforms are
1307 acceptable, or by applying the transforms manually to the content and reverifying the result as consisting
1308 of the same SAML metadata.

3.1.5 [E91] Object

1309

1310 The `<ds:Object>` element is not defined for use with SAML metadata signatures, and SHOULD NOT be
1311 present. Since it can be used in service of an attacker by carrying unsigned data, verifiers SHOULD reject
1312 signatures that contain a `<ds:Object>` element.

3.1.6 KeyInfo

1313

1314 XML Signature [XMLSig] defines usage of the `<ds:KeyInfo>` element. SAML does not require the
1315 use of `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY
1316 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 `<AdditionalMetadataLocation>` 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

1357 subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in [RFC2915]
1358 and [RFC3403].

1359 It is RECOMMENDED that entities publish their resource records in signed zone files using [E66]
1360 [RFC4035] such that relying parties may establish the validity of the published location and authority of
1361 the zone, and integrity of the DNS response. If DNS zone signatures are present, relying parties MUST
1362 properly validate the signature.

4.2.1 Publication

1364 This specification makes use of the NAPTR resource record described in [RFC2915] and [RFC3403].
1365 Familiarity with these documents is encouraged.

1366 Dynamic Delegation Discovery System (DDDS) [RFC3401]is a general purpose system for the retrieval of
1367 information based on an application-specific input string and the application of well known rules to
1368 transform that string until a terminal condition is reached requiring a look-up into an application-specific
1369 defined database or resolution of a URL based on the rules defined by the application. DDDS defines a
1370 specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS
1371 necessary to apply DDDS rules.

1372 Entities MAY publish separate URLs when multiple metadata documents need to be distributed, or when
1373 different metadata documents are required due to multiple trust relationships that require separate keying
1374 material, or when service interfaces require separate metadata declarations. This may be accomplished
1375 through the use of the optional `<AdditionalMetadataLocation>` element, or through the `regexp`
1376 facility and multiple service definition fields in the NAPTR resource record itself.

1377 If the publishing protocol permits MIME-based identification of content types, the content type of the
1378 metadata instance MUST be `application/samlmetadata+xml`.

1379 If the entity's unique identifier is a URN, publication of the corresponding metadata location proceeds as
1380 specified in [RFC3404]. Otherwise, the resolution of the metadata location proceeds as specified below.

1381 The following is the application-specific profile of DDDS for SAML metadata resolution.

4.2.1.1 First Well Known Rule

1383 The "first well-known-rule" for processing SAML metadata resolution is to parse the entity's unique
1384 identifier and extract the fully-qualified domain name (subexpression 3) as described in Section 4.2.3.1.

4.2.1.2 The Order Field

1386 The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY
1387 provide multiple NAPTR resource records which MUST be processed by the resolver application in the
1388 order indicated by this field.

4.2.1.3 The Preference Field

1390 For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving
1391 application. The resolving application MAY ignore this order, in cases where the service field value does
1392 not meet the resolver's requirements (e.g.: the resource record returns a protocol the application does not
1393 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" /
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 `<NameID>` 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

[E94] Document caching MUST be based on the duration indicated by the `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 an instance was obtained.

Note that cache expiration does not imply a lack of validity in the absence of a `validUntil` attribute or other information; failure to update a cached instance (e.g., due to network failure) need not render metadata invalid, although implementations may offer such controls to deployers.

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 [E94] Metadata Instance Validity

Metadata MUST be considered invalid upon reaching the time specified in a `validUntil` attribute of the subject element(s). The effective expiration may be adjusted downward by parent element(s) with earlier expirations. Invalid metadata MUST NOT be used. This contrasts with "stale" metadata that may be beyond its optimum cache duration but is not explicitly invalid. Such metadata remains valid and MAY be used at the discretion of the implementation.

4.3.3 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.

4.3.4 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.

4.3.4.1 Processing Signed DNS Zones

Verification of DNS zone signature SHOULD be processed, if present, as described in [E66][RFC4035].

4.3.4.2 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.

