



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

5 **Committee Draft 04, 15 January 2005**

6 **Document identifier:**

7 sstc-saml-metadata-2.0-cd-04

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

44 Gary Ellison, Sun Microsystems
45 Eve Maler, Sun Microsystems
46 Ron Monzillo, Sun Microsystems
47 Greg Whitehead, Trustgenix

48 **Abstract:**

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

55 **Status:**

56 This is a **Committee Draft** approved by the Security Services Technical Committee on 15
57 January 2005.

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

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

67 Table of Contents

68	1 Introduction.....	5
69	1.1 Notation.....	5
70	2 Metadata for SAML V2.0.....	6
71	2.1 Namespaces	6
72	2.2 Common Types.....	7
73	2.2.1 Simple Type entityIDType.....	7
74	2.2.2 Complex Type EndpointType.....	7
75	2.2.3 Complex Type IndexedEndpointType.....	8
76	2.2.4 Complex Type localizedNameType.....	8
77	2.2.5 Complex Type localizedURIType.....	9
78	2.3 Root Elements.....	9
79	2.3.1 Element <EntitiesDescriptor>.....	9
80	2.3.2 Element <EntityDescriptor>.....	10
81	2.3.2.1 Element <Organization>.....	12
82	2.3.2.2 Element <ContactPerson>.....	12
83	2.3.2.3 Element <AdditionalMetadataLocation>.....	14
84	2.4 Role Descriptor Elements.....	14
85	2.4.1 Element <RoleDescriptor>.....	14
86	2.4.1.1 Element <KeyDescriptor>.....	15
87	2.4.2 Complex Type SSODescriptorType.....	16
88	2.4.3 Element <IDPSSODescriptor>.....	17
89	2.4.4 Element <SPSSODescriptor>.....	18
90	2.4.4.1 Element <AttributeConsumingService>.....	19
91	2.4.4.2 Element <RequestedAttribute>.....	19
92	2.4.5 Element <AuthnAuthorityDescriptor>.....	20
93	2.4.6 Element <PDPDescriptor>.....	20
94	2.4.7 Element <AttributeAuthorityDescriptor>.....	21
95	2.5 Element <AffiliationDescriptor>.....	22
96	2.6 Examples.....	23
97	3 Signature Processing.....	26
98	3.1 XML Signature Profile.....	26
99	3.1.1 Signing Formats and Algorithms.....	26
100	3.1.2 References.....	26
101	3.1.3 Canonicalization Method.....	26
102	3.1.4 Transforms.....	27
103	3.1.5 KeyInfo.....	27
104	4 Metadata Publication and Resolution.....	28
105	4.1 Publication and Resolution via Well-Known Location.....	28
106	4.1.1 Publication.....	28
107	4.1.2 Resolution.....	28
108	4.2 Publishing and Resolution via DNS.....	28
109	4.2.1 Publication.....	29
110	4.2.1.1 First Well Known Rule.....	29
111	4.2.1.2 The Order Field.....	29
112	4.2.1.3 The Preference Field.....	29
113	4.2.1.4 The Flag Field.....	30
114	4.2.1.5 The Service Field.....	30

115	4.2.1.6 The Regex and Replacement Fields.....	30
116	4.2.2 NAPTR Examples.....	31
117	4.2.2.1 Entity Metadata NAPTR Examples.....	31
118	4.2.2.2 Name Identifier Examples.....	31
119	4.2.3 Resolution.....	31
120	4.2.3.1 Parsing the Unique Identifier.....	31
121	4.2.3.2 Obtaining Metadata via the DNS.....	32
122	4.2.4 Metadata Location Caching.....	32
123	4.3 Post-Processing of Metadata.....	32
124	4.3.1 Metadata Instance Caching.....	32
125	4.3.2 Handling of HTTPS Redirects.....	32
126	4.3.3 Processing of XML Signatures and General Trust Processing.....	32
127	4.3.3.1 Processing Signed DNS Zones.....	33
128	4.3.3.2 Processing Signed Documents and Fragments.....	33
129	4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL.....	33
130	5 References.....	34
131	Appendix A.Registration of MIME media type application/samlmetadata+xml.....	35
132	Appendix B. Acknowledgments.....	38
133	Appendix C. Notices.....	40

1 Introduction

134

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

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

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

1.1 Notation

145

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

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

150

151 `Example code listings appear like this.`

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

153 Conventional XML namespace prefixes are used throughout this specification to stand for their respective
154 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

155

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

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

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

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

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

2.1 Namespaces

176

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

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

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

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

```
181 <schema
182   targetNamespace="urn:oasis:names:tc:SAML:2.0:metadata"
183   xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
184   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
185   xmlns:xenc="http://www.w3.org/2001/04/xmenc#"
186   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
187   xmlns="http://www.w3.org/2001/XMLSchema"
188   elementFormDefault="unqualified"
189   attributeFormDefault="unqualified"
190   blockDefault="substitution"
191   version="2.0">
192   <import namespace="http://www.w3.org/2000/09/xmldsig#"
193     schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-
194 core-schema.xsd"/>
195   <import namespace="http://www.w3.org/2001/04/xmenc#"
196     schemaLocation="http://www.w3.org/TR/2002/REC-xmenc-core-20021210/xenc-
197 schema.xsd"/>
198   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
199     schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
200   <import namespace="http://www.w3.org/XML/1998/namespace"
201     schemaLocation="http://www.w3.org/2001/xml.xsd"/>
202   <annotation>
```

```

203     <documentation>
204         Document identifier: sstc-saml-schema-metadata-2.0
205         Location: http://www.oasis-
206 open.org/committees/documents.php?wg_abbrev=security
207         Revision history:
208             V2.0 CD-04 (January, 2005):
209             Schema for SAML metadata, first published in SAML 2.0.
210     </documentation>
211 </annotation>
212 ...
213 </schema>

```

214 2.2 Common Types

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

217 2.2.1 Simple Type `entityIDType`

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

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

```

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

```

229 2.2.2 Complex Type `EndpointType`

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

233 `Binding` [Required]

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

236 `Location` [Required]

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

239 `ResponseLocation` [Optional]

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

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

247 In most contexts, elements of this type appear in unbounded sequences in the schema. This is to permit a
 248 protocol or profile to be offered by an entity at multiple endpoints, usually with different protocol bindings,

249 allowing the metadata consumer to choose an appropriate endpoint for its needs. Multiple endpoints might
250 also offer "client-side" load-balancing or failover, particular in the case of a synchronous protocol binding.

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

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

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

264 2.2.3 Complex Type IndexedEndpointType

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

268 **index** [Required]

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

273 **isDefault** [Optional]

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

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

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

```
281 <complexType name="IndexedEndpointType">  
282   <complexContent>  
283     <extension base="md:EndpointType">  
284       <attribute name="index" type="unsignedShort" use="required"/>  
285       <attribute name="isDefault" type="boolean" use="optional"/>  
286     </extension>  
287   </complexContent>  
288 </complexType>
```

289 2.2.4 Complex Type localizedNameType

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

```
292 <complexType name="localizedNameType">  
293   <simpleContent>  
294     <extension base="string">  
295       <attribute ref="xml:lang" use="required"/>  
296     </extension>  
297   </simpleContent>
```


298 `</complexType>`

299 **2.2.5 Complex Type localizedURIType**

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

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

```
303 <complexType name="localizedURIType">  
304   <simpleContent>  
305     <extension base="anyURI">  
306       <attribute ref="xml:lang" use="required"/>  
307     </extension>  
308   </simpleContent>  
309 </complexType>
```

310 **2.3 Root Elements**

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

314 **2.3.1 Element <EntitiesDescriptor>**

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

318 ID [Optional]

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

320 validUntil [Optional]

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

323 cacheDuration [Optional]

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

326 Name [Optional]

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

328 `<ds:Signature>` [Optional]

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

331 `<Extensions>` [Optional]

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

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

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

337 When used as the root element of a metadata instance, this element **MUST** contain either a `validUntil`

338 or `cacheDuration` attribute. It is RECOMMENDED that only the root element of a metadata instance
339 contain either attribute.

