



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

5 **Committee Draft 02, 24 September 2004**

6 **Document identifier:**

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

8 **Location:**

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

10 **Editors:**

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

15 **SAML V2.0 Contributors:**

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

44 Ron Monzillo, Sun Microsystems
45 Greg Whitehead, Trustgenix

46 **Abstract:**

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

53 **Status:**

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

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

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

65 Table of Contents

66	1 Introduction.....	5
67	1.1 Notation.....	5
68	2 Metadata for SAML V2.0.....	6
69	2.1 Namespaces	6
70	2.2 Common Types.....	6
71	2.2.1 Simple Type entityIDType.....	7
72	2.2.2 Complex Type EndpointType.....	7
73	2.2.3 Complex Type IndexedEndpointType.....	7
74	2.2.4 Complex Type localizedNameType.....	8
75	2.2.5 Complex Type localizedURIType.....	8
76	2.3 Root Elements.....	8
77	2.3.1 Element <EntitiesDescriptor>.....	9
78	2.3.2 Element <EntityDescriptor>.....	10
79	2.3.2.1 Element <Organization>.....	11
80	2.3.2.2 Element <ContactPerson>.....	12
81	2.3.2.3 Element <AdditionalMetadataLocation>.....	13
82	2.4 Role Descriptor Elements.....	13
83	2.4.1 Element <RoleDescriptor>.....	14
84	2.4.1.1 Element <KeyDescriptor>.....	15
85	2.4.2 Complex Type SSODescriptorType.....	16
86	2.4.3 Element <IDPSSODescriptor>.....	16
87	2.4.4 Element <SPSSODescriptor>.....	17
88	2.4.5 Element <AuthnAuthorityDescriptor>.....	18
89	2.4.6 Element <PDPDescriptor>.....	18
90	2.4.7 Element <AttributeAuthorityDescriptor>.....	19
91	2.4.8 Element <AttributeConsumerDescriptor>.....	20
92	2.4.8.1 Element <AttributeConsumingService>.....	21
93	2.4.8.2 Element <RequestedAttribute>.....	21
94	2.5 Element <AffiliationDescriptor>.....	22
95	2.6 Examples.....	23
96	3 Signature Processing.....	26
97	3.1 XML Signature Profile.....	26
98	3.1.1 Signing Formats and Algorithms.....	26
99	3.1.2 References.....	26
100	3.1.3 Canonicalization Method.....	26
101	3.1.4 Transforms.....	27
102	3.1.5 KeyInfo.....	27
103	4 Metadata Publication and Resolution.....	28
104	4.1 Publication and Resolution via Well-Known Location.....	28
105	4.1.1 Publication.....	28
106	4.1.2 Resolution.....	28
107	4.2 Publishing and Resolution via DNS.....	28
108	4.2.1 Publication.....	29
109	4.2.1.1 First Well Known Rule.....	29
110	4.2.1.2 The Order Field.....	29
111	4.2.1.3 The Preference Field.....	29
112	4.2.1.4 The Flag Field.....	30

113	4.2.1.5 The Service Field.....	30
114	4.2.1.6 The Regex and Replacement Fields.....	30
115	4.2.2 NAPTR Examples.....	31
116	4.2.2.1 Entity Metadata NAPTR Examples.....	31
117	4.2.2.2 Name Identifier Examples.....	31
118	4.2.3 Resolution.....	31
119	4.2.3.1 Parsing the Unique Identifier.....	31
120	4.2.3.2 Obtaining Metadata via the DNS.....	32
121	4.2.4 Metadata Location Caching.....	32
122	4.3 Post-Processing of Metadata.....	32
123	4.3.1 Metadata Instance Caching.....	32
124	4.3.2 Handling of HTTPS Redirects.....	32
125	4.3.3 Processing of XML Signatures and General Trust Processing.....	32
126	4.3.3.1 Processing Signed DNS Zones.....	33
127	4.3.3.2 Processing Signed Documents and Fragments.....	33
128	4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL.....	33
129	5 References.....	34
130	6 Registration of MIME media type application/samlmetadata+xml.....	35
131	Appendix A. Acknowledgments.....	39
132	Appendix B. Notices.....	41

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.

1.1 Notation

142

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

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

147

148 `Example code listings appear like this.`

149 **Note:** Non-normative notes and explanations appear like this.

150 Conventional XML namespace prefixes are used throughout this specification to stand for their respective
151 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

152

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

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

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

2.1 Namespaces

163

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

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

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

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

```
168 <schema
169   targetNamespace="urn:oasis:names:tc:SAML:2.0:metadata"
170   xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
171   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
172   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
173   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
174   xmlns="http://www.w3.org/2001/XMLSchema"
175   elementFormDefault="unqualified"
176   attributeFormDefault="unqualified"
177   blockDefault="substitution"
178   version="2.0">
179   <import namespace="http://www.w3.org/2000/09/xmldsig#"
180     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
181 schema.xsd"/>
182   <import namespace="http://www.w3.org/2001/04/xmlenc#"
183     schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-
184 20021210/xenc-schema.xsd"/>
185   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
186     schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
187   <import namespace="http://www.w3.org/XML/1998/namespace"
188     schemaLocation="http://www.w3.org/2001/xml.xsd"/>
189   <annotation>
190     <documentation>
191       Document identifier: sstc-saml-schema-metadata-2.0
192       Location: http://www.oasis-
193 open.org/committees/documents.php?wg_abbrev=security
194       Revision history:
195       V2.0 (August, 2004):
196       Schema for SAML metadata, first published in SAML 2.0.
197     </documentation>
198   </annotation>
199   ...
200 </schema>
```

2.2 Common Types

201

203 These types are used in defining SAML V2.0 Metadata elements and attributes.

204 2.2.1 Simple Type entityIDType

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

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

```
211 <simpleType name="entityIDType">  
212   <restriction base="anyURI">  
213     <maxLength value="1024"/>  
214   </restriction>  
215 </simpleType>
```

216 2.2.2 Complex Type EndpointType

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

220 **Binding** [Required]

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

223 **Location** [Required]

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

226 **ResponseLocation** [Optional]

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

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

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

```
232 <complexType name="EndpointType">  
233   <sequence>  
234     <any namespace="##other" processContents="lax" minOccurs="0"  
235     maxOccurs="unbounded"/>  
236   </sequence>  
237   <attribute name="Binding" type="anyURI" use="required"/>  
238   <attribute name="Location" type="anyURI" use="required"/>  
239   <attribute name="ResponseLocation" type="anyURI" use="optional"/>  
240   <anyAttribute namespace="##other" processContents="lax"/>  
241 </complexType>
```

242 2.2.3 Complex Type IndexedEndpointType

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

246 index [Required]

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

249 isDefault [Optional]

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

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

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

```
257 <complexType name="IndexedEndpointType">  
258   <complexContent>  
259     <extension base="md:EndpointType">  
260       <attribute name="index" type="unsignedShort" use="required"/>  
261       <attribute name="isDefault" type="boolean" use="optional"/>  
262     </extension>  
263   </complexContent>  
264 </complexType>
```

265 2.2.4 Complex Type localizedNameType

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

```
268 <complexType name="localizedNameType">  
269   <simpleContent>  
270     <extension base="string">  
271       <attribute ref="xml:lang" use="required"/>  
272     </extension>  
273   </simpleContent>  
274 </complexType>
```

275 2.2.5 Complex Type localizedURIType

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

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

```
279 <complexType name="localizedURIType">  
280   <simpleContent>  
281     <extension base="anyURI">  
282       <attribute ref="xml:lang" use="required"/>  
283     </extension>  
284   </simpleContent>  
285 </complexType>
```

286 2.3 Root Elements

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

290 2.3.1 Element <EntitiesDescriptor>

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

294 ID [Optional]

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

296 validUntil [Optional]

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

299 cacheDuration [Optional]

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

302 Name [Optional]

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

304 <ds:Signature> [Optional]

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

307 <Extensions> [Optional]

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

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

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

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

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

```
318 <element name="EntitiesDescriptor" type="md:EntitiesDescriptorType"/>  
319 <complexType name="EntitiesDescriptorType">  
320   <sequence>  
321     <element ref="ds:Signature" minOccurs="0"/>  
322     <element ref="md:Extensions" minOccurs="0"/>  
323     <choice minOccurs="1" maxOccurs="unbounded">  
324       <element ref="md:EntityDescriptor"/>  
325       <element ref="md:EntitiesDescriptor"/>  
326     </choice>  
327   </sequence>  
328   <attribute name="validUntil" type="dateTime" use="optional"/>  
329   <attribute name="cacheDuration" type="duration" use="optional"/>  
330   <attribute name="ID" type="ID" use="optional"/>  
331   <attribute name="Name" type="string" use="optional"/>  
332 </complexType>  
333 <element name="Extensions" type="md:ExtensionsType"/>  
334 <complexType final="#all" name="ExtensionsType">  
335   <sequence>  
336     <any namespace="##other" processContents="lax"  
337     maxOccurs="unbounded"/>
```

