



---

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

5 **Committee Draft 03, 14 December 2004**

6 **Document identifier:**

7 sstc-saml-metadata-2.0-cd-03

8 **Location:**

9 [http://www.oasis-open.org/committees/documents.php?wg\\_abbrev=security](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 Paul Madsen, Entrust  
24 Irving Reid, Hewlett-Packard  
25 Paula Austel, IBM  
26 Maryann Hondo, IBM  
27 Michael McIntosh, IBM  
28 Tony Nadalin, IBM  
29 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 Charles Knouse, Oblix  
37 Steve Anderson, OpenNetwork  
38 Prateek Mishra, Principal Identity  
39 John Linn, RSA Security  
40 Rob Philpott, RSA Security  
41 Jahan Moreh, Sigaba  
42 Anne Anderson, Sun Microsystems  
43 Gary Ellison, Sun Microsystems

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

47 **Abstract:**

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

54 **Status:**

55 This is a **Committee Draft** approved by the Security Services Technical Committee on 14  
56 December 2004.

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

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

# 66 Table of Contents

67	1 Introduction.....	5
68	1.1 Notation.....	5
69	2 Metadata for SAML V2.0.....	6
70	2.1 Namespaces .....	6
71	2.2 Common Types.....	7
72	2.2.1 Simple Type entityIDType.....	7
73	2.2.2 Complex Type EndpointType.....	7
74	2.2.3 Complex Type IndexedEndpointType.....	8
75	2.2.4 Complex Type localizedNameType.....	8
76	2.2.5 Complex Type localizedURIType.....	9
77	2.3 Root Elements.....	9
78	2.3.1 Element <EntitiesDescriptor>.....	9
79	2.3.2 Element <EntityDescriptor>.....	10
80	2.3.2.1 Element <Organization>.....	12
81	2.3.2.2 Element <ContactPerson>.....	13
82	2.3.2.3 Element <AdditionalMetadataLocation>.....	14
83	2.4 Role Descriptor Elements.....	14
84	2.4.1 Element <RoleDescriptor>.....	14
85	2.4.1.1 Element <KeyDescriptor>.....	15
86	2.4.2 Complex Type SSODescriptorType.....	16
87	2.4.3 Element <IDPSSODescriptor>.....	17
88	2.4.4 Element <SPSSODescriptor>.....	18
89	2.4.4.1 Element <AttributeConsumingService>.....	19
90	2.4.4.2 Element <RequestedAttribute>.....	20
91	2.4.5 Element <AuthnAuthorityDescriptor>.....	20
92	2.4.6 Element <PDPDescriptor>.....	21
93	2.4.7 Element <AttributeAuthorityDescriptor>.....	21
94	2.5 Element <AffiliationDescriptor>.....	22
95	2.6 Examples.....	23
96	3 Signature Processing.....	27
97	3.1 XML Signature Profile.....	27
98	3.1.1 Signing Formats and Algorithms.....	27
99	3.1.2 References.....	27
100	3.1.3 Canonicalization Method.....	27
101	3.1.4 Transforms.....	29
102	3.1.5 KeyInfo.....	29
103	4 Metadata Publication and Resolution.....	30
104	4.1 Publication and Resolution via Well-Known Location.....	30
105	4.1.1 Publication.....	30
106	4.1.2 Resolution.....	30
107	4.2 Publishing and Resolution via DNS.....	30
108	4.2.1 Publication.....	31
109	4.2.1.1 First Well Known Rule.....	31
110	4.2.1.2 The Order Field.....	31
111	4.2.1.3 The Preference Field.....	31
112	4.2.1.4 The Flag Field.....	32
113	4.2.1.5 The Service Field.....	32

114	4.2.1.6 The Regex and Replacement Fields.....	32
115	4.2.2 NAPTR Examples.....	33
116	4.2.2.1 Entity Metadata NAPTR Examples.....	33
117	4.2.2.2 Name Identifier Examples.....	33
118	4.2.3 Resolution.....	33
119	4.2.3.1 Parsing the Unique Identifier.....	33
120	4.2.3.2 Obtaining Metadata via the DNS.....	34
121	4.2.4 Metadata Location Caching.....	34
122	4.3 Post-Processing of Metadata.....	34
123	4.3.1 Metadata Instance Caching.....	34
124	4.3.2 Handling of HTTPS Redirects.....	34
125	4.3.3 Processing of XML Signatures and General Trust Processing.....	34
126	4.3.3.1 Processing Signed DNS Zones.....	35
127	4.3.3.2 Processing Signed Documents and Fragments.....	35
128	4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL.....	35
129	5 References.....	36
130	Appendix A.Registration of MIME media type application/samlmetadata+xml.....	37
131	Appendix B. Acknowledgments.....	41
132	Appendix C. Notices.....	43

---

# 1 Introduction

133

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

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

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

## 1.1 Notation

144

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

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

149

150 `Example code listings appear like this.`

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

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

154

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

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

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

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

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

### 2.1 Namespaces

175

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

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

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

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

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

```
202     <documentation>
203         Document identifier: sstc-saml-schema-metadata-2.0
204         Location: http://www.oasis-
205 open.org/committees/documents.php?wg_abbrev=security
206         Revision history:
207         V2.0 CD-03 (December, 2004):
208         Schema for SAML metadata, first published in SAML 2.0.
209     </documentation>
210 </annotation>
211 ...
212 </schema>
```

## 213 2.2 Common Types

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

### 216 2.2.1 Simple Type `entityIDType`

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

222 The following schema fragment defines the `entityIDType` simple type:

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

### 228 2.2.2 Complex Type `EndpointType`

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

232 `Binding` [Required]

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

235 `Location` [Required]

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

238 `ResponseLocation` [Optional]

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

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

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

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

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

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

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

### 263 2.2.3 Complex Type IndexedEndpointType

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

267 **index** [Required]

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

272 **isDefault** [Optional]

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

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

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

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

### 288 2.2.4 Complex Type localizedNameType

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

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



297 </complexType>

## 298 2.2.5 Complex Type localizedURIType

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

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

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

## 309 2.3 Root Elements

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

### 313 2.3.1 Element <EntitiesDescriptor>

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

317 ID [Optional]

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

319 validUntil [Optional]

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

322 cacheDuration [Optional]

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

325 Name [Optional]

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

327 <ds:Signature> [Optional]

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

330 <Extensions> [Optional]