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

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

363 2.3.2 Element `<EntityDescriptor>`

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

- 368 • SSO Identity Provider
- 369 • SSO Service Provider
- 370 • Authentication Authority
- 371 • Attribute Authority
- 372 • Policy Decision Point
- 373 • Affiliation

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

375 `entityID` [Required]

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

378 `ID` [Optional]

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

380 `validUntil` [Optional]

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

383 `cacheDuration` [Optional]

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

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

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

393 <RoleDescriptor>, <IDPSSODescriptor>, <SPSSODescriptor>,
 394 <AuthnAuthorityDescriptor>, <AttributeAuthorityDescriptor>, <PDPDescriptor> [One
 395 or More]
 396 **OR**

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

400 <Organization> [Optional]
 401 Optional element identifying the organization responsible for the SAML entity described by the
 402 element.

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

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

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

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

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

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

```

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

```

```

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

```

447 2.3.2.1 Element <Organization>

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

452 <Extensions> [Optional]

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

456 <OrganizationName> [One or More]

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

458 <OrganizationDisplayName> [One or More]

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

460 <OrganizationURL> [One or More]

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

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

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

```

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

```

480 2.3.2.2 Element <ContactPerson>

481 The <ContactPerson> element specifies basic contact information about a person responsible in some
482 capacity for a SAML entity or role. The use of this element is always optional. Its content is informative in

483 nature and does not directly map to any core SAML elements or attributes. Its **ContactType** complex type
484 consists of the following elements and attributes:

485 `contactType` [Required]

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

488 `<Extensions>` [Optional]

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

492 `<Company>` [Optional]

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

494 `<GivenName>` [Optional]

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

496 `<SurName>` [Optional]

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

498 `<EmailAddress>` [Zero or More]

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

501 `<TelephoneNumber>` [Zero or More]

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

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

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

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

533 2.3.2.3 Element <AdditionalMetadataLocation>

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

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

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

550 2.4 Role Descriptor Elements

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

554 2.4.1 Element <RoleDescriptor>

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

559 ID [Optional]

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

561 validUntil [Optional]

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

564 cacheDuration [Optional]

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

567 protocolSupportEnumeration [Required]

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

573 errorURL [Optional]

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

- 576 `<ds:Signature>` [Optional]
 577 An XML signature that authenticates the containing element and its contents, as described in
 578 Section 3.
- 579 `<Extensions>` [Optional]
 580 This contains optional metadata extensions that are agreed upon between a metadata publisher
 581 and consumer. Extension elements MUST be namespace-qualified by a non-SAML-defined
 582 namespace.
- 583 `<KeyDescriptor>` [Zero or More]
 584 Optional sequence of elements that provides information about the cryptographic keys that the
 585 entity uses when acting in this role.
- 586 `<Organization>` [Optional]
 587 Optional element specifies the organization associated with this role. Identical to the element used
 588 within the `<EntityDescriptor>` element.
- 589 `<ContactPerson>` [Zero or More]
 590 Optional sequence of elements specifying contacts associated with this role. Identical to the
 591 element used within the `<EntityDescriptor>` element.
- 592 Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

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

```

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

```

615 2.4.1.1 Element `<KeyDescriptor>`

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

- 619 `use` [Optional]
 620 Optional attribute specifying the purpose of the key being described. Values are drawn from the
 621 **KeyTypes** enumeration, and consist of the values `encryption` and `signing`.
- 622 `<ds:KeyInfo>` [Required]
 623 Optional element that directly or indirectly identifies a key. See [XMLSig] for additional details on

624 the use of this element.

625 <EncryptionMethod> [Zero or More]

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

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

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

646 2.4.2 Complex Type SSODescriptorType

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

651 <ArtifactResolutionService> [Zero or More]

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

655 <SingleLogoutService> [Zero or More]

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

658 <ManageNameIDService> [Zero or More]

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

661 <NameIDFormat> [Zero or More]

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

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

```
666 <complexType name="SSODescriptorType" abstract="true">
667   <complexContent>
668     <extension base="md:RoleDescriptorType">
669       <sequence>
670         <element ref="md:ArtifactResolutionService" minOccurs="0"
671 maxOccurs="unbounded"/>
```



```

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

```

685 2.4.3 Element <IDPSSODescriptor>

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

689 WantAuthnRequestsSigned [Optional]

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

692 <SingleSignOnService> [One or More]

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

696 <NameIDMappingService> [Zero or More]

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

700 <AssertionIDRequestService> [Zero or More]

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

704 <AttributeProfile> [Zero or More]

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

707 <saml:Attribute> [Zero or More]

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

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

```

714 <element name="IDPSSODescriptor" type="md:IDPSSODescriptorType"/>
715 <complexType name="IDPSSODescriptorType">
716     <complexContent>
717         <extension base="md:SSODescriptorType">
718             <sequence>
719                 <element ref="md:SingleSignOnService" maxOccurs="unbounded"/>

```

```

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

```

736 2.4.4 Element <SPSSODescriptor>

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

740 AuthnRequestsSigned [Optional]

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

743 WantAssertionsSigned [Optional]

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

748 <AssertionConsumerService> [One or More]

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

752 <AttributeConsumingService> [Zero or More]

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

755 At most one <AttributeConsumingService> element can have the attribute `isDefault` set to
756 true. It is permissible for none of the included elements to contain an `isDefault` attribute set to true.

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

```

759 <element name="SPSSODescriptor" type="md:SPSSODescriptorType"/>
760 <complexType name="SPSSODescriptorType">
761     <complexContent>
762         <extension base="md:SSODescriptorType">
763             <sequence>
764                 <element ref="md:AssertionConsumerService" maxOccurs="unbounded"/>
765                 <element ref="md:AttributeConsumingService" minOccurs="0"
766 maxOccurs="unbounded"/>
767             </sequence>
768             <attribute name="AuthnRequestsSigned" type="boolean" use="optional"/>
769             <attribute name="WantAssertionsSigned" type="boolean" use="optional"/>
770         </extension>
771     </complexContent>

```

```
772 </complexType>
773 <element name="AssertionConsumerService" type="md:IndexedEndpointType"/>
```

774 2.4.4.1 Element <AttributeConsumingService>

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

778 index [Required]

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

781 isDefault [Optional]

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

784 <ServiceName> [One or More]

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

786 <ServiceDescription> [Zero or More]

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

788 <RequestedAttribute> [One or More]

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

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

```
792 <element name="AttributeConsumingService" type="md:AttributeConsumingServiceType"/>
793 <complexType name="AttributeConsumingServiceType">
794   <sequence>
795     <element ref="md:ServiceName" maxOccurs="unbounded"/>
796     <element ref="md:ServiceDescription" minOccurs="0" maxOccurs="unbounded"/>
797     <element ref="md:RequestedAttribute" maxOccurs="unbounded"/>
798   </sequence>
799   <attribute name="index" type="unsignedShort" use="required"/>
800   <attribute name="isDefault" type="boolean" use="optional"/>
801 </complexType>
802 <element name="ServiceName" type="md:localizedNameType"/>
803 <element name="ServiceDescription" type="md:localizedNameType"/>
```