338
339

```
</sequence>  
</complexType>
```

340 2.3.2 Element <EntityDescriptor>

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

- 345 • SSO Identity Provider
- 346 • SSO Service Provider
- 347 • Authentication Authority
- 348 • Attribute Authority
- 349 • Attribute Consumer
- 350 • Policy Decision Point
- 351 • Affiliation

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

353 `entityID` [Required]

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

356 `ID` [Optional]

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

358 `validUntil` [Optional]

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

361 `cacheDuration` [Optional]

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

364 `<ds:Signature>` [Optional]

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

367 `<Extensions>` [Optional]

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

371 `<RoleDescriptor>`, `<IDPSSODescriptor>`, `<SPSSODescriptor>`,
372 `<AuthnAuthorityDescriptor>`, `<AttributeAuthorityDescriptor>`,
373 `<AttributeConsumerDescriptor>`, `<PDPDescriptor>` [One or More]

374 **OR**

375 `<AffiliationDescriptor>` [Required]

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

378 <Organization> [Optional]
379 Optional element identifying the organization responsible for the SAML entity described by the
380 element.

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

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

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

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

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

```
393 <element name="EntityDescriptor" type="md:EntityDescriptorType"/>  
394 <complexType name="EntityDescriptorType">  
395   <sequence>  
396     <element ref="ds:Signature" minOccurs="0"/>  
397     <element ref="md:Extensions" minOccurs="0"/>  
398     <choice>  
399       <choice maxOccurs="unbounded">  
400         <element ref="md:RoleDescriptor"/>  
401         <element ref="md:IDPSSODescriptor"/>  
402         <element ref="md:SPSSODescriptor"/>  
403         <element ref="md:AuthnAuthorityDescriptor"/>  
404         <element ref="md:AttributeAuthorityDescriptor"/>  
405         <element ref="md:AttributeConsumerDescriptor"/>  
406         <element ref="md:PDPDescriptor"/>  
407       </choice>  
408       <element ref="md:AffiliationDescriptor"/>  
409     </choice>  
410     <element ref="md:Organization" minOccurs="0"/>  
411     <element ref="md:ContactPerson" minOccurs="0"  
412     maxOccurs="unbounded"/>  
413     <element ref="md:AdditionalMetadataLocation" minOccurs="0"  
414     maxOccurs="unbounded"/>  
415   </sequence>  
416   <attribute name="entityID" type="md:entityIDType" use="required"/>  
417   <attribute name="validUntil" type="dateTime" use="optional"/>  
418   <attribute name="cacheDuration" type="duration" use="optional"/>  
419   <attribute name="ID" type="ID" use="optional"/>  
420   <anyAttribute namespace="##other" processContents="lax"/>  
421 </complexType>
```

422 2.3.2.1 Element <Organization>

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

427 <Extensions> [Optional]
428 This contains optional metadata extensions that are agreed upon between a metadata publisher
429 and consumer. Extensions MUST NOT include global (non-namespace-qualified) elements or
430 elements qualified by a SAML-defined namespace within this element.

431 <OrganizationName> [One or More]
 432 One or more language-qualified names that may or may not be suitable for human consumption.

433 <OrganizationDisplayName> [One or More]
 434 One or more language-qualified names that are suitable for human consumption.

435 <OrganizationURL> [One or More]
 436 One or more language-qualified URIs that specify a location to which to direct a principal for
 437 additional information. Note that the language qualifier refers to the content of the material at the
 438 specified location.

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

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

```

442 <element name="Organization" type="md:OrganizationType"/>
443 <complexType name="OrganizationType">
444   <sequence>
445     <element ref="md:Extensions" minOccurs="0"/>
446     <element ref="md:OrganizationName" maxOccurs="unbounded"/>
447     <element ref="md:OrganizationDisplayName" maxOccurs="unbounded"/>
448     <element ref="md:OrganizationURL" maxOccurs="unbounded"/>
449   </sequence>
450   <anyAttribute namespace="##other" processContents="lax"/>
451 </complexType>
452 <element name="OrganizationName" type="md:localizedNameType"/>
453 <element name="OrganizationDisplayName" type="md:localizedNameType"/>
454 <element name="OrganizationURL" type="md:localizedURIType"/>

```

455 2.3.2.2 Element <ContactPerson>

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

460 contactType [Required]
 461 Specifies the type of contact using the **ContactTypeType** enumeration. The possible values are
 462 technical, support, administrative, billing, and other.

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

467 <Company> [Optional]
 468 Optional string element that specifies the name of the company for the contact person.

469 <GivenName> [Optional]
 470 Optional string element that specifies the given (first) name of the contact person.

471 <SurName> [Optional]
 472 Optional string element that specifies the surname of the contact person.

473 <EmailAddress> [Zero or More]
 474 Zero or more elements containing mailto: URIs representing e-mail addresses belonging to the
 475 contact person.

476 <TelephoneNumber> [Zero or More]

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

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

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

```
481 <element name="ContactPerson" type="md:ContactType"/>
482 <complexType name="ContactType">
483   <sequence>
484     <element ref="md:Extensions" minOccurs="0"/>
485     <element ref="md:Company" minOccurs="0"/>
486     <element ref="md:GivenName" minOccurs="0"/>
487     <element ref="md:SurName" minOccurs="0"/>
488     <element ref="md:EmailAddress" minOccurs="0"
489     maxOccurs="unbounded"/>
490     <element ref="md:TelephoneNumber" minOccurs="0"
491     maxOccurs="unbounded"/>
492   </sequence>
493   <attribute name="contactType" type="md:ContactTypeType"
494   use="required"/>
495   <anyAttribute namespace="##other" processContents="lax"/>
496 </complexType>
497 <element name="Company" type="string"/>
498 <element name="GivenName" type="string"/>
499 <element name="SurName" type="string"/>
500 <element name="EmailAddress" type="anyURI"/>
501 <element name="TelephoneNumber" type="string"/>
502 <simpleType name="ContactTypeType">
503   <restriction base="string">
504     <enumeration value="technical"/>
505     <enumeration value="support"/>
506     <enumeration value="administrative"/>
507     <enumeration value="billing"/>
508     <enumeration value="other"/>
509   </restriction>
510 </simpleType>
```

511 2.3.2.3 Element <AdditionalMetadataLocation>

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

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

```
519 <element name="AdditionalMetadataLocation"
520 type="md:AdditionalMetadataLocationType"/>
521 <complexType name="AdditionalMetadataLocationType">
522   <simpleContent>
523     <extension base="anyURI">
524       <attribute name="namespace" type="anyURI" use="required"/>
525     </extension>
526   </simpleContent>
527 </complexType>
```

528 2.4 Role Descriptor Elements

529 The elements in this section make up the bulk of the operational support component of the metadata.
530 Each element (save for the abstract one) define a specific collection of operational behavior in support of

531 SAML profiles defined in [SAMLProf].

532 **2.4.1 Element <RoleDescriptor>**

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

537 ID [Optional]

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

539 validUntil [Optional]

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

542 cacheDuration [Optional]

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

545 protocolSupportEnumeration [Required]

546 A whitespace-delimited set of URIs that identify the set of protocol specifications supported by the
547 role element. For SAML V2.0 entities, this set **MUST** include the SAML protocol namespace URI,
548 urn:oasis:names:tc:SAML:2.0:protocol.