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

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

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

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

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

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

```
341 <element name="EntitiesDescriptor" type="md:EntitiesDescriptorType"/>
342 <complexType name="EntitiesDescriptorType">
343   <sequence>
344     <element ref="ds:Signature" minOccurs="0"/>
345     <element ref="md:Extensions" minOccurs="0"/>
346     <choice minOccurs="1" maxOccurs="unbounded">
347       <element ref="md:EntityDescriptor"/>
348       <element ref="md:EntitiesDescriptor"/>
349     </choice>
350   </sequence>
351   <attribute name="validUntil" type="dateTime" use="optional"/>
352   <attribute name="cacheDuration" type="duration" use="optional"/>
353   <attribute name="ID" type="ID" use="optional"/>
354   <attribute name="Name" type="string" use="optional"/>
355 </complexType>
356 <element name="Extensions" type="md:ExtensionsType"/>
357 <complexType final="#all" name="ExtensionsType">
358   <sequence>
359     <any namespace="##other" processContents="lax"
360     minOccurs="0" 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"
438 maxOccurs="unbounded"/>
439 <element ref="md:AdditionalMetadataLocation" minOccurs="0"
440 maxOccurs="unbounded"/>
441 </sequence>
442 <attribute name="entityID" type="md:entityIDType" use="required"/>
443 <attribute name="validUntil" type="dateTime" use="optional"/>
444 <attribute name="cacheDuration" type="duration" use="optional"/>
445 <attribute name="ID" type="ID" use="optional"/>
446 <anyAttribute namespace="##other" processContents="lax"/>
447 </complexType>

```

### 448 2.3.2.1 Element <Organization>

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

453 <Extensions> [Optional]

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

457 <OrganizationName> [One or More]

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

459 <OrganizationDisplayName> [One or More]

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

461 <OrganizationURL> [One or More]

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

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

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

```

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

```

### 481 2.3.2.2 Element <ContactPerson>

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

486 **contactType** [Required]

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

489 <Extensions> [Optional]

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

493 <Company> [Optional]

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

495 <GivenName> [Optional]

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

497 <SurName> [Optional]

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

499 <EmailAddress> [Zero or More]

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

502 <TelephoneNumber> [Zero or More]

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

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

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

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

```

530     <enumeration value="technical"/>
531     <enumeration value="support"/>
532     <enumeration value="administrative"/>
533     <enumeration value="billing"/>
534     <enumeration value="other"/>
535   </restriction>
536 </simpleType>

```

### 537 2.3.2.3 Element <AdditionalMetadataLocation>

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

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

```

545 <element name="AdditionalMetadataLocation"
546   type="md:AdditionalMetadataLocationType"/>
547 <complexType name="AdditionalMetadataLocationType">
548   <simpleContent>
549     <extension base="anyURI">
550       <attribute name="namespace" type="anyURI" use="required"/>
551     </extension>
552   </simpleContent>
553 </complexType>

```

## 554 2.4 Role Descriptor Elements

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

### 558 2.4.1 Element <RoleDescriptor>

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

563 ID [Optional]

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

565 validUntil [Optional]

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

568 cacheDuration [Optional]

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

571 protocolSupportEnumeration [Required]

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

576 ensure discrimination when necessary.

577 `errorURL` [Optional]

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

580 `<ds:Signature>` [Optional]

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

583 `<Extensions>` [Optional]

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

587 `<KeyDescriptor>` [Zero or More]

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

590 `<Organization>` [Optional]

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

593 `<ContactPerson>` [Zero or More]

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

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

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

```

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

```

#### 621 **2.4.1.1 Element `<KeyDescriptor>`**

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

625 use [Optional]

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

628 `<ds:KeyInfo>` [Required]

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

631 `<EncryptionMethod>` [Zero or More]

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

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

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

## 653 2.4.2 Complex Type **SSODescriptorType**

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

658 `<ArtifactResolutionService>` [Zero or More]

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

662 `<SingleLogoutService>` [Zero or More]

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

665 `<ManageNameIDService>` [Zero or More]

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

668 `<NameIDFormat>` [Zero or More]

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



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

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

### 693 **2.4.3 Element <IDPSSODescriptor>**

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

697 WantAuthnRequestsSigned [Optional]

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

700 <SingleSignOnService> [One or More]

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

704 <NameIDMappingService> [Zero or More]

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

708 <AssertionIDRequestService> [Zero or More]

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

712 <AttributeProfile> [Zero or More]

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

715 <saml:Attribute> [Zero or More]

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

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

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

#### 747 **2.4.4 Element <SPSSODescriptor>**

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

751 AuthnRequestsSigned [Optional]

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

754 WantAssertionsSigned [Optional]

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

759 <AssertionConsumerService> [One or More]

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

763 <AttributeConsumingService> [Zero or More]

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

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

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

```
770 <element name="SPSSODescriptor" type="md:SPSSODescriptorType"/>
```

```

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

```

#### 789 2.4.4.1 Element <AttributeConsumingService>

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

793 index [Required]

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

796 isDefault [Optional]

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

799 <ServiceName> [One or More]

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

801 <ServiceDescription> [Zero or More]

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

803 <RequestedAttribute> [One or More]

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

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

```

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

```

#### 821 **2.4.4.2 Element <RequestedAttribute>**

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

825 **isRequired** [Optional]

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

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

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

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

#### 840 **2.4.5 Element <AuthnAuthorityDescriptor>**

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

844 <AuthnQueryService> [One or More]

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

848 <AssertionIDRequestService> [Zero or More]

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

852 <NameIDFormat> [Zero or More]

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

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

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

```

869     </extension>
870     </complexContent>
871 </complexType>
872 <element name="AuthnQueryService" type="md:EndpointType"/>

```

## 873 2.4.6 Element <PDPDescriptor>

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

877 <AuthzService> [One or More]

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

881 <AssertionIDRequestService> [Zero or More]

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

885 <NameIDFormat> [Zero or More]

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

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

```

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

```

## 905 2.4.7 Element <AttributeAuthorityDescriptor>

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

910 <AttributeService> [One or More]

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

914 <AssertionIDRequestService> [Zero or More]

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

- 917 requests defined in [SAMLBind].
- 918 <NameIDFormat> [Zero or More]
- 919 Zero or more elements of type **anyURI** that enumerate the name identifier formats supported by  
920 this authority. See Section 8.3 of [SAMLCore] for some possible values for this element.
- 921 <AttributeProfile> [Zero or More]
- 922 Zero or more elements of type **anyURI** that enumerate the attribute profiles supported by this  
923 authority. See [SAMLProf] for some possible values for this element.
- 924 <saml:Attribute> [Zero or More]
- 925 Zero or more elements that identify the SAML attributes supported by the authority. Specific  
926 values MAY optionally be included, indicating that only certain values permitted by the attribute's  
927 definition are supported.

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

```

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

```

## 950 2.5 Element <AffiliationDescriptor>

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

- 957 affiliationOwnerID [Required]
- 958 Specifies the unique identifier of the entity responsible for the affiliation. The owner is NOT  
959 presumed to be a member of the affiliation; if it is a member, its identifier MUST also appear in an  
960 <AffiliateMember> element.
- 961 ID [Optional]
- 962 A document-unique identifier for the element, typically used as a reference point when signing.
- 963 validUntil [Optional]
- 964 Optional attribute indicates the expiration time of the metadata contained in the element and any  
965 contained elements.