804 2.4.4.2 Element <RequestedAttribute>

805 The <RequestedAttribute> element specifies a service provider's interest in a specific SAML
806 attribute, optionally including specific values. Its **RequestedAttributeType** complex type extends the
807 **saml:AttributeType** with the following attribute:

808 isRequired [Optional]

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

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

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

```
815 <element name="RequestedAttribute" type="md:RequestedAttributeType"/>
```

```

816 <complexType name="RequestedAttributeType">
817   <complexContent>
818     <extension base="saml:AttributeType">
819       <attribute name="isRequired" type="boolean" use="optional"/>
820     </extension>
821   </complexContent>
822 </complexType>

```

823 2.4.5 Element <AuthnAuthorityDescriptor>

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

827 <AuthnQueryService> [One or More]

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

831 <AssertionIDRequestService> [Zero or More]

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

835 <NameIDFormat> [Zero or More]

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

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

```

840 <element name="AuthnAuthorityDescriptor" type="md:AuthnAuthorityDescriptorType"/>
841 <complexType name="AuthnAuthorityDescriptorType">
842   <complexContent>
843     <extension base="md:RoleDescriptorType">
844       <sequence>
845         <element ref="md:AuthnQueryService" maxOccurs="unbounded"/>
846         <element ref="md:AssertionIDRequestService" minOccurs="0"
847 maxOccurs="unbounded"/>
848         <element ref="md:NameIDFormat" minOccurs="0" maxOccurs="unbounded"/>
849       </sequence>
850     </extension>
851   </complexContent>
852 </complexType>
853 <element name="AuthnQueryService" type="md:EndpointType"/>

```

854 2.4.6 Element <PDPDescriptor>

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

858 <AuthzService> [One or More]

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

862 <AssertionIDRequestService> [Zero or More]

863 Zero or more elements of type **EndpointType** that describe endpoints that support the profile of

864 the Assertion Request protocol defined in [SAMLProf] or the special URI binding for assertion
865 requests defined in [SAMLBind].

866 <NameIDFormat> [Zero or More]

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

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

```
871 <element name="PDPDescriptor" type="md:PDPDescriptorType"/>
872 <complexType name="PDPDescriptorType">
873   <complexContent>
874     <extension base="md:RoleDescriptorType">
875       <sequence>
876         <element ref="md:AuthzService" maxOccurs="unbounded"/>
877         <element ref="md:AssertionIDRequestService" minOccurs="0"
878 maxOccurs="unbounded"/>
879         <element ref="md:NameIDFormat" minOccurs="0" maxOccurs="unbounded"/>
880       </sequence>
881     </extension>
882   </complexContent>
883 </complexType>
884 <element name="AuthzService" type="md:EndpointType"/>
```

885 2.4.7 Element <AttributeAuthorityDescriptor>

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

890 <AttributeService> [One or More]

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

894 <AssertionIDRequestService> [Zero or More]

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

898 <NameIDFormat> [Zero or More]

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

901 <AttributeProfile> [Zero or More]

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

904 <saml:Attribute> [Zero or More]

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

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

```

910 <element name="AttributeAuthorityDescriptor"
911 type="md:AttributeAuthorityDescriptorType"/>
912 <complexType name="AttributeAuthorityDescriptorType">
913   <complexContent>
914     <extension base="md:RoleDescriptorType">
915       <sequence>
916         <element ref="md:AttributeService" maxOccurs="unbounded"/>
917         <element ref="md:AssertionIDRequestService" minOccurs="0"
918 maxOccurs="unbounded"/>
919         <element ref="md:NameIDFormat" minOccurs="0" maxOccurs="unbounded"/>
920         <element ref="md:AttributeProfile" minOccurs="0"
921 maxOccurs="unbounded"/>
922         <element ref="saml:Attribute" minOccurs="0" maxOccurs="unbounded"/>
923       </sequence>
924     </extension>
925   </complexContent>
926 </complexType>
927 <element name="AttributeService" type="md:EndpointType"/>

```

928 2.5 Element <AffiliationDescriptor>

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

935 affiliationOwnerID [Required]

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

939 ID [Optional]

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

941 validUntil [Optional]

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

944 cacheDuration [Optional]

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

947 <ds:Signature> [Optional]

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

950 <Extensions> [Optional]

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

954 <AffiliateMember> [One or More]

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

957 <KeyDescriptor> [Zero or More]

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

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

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

```
964 <element name="AffiliationDescriptor" type="md:AffiliationDescriptorType"/>
965 <complexType name="AffiliationDescriptorType">
966   <sequence>
967     <element ref="ds:Signature" minOccurs="0"/>
968     <element ref="md:Extensions" minOccurs="0"/>
969     <element ref="md:AffiliateMember" maxOccurs="unbounded"/>
970     <element ref="md:KeyDescriptor" minOccurs="0" maxOccurs="unbounded"/>
971   </sequence>
972   <attribute name="affiliationOwnerID" type="md:entityIDType" use="required"/>
973   <attribute name="validUntil" type="dateTime" use="optional"/>
974   <attribute name="cacheDuration" type="duration" use="optional"/>
975   <attribute name="ID" type="ID" use="optional"/>
976   <anyAttribute namespace="##other" processContents="lax"/>
977 </complexType>
978 <element name="AffiliateMember" type="md:entityIDType"/>
```

979 2.6 Examples

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

```
982
983 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
984   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
985   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
986   entityID="https://IdentityProvider.com/SAML">
987   <ds:Signature>...</ds:Signature>
988   <IDPSSODescriptor WantAuthnRequestsSigned="true"
989     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
990     <KeyDescriptor use="signing">
991       <ds:KeyInfo>
992         <ds:KeyName>IdentityProvider.com SSO Key</ds:KeyName>
993       </ds:KeyInfo>
994     </KeyDescriptor>
995     <ArtifactResolutionService isDefault="true" index="0"
996       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
997       Location="https://IdentityProvider.com/SAML/Artifact"/>
998     <SingleLogoutService
999       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1000       Location="https://IdentityProvider.com/SAML/SLO/SOAP"/>
1001     <SingleLogoutService
1002       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1003       Location="https://IdentityProvider.com/SAML/SLO/Browser"
1004       ResponseLocation="https://IdentityProvider.com/SAML/SLO/Response"/>
1005     <NameIDFormat>
1006       urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1007     </NameIDFormat>
1008     <NameIDFormat>
1009       urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1010     </NameIDFormat>
1011     <NameIDFormat>
1012       urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1013     </NameIDFormat>
1014     <SingleSignOnService
1015       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1016       Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
```

```

1017     <SingleSignOnService
1018         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1019         Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1020     <saml:Attribute
1021         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1022         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1023         FriendlyName="eduPersonPrincipalName">
1024     </saml:Attribute>
1025     <saml:Attribute
1026         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1027         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1028         FriendlyName="eduPersonAffiliation">
1029         <saml:AttributeValue>member</saml:AttributeValue>
1030         <saml:AttributeValue>student</saml:AttributeValue>
1031         <saml:AttributeValue>faculty</saml:AttributeValue>
1032         <saml:AttributeValue>employee</saml:AttributeValue>
1033         <saml:AttributeValue>staff</saml:AttributeValue>
1034     </saml:Attribute>
1035 </IDPSSODescriptor>
1036 <AttributeAuthorityDescriptor
1037     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1038     <KeyDescriptor use="signing">
1039         <ds:KeyInfo>
1040             <ds:KeyName>IdentityProvider.com AA Key</ds:KeyName>
1041         </ds:KeyInfo>
1042     </KeyDescriptor>
1043     <AttributeService
1044         Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1045         Location="https://IdentityProvider.com/SAML/AA/SOAP"/>
1046     <AssertionIDRequestService
1047         Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
1048         Location="https://IdentityProvider.com/SAML/AA/URI"/>
1049     <NameIDFormat>
1050         urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1051     </NameIDFormat>
1052     <NameIDFormat>
1053         urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1054     </NameIDFormat>
1055     <NameIDFormat>
1056         urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1057     </NameIDFormat>
1058     <saml:Attribute
1059         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1060         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1061         FriendlyName="eduPersonPrincipalName">
1062     </saml:Attribute>
1063     <saml:Attribute
1064         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1065         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1066         FriendlyName="eduPersonAffiliation">
1067         <saml:AttributeValue>member</saml:AttributeValue>
1068         <saml:AttributeValue>student</saml:AttributeValue>
1069         <saml:AttributeValue>faculty</saml:AttributeValue>
1070         <saml:AttributeValue>employee</saml:AttributeValue>
1071         <saml:AttributeValue>staff</saml:AttributeValue>
1072     </saml:Attribute>
1073 </AttributeAuthorityDescriptor>
1074 <Organization>
1075     <OrganizationName xml:lang="en">Identity Providers R US</OrganizationName>
1076     <OrganizationDisplayName xml:lang="en">
1077         Identity Providers R US, a Division of Lerxst Corp.
1078     </OrganizationDisplayName>
1079     <OrganizationURL
1080     xml:lang="en">https://IdentityProvider.com</OrganizationURL>
1081 </Organization>
1082 </EntityDescriptor>
1083