549 errorURL [Optional]

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

552 <ds:Signature> [Optional]

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

555 <Extensions> [Optional]

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

559 <KeyDescriptor> [Zero or More]

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

562 <Organization> [Optional]

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

565 <ContactPerson> [Zero or More]

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

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

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

```

571 <element name="RoleDescriptor" type="md:RoleDescriptorType"/>
572 <complexType name="RoleDescriptorType" abstract="true">
573   <sequence>
574     <element ref="ds:Signature" minOccurs="0"/>
575     <element ref="md:Extensions" minOccurs="0"/>
576     <element ref="md:KeyDescriptor" minOccurs="0"
577     maxOccurs="unbounded"/>
578     <element ref="md:Organization" minOccurs="0"/>
579     <element ref="md:ContactPerson" minOccurs="0"
580     maxOccurs="unbounded"/>
581   </sequence>
582   <attribute name="ID" type="ID" use="optional"/>
583   <attribute name="validUntil" type="dateTime" use="optional"/>
584   <attribute name="cacheDuration" type="duration" use="optional"/>
585   <attribute name="protocolSupportEnumeration" type="md:anyURLListType"
586   use="required"/>
587   <attribute name="errorURL" type="anyURI" use="optional"/>
588   <anyAttribute namespace="##other" processContents="lax"/>
589 </complexType>
590 <simpleType name="anyURLListType">
591   <list itemType="anyURI"/>
592 </simpleType>

```

593 2.4.1.1 Element <KeyDescriptor>

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

597 use [Optional]

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

600 <ds:KeyInfo> [Required]

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

603 <EncryptionMethod> [Zero or More]

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

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

```

609 <element name="KeyDescriptor" type="md:KeyDescriptorType"/>
610 <complexType name="KeyDescriptorType">
611   <sequence>
612     <element ref="ds:KeyInfo"/>
613     <element ref="md:EncryptionMethod" minOccurs="0"
614     maxOccurs="unbounded"/>
615   </sequence>
616   <attribute name="use" type="md:KeyTypes" use="optional"/>
617 </complexType>
618 <simpleType name="KeyTypes">
619   <restriction base="string">
620     <enumeration value="encryption"/>
621     <enumeration value="signing"/>
622   </restriction>
623 </simpleType>
624 <element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>

```

625 2.4.2 Complex Type **SSODescriptorType**

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

630 <ArtifactResolutionService> [Zero or More]

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

634 <SingleLogoutService> [Zero or More]

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

637 <ManageNameIDService> [Zero or More]

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

640 <NameIDFormat> [Zero or More]

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

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

```
645 <complexType name="SSODescriptorType" abstract="true">
646   <complexContent>
647     <extension base="md:RoleDescriptorType">
648       <sequence>
649         <element ref="md:ArtifactResolutionService" minOccurs="0"
650 maxOccurs="unbounded"/>
651         <element ref="md:SingleLogoutService" minOccurs="0"
652 maxOccurs="unbounded"/>
653         <element ref="md:ManageNameIDService" minOccurs="0"
654 maxOccurs="unbounded"/>
655         <element ref="md:NameIDFormat" minOccurs="0"
656 maxOccurs="unbounded"/>
657       </sequence>
658     </extension>
659   </complexContent>
660 </complexType>
661 <element name="ArtifactResolutionService" type="md:IndexedEndpointType"/>
662 <element name="SingleLogoutService" type="md:EndpointType"/>
663 <element name="ManageNameIDService" type="md:EndpointType"/>
664 <element name="NameIDFormat" type="anyURI"/>
```

665 2.4.3 Element <IDPSSODescriptor>

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

669 `WantAuthnRequestsSigned` [Optional]

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

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

676 <NameIDMappingService> [Zero or More]
677 Zero or more elements of type **EndpointType** that describe endpoints that support the Name
678 Identifier Mapping profile defined in [SAMLProf]. The `ResponseLocation` attribute MUST be
679 omitted.

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

```
682 <element name="IDPSSODescriptor" type="md:IDPSSODescriptorType"/>  
683 <complexType name="IDPSSODescriptorType">  
684   <complexContent>  
685     <extension base="md:SSODescriptorType">  
686       <sequence>  
687         <element ref="md:SingleSignOnService"  
688 maxOccurs="unbounded"/>  
689         <element ref="md:NameIDMappingService" minOccurs="0"  
690 maxOccurs="unbounded"/>  
691       </sequence>  
692       <attribute name="WantAuthnRequestsSigned" type="boolean"  
693 use="optional"/>  
694     </extension>  
695   </complexContent>  
696 </complexType>  
697 <element name="SingleSignOnService" type="md:EndpointType"/>  
698 <element name="NameIDMappingService" type="md:EndpointType"/>
```

699 **2.4.4 Element <SPSSODescriptor>**

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

703 `AuthnRequestsSigned` [Optional]

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

706 `WantAssertionsSigned` [Optional]

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

711 <AssertionConsumerService> [One or More]

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

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

```
717 <element name="SPSSODescriptor" type="md:SPSSODescriptorType"/>  
718 <complexType name="SPSSODescriptorType">  
719   <complexContent>  
720     <extension base="md:SSODescriptorType">  
721       <sequence>
```

```

722         <element ref="md:AssertionConsumerService"
723 maxOccurs="unbounded"/>
724     </sequence>
725     <attribute name="AuthnRequestsSigned" type="boolean"
726 use="optional"/>
727     <attribute name="WantAssertionsSigned" type="boolean"
728 use="optional"/>
729     </extension>
730 </complexContent>
731 </complexType>
732 <element name="AssertionConsumerService"
733 type="md:AssertionConsumerServiceType"/>

```

734 2.4.5 Element <AuthnAuthorityDescriptor>

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

738 <AuthnQueryService> [One or More]

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

742 <AssertionIDRequestService> [Zero or More]

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

746 <NameIDFormat> [Zero or More]

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

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

```

751 <element name="AuthnAuthorityDescriptor"
752 type="md:AuthnAuthorityDescriptorType"/>
753 <complexType name="AuthnAuthorityDescriptorType">
754     <complexContent>
755         <extension base="md:RoleDescriptorType">
756             <sequence>
757                 <element ref="md:AuthnQueryService" maxOccurs="unbounded"/>
758                 <element ref="md:AssertionIDRequestService" minOccurs="0"
759 maxOccurs="unbounded"/>
760                 <element ref="md:NameIDFormat" minOccurs="0"
761 maxOccurs="unbounded"/>
762             </sequence>
763         </extension>
764     </complexContent>
765 </complexType>
766 <element name="AuthnQueryService" type="md:EndpointType"/>
767 <element name="AssertionIDRequestService" type="md:EndpointType"/>

```

768 2.4.6 Element <PDPDescriptor>

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

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

776 <AssertionIDRequestService> [Zero or More]
777 Zero or more elements of type **EndpointType** that describe endpoints that support the profile of
778 the Assertion Request protocol defined in [SAMLProf] or the special URI binding for assertion
779 requests defined in [SAMLBind].

780 <NameIDFormat> [Zero or More]
781 Zero or more elements of type **anyURI** that enumerate the name identifier formats supported by
782 this authority. See section 8.3 of [SAMLCore] for some possible values for this element.

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

```
785 <element name="PDPDescriptor" type="md:PDPDescriptorType"/>  
786 <complexType name="PDPDescriptorType">  
787   <complexContent>  
788     <extension base="md:RoleDescriptorType">  
789       <sequence>  
790         <element ref="md:AuthzService" maxOccurs="unbounded"/>  
791         <element ref="md:AssertionIDRequestService" minOccurs="0"  
792 maxOccurs="unbounded"/>  
793         <element ref="md:NameIDFormat" minOccurs="0"  
794 maxOccurs="unbounded"/>  
795       </sequence>  
796     </extension>  
797   </complexContent>  
798 </complexType>  
799 <element name="AuthzService" type="md:EndpointType"/>
```

800 2.4.7 Element <AttributeAuthorityDescriptor>

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

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

809 <AssertionIDRequestService> [Zero or More]
810 Zero or more elements of type **EndpointType** that describe endpoints that support the profile of
811 the Assertion Request protocol defined in [SAMLProf] or the special URI binding for assertion
812 requests defined in [SAMLBind].

813 <saml:Attribute> [Zero or More]
814 Zero or more elements that identify the SAML attributes supported by the authority. Specific
815 values MAY optionally be included, indicating that only certain values permitted by the attribute's
816 definition are supported.

817 <NameIDFormat> [Zero or More]
818 Zero or more elements of type **anyURI** that enumerate the name identifier formats supported by
819 this authority. See section 8.3 of [SAMLCore] for some possible values for this element.

820 <AttributeProfile> [Zero or More]
821 Zero or more elements of type **anyURI** that enumerate the attribute profiles supported by this
822 authority. See [SAMLProf] for some possible values for this element.

