



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

Last-Call Working Draft 08, 13@@@ July 2004

Document identifier:

sstc-saml-metadata-2.0-draft-08

Location:

<http://www.oasis-open.org/apps/org/workgroup/security/download.php>

Editors:

Jahan Moreh, Sigaba <jmoreh@sigaba.com>
Scott Cantor, Internet2 <cantor.2@osu.edu>
Eve Maler, Sun Microsystems <eve.maler@sun.com>

Abstract:

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

Status:

This is a last-call working draft produced by the Security Services Technical Committee. **See the Revision History for details of changes made in this revision.**

Comments on this last-call draft are solicited by **2 August 2004** so that the TC can subsequently prepare an OASIS Committee Draft. Committee members should send comments on this specification to the security-services@lists.oasis-open.org list. Others should submit them by filling in the form at http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security. The committee will publish vetted errata on the Security Services TC web page (<http://www.oasis-open.org/committees/security/>).

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

34 Table of Contents

35	1 Introduction.....	4
36	1.1 Notation.....	4
37	2 Metadata for SAML 2.0.....	5
38	2.1 Namespaces	5
39	2.2 Common Types.....	5
40	2.2.1 Simple Type entityIDType.....	5
41	2.2.2 Complex Type EndpointType.....	6
42	2.2.3 Complex Type IndexedEndpointType.....	6
43	2.2.4 Complex Type localizedNameType.....	7
44	2.2.5 Complex Type localizedURIType.....	7
45	2.3 Root Elements.....	7
46	2.3.1 Element <EntitiesDescriptor>.....	7
47	2.3.2 Element <EntityDescriptor>.....	8
48	2.3.2.1 Element <Organization>.....	10
49	2.3.2.2 Element <ContactPerson>.....	11
50	2.3.2.3 Element <AdditionalMetadataLocation>.....	12
51	2.4 Role Descriptor Elements.....	12
52	2.4.1 Element <RoleDescriptor>.....	12
53	2.4.1.1 Element <KeyDescriptor>.....	14
54	2.4.2 Complex Type SSODescriptorType.....	14
55	2.4.3 Element <IDPSSODescriptor>.....	15
56	2.4.4 Element <SPSSODescriptor>.....	16
57	2.4.5 Element <AuthnAuthorityDescriptor>.....	16
58	2.4.6 Element <PDPDescriptor>.....	17
59	2.4.7 Element <AttributeAuthorityDescriptor>.....	18
60	2.4.8 Element <AttributeConsumerDescriptor>.....	18
61	2.4.8.1 Element <AttributeConsumingService>.....	19
62	2.4.8.2 Element <RequestedAttribute>.....	20
63	2.5 Element <AffiliationDescriptor>.....	20
64	2.6 Examples.....	21
65	3 Signature Processing.....	22
66	3.1 XML Signature Profile.....	22
67	3.1.1 Signing Formats and Algorithms.....	22
68	3.1.2 References.....	22
69	3.1.3 Canonicalization Method.....	22
70	3.1.4 Transforms.....	23
71	3.1.5 KeyInfo.....	23
72	4 Metadata Publication and Resolution.....	24
73	4.1 Publication and Resolution via Well-Known Location.....	24
74	4.1.1 Publication.....	24
75	4.1.2 Resolution.....	24
76	4.2 Publishing and Resolution via DNS.....	24
77	4.2.1 Publication.....	25
78	4.2.1.1 First Well Known Rule.....	25
79	4.2.1.2 The Order Field.....	25
80	4.2.1.3 The Preference Field.....	25
81	4.2.1.4 The Flag Field.....	25

82	4.2.1.5 The Service Field.....	26
83	4.2.1.6 The Regex and Replacement Fields.....	26
84	4.2.2 NAPTR Examples.....	26
85	4.2.2.1 Entity Metadata NAPTR Examples.....	26
86	4.2.2.2 Name Identifier Examples.....	27
87	4.2.3 Resolution.....	27
88	4.2.3.1 Parsing the Unique Identifier.....	27
89	4.2.3.2 Obtaining Metadata via the DNS.....	27
90	4.2.4 Metadata Location Caching.....	28
91	4.3 Post-Processing of Metadata.....	28
92	4.3.1 Metadata Instance Caching.....	28
93	4.3.2 Handling of HTTPS Redirects.....	28
94	4.3.3 Processing of XML Signatures and General Trust Processing.....	28
95	4.3.3.1 Processing Signed DNS Zones.....	29
96	4.3.3.2 Processing Signed Documents and Fragments.....	29
97	4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL.....	29
98	5 References.....	30
99	Appendix A.Revision History.....	31
100	Appendix B.Notices.....	32