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

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

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

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

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

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

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

```

986 <element name="AffiliationDescriptor"
987 type="md:AffiliationDescriptorType"/>
988 <complexType name="AffiliationDescriptorType">
989   <sequence>
990     <element ref="ds:Signature" minOccurs="0"/>
991     <element ref="md:Extensions" minOccurs="0"/>
992     <element ref="md:AffiliateMember" maxOccurs="unbounded"/>
993     <element ref="md:KeyDescriptor" minOccurs="0"
994 maxOccurs="unbounded"/>
995   </sequence>
996   <attribute name="affiliationOwnerID" type="md:entityIDType"
997 use="required"/>
998   <attribute name="validUntil" type="dateTime" use="optional"/>
999   <attribute name="cacheDuration" type="duration" use="optional"/>
1000   <attribute name="ID" type="ID" use="optional"/>
1001   <anyAttribute namespace="##other" processContents="lax"/>
1002 </complexType>
1003 <element name="AffiliateMember" type="md:entityIDType"/>

```

## 1004 2.6 Examples

1005 The following is an example of metadata for a SAML system entity acting as an identity provider and an  
 1006 attribute authority. A signature is shown as a placeholder, without the actual content.  
 1007

```

1008 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1009 xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1010 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1011 entityID="https://IdentityProvider.com/SAML">
1012   <ds:Signature>...</ds:Signature>
1013   <IDPSSODescriptor WantAuthnRequestsSigned="true"
1014 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1015     <KeyDescriptor use="signing">
1016       <ds:KeyInfo>
1017         <ds:KeyName>IdentityProvider.com SSO Key</ds:KeyName>

```

```

1018     </ds:KeyInfo>
1019 </KeyDescriptor>
1020 <ArtifactResolutionService isDefault="true" index="0"
1021   Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1022   Location="https://IdentityProvider.com/SAML/Artifact"/>
1023 <SingleLogoutService
1024   Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1025   Location="https://IdentityProvider.com/SAML/SLO/SOAP"/>
1026 <SingleLogoutService
1027   Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1028   Location="https://IdentityProvider.com/SAML/SLO/Browser"
1029   ResponseLocation="https://IdentityProvider.com/SAML/SLO/Response"/>
1030 <NameIDFormat>
1031   urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1032 </NameIDFormat>
1033 <NameIDFormat>
1034   urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1035 </NameIDFormat>
1036 <NameIDFormat>
1037   urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1038 </NameIDFormat>
1039 <SingleSignOnService
1040   Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1041   Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1042 <SingleSignOnService
1043   Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1044   Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1045 <saml:Attribute
1046   NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1047   Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1048   FriendlyName="eduPersonPrincipalName">
1049 </saml:Attribute>
1050 <saml:Attribute
1051   NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1052   Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1053   FriendlyName="eduPersonAffiliation">
1054   <saml:AttributeValue>member</saml:AttributeValue>
1055   <saml:AttributeValue>student</saml:AttributeValue>
1056   <saml:AttributeValue>faculty</saml:AttributeValue>
1057   <saml:AttributeValue>employee</saml:AttributeValue>
1058   <saml:AttributeValue>staff</saml:AttributeValue>
1059 </saml:Attribute>
1060 </IDPSSODescriptor>
1061 <AttributeAuthorityDescriptor
1062   protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1063   <KeyDescriptor use="signing">
1064     <ds:KeyInfo>
1065       <ds:KeyName>IdentityProvider.com AA Key</ds:KeyName>
1066     </ds:KeyInfo>
1067   </KeyDescriptor>
1068   <AttributeService
1069     Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1070     Location="https://IdentityProvider.com/SAML/AA/SOAP"/>
1071   <AssertionIDRequestService
1072     Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
1073     Location="https://IdentityProvider.com/SAML/AA/URI"/>
1074   <NameIDFormat>
1075     urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1076   </NameIDFormat>
1077   <NameIDFormat>
1078     urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1079   </NameIDFormat>
1080   <NameIDFormat>
1081     urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1082   </NameIDFormat>
1083   <saml:Attribute
1084     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"

```



```

1085     Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1086     FriendlyName="eduPersonPrincipalName">
1087 </saml:Attribute>
1088 <saml:Attribute
1089     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1090     Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1091     FriendlyName="eduPersonAffiliation">
1092     <saml:AttributeValue>member</saml:AttributeValue>
1093     <saml:AttributeValue>student</saml:AttributeValue>
1094     <saml:AttributeValue>faculty</saml:AttributeValue>
1095     <saml:AttributeValue>employee</saml:AttributeValue>
1096     <saml:AttributeValue>staff</saml:AttributeValue>
1097 </saml:Attribute>
1098 </AttributeAuthorityDescriptor>
1099 <Organization>
1100     <OrganizationName xml:lang="en">Identity Providers R US</OrganizationName>
1101     <OrganizationDisplayName xml:lang="en">
1102     Identity Providers R US, a Division of Lerxst Corp.
1103 </OrganizationDisplayName>
1104     <OrganizationURL xml:lang="en">https://IdentityProvider.com</OrganizationURL>
1105 </Organization>
1106 </EntityDescriptor>
1107

```

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

```

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

```

```
1152         FriendlyName="eduPersonEntitlement">
1153             <saml:AttributeValue>
1154                 https://ServiceProvider.com/entitlements/123456789
1155             </saml:AttributeValue>
1156         </RequestedAttribute>
1157     </AttributeConsumingService>
1158 </SPSSODescriptor>
1159 <Organization>
1160     <OrganizationName xml:lang="en">Academic Journals R US</OrganizationName>
1161     <OrganizationDisplayName xml:lang="en">
1162         Academic Journals R US, a Division of Dirk Corp.
1163     </OrganizationDisplayName>
1164     <OrganizationURL xml:lang="en">https://ServiceProvider.com</OrganizationURL>
1165 </Organization>
1166 </EntityDescriptor>
```

---

## 1167 3 Signature Processing

1168 Various elements in a metadata instance can be digitally signed (as indicated by the element's inclusion of  
1169 a `<ds:Signature>` element), with the following benefits:

- 1170 • Metadata integrity
- 1171 • Authentication of the metadata by a trusted signer

1172 A digital signature is not always required, for example if the relying party obtains the information directly  
1173 from the publishing entity directly (with no intermediaries) through a secure channel, with the entity having  
1174 authenticated to the relying party by some means other than a digital signature.

1175 Many different techniques are available for "direct" authentication and secure channel establishment  
1176 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,  
1177 the applicable security requirements depend on the communicating applications.

1178 Additionally, elements can inherit signatures on enclosing parent elements that are themselves signed.

1179 In the absence of such context, it is RECOMMENDED that at least the root element of a metadata  
1180 instance be signed.

### 1181 3.1 XML Signature Profile

1182 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility  
1183 and many choices. This section details the constraints on these facilities so that metadata processors do  
1184 not have to deal with the full generality of XML Signature processing. This usage makes specific use of  
1185 the **xs:ID**-typed attributes optionally present on the elements to which signatures can apply. These  
1186 attributes are collectively referred to in this section as the identifier attributes.