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

```
825 <element name="AttributeAuthorityDescriptor"  
826 type="md:AttributeAuthorityDescriptorType"/>  
827 <complexType name="AttributeAuthorityDescriptorType">  
828 <complexContent>  
829 <extension base="md:RoleDescriptorType">  
830 <sequence>  
831 <element ref="md:AttributeService" maxOccurs="unbounded"/>  
832 <element ref="md:AssertionIDRequestService" minOccurs="0"  
833 maxOccurs="unbounded"/>  
834 <element ref="saml:Attribute" minOccurs="0"  
835 maxOccurs="unbounded"/>  
836 <element ref="md:NameIDFormat" minOccurs="0"  
837 maxOccurs="unbounded"/>  
838 <element ref="md:AttributeProfile" minOccurs="0"  
839 maxOccurs="unbounded"/>  
840 </sequence>  
841 </extension>  
842 </complexContent>  
843 </complexType>  
844 <element name="AttributeService" type="md:EndpointType"/>  
845 <element name="AttributeProfile" type="anyURI"/>
```

846 2.4.8 Element <AttributeConsumerDescriptor>

847 The <AttributeConsumerDescriptor> element extends **RoleDescriptorType** with content reflecting
848 information specific to consumers of SAML attributes. Its **AttributeConsumerDescriptorType** complex
849 type contains the following additional element:

850 <AttributeConsumingService> [One or More]
851 One or more elements that describe a service provided by the entity that requires or desires the
852 use of SAML attributes.

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

857 The following schema fragment defines the <AttributeConsumerDescriptor> element and its
858 **AttributeConsumerDescriptorType** complex type:

```
859 <element name="AttributeConsumerDescriptor"  
860 type="md:AttributeConsumerDescriptorType"/>  
861 <complexType name="AttributeConsumerDescriptorType">  
862 <complexContent>  
863 <extension base="md:RoleDescriptorType">  
864 <sequence>  
865 <element ref="md:AttributeConsumingService"  
866 maxOccurs="unbounded"/>  
867 </sequence>  
868 </extension>  
869 </complexContent>  
870 </complexType>
```

871 **2.4.8.1 Element <AttributeConsumingService>**

872 The <AttributeConsumingService> element defines a particular service of the attribute consumer in
873 terms of the attributes the service requires or desires. Its **AttributeConsumingServiceType** complex type
874 contains the following elements and attributes:

875 `index` [Required]

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

878 `isDefault` [Optional]

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

881 `WantAssertionsSigned` [Optional]

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

886 <ServiceName> [One or More]

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

888 <ServiceDescription> [Zero or More]

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

890 <RequestedAttribute> [One or More]

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

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

```
894 <element name="AttributeConsumingService"  
895 type="md:AttributeConsumingServiceType"/>  
896 <complexType name="AttributeConsumingServiceType">  
897 <sequence>  
898 <element ref="md:ServiceName" maxOccurs="unbounded"/>  
899 <element ref="md:ServiceDescription" minOccurs="0"  
900 maxOccurs="unbounded"/>  
901 <element ref="md:RequestedAttribute" maxOccurs="unbounded"/>  
902 </sequence>  
903 <attribute name="index" type="unsignedShort" use="required"/>  
904 <attribute name="isDefault" type="boolean" use="optional"/>  
905 <attribute name="WantAssertionsSigned" type="boolean" use="optional"/>  
906 </complexType>  
907 <element name="ServiceName" type="md:localizedNameType"/>  
908 <element name="ServiceDescription" type="md:localizedNameType"/>
```

909 **2.4.8.2 Element <RequestedAttribute>**

910 The <RequestedAttribute> element specifies an attribute consumer's interest in a specific SAML
911 attribute, optionally including specific values. Its **RequestedAttributeType** complex type extends the
912 **saml:AttributeType** with the following attribute:

913 `isRequired` [Optional]

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

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

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

```
920 <element name="RequestedAttribute" type="md:RequestedAttributeType"/>  
921 <complexType name="RequestedAttributeType">  
922   <complexContent>  
923     <extension base="saml:AttributeType">  
924       <attribute name="isRequired" type="boolean" use="optional"/>  
925     </extension>  
926   </complexContent>  
927 </complexType>
```

928 2.5 Element `<AffiliationDescriptor>`

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

935 `affiliationOwnerID` [Required]

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

939 `ID` [Optional]

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

941 `validUntil` [Optional]

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

944 `cacheDuration` [Optional]

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

947 `<ds:Signature>` [Optional]

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

950 `<Extensions>` [Optional]

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

954 `<AffiliateMember>` [One or More]

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

957 `<KeyDescriptor>` [Zero or More]

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

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

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

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

982 2.6 Examples

983 The following is an example of metadata for a SAML system entity acting as an identity provider and an
984 attribute authority. A signature is shown as a placeholder, without the actual content.
985

```
986 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"  
987 xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
988 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
989 entityID="https://IdentityProvider.com/SAML">  
990 <ds:Signature>...</ds:Signature>  
991 <IDPSSODescriptor WantAuthnRequestsSigned="true"  
992 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">  
993   <KeyDescriptor use="signing">  
994     <ds:KeyInfo>  
995       <ds:KeyName>IdentityProvider.com SSO Key</ds:KeyName>  
996     </ds:KeyInfo>  
997   </KeyDescriptor>  
998   <ArtifactResolutionService isDefault="true" index="0"  
999   Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"  
1000   Location="https://IdentityProvider.com/SAML/Artifact"/>  
1001   <SingleLogoutService  
1002   Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"  
1003   Location="https://IdentityProvider.com/SAML/SLO/SOAP"/>  
1004   <SingleLogoutService  
1005   Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"  
1006   Location="https://IdentityProvider.com/SAML/SLO/Browser"  
1007   ResponseLocation="https://IdentityProvider.com/SAML/SLO/Response"/>  
1008   <NameIDFormat>  
1009     urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName  
1010   </NameIDFormat>  
1011   <NameIDFormat>  
1012     urn:oasis:names:tc:SAML:2.0:nameid-format:persistent  
1013   </NameIDFormat>  
1014   <NameIDFormat>  
1015     urn:oasis:names:tc:SAML:2.0:nameid-format:transient  
1016   </NameIDFormat>  
1017   <SingleSignOnService  
1018   Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"  
1019   Location="https://IdentityProvider.com/SAML/SSO/Browser"/>  
1020   <SingleSignOnService  
1021   Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"  
1022   Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
```

```

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

```

1075 The following is an example of metadata for a SAML system entity acting as a service provider and an
1076 attribute consumer. A signature is shown as a placeholder, without the actual content. For illustrative
1077 purposes, the service is one that does not require users to uniquely identify themselves, but rather
1078 authorizes access on the basis of a role-like attribute.

```

1080 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1081     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1082     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1083     entityID="https://ServiceProvider.com/SAML">
1084     <ds:Signature>...</ds:Signature>
1085     <SPSSODescriptor AuthnRequestsSigned="true"
1086     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1087         <KeyDescriptor use="signing">
1088             <ds:KeyInfo>
1089                 <ds:KeyName>ServiceProvider.com SSO Key</ds:KeyName>

```

```

1090         </ds:KeyInfo>
1091     </KeyDescriptor>
1092     <SingleLogoutService
1093         Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1094         Location="https://ServiceProvider.com/SAML/SLO/SOAP"/>
1095     <SingleLogoutService
1096         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1097         Location="https://ServiceProvider.com/SAML/SLO/Browser"
1098     ResponseLocation="https://ServiceProvider.com/SAML/SLO/Response"/>
1099     <NameIDFormat>
1100         urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1101     </NameIDFormat>
1102     <AssertionConsumerService isDefault="true" index="0"
1103         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
1104         Location="https://ServiceProvider.com/SAML/SSO/Artifact"/>
1105     <AssertionConsumerService index="1"
1106         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1107         Location="https://ServiceProvider.com/SAML/SSO/POST"/>
1108 </SPSSODescriptor>
1109 <AttributeConsumerDescriptor
1110     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1111     <KeyDescriptor use="encryption">
1112         <ds:KeyInfo>
1113             <ds:KeyName>ServiceProvider.com Encrypt Key</ds:KeyName>
1114         </ds:KeyInfo>
1115         <EncryptionMethod
1116             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
1117     </KeyDescriptor>
1118     <AttributeConsumingService index="0">
1119         <ServiceName xml:lang="en">
1120             Academic Journals R US
1121         </ServiceName>
1122         <RequestedAttribute
1123             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1124             Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7"
1125             FriendlyName="eduPersonEntitlement">
1126                 <saml:AttributeValue>
1127                     https://ServiceProvider.com/entitlements/123456789
1128                 </saml:AttributeValue>
1129             </RequestedAttribute>
1130         </AttributeConsumingService>
1131     </AttributeConsumerDescriptor>
1132 <Organization>
1133     <OrganizationName xml:lang="en">
1134         Academic Journals R US
1135     </OrganizationName>
1136     <OrganizationDisplayName xml:lang="en">
1137         Academic Journals R US, a Division of Dirk Corp.
1138     </OrganizationDisplayName>
1139     <OrganizationURL xml:lang="en">
1140         https://ServiceProvider.com
1141     </OrganizationURL>
1142 </Organization>
1143 </EntityDescriptor>

```

1144 3 Signature Processing

1145 Various elements in a metadata instance can be digitally signed (as indicated by the element's inclusion of
1146 a `<ds:Signature>` element), with the following benefits:

- 1147 • Metadata integrity
- 1148 • Authentication of the metadata by a trusted signer

1149 A digital signature is not always required, for example if the relying party obtains the information directly
1150 from the publishing entity directly (with no intermediaries) through a secure channel, with the entity having
1151 authenticated to the relying party by some means other than a digital signature.