```


1084 The following is an example of metadata for a SAML system entity acting as a service provider. A
1085 signature is shown as a placeholder, without the actual content. For illustrative purposes, the service is
1086 one that does not require users to uniquely identify themselves, but rather authorizes access on the basis
1087 of a role-like attribute.
1088

```
1089 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"  
1090   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
1091   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
1092   entityID="https://ServiceProvider.com/SAML">  
1093   <ds:Signature>...</ds:Signature>  
1094   <SPSSODescriptor AuthnRequestsSigned="true"  
1095     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">  
1096     <KeyDescriptor use="signing">  
1097       <ds:KeyInfo>  
1098         <ds:KeyName>ServiceProvider.com SSO Key</ds:KeyName>  
1099       </ds:KeyInfo>  
1100     </KeyDescriptor>  
1101     <KeyDescriptor use="encryption">  
1102       <ds:KeyInfo>  
1103         <ds:KeyName>ServiceProvider.com Encrypt Key</ds:KeyName>  
1104       </ds:KeyInfo>  
1105       <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmenc#rsa-1_5"/>  
1106     </KeyDescriptor>  
1107     <SingleLogoutService  
1108       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"  
1109       Location="https://ServiceProvider.com/SAML/SLO/SOAP"/>  
1110     <SingleLogoutService  
1111       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"  
1112       Location="https://ServiceProvider.com/SAML/SLO/Browser"  
1113       ResponseLocation="https://ServiceProvider.com/SAML/SLO/Response"/>  
1114     <NameIDFormat>  
1115       urn:oasis:names:tc:SAML:2.0:nameid-format:transient  
1116     </NameIDFormat>  
1117     <AssertionConsumerService isDefault="true" index="0"  
1118       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"  
1119       Location="https://ServiceProvider.com/SAML/SSO/Artifact"/>  
1120     <AssertionConsumerService index="1"  
1121       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"  
1122       Location="https://ServiceProvider.com/SAML/SSO/POST"/>  
1123     <AttributeConsumingService index="0">  
1124       <ServiceName xml:lang="en">Academic Journals R US</ServiceName>  
1125       <RequestedAttribute  
1126         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1127         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7"  
1128         FriendlyName="eduPersonEntitlement">  
1129         <saml:AttributeValue>  
1130           https://ServiceProvider.com/entitlements/123456789  
1131         </saml:AttributeValue>  
1132       </RequestedAttribute>  
1133     </AttributeConsumingService>  
1134   </SPSSODescriptor>  
1135   <Organization>  
1136     <OrganizationName xml:lang="en">Academic Journals R US</OrganizationName>  
1137     <OrganizationDisplayName xml:lang="en">  
1138       Academic Journals R US, a Division of Dirk Corp.  
1139     </OrganizationDisplayName>  
1140     <OrganizationURL xml:lang="en">https://ServiceProvider.com</OrganizationURL>  
1141   </Organization>  
1142 </EntityDescriptor>
```

1143 3 Signature Processing

1144 Various elements in a metadata instance can be digitally signed (as indicated by the element's inclusion of
1145 a `<ds:Signature>` element), with the following benefits:

- 1146 • Metadata integrity
- 1147 • Authentication of the metadata by a trusted signer

1148 A digital signature is not always required, for example if the relying party obtains the information directly
1149 from the publishing entity directly (with no intermediaries) through a secure channel, with the entity having
1150 authenticated to the relying party by some means other than a digital signature.

1151 Many different techniques are available for "direct" authentication and secure channel establishment
1152 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,
1153 the applicable security requirements depend on the communicating applications.

1154 Additionally, elements can inherit signatures on enclosing parent elements that are themselves signed.

1155 In the absence of such context, it is RECOMMENDED that at least the root element of a metadata
1156 instance be signed.

1157 3.1 XML Signature Profile

1158 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
1159 and many choices. This section details the constraints on these facilities so that metadata processors do
1160 not have to deal with the full generality of XML Signature processing. This usage makes specific use of
1161 the **xs:ID**-typed attributes optionally present on the elements to which signatures can apply. These
1162 attributes are collectively referred to in this section as the identifier attributes.

1163 3.1.1 Signing Formats and Algorithms

1164 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
1165 detached.

1166 SAML metadata MUST use enveloped signatures when signing the elements defined in this specification.
1167 SAML processors SHOULD support the use of RSA signing and verification for public key operations in
1168 accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

1169 3.1.2 References

1170 Signed metadata elements MUST supply a value for the identifier attribute on the signed element. The
1171 element may or may not be the root element of the actual XML document containing the signed metadata
1172 element.

1173 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute
1174 value of the metadata element being signed. For example, if the identifier attribute value is "foo", then the
1175 URI attribute in the `<ds:Reference>` element MUST be "#foo".

1176 As a consequence, a metadata element's signature MUST apply to the content of the signed element and
1177 any child elements it contains.

1178 3.1.3 Canonicalization Method

1179 SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the

1180 <ds:CanonicalizationMethod> element of <ds:SignedInfo>, and as a <ds:Transform>
1181 algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML metadata
1182 embedded in an XML context can be verified independent of that context.

1183 **3.1.4 Transforms**

1184 Signatures in SAML metadata SHOULD NOT contain transforms other than the enveloped signature
1185 transform (with the identifier <http://www.w3.org/2000/09/xmlsig#enveloped-signature>) or the exclusive
1186 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
1187 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

1188 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
1189 not, verifiers MUST ensure that no content of the signed metadata element is excluded from the
1190 signature. This can be accomplished by establishing out-of-band agreement as to what transforms are
1191 acceptable, or by applying the transforms manually to the content and reverifying the result as consisting
1192 of the same SAML metadata.

1193 **3.1.5 KeyInfo**

1194 XML Signature [XMLSig] defines usage of the <ds:KeyInfo> element. SAML does not require the
1195 use of <ds:KeyInfo> nor does it impose any restrictions on its use. Therefore, <ds:KeyInfo> MAY
1196 be absent.