4.3.4.3 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

[RFC1034]	P. Mockapetris. *Domain Names – Concepts and Facilities*. IETF RFC 1034, November 1987. See http://www.ietf.org/rfc/rfc1034.txt.
[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.
[RFC2246]	T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. See http://www.ietf.org/rfc/rfc2246.txt.
[E66]**[RFC2616]**	R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. IETF RFC 2616, June 1999. See http://www.ietf.org/rfc/rfc2616.txt.
[RFC2915]	M. Mealling. *The Naming Authority Pointer (NAPTR) DNS Resource Record*. IETF RFC 2915, September 2000. See http://www.ietf.org/rfc/rfc2915.txt.
[RFC3401]	M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS*. IETF RFC 3401, October 2002. See http://www.ietf.org/rfc/rfc3401.txt.
[RFC3403]	M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database*. IETF RFC 3403, October 2002. See http://www.ietf.org/rfc/rfc3403.txt.
[RFC3404]	M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Four: The Uniform Resource Identifiers (URI) Resolution Application*. IETF RFC 3404, October 2002. See http://www.ietf.org/rfc/rfc3404.txt.
[RFC3546]	S. Blake-Wilson et al. *Transport Layer Security (TLS) Extensions*. IETF RFC 3546, June 2003. See http://www.ietf.org/rfc/rfc3546.txt.
[E66]**[RFC4035]**	R. Arends et al. *Protocol Modifications for the DNS Security Extensions*. IETF RFC 4035, March 2005. See http://www.ietf.org/rfc/rfc4035.txt.
[SAMLBind]	S. Cantor et al. *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os. 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, March 2005. Document ID saml-conformance-2.0-os. 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, March 2005. Document ID saml-core-2.0-os. See http://www.oasis-open.org/committees/security/.
[SAMLMeta-xsd]	S. Cantor et al. SAML metadata schema. OASIS SSTC, March 2005. Document ID 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, March 2005. Document ID saml-profiles-2.0-os. 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, March 2005. Document ID saml-sec-consider-2.0-os. See http://www.oasis-open.org/committees/security/.
[Schema1]	H. S. Thompson et al. *XML Schema Part 1: Structures.* World Wide Web Consortium Recommendation, May 2001. See http://www.w3.org/TR/xmlschema-1/. Note that this specification normatively references [Schema2], listed below.
[Schema2]	P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium Recommendation, May 2001. See http://www.w3.org/TR/xmlschema-

1603   **[XMLEnc]**   D. Eastlake et al. *XML-Encryption Syntax and Processing*,
1604   http://www.w3.org/TR/xmlenc-core/, World Wide Web Consortium.

1605   **[XMLSig]**   D. Eastlake et al. *XML-Signature Syntax and Processing, [E74] Second Edition.*
1606   World Wide Web Consortium, June 2008. See http://www.w3.org/TR/xmldsig-
1607   core/.

Appendix A.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 Section10 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.

| 1646 | [SAMLv2Meta] also defines prescriptive metadata caching directives, as well as guidance on |
| 1647 | handling HTTPS redirects, trust processing, server authentication, and related items. |

1648	For a more detailed discussion of SAML v2.0 metadata and its security considerations, please
1649	see [SAMLv2Meta]. For a discussion of overall SAML v2.0 security considerations and specific
1650	security-related design features, please refer to the SAML v2.0 specifications listed in the below
1651	bibliography. The specifications containing security-specific information are explicitly listed.

Interoperability considerations

| 1652 | |

1653	SAML v2.0 metadata explicitly supports identifying the protocols and versions supported by the
1654	identified entities. For example, an identity provider entity can be denoted as supporting SAML
1655	v2.0 [SAMLv2.0], SAML v1.1 [SAMLv1.1], Liberty ID-FF 1.2 [LAPFF], or even other protocols if
1656	they are unambiguously identifiable via URI [RFC2396]. This protocol support information is
1657	conveyed via the `protocolSupportEnumeration` attribute of metadata objects of the
1658	**RoleDescriptorType**.