1152 Many different techniques are available for "direct" authentication and secure channel establishment
1153 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,
1154 the applicable security requirements depend on the communicating applications.

1155 Additionally, elements can inherit signatures on enclosing parent elements that are themselves signed.

1156 In the absence of such context, it is RECOMMENDED that at least the root element of a metadata
1157 instance be signed.

1158 3.1 XML Signature Profile

1159 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
1160 and many choices. This section details the constraints on these facilities so that metadata processors do
1161 not have to deal with the full generality of XML Signature processing. This usage makes specific use of
1162 the **xs:ID**-typed attributes optionally present on the elements to which signatures can apply. These
1163 attributes are collectively referred to in this section as the identifier attributes.

1164 3.1.1 Signing Formats and Algorithms

1165 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
1166 detached.

1167 SAML metadata MUST use enveloped signatures when signing the elements defined in this specification.
1168 SAML processors SHOULD support the use of RSA signing and verification for public key operations in
1169 accordance with the algorithm identified by <http://www.w3.org/2000/09/xmlsig#rsa-sha1>.

1170 3.1.2 References

1171 Signed metadata elements MUST supply a value for the identifier attribute on the signed element. The
1172 element may or may not be the root element of the actual XML document containing the signed metadata
1173 element.

1174 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute
1175 value of the metadata element being signed. For example, if the identifier attribute value is "foo", then the
1176 URI attribute in the `<ds:Reference>` element MUST be "#foo".

1177 As a consequence, a metadata element's signature MUST apply to the content of the signed element and
1178 any child elements it contains.

1179 3.1.3 Canonicalization Method

1180 SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the
1181 `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a `<ds:Transform>`
1182 algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML metadata

1183 embedded in an XML context can be verified independent of that context.

1184 **3.1.4 Transforms**

1185 Signatures in SAML metadata SHOULD NOT contain transforms other than the enveloped signature
1186 transform (with the identifier <http://www.w3.org/2000/09/xmlsig#enveloped-signature>) or the exclusive
1187 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
1188 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

1189 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
1190 not, verifiers MUST ensure that no content of the signed metadata element is excluded from the
1191 signature. This can be accomplished by establishing out-of-band agreement as to what transforms are
1192 acceptable, or by applying the transforms manually to the content and reverifying the result as consisting
1193 of the same SAML metadata.

1194 **3.1.5 KeyInfo**

1195 XML Signature [XMLSig] defines usage of the `<ds:KeyInfo>` element. SAML does not require the
1196 use of `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY
1197 be absent.

1198 4 Metadata Publication and Resolution

1199 Two mechanisms are provided for an entity to publish (and for a consumer to resolve the location of)
1200 metadata documents: via a "well-known-location" by directly dereferencing the entity's unique identifier (a
1201 URI variously referred to as an *entityID* or *providerID*), or indirectly by publishing the location of metadata
1202 in the DNS. Other out-of-band mechanisms are of course also permitted. A consumer that supports both
1203 approaches defined in this document MUST attempt resolution via DNS before using the "well-known-
1204 location" mechanism.

1205 When retrieval requires network transport of the document, the transport SHOULD be protected with
1206 mechanisms providing server authentication and integrity protection. For example, HTTP-based resolution
1207 SHOULD be protected with TLS/SSL [RFC2246] as amended by [RFC3546].

1208 Various mechanisms are described in this section to aid in establishing trust in the accuracy and
1209 legitimacy of metadata, including use of XML signatures, SSL/TLS server authentication, and DNS
1210 signatures. Regardless of the mechanism(s) used, relying parties SHOULD have some means by which to
1211 establish trust in metadata information before relying on it.

1212 4.1 Publication and Resolution via Well-Known Location

1213 The following sections describe publication and resolution of metadata by means of a well-known location.

1214 4.1.1 Publication

1215 Entities MAY publish their metadata documents at a well known location by placing the document at the
1216 location denoted by its unique identifier, which MUST be in the form of a URL (rather than a URN). See
1217 Section 8.3.6 of [SAMLCore] for more information about such identifiers. It is STRONGLY
1218 RECOMMENDED that `https` URLs be used for this purpose. An indirection mechanism supported by the
1219 URL scheme (such as an HTTP 1.1 302 redirect) MAY be used if the document is not placed directly at
1220 the location. If the publishing protocol permits MIME-based identification of content types, the content type
1221 of the metadata instance MUST be `application/samlmetadata+xml`.

1222 The XML document provided at the well-known location MUST describe the metadata only for the entity
1223 represented by the unique identifier (that is, the root element MUST be an `<EntityDescriptor>` with
1224 an `entityID` matching the location). If other entities need to be described, the
1225 `<AdditionalMetaLocation>` element MUST be used. Thus the `<EntitiesDescriptor>` element
1226 MUST NOT be used in documents published using this mechanism, since a group of entities are not
1227 defined by such an identifier.

1228 4.1.2 Resolution

1229 If an entity's unique identifier is a URL, metadata consumers MAY attempt to resolve an entity's unique
1230 identifier directly, in a scheme-specific manner, by dereferencing the identifier.

1231 4.2 Publishing and Resolution via DNS

1232 To improve the accessibility of metadata documents and provide additional indirection between an entity's
1233 unique identifier and the location of metadata, entities MAY publish their metadata document locations in a
1234 zone of their corresponding DNS [RFC1034]. The entity's unique identifier (a URI) is used as the input to
1235 the process. Since URIs are flexible identifiers, location publication methods and the resolution process
1236 are determined by the URI's scheme and fully-qualified name. URI locations for metadata are

1237 subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in [\[RFC2915\]](#)
1238 and [\[RFC3403\]](#).

1239 It is RECOMMENDED that entities publish their resource records in signed zone files using [\[RFC2535\]](#)
1240 such that relying parties may establish the validity of the published location and authority of the zone, and
1241 integrity of the DNS response. If DNS zone signatures are present, relying parties MUST properly validate
1242 the signature.

1243 **4.2.1 Publication**

1244 This specification makes use of the NAPTR resource record described in [\[RFC2915\]](#) and [\[RFC3403\]](#).
1245 Familiarity with these documents is encouraged.

1246 Dynamic Delegation Discovery System (DDDS) [\[RFC3401\]](#) is a general purpose system for the retrieval of
1247 information based on an application-specific input string and the application of well known rules to
1248 transform that string until a terminal condition is reached requiring a look-up into an application-specific
1249 defined database or resolution of a URL based on the rules defined by the application. DDDS defines a
1250 specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS
1251 necessary to apply DDDS rules.

1252 Entities MAY publish separate URLs when multiple metadata documents need to be distributed, or when
1253 different metadata documents are required due to multiple trust relationships that require separate keying
1254 material, or when service interfaces require separate metadata declarations. This may be accomplished
1255 through the use of the optional `<AdditionalMetaLocation>` element, or through the regexp facility and
1256 multiple service definition fields in the NAPTR resource record itself.

1257 If the publishing protocol permits MIME-based identification of content types, the content type of the
1258 metadata instance MUST be `application/samlmetadata+xml`.

1259 If the entity's unique identifier is a URN, publication of the corresponding metadata location proceeds as
1260 specified in [\[RFC3404\]](#). Otherwise, the resolution of the metadata location proceeds as specified below.

1261 The following is the application-specific profile of DDDS for SAML metadata resolution.

1262 **4.2.1.1 First Well Known Rule**

1263 The "first well-known-rule" for processing SAML metadata resolution is to parse the entity's unique
1264 identifier and extract the fully-qualified domain name (subexpression 3) as described in Section "[Parsing](#)
1265 [the providerID](#)".

1266 **4.2.1.2 The Order Field**

1267 The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY
1268 provide multiple NAPTR resource records which MUST be processed by the resolver application in the
1269 order indicated by this field.

1270 **4.2.1.3 The Preference Field**

1271 For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving
1272 application. The resolving application MAY ignore this order, in cases where the service field value does
1273 not meet the resolver's requirements (e.g.: the resource record returns a protocol the application does not
1274 support).

