



2 Metadata for the OASIS Security 3 Assertion Markup Language (SAML) 4 V2.0

5 **Committee Draft 02e, 11 November 2004**

6 **Document identifier:**

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

8 **Location:**

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

10 **Editors:**

11 Scott Cantor, Internet2
12 Jahan Moreh, Sigaba
13 Rob Philpott, RSA Security
14 Eve Maler, Sun Microsystems

15 **SAML V2.0 Contributors:**

16 Conor P. Cahill, AOL
17 Hal Lockhart, BEA Systems
18 Michael Beach, Boeing
19 Rick Randall, Booze, Allen, Hamilton
20 Tim Alsop, CyberSafe Limited
21 Nick Ragouzis, Enosis
22 John Hughes, Atos Origin
23 Paul Madsen, Entrust
24 Irving Reid, Hewlett-Packard
25 Paula Austel, IBM
26 Maryann Hondo, IBM
27 Michael McIntosh, IBM
28 Tony Nadalin, IBM
29 Scott Cantor, Internet2
30 RL 'Bob' Morgan, Internet2
31 Rebekah Metz, NASA
32 Prateek Mishra, Netegrity
33 Peter C Davis, Neustar
34 Frederick Hirsch, Nokia
35 John Kemp, Nokia
36 Charles Knouse, Oblix
37 Steve Anderson, OpenNetwork
38 John Linn, RSA Security
39 Rob Philpott, RSA Security
40 Jahan Moreh, Sigaba
41 Anne Anderson, Sun Microsystems
42 Jeff Hodges, Sun Microsystems
43 Eve Maler, Sun Microsystems

44 Ron Monzillo, Sun Microsystems
45 Greg Whitehead, Trustgenix

46 **Abstract:**

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

53 **Status:**

54 This is a **second Committee Draft** approved by the Security Services Technical Committee on
55 21 September 2004.

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

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

65 Table of Contents

66	1 Introduction.....	5
67	1.1 Notation.....	5
68	2 Metadata for SAML V2.0.....	6
69	2.1 Namespaces	6
70	2.2 Common Types.....	7
71	2.2.1 Simple Type entityIDType.....	7
72	2.2.2 Complex Type EndpointType.....	7
73	2.2.3 Complex Type IndexedEndpointType.....	8
74	2.2.4 Complex Type localizedNameType.....	8
75	2.2.5 Complex Type localizedURIType.....	9
76	2.3 Root Elements.....	9
77	2.3.1 Element <EntitiesDescriptor>.....	9
78	2.3.2 Element <EntityDescriptor>.....	10
79	2.3.2.1 Element <Organization>.....	12
80	2.3.2.2 Element <ContactPerson>.....	12
81	2.3.2.3 Element <AdditionalMetadataLocation>.....	14
82	2.4 Role Descriptor Elements.....	14
83	2.4.1 Element <RoleDescriptor>.....	14
84	2.4.1.1 Element <KeyDescriptor>.....	15
85	2.4.2 Complex Type SSODescriptorType.....	16
86	2.4.3 Element <IDPSSODescriptor>.....	17
87	2.4.4 Element <SPSSODescriptor>.....	18
88	2.4.4.1 Element <AttributeConsumingService>.....	19
89	2.4.4.2 Element <RequestedAttribute>.....	19
90	2.4.5 Element <AuthnAuthorityDescriptor>.....	20
91	2.4.6 Element <PDPDescriptor>.....	21
92	2.4.7 Element <AttributeAuthorityDescriptor>.....	21
93	2.5 Element <AffiliationDescriptor>.....	22
94	2.6 Examples.....	23
95	3 Signature Processing.....	27
96	3.1 XML Signature Profile.....	27
97	3.1.1 Signing Formats and Algorithms.....	27
98	3.1.2 References.....	27
99	3.1.3 Canonicalization Method.....	27
100	3.1.4 Transforms.....	28
101	3.1.5 KeyInfo.....	28
102	4 Metadata Publication and Resolution.....	29
103	4.1 Publication and Resolution via Well-Known Location.....	29
104	4.1.1 Publication.....	29
105	4.1.2 Resolution.....	29
106	4.2 Publishing and Resolution via DNS.....	29
107	4.2.1 Publication.....	30
108	4.2.1.1 First Well Known Rule.....	30
109	4.2.1.2 The Order Field.....	30
110	4.2.1.3 The Preference Field.....	30
111	4.2.1.4 The Flag Field.....	31
112	4.2.1.5 The Service Field.....	31

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

1 Introduction

132

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

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

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

1.1 Notation

143

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

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

148

149 `Example code listings appear like this.`

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

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

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

2 Metadata for SAML V2.0

153

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

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

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

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

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

2.1 Namespaces

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

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

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

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

```
178 <schema  
179   targetNamespace="urn:oasis:names:tc:SAML:2.0:metadata"  
180   xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"  
181   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
182   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"  
183   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
184   xmlns="http://www.w3.org/2001/XMLSchema"  
185   elementFormDefault="unqualified"  
186   attributeFormDefault="unqualified"  
187   blockDefault="substitution"  
188   version="2.0">  
189   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
190     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-  
191     schema.xsd"/>  
192   <import namespace="http://www.w3.org/2001/04/xmlenc#"  
193     schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-  
194     20021210/xenc-schema.xsd"/>  
195   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"  
196     schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>  
197   <import namespace="http://www.w3.org/XML/1998/namespace"  
198     schemaLocation="http://www.w3.org/2001/xml.xsd"/>  
199   <annotation>
```

```

201     <documentation>
202         Document identifier: sstc-saml-schema-metadata-2.0
203         Location: http://www.oasis-
204 open.org/committees/documents.php?wg_abbrev=security
205         Revision history:
206         V2.0 (August, 2004):
207             Schema for SAML metadata, first published in SAML 2.0.
208     </documentation>
209 </annotation>
210 ...
211 </schema>

```

212 2.2 Common Types

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

215 2.2.1 Simple Type `entityIDType`

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

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

```

222 <simpleType name="entityIDType">
223     <restriction base="anyURI">
224         <maxLength value="1024"/>
225     </restriction>
226 </simpleType>

```

227 2.2.2 Complex Type `EndpointType`

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

231 `Binding` [Required]

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

234 `Location` [Required]

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

237 `ResponseLocation` [Optional]

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

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

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

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

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

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

```
252 <complexType name="EndpointType">  
253   <sequence>  
254     <any namespace="##other" processContents="lax" minOccurs="0"  
255     maxOccurs="unbounded"/>  
256   </sequence>  
257   <attribute name="Binding" type="anyURI" use="required"/>  
258   <attribute name="Location" type="anyURI" use="required"/>  
259   <attribute name="ResponseLocation" type="anyURI" use="optional"/>  
260   <anyAttribute namespace="##other" processContents="lax"/>  
261 </complexType>
```

262 2.2.3 Complex Type IndexedEndpointType

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

266 **index** [Required]

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

269 **isDefault** [Optional]

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

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

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

```
277 <complexType name="IndexedEndpointType">  
278   <complexContent>  
279     <extension base="md:EndpointType">  
280       <attribute name="index" type="unsignedShort" use="required"/>  
281       <attribute name="isDefault" type="boolean" use="optional"/>  
282     </extension>  
283   </complexContent>  
284 </complexType>
```

285 2.2.4 Complex Type localizedNameType

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

```
288 <complexType name="localizedNameType">  
289   <simpleContent>  
290     <extension base="string">  
291       <attribute ref="xml:lang" use="required"/>  
292     </extension>  
293   </simpleContent>  
294 </complexType>
```


295 2.2.5 Complex Type localizedURIType

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

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

```
299 <complexType name="localizedURIType">  
300   <simpleContent>  
301     <extension base="anyURI">  
302       <attribute ref="xml:lang" use="required"/>  
303     </extension>  
304   </simpleContent>  
305 </complexType>
```

306 2.3 Root Elements

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

310 2.3.1 Element `<EntitiesDescriptor>`

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

314 ID [Optional]

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

316 validUntil [Optional]

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

319 cacheDuration [Optional]

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

322 Name [Optional]

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

324 `<ds:Signature>` [Optional]

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

327 `<Extensions>` [Optional]