Published specification

| 1659 | |

| 1660 | [SAMLv2Meta] explicitly specifies use of the `application/samlmetadata+xml` MIME media |
| 1661 | type. |

Applications which use this media type

| 1662 | |

| 1663 | Potentially any application implementing SAML v2.0, as well as those applications implementing |
| 1664 | specifications based on SAML, e.g. those available from the Liberty Alliance [LAP]. |

Additional information

| 1665 | |

Magic number(s)

| 1666 | |

1667	In general, the same as for `application/xml` [RFC3023]. In particular, the XML root element of
1668	the returned object will have a namespace-qualified name with:
1669	

1670	– a local name of: `EntityDescriptor`, or
1671	`AffiliationDescriptor`, or
1672	`EntitiesDescriptor`

| 1674 | – a namespace URI of: `urn:oasis:names:tc:SAML:2.0:metadata` |
| 1675 | (the SAMLv2.0 metadata namespace) |

File extension(s)

| 1676 | |

| 1677 | None |

Macintosh File Type Code(s)

| 1678 | |

| 1679 | None |

Person & email address to contact for further information

| 1680 | |

1681	This registration is made on behalf of the OASIS Security Services Technical Committee (SSTC)
1682	Please refer to the SSTC website for current information on committee chairperson(s) and their
1683	contact addresses: http://www.oasis-open.org/committees/security/. Committee members should
1684	submit comments and potential errata to the securityservices@lists.oasis-open.org list. Others
1685	should submit them by filling out the web form located at http://www.oasis-
1686	open.org/committees/comments/form.php?wg_abbrev=security.

1687

1688 Additionally, the SAML developer community email distribution list, saml-dev@lists.oasis-
1689 open.org, may be employed to discuss usage of the `application/samlmetadata+xml` MIME
1690 media type. The "saml-dev" mailing list is publicly archived here: http://lists.oasis-
1691 open.org/archives/saml-dev/. To post to the "saml-dev" mailing list, one must subscribe to it. To
1692 subscribe, send a message with the single word "subscribe" in the message body, to: saml-dev-
1693 request@lists.oasis-open.org.

Intended usage

1695 COMMON

Author/Change controller

1697 The SAML specification sets are a work product of the OASIS Security Services Technical
1698 Committee (SSTC). OASIS and the SSTC have change control over the SAML specification sets.

Bibliography

1700 [LAP]   "*Liberty Alliance Project*". See http://www.projectliberty.org/
1701 [LAPFF]   "Liberty Alliance Project: Federation Framework". See
1702   http://www.projectliberty.org/resources/specifications.php#box1
1703 [OASIS]   "*Organization for the Advancement of Structured Information Systems*".
1704   See http://www.oasis-open.org/
1705 [RFC2396]   T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifiers
1706   (URI): Generic Syntax*. IETF RFC 2396, August 1998. Available at
1707   http://www.ietf.org/rfc/rfc2396.txt
1708 [RFC3023]   M. Murata, S. St.Laurent, D. Kohn, "*XML Media Types*", IETF Request
1709   for Comments 3023, January 2001. Available as http://www.rfc-
1710   editor.org/rfc/rfc3023.txt
1711 [SAMLv1.1]   OASIS Security Services Technical Committee, "*Security Assertion
1712   Markup Language (SAML) Version 1.1 Specification Set*". OASIS
1713   Standard 200308, August 2003. Available as http://www.oasis-
1714   open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-
1715   xsd.zip
1716 [SAMLv2.0]   OASIS Security Services Technical Committee, "*Security Assertion
1717   Markup Language (SAML) Version 2.0 Specification Set*". OASIS
1718   Standard, 15-Mar-2005. Available at: http://docs.oasis-
1719   open.org/security/saml/v2.0/saml-2.0-os.zip
1720 [SAMLv2Bind]   S. Cantor et al., "*Bindings for the OASIS Security Assertion Markup
1721   Language (SAML) V2.0*". OASIS, March 2005. Document ID saml-
1722   bindings-2.0-os. Available at: http://docs.oasis-
1723   open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf
1724 [SAMLv2Core]   S. Cantor et al., "*Assertions and Protocols for the OASIS Security
1725   Assertion Markup Language (SAML) V2.0*". OASIS, March 2005.
1726   Document ID saml-core-2.0-os. Available at: http://docs.oasis-
1727   open.org/security/saml/v2.0/saml-core-2.0-os.pdf
1728 [SAMLv2Meta]   S. Cantor et al., *Metadata for the OASIS Security Assertion Markup
1729   Language (SAML) V2.0*. OASIS SSTC, August 2004. Document ID saml-
1730   metadata-2.0-os. Available at: http://docs.oasis-
1731   open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf
1732 [SAMLv2Prof]   S. Cantor et al., "*Profiles for the OASIS Security Assertion Markup
1733   Language (SAML) V2.0*". OASIS, March 2005. Document ID saml-
1734   profiles-2.0-os. Available at: http://docs.oasis-
1735   open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