1275 4.2.1.4 The Flag Field

1276 SAML metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying
1277 additional resource records are to be processed). The "U" flag indicates that the output of the rule is a
1278 URI.

1279 4.2.1.5 The Service Field

1280 The SAML-specific service field, as described in the following BNF, declares the modes by which instance
1281 document(s) shall be made available:

```
1282 servicefield = 1("PID2U" / "NID2U") "+" proto [*(":" class) *(":" servicetype)]
1283 proto = 1("https" / "uddi")
1284 class = 1[ "entity" / "entitygroup" ]
1285 servicetype = 1(si / "spssso" / "idpssso" / "authn" / "authnauth" / "pdp" / "attrauth" /
1286 "attrcons" / alphanum )
1287 si = "si" [ ":" alphanum ] [ ":" endpoint ]
1288 alphanum = 1*32 (ALPHA / DIGIT)
```

1289 where:

- 1290 • servicefield PID2U resolves an entity's unique identifier to metadata URL.
- 1291 • servicefield NID2U resolves a principal's <NameIdentifier> into a metadata URL.
- 1292 • proto describes the retrieval protocol (`https` or `uddi`). In the case of UDDI, the URL will be an
1293 http(s) URL referencing a WSDL document.
- 1294 • class identifies whether the referenced metadata document describes a single entity, or multiple.
1295 In the latter case, the referenced document MUST contain the entity defined by the original unique
1296 identifier as a member of a group of entities within the document itself such as an
1297 <AffiliationDescriptor> or <EntitiesDescriptor>.
- 1298 • servicetype allows an entity to publish metadata for distinct roles and services as separate
1299 documents. Resolvers who encounter multiple servicetype declarations will dereference the
1300 appropriate URI, depending on which service is required for an operation (e.g.: an entity operating
1301 both as an identity provider and a service provider can publish metadata for each role at different
1302 locations). The `authn` service type represents a <SingleSignOnService> endpoint.
- 1303 • si (with optional endpoint component) allows the publisher to either directly publish the metadata
1304 for a service instance, or by articulating a SOAP endpoint (using `endpoint`).

1305 For example:

- 1306 • PID2U+https:entity - represents the entity's complete metadata document available via the
1307 https protocol
- 1308 • PID2U+uddi:entity:si:foo - represents the WSDL document location that describes a service
1309 instance "foo"
- 1310 • PID2U+https:entitygroup:idpssso - represents the metadata for a group of entities acting as
1311 SSO identity providers, of which the original entity is a member.
- 1312 • NID2U+https:idp - represents the metadata for the SSO identity provider of a principal

1313 4.2.1.6 The Regexp and Replacement Fields

1314 The expected output after processing the input string through the regex MUST be a valid `https` URL or
1315 UDDI node (WSDL document) address.

1316 4.2.2 NAPTR Examples

1317 4.2.2.1 Entity Metadata NAPTR Examples

1318 Entities publish metadata URLs in the following manner:

```
1319 $ORIGIN provider.biz
1320
1321 ;; order pref f service regexp or replacement
1322
1323 IN NAPTR 100 10 "U" PID2U+https:entity
1324 "!.*!https://host.provider.biz/some/directory/trust.xml!" ""
1325 IN NAPTR 110 10 "U" PID2U+https: entity:trust
1326 "!.*!https://foo.provider.biz:1443/mdtrust.xml!" ""
1327 IN NAPTR 125 10 "U" PID2U+https:"
1328 IN NAPTR 110 10 "U" PID2U+uddi:entity
1329 "!.*!https://this.uddi.node.provider.biz/libmd.wsd1" ""
```

1330 4.2.2.2 Name Identifier Examples

1331 A principal's employer `example.int` operates an identity provider which may be used by an office supply
1332 company to authenticate authorized buyers. The supplier takes a users' email address
1333 `buyer@example.int` as input to the resolution process, and parses the email address to extract the
1334 FQDN (`example.int`). The employer publishes the following NAPTR record in the `example.int` DNS:

```
1335 $ORIGIN example.int
1336
1337 IN NAPTR 100 10 "U" NID2U+https:authn
1338 "!.^([\^@]+)@(\.*)!https://serv.example.int:8000/cgi-bin/getmd?\1!" ""
1339 IN NAPTR 100 10 "U" NID2U+https:idp
1340 "!.^([\^@]+)@(\.*)!https://auth.example.int/app/auth?\1" ""
```

1341 4.2.3 Resolution

1342 When resolving metadata for an entity via the DNS, the unique identifier of the entity is used as the initial
1343 input into the resolution process, rather than as an actual location Proceed as follows:

- 1344 • If the unique identifier is a URN, proceed with the resolution steps as defined in [\[RFC3404\]](#).
- 1345 • Otherwise, parse the identifier to obtain the fully-qualified domain name.
- 1346 • Query the DNS for NAPTR resource records of the domain iteratively until a terminal resource
1347 record is returned.
- 1348 • Identify which resource record to use based on the service fields, then order fields, then preference
1349 fields of the result set.
- 1350 • Obtain the document(s) at the provided location(s) as required by the application.

1351 4.2.3.1 Parsing the Unique Identifier

1352 To initiate the resolution of the location of the metadata information, it will be necessary in some cases to
1353 decompose the entity's unique identifier (expressed as a URI) into one or more atomic elements.

1354 The following regular expression should be used when initiating the decomposition process:

```
1355 ^([\^:/?#]+)?/*([\^:/?#]*@)?(((\^/?#*\.)*((\^/?#:\.]+)\.([\^/?#:\.]+))
1356 (: \d+) ? ([\^?#]*) (\ \ ? [\^#]*) ? (#.*) ? $
1357      1           2           3         4           5           6           7
1358      8           9          10          11
```

1359 Subexpression 3 MUST result in a Fully-Qualified Domain Name (FQDN), which will be the basis for
1360 retrieving metadata locations from this zone.

1361 **4.2.3.2 Obtaining Metadata via the DNS**

1362 Upon completion of the parsing of the identifier, the application then performs a DNS query for the resulting
1363 domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses.
1364 Applications MAY exclude from the result set any service definitions that do not concern the present
1365 request operations.

1366 Resolving applications MUST subsequently order the result set according to the order field, and MAY
1367 order the result set based on the preference set. Resolvers are NOT REQUIRED to follow the ordering of
1368 the preferences field. The resulting NAPTR resource record(s) are operated on iteratively (based on the
1369 order flag) until a terminal NAPTR resource record is reached.

1370 The result will be a well-formed, absolute URL, which is then used to retrieve the metadata document.

1371 **4.2.4 Metadata Location Caching**

1372 Location caching MUST NOT exceed the TTL of the DNS zone from which the location was derived.
1373 Resolvers MUST obtain a fresh copy of the metadata location upon reaching the expiration of the TTL of
1374 the zone.

1375 Publishers of metadata documents should carefully consider the TTL of the zone when making changes
1376 to metadata document locations. Should such a location change occur, a publisher MUST either keep the
1377 document at both the old and new location until all conforming resolvers are certain to have the updated
1378 location (e.g.: time of zone change + TTL), or provide an HTTP Redirect [RFC2616] response at the old
1379 location specifying the new location.

1380 **4.3 Post-Processing of Metadata**

1381 The following sections describe the post-processing of metadata.

1382 **4.3.1 Metadata Instance Caching**

1383 Document caching MUST NOT exceed the `validUntil` or `cacheDuration` attribute of the subject
1384 element(s). If metadata elements have parent elements which contain caching policies, the parent
1385 element takes precedence.

1386 To properly process the `cacheDuration` attribute, consumers MUST retain the date and time when the
1387 document was retrieved.

1388 When a document or element has expired, the consumer MUST retrieve a fresh copy, which may require
1389 a refresh of the document location(s). Consumers SHOULD process document cache processing
1390 according to [RFC2616] Section 13, and MAY request the Last-Modified date and time from the HTTP
1391 server. Publishers SHOULD ensure acceptable cache processing as described in [RFC2616] (Section
1392 10.3.5 304 Not Modified).

1393 **4.3.2 Handling of HTTPS Redirects**

1394 Publishers MAY issue an HTTP Redirect (301 Moved Permanently, 302 or 307 Temporary Redirect)
1395 [RFC2616], and user agents MUST follow the specified URL in the Redirect response. Redirects
1396 SHOULD be of the same protocol as the initial request.