1197 4 Metadata Publication and Resolution

1198 Two mechanisms are provided for an entity to publish (and for a consumer to resolve the location of)
1199 metadata documents: via a "well-known-location" by directly dereferencing the entity's unique identifier (a
1200 URI variously referred to as an *entityID* or *providerID*), or indirectly by publishing the location of metadata
1201 in the DNS. Other out-of-band mechanisms are of course also permitted. A consumer that supports both
1202 approaches defined in this document MUST attempt resolution via DNS before using the "well-known-
1203 location" mechanism.

1204 When retrieval requires network transport of the document, the transport SHOULD be protected with
1205 mechanisms providing server authentication and integrity protection. For example, HTTP-based resolution
1206 SHOULD be protected with *TLS/SSL* [RFC2246] as amended by [RFC3546].

1207 Various mechanisms are described in this section to aid in establishing trust in the accuracy and
1208 legitimacy of metadata, including use of XML signatures, SSL/TLS server authentication, and DNS
1209 signatures. Regardless of the mechanism(s) used, relying parties SHOULD have some means by which to
1210 establish trust in metadata information before relying on it.

1211 4.1 Publication and Resolution via Well-Known Location

1212 The following sections describe publication and resolution of metadata by means of a well-known location.

1213 4.1.1 Publication

1214 Entities MAY publish their metadata documents at a well known location by placing the document at the
1215 location denoted by its unique identifier, which MUST be in the form of a URL (rather than a URN). See
1216 Section 8.3.6 of [SAMLCore] for more information about such identifiers. It is STRONGLY
1217 RECOMMENDED that `https` URLs be used for this purpose. An indirection mechanism supported by the
1218 URL scheme (such as an HTTP 1.1 302 redirect) MAY be used if the document is not placed directly at
1219 the location. If the publishing protocol permits MIME-based identification of content types, the content type
1220 of the metadata instance MUST be `application/samlmetadata+xml`.

1221 The XML document provided at the well-known location MUST describe the metadata only for the entity
1222 represented by the unique identifier (that is, the root element MUST be an `<EntityDescriptor>` with
1223 an `entityID` matching the location). If other entities need to be described, the
1224 `<AdditionalMetaLocation>` element MUST be used. Thus the `<EntitiesDescriptor>` element
1225 MUST NOT be used in documents published using this mechanism, since a group of entities are not
1226 defined by such an identifier.

1227 4.1.2 Resolution

1228 If an entity's unique identifier is a URL, metadata consumers MAY attempt to resolve an entity's unique
1229 identifier directly, in a scheme-specific manner, by dereferencing the identifier.

1230 4.2 Publishing and Resolution via DNS

1231 To improve the accessibility of metadata documents and provide additional indirection between an entity's
1232 unique identifier and the location of metadata, entities MAY publish their metadata document locations in a
1233 zone of their corresponding DNS [RFC1034]. The entity's unique identifier (a URI) is used as the input to
1234 the process. Since URIs are flexible identifiers, location publication methods and the resolution process
1235 are determined by the URI's scheme and fully-qualified name. URI locations for metadata are

1236 subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in [\[RFC2915\]](#)
1237 and [\[RFC3403\]](#).

1238 It is RECOMMENDED that entities publish their resource records in signed zone files using [\[RFC2535\]](#)
1239 such that relying parties may establish the validity of the published location and authority of the zone, and
1240 integrity of the DNS response. If DNS zone signatures are present, relying parties MUST properly validate
1241 the signature.

1242 **4.2.1 Publication**

1243 This specification makes use of the NAPTR resource record described in [\[RFC2915\]](#) and [\[RFC3403\]](#).
1244 Familiarity with these documents is encouraged.

1245 Dynamic Delegation Discovery System (DDDS) [\[RFC3401\]](#) is a general purpose system for the retrieval of
1246 information based on an application-specific input string and the application of well known rules to
1247 transform that string until a terminal condition is reached requiring a look-up into an application-specific
1248 defined database or resolution of a URL based on the rules defined by the application. DDDS defines a
1249 specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS
1250 necessary to apply DDDS rules.

1251 Entities MAY publish separate URLs when multiple metadata documents need to be distributed, or when
1252 different metadata documents are required due to multiple trust relationships that require separate keying
1253 material, or when service interfaces require separate metadata declarations. This may be accomplished
1254 through the use of the optional `<AdditionalMetaLocation>` element, or through the regexp facility and
1255 multiple service definition fields in the NAPTR resource record itself.

1256 If the publishing protocol permits MIME-based identification of content types, the content type of the
1257 metadata instance MUST be `application/samlmetadata+xml`.

1258 If the entity's unique identifier is a URN, publication of the corresponding metadata location proceeds as
1259 specified in [\[RFC3404\]](#). Otherwise, the resolution of the metadata location proceeds as specified below.

1260 The following is the application-specific profile of DDDS for SAML metadata resolution.

1261 **4.2.1.1 First Well Known Rule**

1262 The "first well-known-rule" for processing SAML metadata resolution is to parse the entity's unique
1263 identifier and extract the fully-qualified domain name (subexpression 3) as described in Section "[Parsing](#)
1264 [the providerID](#)".

1265 **4.2.1.2 The Order Field**

1266 The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY
1267 provide multiple NAPTR resource records which MUST be processed by the resolver application in the
1268 order indicated by this field.

1269 **4.2.1.3 The Preference Field**

1270 For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving
1271 application. The resolving application MAY ignore this order, in cases where the service field value does
1272 not meet the resolver's requirements (e.g.: the resource record returns a protocol the application does not
1273 support).

1274 4.2.1.4 The Flag Field

1275 SAML metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying
1276 additional resource records are to be processed). The "U" flag indicates that the output of the rule is a
1277 URI.

1278 4.2.1.5 The Service Field

1279 The SAML-specific service field, as described in the following BNF, declares the modes by which instance
1280 document(s) shall be made available:

```
1281 servicefield = 1("PID2U" / "NID2U") "+" proto [*(":" class) *(":" servicetype)]
1282 proto = 1("https" / "uddi")
1283 class = 1[ "entity" / "entitygroup" ]
1284 servicetype = 1(si / "spssso" / "idpssso" / "authn" / "authnauth" / "pdp" / "attrauth" /
1285 alphanum )
1286 si = "si" [ ":" alphanum ] [ ":" endpoint" ]
1287 alphanum = 1*32( ALPHA / DIGIT )
```

1288 where:

- 1289 • servicefield PID2U resolves an entity's unique identifier to metadata URL.
- 1290 • servicefield NID2U resolves a principal's <NameIdentifier> into a metadata URL.
- 1291 • proto describes the retrieval protocol (https or uddi). In the case of UDDI, the URL will be an
1292 http(s) URL referencing a WSDL document.
- 1293 • class identifies whether the referenced metadata document describes a single entity, or multiple.
1294 In the latter case, the referenced document MUST contain the entity defined by the original unique
1295 identifier as a member of a group of entities within the document itself such as an
1296 <AffiliationDescriptor> or <EntitiesDescriptor>.
- 1297 • servicetype allows an entity to publish metadata for distinct roles and services as separate
1298 documents. Resolvers who encounter multiple servicetype declarations will dereference the
1299 appropriate URI, depending on which service is required for an operation (e.g.: an entity operating
1300 both as an identity provider and a service provider can publish metadata for each role at different
1301 locations). The authn service type represents a <SingleSignOnService> endpoint.
- 1302 • si (with optional endpoint component) allows the publisher to either directly publish the metadata
1303 for a service instance, or by articulating a SOAP endpoint (using endpoint).