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

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

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

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

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

```
338 <element name="EntitiesDescriptor" type="md:EntitiesDescriptorType"/>
339 <complexType name="EntitiesDescriptorType">
340   <sequence>
341     <element ref="ds:Signature" minOccurs="0"/>
342     <element ref="md:Extensions" minOccurs="0"/>
343     <choice minOccurs="1" maxOccurs="unbounded">
344       <element ref="md:EntityDescriptor"/>
345       <element ref="md:EntitiesDescriptor"/>
346     </choice>
347   </sequence>
348   <attribute name="validUntil" type="dateTime" use="optional"/>
349   <attribute name="cacheDuration" type="duration" use="optional"/>
350   <attribute name="ID" type="ID" use="optional"/>
351   <attribute name="Name" type="string" use="optional"/>
352 </complexType>
353 <element name="Extensions" type="md:ExtensionsType"/>
354 <complexType final="#all" name="ExtensionsType">
355   <sequence>
356     <any namespace="##other" processContents="lax"
357     maxOccurs="unbounded"/>
358   </sequence>
359 </complexType>
```

360 2.3.2 Element `<EntityDescriptor>`

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

- 365 • SSO Identity Provider
- 366 • SSO Service Provider
- 367 • Authentication Authority
- 368 • Attribute Authority
- 369 • Policy Decision Point
- 370 • Affiliation

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

372 `entityID` [Required]

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

375 `ID` [Optional]

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

377 `validUntil` [Optional]

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

380 `cacheDuration` [Optional]

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

383 <ds:Signature> [Optional]
 384 An XML signature that authenticates the containing element and its contents, as described in
 385 Section 3.

386 <Extensions> [Optional]
 387 This contains optional metadata extensions that are agreed upon between a metadata publisher
 388 and consumer. Extension elements MUST be namespace-qualified by a non-SAML-defined
 389 namespace.

390 <RoleDescriptor>, <IDPSSODescriptor>, <SPSSODescriptor>,
 391 <AuthnAuthorityDescriptor>, <AttributeAuthorityDescriptor>, <PDPDescriptor> [One
 392 or More]
 393 **OR**

394 <AffiliationDescriptor> [Required]
 395 The primary content of the element is either a sequence of one or more role descriptor elements,
 396 or a specialized descriptor that defines an affiliation.

397 <Organization> [Optional]
 398 Optional element identifying the organization responsible for the SAML entity described by the
 399 element.

400 <ContactPerson> [Zero or More]
 401 Optional sequence of elements identifying various kinds of contact personnel.

402 <AdditionalMetadataLocation> [Zero or More]
 403 Optional sequence of namespace-qualified locations where additional metadata exists for the
 404 SAML entity. This may include metadata in alternate formats or describing adherence to other
 405 non-SAML specifications.

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

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

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

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

```

417 <element name="EntityDescriptor" type="md:EntityDescriptorType"/>
418 <complexType name="EntityDescriptorType">
419   <sequence>
420     <element ref="ds:Signature" minOccurs="0"/>
421     <element ref="md:Extensions" minOccurs="0"/>
422     <choice>
423       <choice maxOccurs="unbounded">
424         <element ref="md:RoleDescriptor"/>
425         <element ref="md:IDPSSODescriptor"/>
426         <element ref="md:SPSSODescriptor"/>
427         <element ref="md:AuthnAuthorityDescriptor"/>
428         <element ref="md:AttributeAuthorityDescriptor"/>
429         <element ref="md:PDPDescriptor"/>
430       </choice>
431     <element ref="md:AffiliationDescriptor"/>
  
```

```

432     </choice>
433     <element ref="md:Organization" minOccurs="0"/>
434     <element ref="md>ContactPerson" minOccurs="0"
435 maxOccurs="unbounded"/>
436     <element ref="md:AdditionalMetadataLocation" minOccurs="0"
437 maxOccurs="unbounded"/>
438 </sequence>
439 <attribute name="entityID" type="md:entityIDType" use="required"/>
440 <attribute name="validUntil" type="dateTime" use="optional"/>
441 <attribute name="cacheDuration" type="duration" use="optional"/>
442 <attribute name="ID" type="ID" use="optional"/>
443 <anyAttribute namespace="##other" processContents="lax"/>
444 </complexType>

```

445 2.3.2.1 Element <Organization>

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

450 <Extensions> [Optional]

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

454 <OrganizationName> [One or More]

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

456 <OrganizationDisplayName> [One or More]

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

458 <OrganizationURL> [One or More]

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

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

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

```

465 <element name="Organization" type="md:OrganizationType"/>
466 <complexType name="OrganizationType">
467   <sequence>
468     <element ref="md:Extensions" minOccurs="0"/>
469     <element ref="md:OrganizationName" maxOccurs="unbounded"/>
470     <element ref="md:OrganizationDisplayName" maxOccurs="unbounded"/>
471     <element ref="md:OrganizationURL" maxOccurs="unbounded"/>
472   </sequence>
473   <anyAttribute namespace="##other" processContents="lax"/>
474 </complexType>
475 <element name="OrganizationName" type="md:localizedNameType"/>
476 <element name="OrganizationDisplayName" type="md:localizedNameType"/>
477 <element name="OrganizationURL" type="md:localizedURIType"/>

```

478 2.3.2.2 Element <ContactPerson>

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

482 consists of the following elements and attributes:

483 `contactType` [Required]

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

486 `<Extensions>` [Optional]

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

490 `<Company>` [Optional]

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

492 `<GivenName>` [Optional]

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

494 `<SurName>` [Optional]

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

496 `<EmailAddress>` [Zero or More]

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

499 `<TelephoneNumber>` [Zero or More]

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

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

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

```
504 <element name="ContactPerson" type="md:ContactType"/>
505 <complexType name="ContactType">
506   <sequence>
507     <element ref="md:Extensions" minOccurs="0"/>
508     <element ref="md:Company" minOccurs="0"/>
509     <element ref="md:GivenName" minOccurs="0"/>
510     <element ref="md:SurName" minOccurs="0"/>
511     <element ref="md:EmailAddress" minOccurs="0"
512 maxOccurs="unbounded"/>
513     <element ref="md:TelephoneNumber" minOccurs="0"
514 maxOccurs="unbounded"/>
515   </sequence>
516   <attribute name="contactType" type="md:ContactTypeType"
517 use="required"/>
518   <anyAttribute namespace="##other" processContents="lax"/>
519 </complexType>
520 <element name="Company" type="string"/>
521 <element name="GivenName" type="string"/>
522 <element name="SurName" type="string"/>
523 <element name="EmailAddress" type="anyURI"/>
524 <element name="TelephoneNumber" type="string"/>
525 <simpleType name="ContactTypeType">
526   <restriction base="string">
527     <enumeration value="technical"/>
528     <enumeration value="support"/>
529     <enumeration value="administrative"/>
530     <enumeration value="billing"/>
531     <enumeration value="other"/>
532   </restriction>
```

533 </simpleType>

534 2.3.2.3 Element <AdditionalMetadataLocation>

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

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

```
542 <element name="AdditionalMetadataLocation"  
543 type="md:AdditionalMetadataLocationType"/>  
544 <complexType name="AdditionalMetadataLocationType">  
545   <simpleContent>  
546     <extension base="anyURI">  
547       <attribute name="namespace" type="anyURI" use="required"/>  
548     </extension>  
549   </simpleContent>  
550 </complexType>
```

551 2.4 Role Descriptor Elements

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

555 2.4.1 Element <RoleDescriptor>

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

560 ID [Optional]

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

562 validUntil [Optional]

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

565 cacheDuration [Optional]

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

568 protocolSupportEnumeration [Required]

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

574 errorURL [Optional]

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

577 <ds:Signature> [Optional]
578 An XML signature that authenticates the containing element and its contents, as described in
579 Section 3.

580 <Extensions> [Optional]
581 This contains optional metadata extensions that are agreed upon between a metadata publisher
582 and consumer. Extension elements MUST be namespace-qualified by a non-SAML-defined
583 namespace.

584 <KeyDescriptor> [Zero or More]
585 Optional sequence of elements that provides information about the cryptographic keys that the
586 entity uses when acting in this role.

587 <Organization> [Optional]
588 Optional element specifies the organization associated with this role. Identical to the element used
589 within the <EntityDescriptor> element.

590 <ContactPerson> [Zero or More]
591 Optional sequence of elements specifying contacts associated with this role. Identical to the
592 element used within the <EntityDescriptor> element.