1397 **4.3.3 Processing of XML Signatures and General Trust Processing**

1398 Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and
1399 for the trust ascribed to the entity described by such metadata:

- 1400 • Trust derived from the signature of the DNS zone from which the metadata location URL was

1401 resolved, ensuring accuracy of the metadata document location(s)

1402 • Trust derived from signature processing of the metadata document itself, ensuring the integrity of
1403 the XML document

1404 • Trust derived from the SSL/TLS server authentication of the metadata location URL, ensuring the
1405 identity of the publisher of the metadata

1406 Post-processing of the metadata document MUST include signature processing at the XML-document
1407 level and MAY include one of the other two processes. Specifically, the relying party MAY choose to trust
1408 any of the cited authorities in the resolution and parsing process. Publishers of metadata MUST employ a
1409 document-integrity mechanism and MAY employ any of the other two processing profiles to establish trust
1410 in the metadata document, governed by implementation policies.

1411 **4.3.3.1 Processing Signed DNS Zones**

1412 Verification of DNS zone signature SHOULD be processed, if present, as described in [\[RFC2535\]](#).

1413 **4.3.3.2 Processing Signed Documents and Fragments**

1414 Published metadata documents SHOULD be signed, as described in Section 3, either by a certificate
1415 issued to the subject of the document, or by another trusted party. Publishers MAY consider signatures of
1416 other parties as a means of trust conveyance.

1417 Metadata consumers MUST validate signatures, when present, on the metadata document as described
1418 by Section 3.

1419 **4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL**

1420 It is STRONGLY RECOMMENDED that publishers implement TLS/SSL URLs; therefore, consumers
1421 SHOULD consider the trust inherited from the issuer of the TLS/SSL certificate. Publication URLs may not
1422 always be located in the domain of the subject of the metadata document; therefore, consumers SHOULD
1423 NOT presume certificates whose subject is the entity in question, as it may be hosted by another trusted
1424 party.

1425 As the basis of this trust may not be available against a cached document, other mechanisms SHOULD
1426 be used under such circumstances.

1427

5 References

- 1428 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
1429 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1430 **[SAMLBind]** S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language*
1431 *(SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-bindings-
1432 2.0-cd-02. See <http://www.oasis-open.org/committees/security/>.
- 1433 **[SAMLCore]** S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion*
1434 *Markup Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID
1435 sstc-saml-core-2.0-cd-02. See <http://www.oasis-open.org/committees/security/>.
- 1436 **[SAMLMeta-xsd]** S. Cantor et al., SAML metadata schema. OASIS SSTC, September 2004.
1437 Document ID sstc-saml-schema-metadata-2.0. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1438 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1439 **[SAMLProf]** S. Cantor et al., *Profiles for the OASIS Security Assertion Markup Language*
1440 *(SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-profiles-
1441 2.0-cd-02. See <http://www.oasis-open.org/committees/security/>.
- 1442 **[SAMLSec]** F. Hirsch et al., *Security and Privacy Considerations for the OASIS Security*
1443 *Assertion Markup Language (SAML) V2.0*. OASIS SSTC, September 2004.
1444 Document ID sstc-saml-sec-consider-2.0-cd-02. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1445 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1446 **[XMLEnc]** D. Eastlake et al., XML-Encryption Syntax and Processing,
1447 <http://www.w3.org/TR/xmlenc-core/>, World Wide Web Consortium.
- 1448 **[XMLSig]** D. Eastlake et al., XML-Signature Syntax and Processing,
1449 <http://www.w3.org/TR/xmlsig-core/>, World Wide Web Consortium.

6 Registration of MIME media type application/samlmetadata+xml

1450

1451

1452 To: ietf-types@iana.org

1453 Subject: Registration of MIME media type application/samlmetadata+xml

1454

1455 Introduction

1456 This document defines a MIME media type -- application/samlmetadata+xml -- for use with the
1457 XML serialization of Security Assertion Markup Language metadata.

1458

1459 SAML is a work product of the OASIS Security Services Technical Committee [SSTC]. The SAML
1460 specifications define XML-based constructs with which one may make, and convey, security
1461 assertions. Using SAML, one can assert that an authentication event pertaining to some subject
1462 has occurred and convey said assertion to a relying party, for example.

1463

1464 SAML profiles require agreements between system entities regarding identifiers, binding support,
1465 endpoints, certificates, keys, and so forth. Such information is treated as metadata by SAML v2.0.
1466 [SAMLv2Meta] specifies this metadata, as well as specifying metadata publication and resolution
1467 mechanisms. If the publishing protocol permits MIME-based identification of content types, then
1468 use of the application/samlmetadata+xml MIME media type is required.

1469

1470 MIME media type name: application

1471

1472 MIME subtype name: samlmetadata+xml

1473

1474 Required parameters: none

1475

1476 Optional parameters: charset

1477 Same as charset parameter of application/xml [RFC3023].

1478

1479 Encoding considerations:

1480 Same as for application/xml [RFC3023].

1481

1482 Security considerations:

1483 Per their specification, samlmetadata+xml typed objects do not contain executable content.
1484 However, these objects are XML-based [XML], and thus they have all of the general security
1485 considerations presented in section 10 of [RFC3023].

1486

1487 SAML metadata [SAMLv2Meta] contains information whose integrity and authenticity is important
1488 – identity provider and service provider public keys and endpoint addresses, for example.

1489

1490 To counter potential issues, the publisher may sign samlmetadata+xml typed objects. Any such
1491 signature should be verified by the recipient of the data - both as a valid signature, and as being
1492 the signature of the publisher.

1493

1494 Additionally, various of the publication protocols, e.g. HTTP-over-TLS/SSL, offer means for
1495 ensuring the authenticity of the publishing party and for protecting the metadata in transit.
1496 [SAMLv2Meta] also defines prescriptive metadata caching directives, as well as guidance on
1497 handling HTTPS redirects, trust processing, server authentication, and related items.

1498 For a more detailed discussion of SAML v2.0 metadata and its security considerations, please
1499 see [SAMLv2Meta]. For a discussion of overall SAML v2.0 security considerations and specific
1500 security-related design features, please refer to the SAML v2.0 specifications listed in the below
1501 bibliography. The specifications containing security-specific information are explicitly listed.
1502

1503 Interoperability considerations:

1504 SAML v2.0 metadata explicitly supports identifying the protocols and versions supported by the
1505 identified entities. For example, an identity provider entity can be denoted as supporting SAML
1506 v2.0, SAML v1.1 [SAMLv1.1], Liberty ID-FF 1.2 [LAPFF], or even other protocols if they are
1507 unambiguously identifiable via URI [RFC2396]. This protocol support information is conveyed via
1508 the `protocolSupportEnumeration` attribute of metadata objects of the
1509 `RoleDescriptorType`.
1510

1511 Published specification:

1512 [SAMLv2Meta] explicitly specifies use of the `application/samlmetadata+xml` MIME media type.
1513

1514 Applications which use this media type:

1515 Potentially any application implementing SAML v2.0, as well as those applications implementing
1516 specifications based on SAML, e.g. those available from the Liberty Alliance [LAP].
1517

1518 Additional information:

1520 Magic number(s):

1521 In general, the same as for `application/xml` [RFC3023]. In particular, the XML root
1522 element of the returned object will have a namespace-qualified name with:
1523

1524 – a local name of: `EntityDescriptor`, `AffiliationDescriptor`, or
1525 `EntitiesDescriptor`

1527 – a namespace URI of: `urn:oasis:names:tc:SAML:2.0:metadata`
1528 (the SAMLv2.0 metadata namespace)
1529
1530

1531 File extension(s): none

1533 Macintosh File Type Code(s): none
1534

1535 Person & email address to contact for further information:

1536 This registration is made on behalf of the OASIS Security Services Technical Committee (SSTC)
1537 Please refer to the SSTC website for current information on committee chairperson(s) and their
1538 contact addresses: <http://www.oasis-open.org/committees/security/>. Committee members should
1539 submit comments and potential errata to the securityservices@lists.oasis-open.org list. Others
1540 should submit them by filling out the web form located at [http://www.oasis-
1541 open.org/committees/comments/form.php?wg_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security).
1542

1543 Additionally, the SAML developer community email distribution list, [saml-dev@lists.oasis-
1544 open.org](mailto:saml-dev@lists.oasis-open.org), may be employed to discuss usage of the `application/samlmetadata+xml` MIME media
1545 type. The "saml-dev" mailing list is publicly archived here: [http://lists.oasis-
1546 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
1547 subscribe, send a message with the single word "subscribe" in the message body, to: [saml-dev-
1548 request@lists.oasis-open.org](mailto:saml-dev-request@lists.oasis-open.org).
1549

1550 Intended usage: COMMON
1551
1552

1553 Author/Change controller:
1554 The SAML specification sets are a work product of the OASIS Security Services Technical
1555 Committee (SSTC). OASIS and the SSTC have change control over the SAML specification sets.
1556