#### 1187 3.1.1 Signing Formats and Algorithms

1188 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and  
1189 detached.

1190 SAML metadata MUST use enveloped signatures when signing the elements defined in this specification.  
1191 SAML processors SHOULD support the use of RSA signing and verification for public key operations in  
1192 accordance with the algorithm identified by <http://www.w3.org/2000/09/xmlsig#rsa-sha1>.

#### 1193 3.1.2 References

1194 Signed metadata elements MUST supply a value for the identifier attribute on the signed element. The  
1195 element may or may not be the root element of the actual XML document containing the signed metadata  
1196 element.

1197 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute  
1198 value of the metadata element being signed. For example, if the identifier attribute value is "foo", then the  
1199 URI attribute in the `<ds:Reference>` element MUST be "#foo".

1200 As a consequence, a metadata element's signature MUST apply to the content of the signed element and  
1201 any child elements it contains.

#### 1202 3.1.3 Canonicalization Method

1203 SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the  
1204 `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a `<ds:Transform>`

1206 embedded in an XML context can be verified independent of that context.

### 1207 **3.1.4 Transforms**

1208 Signatures in SAML metadata SHOULD NOT contain transforms other than the enveloped signature  
1209 transform (with the identifier <http://www.w3.org/2000/09/xmlsig#enveloped-signature>) or the exclusive  
1210 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or  
1211 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

1212 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do  
1213 not, verifiers MUST ensure that no content of the signed metadata element is excluded from the  
1214 signature. This can be accomplished by establishing out-of-band agreement as to what transforms are  
1215 acceptable, or by applying the transforms manually to the content and reverifying the result as consisting  
1216 of the same SAML metadata.

### 1217 **3.1.5 KeyInfo**

1218 XML Signature [XMLSig] defines usage of the `<ds:KeyInfo>` element. SAML does not require the  
1219 use of `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY  
1220 be absent.

---

## 1221 4 Metadata Publication and Resolution

1222 Two mechanisms are provided for an entity to publish (and for a consumer to resolve the location of)  
1223 metadata documents: via a "well-known-location" by directly dereferencing the entity's unique identifier (a  
1224 URI variously referred to as an *entityID* or *providerID*), or indirectly by publishing the location of metadata  
1225 in the DNS. Other out-of-band mechanisms are of course also permitted. A consumer that supports both  
1226 approaches defined in this document MUST attempt resolution via DNS before using the "well-known-  
1227 location" mechanism.

1228 When retrieval requires network transport of the document, the transport SHOULD be protected with  
1229 mechanisms providing server authentication and integrity protection. For example, HTTP-based resolution  
1230 SHOULD be protected with TLS/SSL [RFC2246] as amended by [RFC3546].

1231 Various mechanisms are described in this section to aid in establishing trust in the accuracy and  
1232 legitimacy of metadata, including use of XML signatures, SSL/TLS server authentication, and DNS  
1233 signatures. Regardless of the mechanism(s) used, relying parties SHOULD have some means by which to  
1234 establish trust in metadata information before relying on it.

### 1235 4.1 Publication and Resolution via Well-Known Location

1236 The following sections describe publication and resolution of metadata by means of a well-known location.

#### 1237 4.1.1 Publication

1238 Entities MAY publish their metadata documents at a well known location by placing the document at the  
1239 location denoted by its unique identifier, which MUST be in the form of a URL (rather than a URN). See  
1240 Section 8.3.6 of [SAMLCore] for more information about such identifiers. It is STRONGLY  
1241 RECOMMENDED that `https` URLs be used for this purpose. An indirection mechanism supported by the  
1242 URL scheme (such as an HTTP 1.1 302 redirect) MAY be used if the document is not placed directly at  
1243 the location. If the publishing protocol permits MIME-based identification of content types, the content type  
1244 of the metadata instance MUST be `application/samlmetadata+xml`.

1245 The XML document provided at the well-known location MUST describe the metadata only for the entity  
1246 represented by the unique identifier (that is, the root element MUST be an `<EntityDescriptor>` with  
1247 an `entityID` matching the location). If other entities need to be described, the  
1248 `<AdditionalMetaLocation>` element MUST be used. Thus the `<EntitiesDescriptor>` element  
1249 MUST NOT be used in documents published using this mechanism, since a group of entities are not  
1250 defined by such an identifier.

#### 1251 4.1.2 Resolution

1252 If an entity's unique identifier is a URL, metadata consumers MAY attempt to resolve an entity's unique  
1253 identifier directly, in a scheme-specific manner, by dereferencing the identifier.

### 1254 4.2 Publishing and Resolution via DNS

1255 To improve the accessibility of metadata documents and provide additional indirection between an entity's  
1256 unique identifier and the location of metadata, entities MAY publish their metadata document locations in a  
1257 zone of their corresponding DNS [RFC1034]. The entity's unique identifier (a URI) is used as the input to  
1258 the process. Since URIs are flexible identifiers, location publication methods and the resolution process  
1259 are determined by the URI's scheme and fully-qualified name. URI locations for metadata are

1260 subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in [\[RFC2915\]](#)  
1261 and [\[RFC3403\]](#).

1262 It is RECOMMENDED that entities publish their resource records in signed zone files using [\[RFC2535\]](#)  
1263 such that relying parties may establish the validity of the published location and authority of the zone, and  
1264 integrity of the DNS response. If DNS zone signatures are present, relying parties MUST properly validate  
1265 the signature.

## 1266 **4.2.1 Publication**

1267 This specification makes use of the NAPTR resource record described in [\[RFC2915\]](#) and [\[RFC3403\]](#).  
1268 Familiarity with these documents is encouraged.

1269 Dynamic Delegation Discovery System (DDDS) [\[RFC3401\]](#) is a general purpose system for the retrieval of  
1270 information based on an application-specific input string and the application of well known rules to  
1271 transform that string until a terminal condition is reached requiring a look-up into an application-specific  
1272 defined database or resolution of a URL based on the rules defined by the application. DDDS defines a  
1273 specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS  
1274 necessary to apply DDDS rules.

1275 Entities MAY publish separate URLs when multiple metadata documents need to be distributed, or when  
1276 different metadata documents are required due to multiple trust relationships that require separate keying  
1277 material, or when service interfaces require separate metadata declarations. This may be accomplished  
1278 through the use of the optional `<AdditionalMetaLocation>` element, or through the regexp facility and  
1279 multiple service definition fields in the NAPTR resource record itself.

1280 If the publishing protocol permits MIME-based identification of content types, the content type of the  
1281 metadata instance MUST be `application/samlmetadata+xml`.

1282 If the entity's unique identifier is a URN, publication of the corresponding metadata location proceeds as  
1283 specified in [\[RFC3404\]](#). Otherwise, the resolution of the metadata location proceeds as specified below.