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

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

```

596 <element name="RoleDescriptor" type="md:RoleDescriptorType"/>
597 <complexType name="RoleDescriptorType" abstract="true">
598   <sequence>
599     <element ref="ds:Signature" minOccurs="0"/>
600     <element ref="md:Extensions" minOccurs="0"/>
601     <element ref="md:KeyDescriptor" minOccurs="0"
602     maxOccurs="unbounded"/>
603     <element ref="md:Organization" minOccurs="0"/>
604     <element ref="md:ContactPerson" minOccurs="0"
605     maxOccurs="unbounded"/>
606   </sequence>
607   <attribute name="ID" type="ID" use="optional"/>
608   <attribute name="validUntil" type="dateTime" use="optional"/>
609   <attribute name="cacheDuration" type="duration" use="optional"/>
610   <attribute name="protocolSupportEnumeration" type="md:anyURIListType"
611   use="required"/>
612   <attribute name="errorURL" type="anyURI" use="optional"/>
613   <anyAttribute namespace="##other" processContents="lax"/>
614 </complexType>
615 <simpleType name="anyURIListType">
616   <list itemType="anyURI"/>
617 </simpleType>

```

618 2.4.1.1 Element <KeyDescriptor>

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

622 use [Optional]
623 Optional attribute specifying the purpose of the key being described. Values are drawn from the
624 **KeyTypes** enumeration, and consist of the values **encryption** and **signing**.

625 <ds:KeyInfo> [Required]
 626 Optional element that directly or indirectly identifies a key. See [XMLSig] for additional details on
 627 the use of this element.

628 <EncryptionMethod> [Zero or More]
 629 Optional element specifying an algorithm and algorithm-specific settings supported by the entity.
 630 The exact content varies based on the algorithm supported. See [XMLEnc] for the definition of this
 631 element's **xenc:EncryptionMethodType** complex type.

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

```

634 <element name="KeyDescriptor" type="md:KeyDescriptorType"/>
635 <complexType name="KeyDescriptorType">
636   <sequence>
637     <element ref="ds:KeyInfo"/>
638     <element ref="md:EncryptionMethod minOccurs="0"
639 maxOccurs="unbounded"/>
640   </sequence>
641   <attribute name="use" type="md:KeyTypes" use="optional"/>
642 </complexType>
643 <simpleType name="KeyTypes">
644   <restriction base="string">
645     <enumeration value="encryption"/>
646     <enumeration value="signing"/>
647   </restriction>
648 </simpleType>
649 <element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>
  
```

650 2.4.2 Complex Type SSODescriptorType

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

655 <ArtifactResolutionService> [Zero or More]
 656 Zero or more elements of type **IndexedEndpointType** that describe indexed endpoints that
 657 support the Artifact Resolution profile defined in [SAMLProf]. The *ResponseLocation* attribute
 658 MUST be omitted.

659 <SingleLogoutService> [Zero or More]
 660 Zero or more elements of type **EndpointType** that describe endpoints that support the Single
 661 Logout profiles defined in [SAMLProf].

662 <ManageNameIDService> [Zero or More]
 663 Zero or more elements of type **EndpointType** that describe endpoints that support the Name
 664 Identifier Management profiles defined in [SAMLProf].

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

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

```

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



```

674         <element ref="md:ArtifactResolutionService" minOccurs="0"
675 maxOccurs="unbounded"/>
676         <element ref="md:SingleLogoutService" minOccurs="0"
677 maxOccurs="unbounded"/>
678         <element ref="md:ManageNameIDService" minOccurs="0"
679 maxOccurs="unbounded"/>
680         <element ref="md:NameIDFormat" minOccurs="0"
681 maxOccurs="unbounded"/>
682     </sequence>
683 </extension>
684 </complexContent>
685 </complexType>
686 <element name="ArtifactResolutionService" type="md:IndexedEndpointType"/>
687 <element name="SingleLogoutService" type="md:EndpointType"/>
688 <element name="ManageNameIDService" type="md:EndpointType"/>
689 <element name="NameIDFormat" type="anyURI"/>

```

690 2.4.3 Element <IDPSSODescriptor>

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

694 WantAuthnRequestsSigned [Optional]

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

697 <SingleSignOnService> [One or More]

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

701 <NameIDMappingService> [Zero or More]

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

705 <AssertionIDRequestService> [Zero or More]

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

709 <AttributeProfile> [Zero or More]

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

712 <saml:Attribute> [Zero or More]

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

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

```

719 <element name="IDPSSODescriptor" type="md:IDPSSODescriptorType"/>
720 <complexType name="IDPSSODescriptorType">
721     <complexContent>
722         <extension base="md:SSODescriptorType">

```

```

723         <sequence>
724             <element ref="md:SingleSignOnService"
725 maxOccurs="unbounded"/>
726             <element ref="md:NameIDMappingService" minOccurs="0"
727 maxOccurs="unbounded"/>
728             <element ref="md:AssertionIDRequestService" minOccurs="0"
729 maxOccurs="unbounded"/>
730             <element ref="md:AttributeProfile" minOccurs="0"
731 maxOccurs="unbounded"/>
732             <element ref="saml:Attribute" minOccurs="0"
733 maxOccurs="unbounded"/>
734         </sequence>
735         <attribute name="WantAuthnRequestsSigned" type="boolean"
736 use="optional"/>
737     </extension>
738 </complexContent>
739 </complexType>
740 <element name="SingleSignOnService" type="md:EndpointType"/>
741 <element name="NameIDMappingService" type="md:EndpointType"/>
742 <element name="AssertionIDRequestService" type="md:EndpointType"/>
743 <element name="AttributeProfile" type="anyURI"/>

```

744 2.4.4 Element <SPSSODescriptor>

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

748 AuthnRequestsSigned [Optional]

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

751 WantAssertionsSigned [Optional]

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

756 <AssertionConsumerService> [One or More]

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

760 <AttributeConsumingService> [Zero or More]

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

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

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

```

769 <element name="SPSSODescriptor" type="md:SPSSODescriptorType"/>
770 <complexType name="SPSSODescriptorType">
771     <complexContent>
772         <extension base="md:SSODescriptorType">
773             <sequence>

```

```

774         <element ref="md:AssertionConsumerService"
775 maxOccurs="unbounded"/>
776         <element ref="md:AttributeConsumingService" minOccurs="0"
777 maxOccurs="unbounded"/>
778     </sequence>
779     <attribute name="AuthnRequestsSigned" type="boolean"
780 use="optional"/>
781     <attribute name="WantAssertionsSigned" type="boolean"
782 use="optional"/>
783 </extension>
784 </complexContent>
785 </complexType>
786 <element name="AssertionConsumerService"
787 type="md:IndexedEndpointType"/>

```

788 2.4.4.1 Element <AttributeConsumingService>

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

792 index [Required]

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

795 isDefault [Optional]

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

798 <ServiceName> [One or More]

799 One or more language-qualified names for the service.

800 <ServiceDescription> [Zero or More]

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

802 <RequestedAttribute> [One or More]

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

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

```

806 <element name="AttributeConsumingService"
807 type="md:AttributeConsumingServiceType"/>
808 <complexType name="AttributeConsumingServiceType">
809     <sequence>
810         <element ref="md:ServiceName" maxOccurs="unbounded"/>
811         <element ref="md:ServiceDescription" minOccurs="0"
812 maxOccurs="unbounded"/>
813         <element ref="md:RequestedAttribute" maxOccurs="unbounded"/>
814     </sequence>
815     <attribute name="index" type="unsignedShort" use="required"/>
816     <attribute name="isDefault" type="boolean" use="optional"/>
817 </complexType>
818 <element name="ServiceName" type="md:localizedNameType"/>
819 <element name="ServiceDescription" type="md:localizedNameType"/>

```

820 2.4.4.2 Element <RequestedAttribute>

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

823 **saml:AttributeType** with the following attribute:

824 `isRequired` [Optional]

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

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

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

```
831 <element name="RequestedAttribute" type="md:RequestedAttributeType"/>
832 <complexType name="RequestedAttributeType">
833   <complexContent>
834     <extension base="saml:AttributeType">
835       <attribute name="isRequired" type="boolean" use="optional"/>
836     </extension>
837   </complexContent>
838 </complexType>
```

839 2.4.5 Element `<AuthnAuthorityDescriptor>`

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

843 `<AuthnQueryService>` [One or More]

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

847 `<AssertionIDRequestService>` [Zero or More]

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

851 `<NameIDFormat>` [Zero or More]

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

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