1557 Bibliography

- 1558 [LAP] “*Liberty Alliance Project*”. See <http://www.projectliberty.org/>
- 1561 [LAPFF] “*Liberty Alliance Project: Federation Framework*”. See
1562 <http://www.projectliberty.org/resources/specifications.php#box1>
1563
- 1564 [OASIS] “*Organization for the Advancement of Structured Information Systems*”.
1565 See <http://www.oasis-open.org/>
1566
- 1567 [RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifiers*
1568 (*URI*): *Generic Syntax*. IETF RFC 2396, August 1998. Available at
1569 <http://www.ietf.org/rfc/rfc2396.txt>
1570
- 1571 [RFC3023] M. Murata, S. St.Laurent, D. Kohn, “*XML Media Types*”, IETF Request for
1572 Comments 3023, January 2001. Available as [http://www.rfc-](http://www.rfc-editor.org/rfc/rfc3023.txt)
1573 [editor.org/rfc/rfc3023.txt](http://www.rfc-editor.org/rfc/rfc3023.txt)
1574
- 1575 [SAMLv1.1] OASIS Security Services Technical Committee, “*Security Assertion*
1576 *Markup Language (SAML) Version 1.1 Specification Set*”. OASIS
1577 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)
1578 [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)
1579
- 1580 [SAMLv2.0] OASIS Security Services Technical Committee, “*Security Assertion*
1581 *Markup Language (SAML) Version 2.0 Specification Set*”. Committee
1582 Draft. Available at <http://www.oasis-open.org/committees/security/>
1583
- 1584 [SAMLv2Bind] S. Cantor et al., “*Bindings for the OASIS Security Assertion Markup*
1585 *Language (SAML) V2.0*”. OASIS SSTC, September 2004. Document ID
1586 sstc-saml-bindings-2.0-cd-02. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1587 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
1588
- 1589 [SAMLv2Core] S. Cantor et al., “*Assertions and Protocols for the OASIS Security*
1590 *Assertion Markup Language (SAML) V2.0*”. OASIS SSTC, September
1591 2004. Document ID sstc-saml-core-2.0-cd-02. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1592 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
1593
- 1594 [SAMLv2Meta] S. Cantor et al., “*Metadata for the OASIS Security Assertion Markup*
1595 *Language (SAML) V2.0*”. OASIS SSTC, September 2004. Document ID
1596 sstc-saml-metadata-2.0-cd-02. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1597 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
1598
- 1599 [SAMLv2Prof] S. Cantor et al., “*Profiles for the OASIS Security Assertion Markup*
1600 *Language (SAML) V2.0*”. OASIS SSTC, September 2004. Document ID
1601 sstc-saml-profiles-2.0-cd-02. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1602 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
1603
- 1604 [SAMLv2Sec] F. Hirsch et al., “*Security and Privacy Considerations for the OASIS*
1605 *Security Assertion Markup Language (SAML) V2.0*”. OASIS SSTC,
1606 September 2004. Document ID sstc-saml-sec-consider-2.0-cd-02. See
1607 <http://www.oasis-open.org/committees/security/>

1608 [SSTC] "OASIS Security Services Technical Committee". See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1609 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
1610
1611 [XML] Bray, T., Paoli, J., Sperberg-McQueen, C.M. and E. Maler, François
1612 Yergeau, "*Extensible Markup Language (XML) 1.0 (Third Edition)*", World
1613 Wide Web Consortium Recommendation REC-xml, Feb 2004, Available
1614 as <http://www.w3.org/TR/REC-xml/>
1615

1616 Appendix A. Acknowledgments

1617 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
1618 Committee, whose voting members at the time of publication were:

- 1619 • Conor Cahill, AOL
- 1620 • John Hughes, ATOS Origin
- 1621 • Hal Lockhart, BEA Systems
- 1622 • Rick Randall, Booz Allen Hamilton
- 1623 • Ronald Jacobson, Computer Associates
- 1624 • Gavenraj Sodhi, Computer Associates
- 1625 • Tim Alsop, CyberSafe Limited
- 1626 • Paul Madsen, Entrust
- 1627 • Carolina Canales-Valenzuela, Ericsson
- 1628 • Dana Kaufman, Forum Systems
- 1629 • Irving Reid, Hewlett-Packard
- 1630 • Paula Austel, IBM
- 1631 • Maryann Hondo, IBM
- 1632 • Michael McIntosh, IBM
- 1633 • Anthony Nadalin, IBM
- 1634 • Nick Ragouzis, Individual
- 1635 • Scott Cantor, Internet2
- 1636 • Bob Morgan, Internet2
- 1637 • Prateek Mishra, Netegrity
- 1638 • Forest Yin, Netegrity
- 1639 • Peter Davis, Neustar
- 1640 • Frederick Hirsch, Nokia
- 1641 • John Kemp, Nokia
- 1642 • Senthil Sengodan, Nokia
- 1643 • Scott Kiestler, Novell
- 1644 • Cameron Morris, Novell
- 1645 • Charles Knouse, Oblix
- 1646 • Steve Anderson, OpenNetwork
- 1647 • Ari Kermaier, Oracle
- 1648 • Vamsi Motukuru, Oracle
- 1649 • Darren Platt, Ping Identity
- 1650 • Jim Lien, RSA Security
- 1651 • John Linn, RSA Security
- 1652 • Rob Philpott, RSA Security
- 1653 • Dipak Chopra, SAP
- 1654 • Jahan Moreh, Sigaba
- 1655 • Bhavna Bhatnagar, Sun Microsystems
- 1656 • Jeff Hodges, Sun Microsystems
- 1657 • Eve Maler, Sun Microsystems

1658

- 1659 • Ronald Monzillo, Sun Microsystems
- 1660 • Emily Xu, Sun Microsystems
- 1661 • Mike Beach, Boeing
- 1662 • Greg Whitehead, Trustgenix

1663

1664 The editors also would like to acknowledge the following people for their contributions to previous versions
1665 of the OASIS Security Assertions Markup Language Standard:

- 1666 • Stephen Farrell, Baltimore Technologies
- 1667 • David Orchard, BEA Systems
- 1668 • Krishna Sankar, Cisco Systems
- 1669 • Zahid Ahmed, CommerceOne
- 1670 • Carlisle Adams, Entrust
- 1671 • Tim Moses, Entrust
- 1672 • Nigel Edwards, Hewlett-Packard
- 1673 • Joe Pato, Hewlett-Packard
- 1674 • Bob Blakley, IBM
- 1675 • Marlena Erdos, IBM
- 1676 • Marc Chanliau, Netegrity
- 1677 • Chris McLaren, Netegrity
- 1678 • Lynne Rosenthal, NIST
- 1679 • Mark Skall, NIST
- 1680 • Simon Godik, Overxeer
- 1681 • Charles Norwood, SAIC
- 1682 • Evan Prodromou, Securant
- 1683 • Robert Griffin, RSA Security (former editor)
- 1684 • Sai Allarvarpu, Sun Microsystems
- 1685 • Chris Ferris, Sun Microsystems
- 1686 • Emily Xu, Sun Microsystems
- 1687 • Mike Myers, Traceroute Security
- 1688 • Phillip Hallam-Baker, VeriSign (former editor)
- 1689 • James Vanderbeek, Vodafone
- 1690 • Mark O'Neill, Vordel
- 1691 • Tony Palmer, Vordel

1692

1693 Finally, the editors wish to acknowledge the following people for their contributions of material used as
1694 input to the OASIS Security Assertions Markup Language specifications:

- 1695 • Thomas Gross, IBM
- 1696 • Birgit Pfitzmann, IBM

Appendix B. Notices

1698 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
1699 might be claimed to pertain to the implementation or use of the technology described in this document or
1700 the extent to which any license under such rights might or might not be available; neither does it represent
1701 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
1702 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
1703 available for publication and any assurances of licenses to be made available, or the result of an attempt
1704 made to obtain a general license or permission for the use of such proprietary rights by implementors or
1705 users of this specification, can be obtained from the OASIS Executive Director.

1706 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
1707 other proprietary rights which may cover technology that may be required to implement this specification.
1708 Please address the information to the OASIS Executive Director.

1709 **Copyright © OASIS Open 2004. All Rights Reserved.**

1710 This document and translations of it may be copied and furnished to others, and derivative works that
1711 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
1712 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
1713 this paragraph are included on all such copies and derivative works. However, this document itself may
1714 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
1715 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
1716 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
1717 into languages other than English.

1718 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
1719 or assigns.

1720 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1721 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1722 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1723 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.