1736	[SAMLv2Sec]	F. Hirsch et al., "*Security and Privacy Considerations for the OASIS*
1737		*Security Assertion Markup Language (SAML) V2.0"*. OASIS, March
1738		2005. Document ID saml-sec-consider-2.0-os. Available at:
1739		http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf
1740	[SSTC]	"*OASIS Security Services Technical Committee"*. See http://www.oasis-
1741		open.org/committees/security/
1742	[XML]	Bray, T., Paoli, J., Sperberg-McQueen, C.M. and E. Maler, François
1743		Yergeau, "*Extensible Markup Language (XML) 1.0 (Third Edition)*", World
1744		Wide Web Consortium Recommendation REC-xml, Feb 2004, Available
1745		as http://www.w3.org/TR/REC-xml/
1746		

Appendix B. 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
- Mike Beach, Boeing
- Rebekah Metz, Booz Allen Hamilton
- Rick Randall, Booz Allen Hamilton
- Ronald Jacobson, Computer Associates
- Gavenraj Sodhi, Computer Associates
- Thomas Wisniewski, Entrust
- Carolina Canales-Valenzuela, Ericsson
- Dana Kaufman, Forum Systems
- Irving Reid, Hewlett-Packard
- Guy Denton, IBM
- Heather Hinton, IBM
- Maryann Hondo, IBM
- Michael McIntosh, IBM
- Anthony Nadalin, IBM
- Nick Ragouzis, Individual
- Scott Cantor, Internet2
- Bob Morgan, Internet2
- Peter Davis, Neustar
- Jeff Hodges, Neustar
- Frederick Hirsch, Nokia
- Senthil Sengodan, Nokia
- Abbie Barbir, Nortel Networks
- Scott Kiester, Novell
- Cameron Morris, Novell
- Paul Madsen, NTT
- Steve Anderson, OpenNetwork
- Ari Kermaier, Oracle
- Vamsi Motukuru, Oracle
- Darren Platt, Ping Identity
- Prateek Mishra, Principal Identity
- Jim Lien, RSA Security
- John Linn, RSA Security
- Rob Philpott, RSA Security
- Dipak Chopra, SAP
- Jahan Moreh, Sigaba
- Bhavna Bhatnagar, Sun Microsystems
- Eve Maler, Sun Microsystems

- Ronald Monzillo, Sun Microsystems
- Emily Xu, Sun Microsystems
- Greg Whitehead, Trustgenix

The editors also would like to acknowledge the following former SSTC members for their contributions to this or 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
- Tim Alsop, CyberSafe Limited
- 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
- Charles Knouse, Oblix
- Simon Godik, Overxeer
- Charles Norwood, SAIC
- Evan Prodromou, Securant
- Robert Griffin, RSA Security (former editor)
- Sai Allarvarpu, Sun Microsystems
- Gary Ellison, Sun Microsystems
- Chris Ferris, 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

The editors also would like to gratefully acknowledge Jahan Moreh of Sigaba, who during his tenure on the SSTC was the primary editor of the errata working document and who made major substantive contributions to all of the errata materials.

Appendix C. 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.

Copyright © OASIS Open 2005. *All Rights Reserved.*

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.