```
856 <element name="AuthnAuthorityDescriptor"
857 type="md:AuthnAuthorityDescriptorType"/>
858 <complexType name="AuthnAuthorityDescriptorType">
859   <complexContent>
860     <extension base="md:RoleDescriptorType">
861       <sequence>
862         <element ref="md:AuthnQueryService" maxOccurs="unbounded"/>
863         <element ref="md:AssertionIDRequestService" minOccurs="0"
864 maxOccurs="unbounded"/>
865         <element ref="md:NameIDFormat" minOccurs="0"
866 maxOccurs="unbounded"/>
867       </sequence>
868     </extension>
869   </complexContent>
870 </complexType>
871 <element name="AuthnQueryService" type="md:EndpointType"/>
```

872 2.4.6 Element <PDPDescriptor>

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

876 <AuthzService> [One or More]

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

880 <AssertionIDRequestService> [Zero or More]

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

884 <NameIDFormat> [Zero or More]

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

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

```
889 <element name="PDPDescriptor" type="md:PDPDescriptorType"/>  
890 <complexType name="PDPDescriptorType">  
891   <complexContent>  
892     <extension base="md:RoleDescriptorType">  
893       <sequence>  
894         <element ref="md:AuthzService" maxOccurs="unbounded"/>  
895         <element ref="md:AssertionIDRequestService" minOccurs="0"  
896 maxOccurs="unbounded"/>  
897         <element ref="md:NameIDFormat" minOccurs="0"  
898 maxOccurs="unbounded"/>  
899       </sequence>  
900     </extension>  
901   </complexContent>  
902 </complexType>  
903 <element name="AuthzService" type="md:EndpointType"/>
```

904 2.4.7 Element <AttributeAuthorityDescriptor>

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

909 <AttributeService> [One or More]

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

913 <AssertionIDRequestService> [Zero or More]

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

917 <NameIDFormat> [Zero or More]

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

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

920 <AttributeProfile> [Zero or More]

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

923 <saml:Attribute> [Zero or More]

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

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

```
929 <element name="AttributeAuthorityDescriptor"  
930 type="md:AttributeAuthorityDescriptorType"/>  
931 <complexType name="AttributeAuthorityDescriptorType">  
932 <complexContent>  
933 <extension base="md:RoleDescriptorType">  
934 <sequence>  
935 <element ref="md:AttributeService" maxOccurs="unbounded"/>  
936 <element ref="md:AssertionIDRequestService" minOccurs="0"  
937 maxOccurs="unbounded"/>  
938 <element ref="md:NameIDFormat" minOccurs="0"  
939 maxOccurs="unbounded"/>  
940 <element ref="md:AttributeProfile" minOccurs="0"  
941 maxOccurs="unbounded"/>  
942 <element ref="saml:Attribute" minOccurs="0"  
943 maxOccurs="unbounded"/>  
944 </sequence>  
945 </extension>  
946 </complexContent>  
947 </complexType>  
948 <element name="AttributeService" type="md:EndpointType"/>
```

949 2.5 Element <AffiliationDescriptor>

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

956 affiliationOwnerID [Required]

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

960 ID [Optional]

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

962 validUntil [Optional]

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

965 cacheDuration [Optional]

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

968 <ds:Signature> [Optional]
 969 An XML signature that authenticates the containing element and its contents, as described in
 970 Section 3.

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

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

978 <KeyDescriptor> [Zero or More]
 979 Optional sequence of elements that provides information about the cryptographic keys that the
 980 affiliation uses as a whole, as distinct from keys used by individual members of the affiliation,
 981 which are published in the metadata for those entities.

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

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

```

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

```

1003 2.6 Examples

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

```

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

```

```

1022     <SingleLogoutService
1023         Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1024         Location="https://IdentityProvider.com/SAML/SLO/SOAP"/>
1025     <SingleLogoutService
1026         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1027         Location="https://IdentityProvider.com/SAML/SLO/Browser"
1028         ResponseLocation="https://IdentityProvider.com/SAML/SLO/Response"/>
1029     <NameIDFormat>
1030         urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1031     </NameIDFormat>
1032     <NameIDFormat>
1033         urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1034     </NameIDFormat>
1035     <NameIDFormat>
1036         urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1037     </NameIDFormat>
1038     <SingleSignOnService
1039         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1040         Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1041     <SingleSignOnService
1042         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1043         Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1044     <saml:Attribute
1045         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1046         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1047         FriendlyName="eduPersonPrincipalName">
1048     </saml:Attribute>
1049     <saml:Attribute
1050         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1051         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1052         FriendlyName="eduPersonAffiliation">
1053         <saml:AttributeValue>member</saml:AttributeValue>
1054         <saml:AttributeValue>student</saml:AttributeValue>
1055         <saml:AttributeValue>faculty</saml:AttributeValue>
1056         <saml:AttributeValue>employee</saml:AttributeValue>
1057         <saml:AttributeValue>staff</saml:AttributeValue>
1058     </saml:Attribute>
1059 </IDPSSODescriptor>
1060 <AttributeAuthorityDescriptor
1061     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1062     <KeyDescriptor use="signing">
1063         <ds:KeyInfo>
1064             <ds:KeyName>IdentityProvider.com AA Key</ds:KeyName>
1065         </ds:KeyInfo>
1066     </KeyDescriptor>
1067     <AttributeService
1068         Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1069         Location="https://IdentityProvider.com/SAML/AA/SOAP"/>
1070     <AssertionIDRequestService
1071         Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
1072         Location="https://IdentityProvider.com/SAML/AA/URI"/>
1073     <NameIDFormat>
1074         urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1075     </NameIDFormat>
1076     <NameIDFormat>
1077         urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1078     </NameIDFormat>
1079     <NameIDFormat>
1080         urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1081     </NameIDFormat>
1082     <saml:Attribute
1083         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1084         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1085         FriendlyName="eduPersonPrincipalName">
1086     </saml:Attribute>
1087     <saml:Attribute
1088         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"

```



```

1089     Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1090     FriendlyName="eduPersonAffiliation">
1091         <saml:AttributeValue>member</saml:AttributeValue>
1092         <saml:AttributeValue>student</saml:AttributeValue>
1093         <saml:AttributeValue>faculty</saml:AttributeValue>
1094         <saml:AttributeValue>employee</saml:AttributeValue>
1095         <saml:AttributeValue>staff</saml:AttributeValue>
1096     </saml:Attribute>
1097 </AttributeAuthorityDescriptor>
1098 <Organization>
1099     <OrganizationName xml:lang="en">Identity Providers R US</OrganizationName>
1100     <OrganizationDisplayName xml:lang="en">
1101         Identity Providers R US, a Division of Leroxst Corp.
1102     </OrganizationDisplayName>
1103     <OrganizationURL xml:lang="en">https://IdentityProvider.com</OrganizationURL>
1104 </Organization>
1105 </EntityDescriptor>
1106

```

1107 The following is an example of metadata for a SAML system entity acting as a service provider. A
1108 signature is shown as a placeholder, without the actual content. For illustrative purposes, the service is
1109 one that does not require users to uniquely identify themselves, but rather authorizes access on the basis
1110 of a role-like attribute.
1111

```

1112 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1113     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1114     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1115     entityID="https://ServiceProvider.com/SAML">
1116     <ds:Signature>...</ds:Signature>
1117     <SPSSODescriptor AuthnRequestsSigned="true"
1118         protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1119         <KeyDescriptor use="signing">
1120             <ds:KeyInfo>
1121                 <ds:KeyName>ServiceProvider.com SSO Key</ds:KeyName>
1122             </ds:KeyInfo>
1123         </KeyDescriptor>
1124         <KeyDescriptor use="encryption">
1125             <ds:KeyInfo>
1126                 <ds:KeyName>ServiceProvider.com Encrypt Key</ds:KeyName>
1127             </ds:KeyInfo>
1128             <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
1129         </KeyDescriptor>
1130         <SingleLogoutService
1131             Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1132             Location="https://ServiceProvider.com/SAML/SLO/SOAP"/>
1133         <SingleLogoutService
1134             Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1135             Location="https://ServiceProvider.com/SAML/SLO/Browser"
1136             ResponseLocation="https://ServiceProvider.com/SAML/SLO/Response"/>
1137         <NameIDFormat>
1138             urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1139         </NameIDFormat>
1140         <AssertionConsumerService isDefault="true" index="0"
1141             Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
1142             Location="https://ServiceProvider.com/SAML/SSO/Artifact"/>
1143         <AssertionConsumerService index="1"
1144             Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1145             Location="https://ServiceProvider.com/SAML/SSO/POST"/>
1146         <AttributeConsumingService index="0">
1147             <ServiceName xml:lang="en">Academic Journals R US</ServiceName>
1148             <RequestedAttribute
1149                 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1150                 Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7"
1151                 FriendlyName="eduPersonEntitlement">
1152                 <saml:AttributeValue>
1153                     https://ServiceProvider.com/entitlements/123456789
1154                 </saml:AttributeValue>
1155             </RequestedAttribute>

```