1304 For example:

- 1305 • PID2U+https:entity - represents the entity's complete metadata document available via the
1306 https protocol
- 1307 • PID2U+uddi:entity:si:foo - represents the WSDL document location that describes a service
1308 instance "foo"
- 1309 • PID2U+https:entitygroup:idpssso - represents the metadata for a group of entities acting as
1310 SSO identity providers, of which the original entity is a member.
- 1311 • NID2U+https:idp - represents the metadata for the SSO identity provider of a principal

1312 4.2.1.6 The Regex and Replacement Fields

1313 The expected output after processing the input string through the regex MUST be a valid https URL or
1314 UDDI node (WSDL document) address.

1315 4.2.2 NAPTR Examples

1316 4.2.2.1 Entity Metadata NAPTR Examples

1317 Entities publish metadata URLs in the following manner:

```
1318 $ORIGIN provider.biz
1319
1320 ;; order pref f service regexp or replacement
1321
1322 IN NAPTR 100 10 "U" PID2U+https:entity
1323     "!^.*$!https://host.provider.biz/some/directory/trust.xml!" ""
1324 IN NAPTR 110 10 "U" PID2U+https: entity:trust
1325     "!^.*$!https://foo.provider.biz:1443/mdtrust.xml!" ""
1326 IN NAPTR 125 10 "U" PID2U+https:"
1327 IN NAPTR 110 10 "U" PID2U+uddi:entity
1328     "!^.*$!https://this.uddi.node.provider.biz/libmd.wsdl" ""
```

1329 4.2.2.2 Name Identifier Examples

1330 A principal's employer `example.int` operates an identity provider which may be used by an office supply
1331 company to authenticate authorized buyers. The supplier takes a users' email address
1332 `buyer@example.int` as input to the resolution process, and parses the email address to extract the
1333 FQDN (`example.int`). The employer publishes the following NAPTR record in the `example.int` DNS:

```
1334 $ORIGIN example.int
1335
1336 IN NAPTR 100 10 "U" NID2U+https:authn
1337     "!^([\^@]+)@(.*)$!https://serv.example.int:8000/cgi-bin/getmd?\1!" ""
1338 IN NAPTR 100 10 "U" NID2U+https:idp
1339     "!^([\^@]+)@(.*)$!https://auth.example.int/app/auth?\1" ""
```

1340 4.2.3 Resolution

1341 When resolving metadata for an entity via the DNS, the unique identifier of the entity is used as the initial
1342 input into the resolution process, rather than as an actual location Proceed as follows:

- 1343 • If the unique identifier is a URN, proceed with the resolution steps as defined in [\[RFC3404\]](#).
- 1344 • Otherwise, parse the identifier to obtain the fully-qualified domain name.
- 1345 • Query the DNS for NAPTR resource records of the domain iteratively until a terminal resource
1346 record is returned.
- 1347 • Identify which resource record to use based on the service fields, then order fields, then preference
1348 fields of the result set.
- 1349 • Obtain the document(s) at the provided location(s) as required by the application.

1350 4.2.3.1 Parsing the Unique Identifier

1351 To initiate the resolution of the location of the metadata information, it will be necessary in some cases to
1352 decompose the entity's unique identifier (expressed as a URI) into one or more atomic elements.

1353 The following regular expression should be used when initiating the decomposition process:

```
1354 ^([\^:/?#]+)?/*([\^:/?#]*@)?(((\^/??:#*\.)*((\^/?#:\.]+\.)?([\^/?#:\.]+)))?(:\d+)?
1355 ([\^?#]*)(\?[\^#]*)?(#.*)?$
1356     1           2           34           56           7           8           9
1357 10          11
```

1358 Subexpression 3 MUST result in a Fully-Qualified Domain Name (FQDN), which will be the basis for
1359 retrieving metadata locations from this zone.

1360 **4.2.3.2 Obtaining Metadata via the DNS**

1361 Upon completion of the parsing of the identifier, the application then performs a DNS query for the resulting
1362 domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses.
1363 Applications MAY exclude from the result set any service definitions that do not concern the present
1364 request operations.

1365 Resolving applications MUST subsequently order the result set according to the order field, and MAY
1366 order the result set based on the preference set. Resolvers are NOT REQUIRED to follow the ordering of
1367 the preferences field. The resulting NAPTR resource record(s) are operated on iteratively (based on the
1368 order flag) until a terminal NAPTR resource record is reached.

1369 The result will be a well-formed, absolute URL, which is then used to retrieve the metadata document.

1370 **4.2.4 Metadata Location Caching**

1371 Location caching MUST NOT exceed the TTL of the DNS zone from which the location was derived.
1372 Resolvers MUST obtain a fresh copy of the metadata location upon reaching the expiration of the TTL of
1373 the zone.

1374 Publishers of metadata documents should carefully consider the TTL of the zone when making changes
1375 to metadata document locations. Should such a location change occur, a publisher MUST either keep the
1376 document at both the old and new location until all conforming resolvers are certain to have the updated
1377 location (e.g.: time of zone change + TTL), or provide an HTTP Redirect [RFC2616] response at the old
1378 location specifying the new location.

1379 **4.3 Post-Processing of Metadata**

1380 The following sections describe the post-processing of metadata.

1381 **4.3.1 Metadata Instance Caching**

1382 Document caching MUST NOT exceed the `validUntil` or `cacheDuration` attribute of the subject
1383 element(s). If metadata elements have parent elements which contain caching policies, the parent
1384 element takes precedence.

1385 To properly process the `cacheDuration` attribute, consumers MUST retain the date and time when the
1386 document was retrieved.

1387 When a document or element has expired, the consumer MUST retrieve a fresh copy, which may require
1388 a refresh of the document location(s). Consumers SHOULD process document cache processing
1389 according to [RFC2616] Section 13, and MAY request the Last-Modified date and time from the HTTP
1390 server. Publishers SHOULD ensure acceptable cache processing as described in [RFC2616] (Section
1391 10.3.5 304 Not Modified).

1392 **4.3.2 Handling of HTTPS Redirects**

1393 Publishers MAY issue an HTTP Redirect (301 Moved Permanently, 302 or 307 Temporary Redirect)
1394 [RFC2616], and user agents MUST follow the specified URL in the Redirect response. Redirects
1395 SHOULD be of the same protocol as the initial request.

1396 **4.3.3 Processing of XML Signatures and General Trust Processing**

1397 Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and
1398 for the trust ascribed to the entity described by such metadata:

- 1399 • Trust derived from the signature of the DNS zone from which the metadata location URL was

1400 resolved, ensuring accuracy of the metadata document location(s)

- 1401 • Trust derived from signature processing of the metadata document itself, ensuring the integrity of
- 1402 the XML document
- 1403 • Trust derived from the SSL/TLS server authentication of the metadata location URL, ensuring the
- 1404 identity of the publisher of the metadata

1405 Post-processing of the metadata document MUST include signature processing at the XML-document

1406 level and MAY include one of the other two processes. Specifically, the relying party MAY choose to trust

1407 any of the cited authorities in the resolution and parsing process. Publishers of metadata MUST employ a

1408 document-integrity mechanism and MAY employ any of the other two processing profiles to establish trust

1409 in the metadata document, governed by implementation policies.

1410 **4.3.3.1 Processing Signed DNS Zones**

1411 Verification of DNS zone signature SHOULD be processed, if present, as described in [\[RFC2535\]](#).

1412 **4.3.3.2 Processing Signed Documents and Fragments**

1413 Published metadata documents SHOULD be signed, as described in Section 3, either by a certificate

1414 issued to the subject of the document, or by another trusted party. Publishers MAY consider signatures of

1415 other parties as a means of trust conveyance.

1416 Metadata consumers MUST validate signatures, when present, on the metadata document as described

1417 by Section 3.