1284 The following is the application-specific profile of DDDS for SAML metadata resolution.

### 1285 **4.2.1.1 First Well Known Rule**

1286 The "first well-known-rule" for processing SAML metadata resolution is to parse the entity's unique  
1287 identifier and extract the fully-qualified domain name (subexpression 3) as described in Section "[Parsing](#)  
1288 [the providerID](#)".

### 1289 **4.2.1.2 The Order Field**

1290 The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY  
1291 provide multiple NAPTR resource records which MUST be processed by the resolver application in the  
1292 order indicated by this field.

### 1293 **4.2.1.3 The Preference Field**

1294 For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving  
1295 application. The resolving application MAY ignore this order, in cases where the service field value does  
1296 not meet the resolver's requirements (e.g.: the resource record returns a protocol the application does not  
1297 support).

#### 1298 4.2.1.4 The Flag Field

1299 SAML metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying  
1300 additional resource records are to be processed). The "U" flag indicates that the output of the rule is a  
1301 URI.

#### 1302 4.2.1.5 The Service Field

1303 The SAML-specific service field, as described in the following BNF, declares the modes by which instance  
1304 document(s) shall be made available:

```
1305 servicefield = 1("PID2U" / "NID2U") "+" proto [*(":" class) *(":" servicetype)]  
1306 proto = 1("https" / "uddi")  
1307 class = 1[ "entity" / "entitygroup" ]  
1308 servicetype = 1(si / "spssso" / "idpssso" / "authn" / "authnauth" / "pdp" / "attrauth" /  
1309 alphanum )  
1310 si = "si" [ ":" alphanum ] [ ":" endpoint ]  
1311 alphanum = 1*32 (ALPHA / DIGIT)
```

1312 where:

- 1313 • servicefield PID2U resolves an entity's unique identifier to metadata URL.
- 1314 • servicefield NID2U resolves a principal's <NameIdentifier> into a metadata URL.
- 1315 • proto describes the retrieval protocol (https or uddi). In the case of UDDI, the URL will be an  
1316 http(s) URL referencing a WSDL document.
- 1317 • class identifies whether the referenced metadata document describes a single entity, or multiple.  
1318 In the latter case, the referenced document MUST contain the entity defined by the original unique  
1319 identifier as a member of a group of entities within the document itself such as an  
1320 <AffiliationDescriptor> or <EntitiesDescriptor>.
- 1321 • servicetype allows an entity to publish metadata for distinct roles and services as separate  
1322 documents. Resolvers who encounter multiple servicetype declarations will dereference the  
1323 appropriate URI, depending on which service is required for an operation (e.g.: an entity operating  
1324 both as an identity provider and a service provider can publish metadata for each role at different  
1325 locations). The authn service type represents a <SingleSignOnService> endpoint.
- 1326 • si (with optional endpoint component) allows the publisher to either directly publish the metadata  
1327 for a service instance, or by articulating a SOAP endpoint (using endpoint).

1328 For example:

- 1329 • PID2U+https:entity - represents the entity's complete metadata document available via the  
1330 https protocol
- 1331 • PID2U+uddi:entity:si:foo - represents the WSDL document location that describes a service  
1332 instance "foo"
- 1333 • PID2U+https:entitygroup:idpssso - represents the metadata for a group of entities acting as  
1334 SSO identity providers, of which the original entity is a member.
- 1335 • NID2U+https:idp - represents the metadata for the SSO identity provider of a principal

#### 1336 4.2.1.6 The Regex and Replacement Fields

1337 The expected output after processing the input string through the regex MUST be a valid https URL or  
1338 UDDI node (WSDL document) address.

## 1339 4.2.2 NAPTR Examples

### 1340 4.2.2.1 Entity Metadata NAPTR Examples

1341 Entities publish metadata URLs in the following manner:

```
1342 $ORIGIN provider.biz
1343
1344 ;; order pref f service regexp or replacement
1345
1346 IN NAPTR 100 10 "U" PID2U+https:entity
1347     "!^.*$!https://host.provider.biz/some/directory/trust.xml!" ""
1348 IN NAPTR 110 10 "U" PID2U+https: entity:trust
1349     "!^.*$!https://foo.provider.biz:1443/mdtrust.xml!" ""
1350 IN NAPTR 125 10 "U" PID2U+https:"
1351 IN NAPTR 110 10 "U" PID2U+uddi:entity
1352     "!^.*$!https://this.uddi.node.provider.biz/libmd.wsdl" ""
```

### 1353 4.2.2.2 Name Identifier Examples

1354 A principal's employer `example.int` operates an identity provider which may be used by an office supply  
1355 company to authenticate authorized buyers. The supplier takes a users' email address  
1356 `buyer@example.int` as input to the resolution process, and parses the email address to extract the  
1357 FQDN (`example.int`). The employer publishes the following NAPTR record in the `example.int` DNS:

```
1358 $ORIGIN example.int
1359
1360 IN NAPTR 100 10 "U" NID2U+https:authn
1361     "!^([\^@]+)@(\.*)$!https://serv.example.int:8000/cgi-bin/getmd?\1!" ""
1362 IN NAPTR 100 10 "U" NID2U+https:idp
1363     "!^([\^@]+)@(\.*)$!https://auth.example.int/app/auth?\1" ""
```

## 1364 4.2.3 Resolution

1365 When resolving metadata for an entity via the DNS, the unique identifier of the entity is used as the initial  
1366 input into the resolution process, rather than as an actual location Proceed as follows:

- 1367 • If the unique identifier is a URN, proceed with the resolution steps as defined in [\[RFC3404\]](#).
- 1368 • Otherwise, parse the identifier to obtain the fully-qualified domain name.
- 1369 • Query the DNS for NAPTR resource records of the domain iteratively until a terminal resource  
1370 record is returned.
- 1371 • Identify which resource record to use based on the service fields, then order fields, then preference  
1372 fields of the result set.
- 1373 • Obtain the document(s) at the provided location(s) as required by the application.

### 1374 4.2.3.1 Parsing the Unique Identifier

1375 To initiate the resolution of the location of the metadata information, it will be necessary in some cases to  
1376 decompose the entity's unique identifier (expressed as a URI) into one or more atomic elements.

1377 The following regular expression should be used when initiating the decomposition process:

```
1378 ^([\^:/?#+:)?/*([\^:/?#]*@)?(((\^/?:#*\.)*((\^/?#:\.]+)\.([\^/?#:\.]+)))
1379 (: \d+)?([\^?#]*)(\?[\^#]*)?(#[.]*)?$
1380      1           2           3         4           5           6           7
1381      8           9          10          11
```

1382 Subexpression 3 MUST result in a Fully-Qualified Domain Name (FQDN), which will be the basis for  
1383 retrieving metadata locations from this zone.



#### 1384 **4.2.3.2 Obtaining Metadata via the DNS**

1385 Upon completion of the parsing of the identifier, the application then performs a DNS query for the resulting  
1386 domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses.  
1387 Applications MAY exclude from the result set any service definitions that do not concern the present  
1388 request operations.