```
1156     </AttributeConsumingService>
1157 </SPSSODescriptor>
1158 <Organization>
1159     <OrganizationName xml:lang="en">Academic Journals R US</OrganizationName>
1160     <OrganizationDisplayName xml:lang="en">
1161         Academic Journals R US, a Division of Dirk Corp.
1162     </OrganizationDisplayName>
1163     <OrganizationURL xml:lang="en">https://ServiceProvider.com</OrganizationURL>
1164 </Organization>
1165 </EntityDescriptor>
```

1166 3 Signature Processing

1167 Various elements in a metadata instance can be digitally signed (as indicated by the element's inclusion of
1168 a `<ds:Signature>` element), with the following benefits:

- 1169 • Metadata integrity
- 1170 • Authentication of the metadata by a trusted signer

1171 A digital signature is not always required, for example if the relying party obtains the information directly
1172 from the publishing entity directly (with no intermediaries) through a secure channel, with the entity having
1173 authenticated to the relying party by some means other than a digital signature.

1174 Many different techniques are available for "direct" authentication and secure channel establishment
1175 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,
1176 the applicable security requirements depend on the communicating applications.

1177 Additionally, elements can inherit signatures on enclosing parent elements that are themselves signed.

1178 In the absence of such context, it is RECOMMENDED that at least the root element of a metadata
1179 instance be signed.

1180 3.1 XML Signature Profile

1181 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
1182 and many choices. This section details the constraints on these facilities so that metadata processors do
1183 not have to deal with the full generality of XML Signature processing. This usage makes specific use of
1184 the **xs:ID**-typed attributes optionally present on the elements to which signatures can apply. These
1185 attributes are collectively referred to in this section as the identifier attributes.

1186 3.1.1 Signing Formats and Algorithms

1187 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
1188 detached.

1189 SAML metadata MUST use enveloped signatures when signing the elements defined in this specification.
1190 SAML processors SHOULD support the use of RSA signing and verification for public key operations in
1191 accordance with the algorithm identified by <http://www.w3.org/2000/09/xmlsig#rsa-sha1>.

1192 3.1.2 References

1193 Signed metadata elements MUST supply a value for the identifier attribute on the signed element. The
1194 element may or may not be the root element of the actual XML document containing the signed metadata
1195 element.

1196 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute
1197 value of the metadata element being signed. For example, if the identifier attribute value is "foo", then the
1198 URI attribute in the `<ds:Reference>` element MUST be "#foo".

1199 As a consequence, a metadata element's signature MUST apply to the content of the signed element and
1200 any child elements it contains.

1201 3.1.3 Canonicalization Method

1202 SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the
1203 `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a `<ds:Transform>`
1204 algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML metadata

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

1206 **3.1.4 Transforms**

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

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

1216 **3.1.5 KeyInfo**

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

1220 4 Metadata Publication and Resolution

1221 Two mechanisms are provided for an entity to publish (and for a consumer to resolve the location of)
1222 metadata documents: via a "well-known-location" by directly dereferencing the entity's unique identifier (a
1223 URI variously referred to as an *entityID* or *providerID*), or indirectly by publishing the location of metadata
1224 in the DNS. Other out-of-band mechanisms are of course also permitted. A consumer that supports both
1225 approaches defined in this document MUST attempt resolution via DNS before using the "well-known-
1226 location" mechanism.

1227 When retrieval requires network transport of the document, the transport SHOULD be protected with
1228 mechanisms providing server authentication and integrity protection. For example, HTTP-based resolution
1229 SHOULD be protected with *TLS/SSL* [RFC2246] as amended by [RFC3546].

1230 Various mechanisms are described in this section to aid in establishing trust in the accuracy and
1231 legitimacy of metadata, including use of XML signatures, SSL/TLS server authentication, and DNS
1232 signatures. Regardless of the mechanism(s) used, relying parties SHOULD have some means by which to
1233 establish trust in metadata information before relying on it.

1234 4.1 Publication and Resolution via Well-Known Location

1235 The following sections describe publication and resolution of metadata by means of a well-known location.

1236 4.1.1 Publication

1237 Entities MAY publish their metadata documents at a well known location by placing the document at the
1238 location denoted by its unique identifier, which MUST be in the form of a URL (rather than a URN). See
1239 Section 8.3.6 of [SAMLCore] for more information about such identifiers. It is STRONGLY
1240 RECOMMENDED that `https` URLs be used for this purpose. An indirection mechanism supported by the
1241 URL scheme (such as an HTTP 1.1 302 redirect) MAY be used if the document is not placed directly at
1242 the location. If the publishing protocol permits MIME-based identification of content types, the content type
1243 of the metadata instance MUST be `application/samlmetadata+xml`.

1244 The XML document provided at the well-known location MUST describe the metadata only for the entity
1245 represented by the unique identifier (that is, the root element MUST be an `<EntityDescriptor>` with
1246 an `entityID` matching the location). If other entities need to be described, the
1247 `<AdditionalMetaLocation>` element MUST be used. Thus the `<EntitiesDescriptor>` element
1248 MUST NOT be used in documents published using this mechanism, since a group of entities are not
1249 defined by such an identifier.

1250 4.1.2 Resolution

1251 If an entity's unique identifier is a URL, metadata consumers MAY attempt to resolve an entity's unique
1252 identifier directly, in a scheme-specific manner, by dereferencing the identifier.

1253 4.2 Publishing and Resolution via DNS

1254 To improve the accessibility of metadata documents and provide additional indirection between an entity's
1255 unique identifier and the location of metadata, entities MAY publish their metadata document locations in a
1256 zone of their corresponding DNS [RFC1034]. The entity's unique identifier (a URI) is used as the input to
1257 the process. Since URIs are flexible identifiers, location publication methods and the resolution process
1258 are determined by the URI's scheme and fully-qualified name. URI locations for metadata are

1259 subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in [\[RFC2915\]](#)
1260 and [\[RFC3403\]](#).

1261 It is RECOMMENDED that entities publish their resource records in signed zone files using [\[RFC2535\]](#)
1262 such that relying parties may establish the validity of the published location and authority of the zone, and
1263 integrity of the DNS response. If DNS zone signatures are present, relying parties MUST properly validate
1264 the signature.

1265 **4.2.1 Publication**

1266 This specification makes use of the NAPTR resource record described in [\[RFC2915\]](#) and [\[RFC3403\]](#).
1267 Familiarity with these documents is encouraged.

1268 Dynamic Delegation Discovery System (DDDS) [\[RFC3401\]](#) is a general purpose system for the retrieval of
1269 information based on an application-specific input string and the application of well known rules to
1270 transform that string until a terminal condition is reached requiring a look-up into an application-specific
1271 defined database or resolution of a URL based on the rules defined by the application. DDDS defines a
1272 specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS
1273 necessary to apply DDDS rules.

1274 Entities MAY publish separate URLs when multiple metadata documents need to be distributed, or when
1275 different metadata documents are required due to multiple trust relationships that require separate keying
1276 material, or when service interfaces require separate metadata declarations. This may be accomplished
1277 through the use of the optional `<AdditionalMetaLocation>` element, or through the regexp facility and
1278 multiple service definition fields in the NAPTR resource record itself.

1279 If the publishing protocol permits MIME-based identification of content types, the content type of the
1280 metadata instance MUST be `application/samlmetadata+xml`.

1281 If the entity's unique identifier is a URN, publication of the corresponding metadata location proceeds as
1282 specified in [\[RFC3404\]](#). Otherwise, the resolution of the metadata location proceeds as specified below.

1283 The following is the application-specific profile of DDDS for SAML metadata resolution.

1284 **4.2.1.1 First Well Known Rule**

1285 The "first well-known-rule" for processing SAML metadata resolution is to parse the entity's unique
1286 identifier and extract the fully-qualified domain name (subexpression 3) as described in Section "[Parsing](#)
1287 [the providerID](#)".

1288 **4.2.1.2 The Order Field**

1289 The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY
1290 provide multiple NAPTR resource records which MUST be processed by the resolver application in the
1291 order indicated by this field.

1292 **4.2.1.3 The Preference Field**

1293 For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving
1294 application. The resolving application MAY ignore this order, in cases where the service field value does
1295 not meet the resolver's requirements (e.g.: the resource record returns a protocol the application does not
1296 support).

1297 4.2.1.4 The Flag Field

1298 SAML metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying
1299 additional resource records are to be processed). The "U" flag indicates that the output of the rule is a
1300 URI.

1301 4.2.1.5 The Service Field

1302 The SAML-specific service field, as described in the following BNF, declares the modes by which instance
1303 document(s) shall be made available:

```
1304 servicefield = 1("PID2U" / "NID2U") "+" proto [*(":" class) *(":" servicetype)]
1305 proto = 1("https" / "uddi")
1306 class = 1[ "entity" / "entitygroup" ]
1307 servicetype = 1(si / "spssso" / "idpssso" / "authn" / "authnauth" / "pdp" / "attrauth" /
1308 "attrcons" / alphanum )
1309 si = "si" [ ":" alphanum ] [ ":" endpoint ]
1310 alphanum = 1*32 (ALPHA / DIGIT)
```

1311 where:

- 1312 • servicefield PID2U resolves an entity's unique identifier to metadata URL.
- 1313 • servicefield NID2U resolves a principal's <NameIdentifier> into a metadata URL.
- 1314 • proto describes the retrieval protocol (`https` or `uddi`). In the case of UDDI, the URL will be an
1315 http(s) URL referencing a WSDL document.
- 1316 • class identifies whether the referenced metadata document describes a single entity, or multiple.
1317 In the latter case, the referenced document MUST contain the entity defined by the original unique
1318 identifier as a member of a group of entities within the document itself such as an
1319 <AffiliationDescriptor> or <EntitiesDescriptor>.
- 1320 • servicetype allows an entity to publish metadata for distinct roles and services as separate
1321 documents. Resolvers who encounter multiple servicetype declarations will dereference the
1322 appropriate URI, depending on which service is required for an operation (e.g.: an entity operating
1323 both as an identity provider and a service provider can publish metadata for each role at different
1324 locations). The `authn` service type represents a <SingleSignOnService> endpoint.
- 1325 • si (with optional endpoint component) allows the publisher to either directly publish the metadata
1326 for a service instance, or by articulating a SOAP endpoint (using `endpoint`).

1327 For example:

- 1328 • PID2U+https:entity - represents the entity's complete metadata document available via the
1329 https protocol
- 1330 • PID2U+uddi:entity:si:foo - represents the WSDL document location that describes a service
1331 instance "foo"
- 1332 • PID2U+https:entitygroup:idpssso - represents the metadata for a group of entities acting as
1333 SSO identity providers, of which the original entity is a member.
- 1334 • NID2U+https:idp - represents the metadata for the SSO identity provider of a principal

1335 4.2.1.6 The Regex and Replacement Fields

1336 The expected output after processing the input string through the regex MUST be a valid `https` URL or
1337 UDDI node (WSDL document) address.

1338 4.2.2 NAPTR Examples

1339 4.2.2.1 Entity Metadata NAPTR Examples

1340 Entities publish metadata URLs in the following manner:

```
1341 $ORIGIN provider.biz
1342
1343 ;; order pref f service regexp or replacement
1344
1345 IN NAPTR 100 10 "U" PID2U+https:entity
1346     "!^.*$!https://host.provider.biz/some/directory/trust.xml!" ""
1347 IN NAPTR 110 10 "U" PID2U+https: entity:trust
1348     "!^.*$!https://foo.provider.biz:1443/mdtrust.xml!" ""
1349 IN NAPTR 125 10 "U" PID2U+https:"
1350 IN NAPTR 110 10 "U" PID2U+uddi:entity
1351     "!^.*$!https://this.uddi.node.provider.biz/libmd.wsdl" ""
```

1352 4.2.2.2 Name Identifier Examples

1353 A principal's employer `example.int` operates an identity provider which may be used by an office supply
1354 company to authenticate authorized buyers. The supplier takes a users' email address
1355 `buyer@example.int` as input to the resolution process, and parses the email address to extract the
1356 FQDN (`example.int`). The employer publishes the following NAPTR record in the `example.int` DNS:

```
1357 $ORIGIN example.int
1358
1359 IN NAPTR 100 10 "U" NID2U+https:authn
1360     "!^([\^@]+)@(.*)$!https://serv.example.int:8000/cgi-bin/getmd?\1!" ""
1361 IN NAPTR 100 10 "U" NID2U+https:idp
1362     "!^([\^@]+)@(.*)$!https://auth.example.int/app/auth?\1" ""
```

1363 4.2.3 Resolution

1364 When resolving metadata for an entity via the DNS, the unique identifier of the entity is used as the initial
1365 input into the resolution process, rather than as an actual location Proceed as follows:

- 1366 • If the unique identifier is a URN, proceed with the resolution steps as defined in [\[RFC3404\]](#).
- 1367 • Otherwise, parse the identifier to obtain the fully-qualified domain name.
- 1368 • Query the DNS for NAPTR resource records of the domain iteratively until a terminal resource
1369 record is returned.
- 1370 • Identify which resource record to use based on the service fields, then order fields, then preference
1371 fields of the result set.
- 1372 • Obtain the document(s) at the provided location(s) as required by the application.

1373 4.2.3.1 Parsing the Unique Identifier

1374 To initiate the resolution of the location of the metadata information, it will be necessary in some cases to
1375 decompose the entity's unique identifier (expressed as a URI) into one or more atomic elements.

1376 The following regular expression should be used when initiating the decomposition process:

```
1377     ^([\^:/?#+:)?/*([\^:/?#]*@)?(((\^/?[:#]*\.)*(\^/?#:\.)(\^/?#:\.+)?)
1378     (: \d+)?([\^?#]*)(\?[\^#]*)?(\#.*)?$
1379     8           1           2           3         4           5           6           7
1380     8           9           10          11
```

1381 Subexpression 3 MUST result in a Fully-Qualified Domain Name (FQDN), which will be the basis for
1382 retrieving metadata locations from this zone.

1383 **4.2.3.2 Obtaining Metadata via the DNS**

1384 Upon completion of the parsing of the identifier, the application then performs a DNS query for the resulting
1385 domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses.
1386 Applications MAY exclude from the result set any service definitions that do not concern the present
1387 request operations.

1388 Resolving applications MUST subsequently order the result set according to the order field, and MAY
1389 order the result set based on the preference set. Resolvers are NOT REQUIRED to follow the ordering of
1390 the preferences field. The resulting NAPTR resource record(s) are operated on iteratively (based on the
1391 order flag) until a terminal NAPTR resource record is reached.

1392 The result will be a well-formed, absolute URL, which is then used to retrieve the metadata document.

1393 **4.2.4 Metadata Location Caching**

1394 Location caching MUST NOT exceed the TTL of the DNS zone from which the location was derived.
1395 Resolvers MUST obtain a fresh copy of the metadata location upon reaching the expiration of the TTL of
1396 the zone.

1397 Publishers of metadata documents should carefully consider the TTL of the zone when making changes
1398 to metadata document locations. Should such a location change occur, a publisher MUST either keep the
1399 document at both the old and new location until all conforming resolvers are certain to have the updated
1400 location (e.g.: time of zone change + TTL), or provide an HTTP Redirect [RFC2616] response at the old
1401 location specifying the new location.

1402 **4.3 Post-Processing of Metadata**

1403 The following sections describe the post-processing of metadata.

1404 **4.3.1 Metadata Instance Caching**

1405 Document caching MUST NOT exceed the `validUntil` or `cacheDuration` attribute of the subject
1406 element(s). If metadata elements have parent elements which contain caching policies, the parent
1407 element takes precedence.

1408 To properly process the `cacheDuration` attribute, consumers MUST retain the date and time when the
1409 document was retrieved.

1410 When a document or element has expired, the consumer MUST retrieve a fresh copy, which may require
1411 a refresh of the document location(s). Consumers SHOULD process document cache processing
1412 according to [RFC2616] Section 13, and MAY request the Last-Modified date and time from the HTTP
1413 server. Publishers SHOULD ensure acceptable cache processing as described in [RFC2616] (Section
1414 10.3.5 304 Not Modified).

1415 **4.3.2 Handling of HTTPS Redirects**

1416 Publishers MAY issue an HTTP Redirect (301 Moved Permanently, 302 or 307 Temporary Redirect)
1417 [RFC2616], and user agents MUST follow the specified URL in the Redirect response. Redirects
1418 SHOULD be of the same protocol as the initial request.

1419 **4.3.3 Processing of XML Signatures and General Trust Processing**

1420 Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and
1421 for the trust ascribed to the entity described by such metadata:

- 1422 • Trust derived from the signature of the DNS zone from which the metadata location URL was

- 1423 resolved, ensuring accuracy of the metadata document location(s)
- 1424 • Trust derived from signature processing of the metadata document itself, ensuring the integrity of
1425 the XML document
- 1426 • Trust derived from the SSL/TLS server authentication of the metadata location URL, ensuring the
1427 identity of the publisher of the metadata
- 1428 Post-processing of the metadata document MUST include signature processing at the XML-document
1429 level and MAY include one of the other two processes. Specifically, the relying party MAY choose to trust
1430 any of the cited authorities in the resolution and parsing process. Publishers of metadata MUST employ a
1431 document-integrity mechanism and MAY employ any of the other two processing profiles to establish trust
1432 in the metadata document, governed by implementation policies.

1433 **4.3.3.1 Processing Signed DNS Zones**

1434 Verification of DNS zone signature SHOULD be processed, if present, as described in [\[RFC2535\]](#).

1435 **4.3.3.2 Processing Signed Documents and Fragments**

1436 Published metadata documents SHOULD be signed, as described in Section 3, either by a certificate
1437 issued to the subject of the document, or by another trusted party. Publishers MAY consider signatures of
1438 other parties as a means of trust conveyance.

1439 Metadata consumers MUST validate signatures, when present, on the metadata document as described
1440 by Section 3.

1441 **4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL**

1442 It is STRONGLY RECOMMENDED that publishers implement TLS/SSL URLs; therefore, consumers
1443 SHOULD consider the trust inherited from the issuer of the TLS/SSL certificate. Publication URLs may not
1444 always be located in the domain of the subject of the metadata document; therefore, consumers SHOULD
1445 NOT presume certificates whose subject is the entity in question, as it may be hosted by another trusted
1446 party.

1447 As the basis of this trust may not be available against a cached document, other mechanisms SHOULD
1448 be used under such circumstances.

5 References

1449

- 1450 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
1451 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1452 **[SAMLBind]** S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language*
1453 *(SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-bindings-
1454 2.0-cd-02. See <http://www.oasis-open.org/committees/security/>.
- 1455 **[SAMLConform]** P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion*
1456 *Markup Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID
1457 sstc-saml-conformance-2.0-cd-02. [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1458 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1459 **[SAMLCore]** S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion*
1460 *Markup Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID
1461 sstc-saml-core-2.0-cd-02. See <http://www.oasis-open.org/committees/security/>.
- 1462 **[SAMLMeta-xsd]** S. Cantor et al., *SAML metadata schema*. OASIS SSTC, September 2004.
1463 Document ID sstc-saml-schema-metadata-2.0. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1464 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1465 **[SAMLProf]** S. Cantor et al., *Profiles for the OASIS Security Assertion Markup Language*
1466 *(SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-profiles-
1467 2.0-cd-02. See <http://www.oasis-open.org/committees/security/>.
- 1468 **[SAMLSec]** F. Hirsch et al., *Security and Privacy Considerations for the OASIS Security*
1469 *Assertion Markup Language (SAML) V2.0*. OASIS SSTC, September 2004.
1470 Document ID sstc-saml-sec-consider-2.0-cd-02. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1471 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1472 **[XMLEnc]** D. Eastlake et al., *XML-Encryption Syntax and Processing*,
1473 <http://www.w3.org/TR/xmlenc-core/>, World Wide Web Consortium.
- 1474 **[XMLSig]** D. Eastlake et al., *XML-Signature Syntax and Processing*,
1475 <http://www.w3.org/TR/xmlsig-core/>, World Wide Web Consortium.

1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523

6 Registration of MIME media type application/samlmetadata+xml

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

Introduction

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

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

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

MIME media type name: application

MIME subtype name: samlmetadata+xml

Required parameters: none

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

Encoding considerations:
Same as for application/xml [RFC3023].

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

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

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

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

1524 For a more detailed discussion of SAML v2.0 metadata and its security considerations, please
1525 see [SAMLv2Meta]. For a discussion of overall SAML v2.0 security considerations and specific
1526 security-related design features, please refer to the SAML v2.0 specifications listed in the below
1527 bibliography. The specifications containing security-specific information are explicitly listed.
1528

1529 Interoperability considerations:

1530 SAML v2.0 metadata explicitly supports identifying the protocols and versions supported by the
1531 identified entities. For example, an identity provider entity can be denoted as supporting SAML
1532 v2.0, SAML v1.1 [SAMLv1.1], Liberty ID-FF 1.2 [LAPFF], or even other protocols if they are
1533 unambiguously identifiable via URI [RFC2396]. This protocol support information is conveyed via
1534 the `protocolSupportEnumeration` attribute of metadata objects of the
1535 `RoleDescriptorType`.
1536

1537 Published specification:

1538 [SAMLv2Meta] explicitly specifies use of the `application/samlmetadata+xml` MIME media type.
1539

1540 Applications which use this media type:

1541 Potentially any application implementing SAML v2.0, as well as those applications implementing
1542 specifications based on SAML, e.g. those available from the Liberty Alliance [LAP].
1543

1544 Additional information:

1545

1546 Magic number(s):

1547 In general, the same as for `application/xml` [RFC3023]. In particular, the XML root
1548 element of the returned object will have a namespace-qualified name with:
1549

1550 - a local name of: `EntityDescriptor`, `AffiliationDescriptor`, or
1551 `EntitiesDescriptor`