1418 **4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL**

1419 It is STRONGLY RECOMMENDED that publishers implement TLS/SSL URLs; therefore, consumers

1420 SHOULD consider the trust inherited from the issuer of the TLS/SSL certificate. Publication URLs may not

1421 always be located in the domain of the subject of the metadata document; therefore, consumers SHOULD

1422 NOT presume certificates whose subject is the entity in question, as it may be hosted by another trusted

1423 party.

1424 As the basis of this trust may not be available against a cached document, other mechanisms SHOULD

1425 be used under such circumstances.

5 References

1426

- 1427 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
1428 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1429 **[SAMLBind]** S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language*
1430 *(SAML) V2.0*. OASIS SSTC, January 2005. Document ID sstc-saml-bindings-2.0-
1431 cd-04. See <http://www.oasis-open.org/committees/security/>.
- 1432 **[SAMLConform]** P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion*
1433 *Markup Language (SAML) V2.0*. OASIS SSTC, January 2005. Document ID sstc-
1434 saml-conformance-2.0-cd-04. <http://www.oasis-open.org/committees/security/>.
- 1435 **[SAMLCore]** S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion*
1436 *Markup Language (SAML) V2.0*. OASIS SSTC, January 2005. Document ID sstc-
1437 saml-core-2.0-cd-04. See <http://www.oasis-open.org/committees/security/>.
- 1438 **[SAMLMeta-xsd]** S. Cantor et al., SAML metadata schema. OASIS SSTC, January 2005.
1439 Document ID sstc-saml-schema-metadata-2.0. See [http://www.oasis-
1440 open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1441 **[SAMLProf]** S. Cantor et al., *Profiles for the OASIS Security Assertion Markup Language*
1442 *(SAML) V2.0*. OASIS SSTC, January 2005. Document ID sstc-saml-profiles-2.0-
1443 cd-04. See <http://www.oasis-open.org/committees/security/>.
- 1444 **[SAMLSec]** F. Hirsch et al., *Security and Privacy Considerations for the OASIS Security*
1445 *Assertion Markup Language (SAML) V2.0*. OASIS SSTC, January 2005.
1446 Document ID sstc-saml-sec-consider-2.0-cd-04. See [http://www.oasis-
1447 open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1448 **[XMLEnc]** D. Eastlake et al., XML-Encryption Syntax and Processing,
1449 <http://www.w3.org/TR/xmlenc-core/>, World Wide Web Consortium.
- 1450 **[XMLSig]** D. Eastlake et al., XML-Signature Syntax and Processing,
1451 <http://www.w3.org/TR/xmlsig-core/>, World Wide Web Consortium.

1452 **Appendix A.Registration of MIME media type**
1453 **application/samlmetadata+xml**

1454 **Introduction**

1455 The IESG has approved a request to register the "application/samlmetadata+xml" MIME
1456 media type in the standards tree. This media type is a product of the Organization for the
1457 Advancement of Structured Information Systems (OASIS).
1458

1459 The IESG contact persons are Ted Hardie and Scott Hollenbeck.

1460 **MIME media type name**

1461 application

1462 **MIME subtype name**

1463 samlmetadata+xml

1464 **Required parameters**

1465 None

1466 **Optional parameters**

1467 charset

1468 Same as charset parameter of application/xml [RFC3023].

1469 **Encoding considerations**

1470 Same as for application/xml [RFC3023].

1471 **Security considerations**

1472 Per their specification, samlmetadata+xml typed objects do not contain executable content.
1473 However, these objects are XML-based [XML], and thus they have all of the general security
1474 considerations presented in section 10 of [RFC3023].

1475 SAML metadata [SAMLv2Meta] contains information whose integrity and authenticity is important
1476 – identity provider and service provider public keys and endpoint addresses, for example.

1477 To counter potential issues, the publisher may sign samlmetadata+xml typed objects. Any such
1478 signature should be verified by the recipient of the data - both as a valid signature, and as being
1479 the signature of the publisher.

1480 Additionally, various of the publication protocols, e.g. HTTP-over-TLS/SSL, offer means for
1481 ensuring the authenticity of the publishing party and for protecting the metadata in transit.
1482 [SAMLv2Meta] also defines prescriptive metadata caching directives, as well as guidance on
1483 handling HTTPS redirects, trust processing, server authentication, and related items.

1484 For a more detailed discussion of SAML v2.0 metadata and its security considerations, please
1485 see [SAMLv2Meta]. For a discussion of overall SAML v2.0 security considerations and specific
1486 security-related design features, please refer to the SAML v2.0 specifications listed in the below
1487 bibliography. The specifications containing security-specific information are explicitly listed.

1488 Interoperability considerations

1489 SAML v2.0 metadata explicitly supports identifying the protocols and versions supported by the
1490 identified entities. For example, an identity provider entity can be denoted as supporting SAML
1491 v2.0 [SAMLv2.0], SAML v1.1 [SAMLv1.1], Liberty ID-FF 1.2 [LAPFF], or even other protocols if
1492 they are unambiguously identifiable via URI [RFC2396]. This protocol support information is
1493 conveyed via the `protocolSupportEnumeration` attribute of metadata objects of the
1494 **RoleDescriptorType**.

1495 Published specification

1496 [SAMLv2Meta] explicitly specifies use of the `application/samlmetadata+xml` MIME media
1497 type.

1498 Applications which use this media type

1499 Potentially any application implementing SAML v2.0, as well as those applications implementing
1500 specifications based on SAML, e.g. those available from the Liberty Alliance [LAP].

1501 Additional information

1502 Magic number(s)

1503 In general, the same as for `application/xml` [RFC3023]. In particular, the XML root element of
1504 the returned object will be one of `<md:EntityDescriptor>`,
1505 `<md:AffiliationDescriptor>`, or `<md:EntitiesDescriptor>`, where "md" maps to the
1506 SAML v2.0 metadata namespace: `urn:oasis:names:tc:SAML:2.0:metadata`

1507 File extension(s)

1508 None

1509 Macintosh File Type Code(s)

1510 None

1511 Person & email address to contact for further information

1512 This registration is made on behalf of the OASIS Security Services Technical Committee (SSTC)
1513 Please refer to the SSTC website for current information on committee chairperson(s) and their
1514 contact addresses: <http://www.oasis-open.org/committees/security/>. Committee members should
1515 submit comments and potential errata to the securityservices@lists.oasis-open.org list. Others
1516 should submit them by filling out the web form located at [http://www.oasis-
1517 open.org/committees/comments/form.php?wg_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security).
1518

1519 Additionally, the SAML developer community email distribution list, [saml-dev@lists.oasis-
1520 open.org](mailto:saml-dev@lists.oasis-open.org), may be employed to discuss usage of the `application/samlmetadata+xml` MIME
1521 media type. The "saml-dev" mailing list is publicly archived here: [http://lists.oasis-
1522 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
1523 subscribe, send a message with the single word "subscribe" in the message body, to: [saml-dev-
1524 request@lists.oasis-open.org](mailto:saml-dev-request@lists.oasis-open.org).

1525 Intended usage

1526 COMMON

1527 Author/Change controller

1528 The SAML specification sets are a work product of the OASIS Security Services Technical
1529 Committee (SSTC). OASIS and the SSTC have change control over the SAML specification sets.