1389 Resolving applications MUST subsequently order the result set according to the order field, and MAY  
1390 order the result set based on the preference set. Resolvers are NOT REQUIRED to follow the ordering of  
1391 the preferences field. The resulting NAPTR resource record(s) are operated on iteratively (based on the  
1392 order flag) until a terminal NAPTR resource record is reached.

1393 The result will be a well-formed, absolute URL, which is then used to retrieve the metadata document.

#### 1394 **4.2.4 Metadata Location Caching**

1395 Location caching MUST NOT exceed the TTL of the DNS zone from which the location was derived.  
1396 Resolvers MUST obtain a fresh copy of the metadata location upon reaching the expiration of the TTL of  
1397 the zone.

1398 Publishers of metadata documents should carefully consider the TTL of the zone when making changes  
1399 to metadata document locations. Should such a location change occur, a publisher MUST either keep the  
1400 document at both the old and new location until all conforming resolvers are certain to have the updated  
1401 location (e.g.: time of zone change + TTL), or provide an HTTP Redirect [RFC2616] response at the old  
1402 location specifying the new location.

### 1403 **4.3 Post-Processing of Metadata**

1404 The following sections describe the post-processing of metadata.

#### 1405 **4.3.1 Metadata Instance Caching**

1406 Document caching MUST NOT exceed the `validUntil` or `cacheDuration` attribute of the subject  
1407 element(s). If metadata elements have parent elements which contain caching policies, the parent  
1408 element takes precedence.

1409 To properly process the `cacheDuration` attribute, consumers MUST retain the date and time when the  
1410 document was retrieved.

1411 When a document or element has expired, the consumer MUST retrieve a fresh copy, which may require  
1412 a refresh of the document location(s). Consumers SHOULD process document cache processing  
1413 according to [RFC2616] Section 13, and MAY request the Last-Modified date and time from the HTTP  
1414 server. Publishers SHOULD ensure acceptable cache processing as described in [RFC2616] (Section  
1415 10.3.5 304 Not Modified).

#### 1416 **4.3.2 Handling of HTTPS Redirects**

1417 Publishers MAY issue an HTTP Redirect (301 Moved Permanently, 302 or 307 Temporary Redirect)  
1418 [RFC2616], and user agents MUST follow the specified URL in the Redirect response. Redirects  
1419 SHOULD be of the same protocol as the initial request.

#### 1420 **4.3.3 Processing of XML Signatures and General Trust Processing**

1421 Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and  
1422 for the trust ascribed to the entity described by such metadata:

- 1423 • Trust derived from the signature of the DNS zone from which the metadata location URL was

- 1424 resolved, ensuring accuracy of the metadata document location(s)
- 1425 • Trust derived from signature processing of the metadata document itself, ensuring the integrity of
  - 1426 the XML document
  - 1427 • Trust derived from the SSL/TLS server authentication of the metadata location URL, ensuring the
  - 1428 identity of the publisher of the metadata
- 1429 Post-processing of the metadata document MUST include signature processing at the XML-document
- 1430 level and MAY include one of the other two processes. Specifically, the relying party MAY choose to trust
- 1431 any of the cited authorities in the resolution and parsing process. Publishers of metadata MUST employ a
- 1432 document-integrity mechanism and MAY employ any of the other two processing profiles to establish trust
- 1433 in the metadata document, governed by implementation policies.

#### 1434 **4.3.3.1 Processing Signed DNS Zones**

1435 Verification of DNS zone signature SHOULD be processed, if present, as described in [\[RFC2535\]](#).

#### 1436 **4.3.3.2 Processing Signed Documents and Fragments**

1437 Published metadata documents SHOULD be signed, as described in Section 3, either by a certificate

1438 issued to the subject of the document, or by another trusted party. Publishers MAY consider signatures of

1439 other parties as a means of trust conveyance.

1440 Metadata consumers MUST validate signatures, when present, on the metadata document as described

1441 by Section 3.

#### 1442 **4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL**

1443 It is STRONGLY RECOMMENDED that publishers implement TLS/SSL URLs; therefore, consumers

1444 SHOULD consider the trust inherited from the issuer of the TLS/SSL certificate. Publication URLs may not

1445 always be located in the domain of the subject of the metadata document; therefore, consumers SHOULD

1446 NOT presume certificates whose subject is the entity in question, as it may be hosted by another trusted

1447 party.

1448 As the basis of this trust may not be available against a cached document, other mechanisms SHOULD

1449 be used under such circumstances.

---

## 5 References

1450

- 1451       **[RFC2119]**       S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,  
1452                   <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1453       **[SAMLBind]**       S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language*  
1454                   *(SAML) V2.0*. OASIS SSTC, December 2004. Document ID sstc-saml-bindings-  
1455                   2.0-cd-03. See <http://www.oasis-open.org/committees/security/>.
- 1456       **[SAMLConform]** P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion*  
1457                   *Markup Language (SAML) V2.0*. OASIS SSTC, December 2004. Document ID  
1458                   sstc-saml-conformance-2.0-cd-03. [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
1459                   [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1460       **[SAMLCore]**       S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion*  
1461                   *Markup Language (SAML) V2.0*. OASIS SSTC, December 2004. Document ID  
1462                   sstc-saml-core-2.0-cd-03. See <http://www.oasis-open.org/committees/security/>.
- 1463       **[SAMLMeta-xsd]** S. Cantor et al., *SAML metadata schema*. OASIS SSTC, December 2004.  
1464                   Document ID sstc-saml-schema-metadata-2.0. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
1465                   [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1466       **[SAMLProf]**       S. Cantor et al., *Profiles for the OASIS Security Assertion Markup Language*  
1467                   *(SAML) V2.0*. OASIS SSTC, December 2004. Document ID sstc-saml-profiles-  
1468                   2.0-cd-03. See <http://www.oasis-open.org/committees/security/>.
- 1469       **[SAMLSec]**       F. Hirsch et al., *Security and Privacy Considerations for the OASIS Security*  
1470                   *Assertion Markup Language (SAML) V2.0*. OASIS SSTC, December 2004.  
1471                   Document ID sstc-saml-sec-consider-2.0-cd-03. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
1472                   [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1473       **[XMLEnc]**       D. Eastlake et al., *XML-Encryption Syntax and Processing*,  
1474                   <http://www.w3.org/TR/xmlenc-core/>, World Wide Web Consortium.
- 1475       **[XMLSig]**       D. Eastlake et al., *XML-Signature Syntax and Processing*,  
1476                   <http://www.w3.org/TR/xmlsig-core/>, World Wide Web Consortium.