1553 - a namespace URI of: `urn:oasis:names:tc:SAML:2.0:metadata`
1554 (the SAMLv2.0 metadata namespace)
1555
1556

1557 File extension(s): none
1558

1559 Macintosh File Type Code(s): none
1560

1561 Person & email address to contact for further information:

1562 This registration is made on behalf of the OASIS Security Services Technical Committee (SSTC)
1563 Please refer to the SSTC website for current information on committee chairperson(s) and their
1564 contact addresses: <http://www.oasis-open.org/committees/security/>. Committee members should
1565 submit comments and potential errata to the securityservices@lists.oasis-open.org list. Others
1566 should submit them by filling out the web form located at [http://www.oasis-
1567 open.org/committees/comments/form.php?wg_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security).
1568

1569 Additionally, the SAML developer community email distribution list, [saml-dev@lists.oasis-
1570 open.org](mailto:saml-dev@lists.oasis-open.org), may be employed to discuss usage of the `application/samlmetadata+xml` MIME media
1571 type. The "saml-dev" mailing list is publicly archived here: [http://lists.oasis-
1572 open.org/archives/saml-dev/](http://lists.oasis-open.org/archives/saml-dev/). To post to the "saml-dev" mailing list, one must subscribe to it. To
1573 subscribe, send a message with the single word "subscribe" in the message body, to: [saml-dev-
1574 request@lists.oasis-open.org](mailto:saml-dev-request@lists.oasis-open.org).
1575

1576 Intended usage: COMMON
1577
1578

1579 Author/Change controller:
1580 The SAML specification sets are a work product of the OASIS Security Services Technical
1581 Committee (SSTC). OASIS and the SSTC have change control over the SAML specification sets.
1582

1583 Bibliography

- 1584 [LAP] “*Liberty Alliance Project*”. See <http://www.projectliberty.org/>
1585
1586
- 1587 [LAPFF] “Liberty Alliance Project: Federation Framework”. See
1588 <http://www.projectliberty.org/resources/specifications.php#box1>
1589
- 1590 [OASIS] “*Organization for the Advancement of Structured Information Systems*”.
1591 See <http://www.oasis-open.org/>
1592
- 1593 [RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifiers*
1594 *(URI): Generic Syntax*. IETF RFC 2396, August 1998. Available at
1595 <http://www.ietf.org/rfc/rfc2396.txt>
1596
- 1597 [RFC3023] M. Murata, S. St.Laurent, D. Kohn, “*XML Media Types*”, IETF Request for
1598 Comments 3023, January 2001. Available as [http://www.rfc-](http://www.rfc-editor.org/rfc/rfc3023.txt)
1599 [editor.org/rfc/rfc3023.txt](http://www.rfc-editor.org/rfc/rfc3023.txt)
1600
- 1601 [SAMLv1.1] OASIS Security Services Technical Committee, “*Security Assertion*
1602 *Markup Language (SAML) Version 1.1 Specification Set*”. OASIS
1603 Standard 200308, August 2003. Available as [http://www.oasis-](http://www.oasis-open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip)
1604 [open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip](http://www.oasis-open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip)
1605
- 1606 [SAMLv2.0] OASIS Security Services Technical Committee, “*Security Assertion*
1607 *Markup Language (SAML) Version 2.0 Specification Set*”. Committee
1608 Draft. Available at <http://www.oasis-open.org/committees/security/>
1609
- 1610 [SAMLv2Bind] S. Cantor et al., “*Bindings for the OASIS Security Assertion Markup*
1611 *Language (SAML) V2.0*”. OASIS SSTC, September 2004. Document ID
1612 sstc-saml-bindings-2.0-cd-02. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1613 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
1614
- 1615 [SAMLv2Core] S. Cantor et al., “*Assertions and Protocols for the OASIS Security*
1616 *Assertion Markup Language (SAML) V2.0*”. OASIS SSTC, September
1617 2004. Document ID sstc-saml-core-2.0-cd-02. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1618 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
1619
- 1620 [SAMLv2Meta] S. Cantor et al., *Metadata for the OASIS Security Assertion Markup*
1621 *Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID
1622 sstc-saml-metadata-2.0-cd-02. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1623 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
1624
- 1625 [SAMLv2Prof] S. Cantor et al., “*Profiles for the OASIS Security Assertion Markup*
1626 *Language (SAML) V2.0*”. OASIS SSTC, September 2004. Document ID
1627 sstc-saml-profiles-2.0-cd-02. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1628 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
1629
- 1630 [SAMLv2Sec] F. Hirsch et al., “*Security and Privacy Considerations for the OASIS*
1631 *Security Assertion Markup Language (SAML) V2.0*”. OASIS SSTC,
1632 September 2004. Document ID sstc-saml-sec-consider-2.0-cd-02. See
1633 <http://www.oasis-open.org/committees/security/>

1634 [SSTC] "OASIS Security Services Technical Committee". See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1635 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
1636
1637 [XML] Bray, T., Paoli, J., Sperberg-McQueen, C.M. and E. Maler, François
1638 Yergeau, "*Extensible Markup Language (XML) 1.0 (Third Edition)*", World
1639 Wide Web Consortium Recommendation REC-xml, Feb 2004, Available
1640 as <http://www.w3.org/TR/REC-xml/>
1641

1642 Appendix A. Acknowledgments

1643 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
1644 Committee, whose voting members at the time of publication were:

- 1645 • Conor Cahill, AOL
- 1646 • John Hughes, ATOS Origin
- 1647 • Hal Lockhart, BEA Systems
- 1648 • Rick Randall, Booz Allen Hamilton
- 1649 • Ronald Jacobson, Computer Associates
- 1650 • Gavenraj Sodhi, Computer Associates
- 1651 • Tim Alsop, CyberSafe Limited
- 1652 • Paul Madsen, Entrust
- 1653 • Carolina Canales-Valenzuela, Ericsson
- 1654 • Dana Kaufman, Forum Systems
- 1655 • Irving Reid, Hewlett-Packard
- 1656 • Paula Austel, IBM
- 1657 • Maryann Hondo, IBM
- 1658 • Michael McIntosh, IBM
- 1659 • Anthony Nadalin, IBM
- 1660 • Nick Ragouzis, Individual
- 1661 • Scott Cantor, Internet2
- 1662 • Bob Morgan, Internet2
- 1663 • Prateek Mishra, Netegrity
- 1664 • Forest Yin, Netegrity
- 1665 • Peter Davis, Neustar
- 1666 • Frederick Hirsch, Nokia
- 1667 • John Kemp, Nokia
- 1668 • Senthil Sengodan, Nokia
- 1669 • Scott Kiestler, Novell
- 1670 • Cameron Morris, Novell
- 1671 • Charles Knouse, Oblix
- 1672 • Steve Anderson, OpenNetwork
- 1673 • Ari Kermaier, Oracle
- 1674 • Vamsi Motukuru, Oracle
- 1675 • Darren Platt, Ping Identity
- 1676 • Jim Lien, RSA Security
- 1677 • John Linn, RSA Security
- 1678 • Rob Philpott, RSA Security
- 1679 • Dipak Chopra, SAP
- 1680 • Jahan Moreh, Sigaba
- 1681 • Bhavna Bhatnagar, Sun Microsystems
- 1682 • Jeff Hodges, Sun Microsystems
- 1683 • Eve Maler, Sun Microsystems

1684

- 1685 • Ronald Monzillo, Sun Microsystems
- 1686 • Emily Xu, Sun Microsystems
- 1687 • Mike Beach, Boeing
- 1688 • Greg Whitehead, Trustgenix

1689

1690 The editors also would like to acknowledge the following people for their contributions to previous versions
1691 of the OASIS Security Assertions Markup Language Standard:

- 1692 • Stephen Farrell, Baltimore Technologies
- 1693 • David Orchard, BEA Systems
- 1694 • Krishna Sankar, Cisco Systems
- 1695 • Zahid Ahmed, CommerceOne
- 1696 • Carlisle Adams, Entrust
- 1697 • Tim Moses, Entrust
- 1698 • Nigel Edwards, Hewlett-Packard
- 1699 • Joe Pato, Hewlett-Packard
- 1700 • Bob Blakley, IBM
- 1701 • Marlena Erdos, IBM
- 1702 • Marc Chanliau, Netegrity
- 1703 • Chris McLaren, Netegrity
- 1704 • Lynne Rosenthal, NIST
- 1705 • Mark Skall, NIST
- 1706 • Simon Godik, Overxeer
- 1707 • Charles Norwood, SAIC
- 1708 • Evan Prodromou, Securant
- 1709 • Robert Griffin, RSA Security (former editor)
- 1710 • Sai Allarvarpu, Sun Microsystems
- 1711 • Chris Ferris, Sun Microsystems
- 1712 • Emily Xu, Sun Microsystems
- 1713 • Mike Myers, Traceroute Security
- 1714 • Phillip Hallam-Baker, VeriSign (former editor)
- 1715 • James Vanderbeek, Vodafone
- 1716 • Mark O'Neill, Vordel
- 1717 • Tony Palmer, Vordel

1718

1719 Finally, the editors wish to acknowledge the following people for their contributions of material used as
1720 input to the OASIS Security Assertions Markup Language specifications:

- 1721 • Thomas Gross, IBM
- 1722 • Birgit Pfitzmann, IBM

1723 Appendix B. Notices

1724 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
1725 might be claimed to pertain to the implementation or use of the technology described in this document or
1726 the extent to which any license under such rights might or might not be available; neither does it represent
1727 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
1728 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
1729 available for publication and any assurances of licenses to be made available, or the result of an attempt
1730 made to obtain a general license or permission for the use of such proprietary rights by implementors or
1731 users of this specification, can be obtained from the OASIS Executive Director.

1732 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
1733 other proprietary rights which may cover technology that may be required to implement this specification.
1734 Please address the information to the OASIS Executive Director.

1735 **Copyright © OASIS Open 2004. All Rights Reserved.**

1736 This document and translations of it may be copied and furnished to others, and derivative works that
1737 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
1738 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
1739 this paragraph are included on all such copies and derivative works. However, this document itself may
1740 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
1741 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
1742 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
1743 into languages other than English.

1744 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
1745 or assigns.

1746 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1747 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1748 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1749 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.