1530 Bibliography

- 1531 [LAP] “Liberty Alliance Project”. See <http://www.projectliberty.org/>
1532 [LAPFF] “Liberty Alliance Project: Federation Framework”. See
1533 <http://www.projectliberty.org/resources/specifications.php#box1>
- 1534 [OASIS] “Organization for the Advancement of Structured Information Systems”.
1535 See <http://www.oasis-open.org/>
- 1536 [RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifiers*
1537 *(URI): Generic Syntax*. IETF RFC 2396, August 1998. Available at
1538 <http://www.ietf.org/rfc/rfc2396.txt>
- 1539 [RFC3023] M. Murata, S. St.Laurent, D. Kohn, “XML Media Types”, IETF Request for
1540 Comments 3023, January 2001. Available as [http://www.rfc-](http://www.rfc-editor.org/rfc/rfc3023.txt)
1541 [editor.org/rfc/rfc3023.txt](http://www.rfc-editor.org/rfc/rfc3023.txt)
- 1542 [SAMLv1.1] OASIS Security Services Technical Committee, “Security Assertion
1543 Markup Language (SAML) Version 1.1 Specification Set”. OASIS
1544 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)
1545 [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)
- 1546 [SAMLv2.0] OASIS Security Services Technical Committee, “Security Assertion
1547 Markup Language (SAML) Version 2.0 Specification Set”. Committee
1548 Draft. Available at <http://www.oasis-open.org/committees/security/>
- 1549 [SAMLv2Bind] S. Cantor et al., “Bindings for the OASIS Security Assertion Markup
1550 Language (SAML) V2.0”. OASIS SSTC, September 2004. Document ID
1551 sstc-saml-bindings-2.0-cd-03. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1552 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
- 1553 [SAMLv2Core] S. Cantor et al., “Assertions and Protocols for the OASIS Security
1554 Assertion Markup Language (SAML) V2.0”. OASIS SSTC, September
1555 2004. Document ID sstc-saml-core-2.0-cd-03. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1556 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
- 1557 [SAMLv2Meta] S. Cantor et al., *Metadata for the OASIS Security Assertion Markup*
1558 *Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID
1559 sstc-saml-metadata-2.0-cd-03. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1560 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
- 1561 [SAMLv2Prof] S. Cantor et al., “Profiles for the OASIS Security Assertion Markup
1562 Language (SAML) V2.0”. OASIS SSTC, September 2004. Document ID
1563 sstc-saml-profiles-2.0-cd-03. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1564 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
- 1565 [SAMLv2Sec] F. Hirsch et al., “Security and Privacy Considerations for the OASIS
1566 Security Assertion Markup Language (SAML) V2.0”. OASIS SSTC,
1567 September 2004. Document ID sstc-saml-sec-consider-2.0-cd-03. See
1568 <http://www.oasis-open.org/committees/security/>
- 1569 [SSTC] “OASIS Security Services Technical Committee”. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1570 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
- 1571 [XML] Bray, T., Paoli, J., Sperberg-McQueen, C.M. and E. Maler, François
1572 Yergeau, “Extensible Markup Language (XML) 1.0 (Third Edition)”, World
1573 Wide Web Consortium Recommendation REC-xml, Feb 2004, Available
1574 as <http://www.w3.org/TR/REC-xml/>
1575

1576 Appendix B. Acknowledgments

1577 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
1578 Committee, whose voting members at the time of publication were:

- 1579 • Conor Cahill, AOL
- 1580 • John Hughes, Atos Origin
- 1581 • Hal Lockhart, BEA Systems
- 1582 • Mike Beach, Boeing
- 1583 • Rebekah Metz, Booz Allen Hamilton
- 1584 • Rick Randall, Booz Allen Hamilton
- 1585 • Ronald Jacobson, Computer Associates
- 1586 • Carolina Canales-Valenzuela, Ericsson
- 1587 • Dana Kaufman, Forum Systems
- 1588 • Irving Reid, Hewlett-Packard
- 1589 • Paula Austel, IBM
- 1590 • Michael McIntosh, IBM
- 1591 • Anthony Nadalin, IBM
- 1592 • Nick Ragouzis, Individual
- 1593 • Scott Cantor, Internet2
- 1594 • Bob Morgan, Internet2
- 1595 • Peter Davis, Neustar
- 1596 • Jeff Hodges, Neustar
- 1597 • Frederick Hirsch, Nokia
- 1598 • Senthil Sengodan, Nokia
- 1599 • Abbie Barbir, Nortel Networks
- 1600 • Scott Kiestler, Novell
- 1601 • Cameron Morris, Novell
- 1602 • Paul Madsen, NTT
- 1603 • Steve Anderson, OpenNetwork
- 1604 • Ari Kermaier, Oracle
- 1605 • Vamsi Motukuru, Oracle
- 1606 • Darren Platt, Ping Identity
- 1607 • Prateek Mishra, Principal Identity
- 1608 • Jim Lien, RSA Security
- 1609 • John Linn, RSA Security
- 1610 • Rob Philpott, RSA Security
- 1611 • Dipak Chopra, SAP
- 1612 • Jahan Moreh, Sigaba
- 1613 • Bhavna Bhatnagar, Sun Microsystems
- 1614 • Eve Maler, Sun Microsystems
- 1615 • Ronald Monzillo, Sun Microsystems
- 1616 • Emily Xu, Sun Microsystems
- 1617 • Greg Whitehead, Trustgenix

1618 The editors also would like to acknowledge the following people for their contributions to previous
1619 versions of the OASIS Security Assertions Markup Language Standard:

- 1620 • Stephen Farrell, Baltimore Technologies
- 1621 • David Orchard, BEA Systems
- 1622 • Krishna Sankar, Cisco Systems
- 1623 • Zahid Ahmed, CommerceOne
- 1624 • Carlisle Adams, Entrust
- 1625 • Tim Moses, Entrust
- 1626 • Nigel Edwards, Hewlett-Packard
- 1627 • Joe Pato, Hewlett-Packard
- 1628 • Bob Blakley, IBM
- 1629 • Marlena Erdos, IBM
- 1630 • Marc Chanliau, Netegrity
- 1631 • Chris McLaren, Netegrity
- 1632 • Lynne Rosenthal, NIST
- 1633 • Mark Skall, NIST
- 1634 • Simon Godik, Overxeer
- 1635 • Charles Norwood, SAIC
- 1636 • Evan Prodromou, Securant
- 1637 • Robert Griffin, RSA Security (former editor)
- 1638 • Sai Allarvarpu, Sun Microsystems
- 1639 • Chris Ferris, Sun Microsystems
- 1640 • Mike Myers, Traceroute Security
- 1641 • Phillip Hallam-Baker, VeriSign (former editor)
- 1642 • James Vanderbeek, Vodafone
- 1643 • Mark O'Neill, Vordel
- 1644 • Tony Palmer, Vordel

1645
1646 Finally, the editors wish to acknowledge the following people for their contributions of material used as
1647 input to the OASIS Security Assertions Markup Language specifications:

- 1648 • Thomas Gross, IBM
- 1649 • Birgit Pfitzmann, IBM

1650 Appendix C. Notices

1651 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
1652 might be claimed to pertain to the implementation or use of the technology described in this document or
1653 the extent to which any license under such rights might or might not be available; neither does it represent
1654 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
1655 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
1656 available for publication and any assurances of licenses to be made available, or the result of an attempt
1657 made to obtain a general license or permission for the use of such proprietary rights by implementors or
1658 users of this specification, can be obtained from the OASIS Executive Director.

1659 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
1660 other proprietary rights which may cover technology that may be required to implement this specification.
1661 Please address the information to the OASIS Executive Director.

1662 **Copyright © OASIS Open 2005. All Rights Reserved.**

1663 This document and translations of it may be copied and furnished to others, and derivative works that
1664 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
1665 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
1666 this paragraph are included on all such copies and derivative works. However, this document itself may
1667 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
1668 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
1669 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
1670 into languages other than English.

1671 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
1672 or assigns.

1673 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1674 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1675 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1676 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.