---

1477 **Appendix A.Registration of MIME media type**  
1478 **application/samlmetadata+xml**

1479 To: ietf-types@iana.org  
1480 Subject: Registration of MIME media type application/samlmetadata+xml  
1481

1482 Introduction

1483 This document defines a MIME media type -- application/samlmetadata+xml -- for use with the  
1484 XML serialization of Security Assertion Markup Language metadata.  
1485

1486 SAML is a work product of the OASIS Security Services Technical Committee [SSTC]. The SAML  
1487 specifications define XML-based constructs with which one may make, and convey, security  
1488 assertions. Using SAML, one can assert that an authentication event pertaining to some subject  
1489 has occurred and convey said assertion to a relying party, for example.  
1490

1491 SAML profiles require agreements between system entities regarding identifiers, binding support,  
1492 endpoints, certificates, keys, and so forth. Such information is treated as metadata by SAML v2.0.  
1493 [SAMLv2Meta] specifies this metadata, as well as specifying metadata publication and resolution  
1494 mechanisms. If the publishing protocol permits MIME-based identification of content types, then  
1495 use of the application/samlmetadata+xml MIME media type is required.  
1496

1497 MIME media type name: application  
1498

1499 MIME subtype name: samlmetadata+xml  
1500

1501 Required parameters: none  
1502

1503 Optional parameters: charset  
1504 Same as charset parameter of application/xml [RFC3023].  
1505

1506 Encoding considerations:  
1507 Same as for application/xml [RFC3023].  
1508

1509 Security considerations:  
1510 Per their specification, samlmetadata+xml typed objects do not contain executable content.  
1511 However, these objects are XML-based [XML], and thus they have all of the general security  
1512 considerations presented in section 10 of [RFC3023].  
1513

1514 SAML metadata [SAMLv2Meta] contains information whose integrity and authenticity is important  
1515 – identity provider and service provider public keys and endpoint addresses, for example.  
1516

1517 To counter potential issues, the publisher may sign samlmetadata+xml typed objects. Any such  
1518 signature should be verified by the recipient of the data - both as a valid signature, and as being  
1519 the signature of the publisher.  
1520

1521 Additionally, various of the publication protocols, e.g. HTTP-over-TLS/SSL, offer means for  
1522 ensuring the authenticity of the publishing party and for protecting the metadata in transit.  
1523 [SAMLv2Meta] also defines prescriptive metadata caching directives, as well as guidance on  
1524 handling HTTPS redirects, trust processing, server authentication, and related items.

1525 For a more detailed discussion of SAML v2.0 metadata and its security considerations, please  
1526 see [SAMLv2Meta]. For a discussion of overall SAML v2.0 security considerations and specific  
1527 security-related design features, please refer to the SAML v2.0 specifications listed in the below  
1528 bibliography. The specifications containing security-specific information are explicitly listed.  
1529

1530 Interoperability considerations:

1531 SAML v2.0 metadata explicitly supports identifying the protocols and versions supported by the  
1532 identified entities. For example, an identity provider entity can be denoted as supporting SAML  
1533 v2.0, SAML v1.1 [SAMLv1.1], Liberty ID-FF 1.2 [LAPFF], or even other protocols if they are  
1534 unambiguously identifiable via URI [RFC2396]. This protocol support information is conveyed via  
1535 the `protocolSupportEnumeration` attribute of metadata objects of the  
1536 `RoleDescriptorType`.  
1537

1538 Published specification:

1539 [SAMLv2Meta] explicitly specifies use of the `application/samlmetadata+xml` MIME media type.  
1540

1541 Applications which use this media type:

1542 Potentially any application implementing SAML v2.0, as well as those applications implementing  
1543 specifications based on SAML, e.g. those available from the Liberty Alliance [LAP].  
1544

1545 Additional information:

1547 Magic number(s):

1548 In general, the same as for `application/xml` [RFC3023]. In particular, the XML root  
1549 element of the returned object will have a namespace-qualified name with:  
1550

1551 – a local name of: `EntityDescriptor`, `AffiliationDescriptor`, or  
1552 `EntitiesDescriptor`

1554 – a namespace URI of: `urn:oasis:names:tc:SAML:2.0:metadata`  
1555 (the SAMLv2.0 metadata namespace)  
1556  
1557

1558 File extension(s): none  
1559

1560 Macintosh File Type Code(s): none  
1561

1562 Person & email address to contact for further information:

1563 This registration is made on behalf of the OASIS Security Services Technical Committee (SSTC)  
1564 Please refer to the SSTC website for current information on committee chairperson(s) and their  
1565 contact addresses: <http://www.oasis-open.org/committees/security/>. Committee members should  
1566 submit comments and potential errata to the [securityservices@lists.oasis-open.org](mailto:securityservices@lists.oasis-open.org) list. Others  
1567 should submit them by filling out the web form located at [http://www.oasis-  
1568 open.org/committees/comments/form.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security).  
1569

1570 Additionally, the SAML developer community email distribution list, [saml-dev@lists.oasis-  
1571 open.org](mailto:saml-dev@lists.oasis-open.org), may be employed to discuss usage of the `application/samlmetadata+xml` MIME media  
1572 type. The "saml-dev" mailing list is publicly archived here: [http://lists.oasis-  
1573 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  
1574 subscribe, send a message with the single word "subscribe" in the message body, to: [saml-dev-  
1575 request@lists.oasis-open.org](mailto:saml-dev-request@lists.oasis-open.org).  
1576

1577 Intended usage: COMMON  
1578  
1579

1580 Author/Change controller:  
1581 The SAML specification sets are a work product of the OASIS Security Services Technical  
1582 Committee (SSTC). OASIS and the SSTC have change control over the SAML specification sets.  
1583

#### 1584 Bibliography

- 1585
- 1586 [LAP] “*Liberty Alliance Project*”. See <http://www.projectliberty.org/>
- 1587
- 1588 [LAPFF] “*Liberty Alliance Project: Federation Framework*”. See  
1589 <http://www.projectliberty.org/resources/specifications.php#box1>  
1590
- 1591 [OASIS] “*Organization for the Advancement of Structured Information Systems*”.  
1592 See <http://www.oasis-open.org/>  
1593
- 1594 [RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifiers*  
1595 *(URI): Generic Syntax*. IETF RFC 2396, August 1998. Available at  
1596 <http://www.ietf.org/rfc/rfc2396.txt>  
1597
- 1598 [RFC3023] M. Murata, S. St.Laurent, D. Kohn, “*XML Media Types*”, IETF Request for  
1599 Comments 3023, January 2001. Available as [http://www.rfc-](http://www.rfc-editor.org/rfc/rfc3023.txt)  
1600 [editor.org/rfc/rfc3023.txt](http://www.rfc-editor.org/rfc/rfc3023.txt)  
1601
- 1602 [SAMLv1.1] OASIS Security Services Technical Committee, “*Security Assertion*  
1603 *Markup Language (SAML) Version 1.1 Specification Set*”. OASIS  
1604 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)  
1605 [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)  
1606
- 1607 [SAMLv2.0] OASIS Security Services Technical Committee, “*Security Assertion*  
1608 *Markup Language (SAML) Version 2.0 Specification Set*”. Committee  
1609 Draft. Available at <http://www.oasis-open.org/committees/security/>  
1610
- 1611 [SAMLv2Bind] S. Cantor et al., “*Bindings for the OASIS Security Assertion Markup*  
1612 *Language (SAML) V2.0*”. OASIS SSTC, September 2004. Document ID  
1613 sstc-saml-bindings-2.0-cd-03. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
1614 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)  
1615
- 1616 [SAMLv2Core] S. Cantor et al., “*Assertions and Protocols for the OASIS Security*  
1617 *Assertion Markup Language (SAML) V2.0*”. OASIS SSTC, September  
1618 2004. Document ID sstc-saml-core-2.0-cd-03. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
1619 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)  
1620
- 1621 [SAMLv2Meta] S. Cantor et al., “*Metadata for the OASIS Security Assertion Markup*  
1622 *Language (SAML) V2.0*”. OASIS SSTC, September 2004. Document ID  
1623 sstc-saml-metadata-2.0-cd-03. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
1624 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)  
1625
- 1626 [SAMLv2Prof] S. Cantor et al., “*Profiles for the OASIS Security Assertion Markup*  
1627 *Language (SAML) V2.0*”. OASIS SSTC, September 2004. Document ID  
1628 sstc-saml-profiles-2.0-cd-03. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
1629 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)  
1630
- 1631 [SAMLv2Sec] F. Hirsch et al., “*Security and Privacy Considerations for the OASIS*  
1632 *Security Assertion Markup Language (SAML) V2.0*”. OASIS SSTC,  
1633 September 2004. Document ID sstc-saml-sec-consider-2.0-cd-03. See  
1634 <http://www.oasis-open.org/committees/security/>

1635 [SSTC] "OASIS Security Services Technical Committee". See [http://www.oasis-](http://www.oasis-open.org/committees/security/)  
1636 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)  
1637  
1638 [XML] Bray, T., Paoli, J., Sperberg-McQueen, C.M. and E. Maler, François  
1639 Yergeau, "*Extensible Markup Language (XML) 1.0 (Third Edition)*", World  
1640 Wide Web Consortium Recommendation REC-xml, Feb 2004, Available  
1641 as <http://www.w3.org/TR/REC-xml/>  
1642

---

## 1643 Appendix B. Acknowledgments

1644 The editors would like to acknowledge the contributions of the OASIS Security Services Technical  
1645 Committee, whose voting members at the time of publication were:

- 1646 • Conor Cahill, AOL
- 1647 • John Hughes, Atos Origin
- 1648 • Hal Lockhart, BEA Systems
- 1649 • Mike Beach, Boeing
- 1650 • Rebekah Metz, Booz Allen Hamilton
- 1651 • Rick Randall, Booz Allen Hamilton
- 1652 • Ronald Jacobson, Computer Associates
- 1653 • Paul Madsen, Entrust
- 1654 • Dana Kaufman, Forum Systems
- 1655 • Paula Austel, IBM
- 1656 • Michael McIntosh, IBM
- 1657 • Anthony Nadalin, IBM
- 1658 • Nick Ragouzis, Individual
- 1659 • Scott Cantor, Internet2
- 1660 • Bob Morgan, Internet2
- 1661 • Peter Davis, Neustar
- 1662 • Jeff Hodges, Neustar
- 1663 • Frederick Hirsch, Nokia
- 1664 • John Kemp, Nokia
- 1665 • Abbie Barbir, Nortel Networks
- 1666 • Scott Kiestler, Novell
- 1667 • Cameron Morris, Novell
- 1668 • Charles Knouse, Oblix
- 1669 • Steve Anderson, OpenNetwork
- 1670 • Ari Kermaier, Oracle
- 1671 • Vamsi Motukuru, Oracle
- 1672 • Darren Platt, Ping Identity
- 1673 • Prateek Mishra, Principal Identity
- 1674 • Jim Lien, RSA Security
- 1675 • Rob Philpott, RSA Security
- 1676 • Dipak Chopra, SAP
- 1677 • Jahan Moreh, Sigaba
- 1678 • Bhavna Bhatnagar, Sun Microsystems
- 1679 • Eve Maler, Sun Microsystems
- 1680 • Ronald Monzillo, Sun Microsystems
- 1681 • Emily Xu, Sun Microsystems
- 1682 • Greg Whitehead, Trustgenix



1683

1684 The editors also would like to acknowledge the following people for their contributions to previous  
1685 versions of the OASIS Security Assertions Markup Language Standard:

- 1686 • Stephen Farrell, Baltimore Technologies
- 1687 • David Orchard, BEA Systems
- 1688 • Krishna Sankar, Cisco Systems
- 1689 • Zahid Ahmed, CommerceOne
- 1690 • Carlisle Adams, Entrust
- 1691 • Tim Moses, Entrust
- 1692 • Nigel Edwards, Hewlett-Packard
- 1693 • Joe Pato, Hewlett-Packard
- 1694 • Bob Blakley, IBM
- 1695 • Marlena Erdos, IBM
- 1696 • Marc Chanliau, Netegrity
- 1697 • Chris McLaren, Netegrity
- 1698 • Lynne Rosenthal, NIST
- 1699 • Mark Skall, NIST
- 1700 • Simon Godik, Overxeer
- 1701 • Charles Norwood, SAIC
- 1702 • Evan Prodromou, Securant
- 1703 • Robert Griffin, RSA Security (former editor)
- 1704 • Sai Allarvarpu, Sun Microsystems
- 1705 • Chris Ferris, Sun Microsystems
- 1706 • Emily Xu, Sun Microsystems
- 1707 • Mike Myers, Traceroute Security
- 1708 • Phillip Hallam-Baker, VeriSign (former editor)
- 1709 • James Vanderbeek, Vodafone
- 1710 • Mark O'Neill, Vordel
- 1711 • Tony Palmer, Vordel

1712

1713 Finally, the editors wish to acknowledge the following people for their contributions of material used as  
1714 input to the OASIS Security Assertions Markup Language specifications:

- 1715 • Thomas Gross, IBM
- 1716 • Birgit Pfitzmann, IBM

---

## 1717 Appendix C. Notices

1718 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
1719 might be claimed to pertain to the implementation or use of the technology described in this document or  
1720 the extent to which any license under such rights might or might not be available; neither does it represent  
1721 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to  
1722 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made  
1723 available for publication and any assurances of licenses to be made available, or the result of an attempt  
1724 made to obtain a general license or permission for the use of such proprietary rights by implementors or  
1725 users of this specification, can be obtained from the OASIS Executive Director.

1726 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or  
1727 other proprietary rights which may cover technology that may be required to implement this specification.  
1728 Please address the information to the OASIS Executive Director.

1729 **Copyright © OASIS Open 2004. All Rights Reserved.**

1730 This document and translations of it may be copied and furnished to others, and derivative works that  
1731 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and  
1732 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and  
1733 this paragraph are included on all such copies and derivative works. However, this document itself may  
1734 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as  
1735 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights  
1736 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it  
1737 into languages other than English.

1738 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
1739 or assigns.

1740 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
1741 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
1742 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR  
1743 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.