101 1 Introduction

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

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

110 1.1 Notation

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

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

115 `Example code listings appear like this.`

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

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

- 120 • The prefix `xsd`: stands for the W3C XML Schema namespace. [XMLSig]
- 121 • The prefix `xenc`: stands for W3C XML Encryption namespace. [XMLEnc]
- 122 • The prefix `md`: stands for the SAML metadata namespace, defined by this specification. If not
123 otherwise specified, elements are assumed to be in this namespace.
- 124 • The prefix `saml`: stands for the SAML assertion namespace [SAMLCore].
- 125 • The prefix `samlp`: stands for the SAML request-response protocol namespace [SAMLCore].
- 126 • The prefix `ds`: stands for the W3C XML Signature namespace,
127 `http://www.w3.org/2000/09/xmldsig#` [XMLSig].

2 Metadata for SAML 2.0

128

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

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

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

2.1 Namespaces

139

140 SAML Metadata defines the following namespace:

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

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

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

```
144 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:metadata"  
145 xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
146 elementFormDefault="unqualified" attributeFormDefault="unqualified"  
147 blockDefault="substitution" version="2.0">  
148  
149 <import namespace="http://www.w3.org/2000/09/xmldsig#"  
150 schemaLocation="xmldsig-core-schema.xsd"/>  
151  
152 <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"  
153 schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>  
154  
155 <import namespace="http://www.w3.org/XML/1998/namespace"  
156 schemaLocation="xml.xsd"/>
```

2.2 Common Types

157

158 The SAML 2.0 Metadata specification defines several types as described in the following subsections.
159 These types are used in defining SAML 2.0 Metadata elements and attributes.

2.2.1 Simple Type `entityIDType`

160

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

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

```
167 <simpleType name="entityIDType">  
168 <restriction base="anyURI">  
169 <maxLength value="1024"/>
```

```
170     </restriction>
171 </simpleType>
```

172 2.2.2 Complex Type EndpointType

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

176 Binding [Required]

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

179 Location [Required]

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

182 ResponseLocation [Optional]

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

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

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

```
188 <complexType name="EndpointType">
189   <sequence>
190     <any namespace="##other" processContents="lax"
191     minOccurs="0" maxOccurs="unbounded"/>
192   </sequence>
193   <attribute name="Binding" type="anyURI" use="required"/>
194   <attribute name="Location" type="anyURI" use="required"/>
195   <attribute name="ResponseLocation" type="anyURI" use="optional"/>
196   <anyAttribute namespace="##other" processContents="lax"/>
197 </complexType>
```

198 2.2.3 Complex Type IndexedEndpointType

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

202 index [Required]

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

205 isDefault [Optional]

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

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

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

```
213 <complexType name="IndexedEndpointType">
```

```

214     <complexContent>
215         <extension base="md:EndpointType">
216             <attribute name="index" type="unsignedShort"
217 use="required"/>
218             <attribute name="isDefault" type="boolean"
219 use="optional"/>
220         </extension>
221     </complexContent>
222 </complexType>

```

223 2.2.4 Complex Type localizedNameType

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

```

226 <complexType name="localizedNameType">
227     <simpleContent>
228         <extension base="string">
229             <attribute ref="xml:lang" use="required"/>
230         </extension>
231     </simpleContent>
232 </complexType>

```

233 2.2.5 Complex Type localizedURIType

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

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

```

237 <complexType name="localizedURIType">
238     <simpleContent>
239         <extension base="anyURI">
240             <attribute ref="xml:lang" use="required"/>
241         </extension>
242     </simpleContent>
243 </complexType>

```

244 2.3 Root Elements

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

248 2.3.1 Element `<EntitiesDescriptor>`

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

252 ID [Optional]

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

254 validUntil [Optional]

255 Optional attribute indicates the expiration time of the metadata contained in the element and any
256 contained elements.

257 cacheDuration [Optional]

258 Optional attribute indicates the maximum length of time a consumer should cache the metadata

259 contained in the element and any contained elements.

260 Name [Optional]

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

262 <ds:Signature> [Optional]

263 An XML signature that authenticates the containing element and its contents, as described in

264 Section 3.

265 <Extensions> [Optional]

266 This contains optional metadata extensions that are agreed upon between a metadata publisher

267 and consumer. Extensions MUST NOT include local (non-namespace-qualified) elements or

268 elements qualified by a SAML-defined namespace within this element.

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

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

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

272 or `cacheDuration` attribute. It is RECOMMENDED that only the root element of a metadata instance

273 contain either attribute.

274 The following schema fragment defines the <EntitiesDescriptor> element and its

275 **EntitiesDescriptorType** complex type:

```

276 <element name="EntitiesDescriptor" type="md:EntitiesDescriptorType"/>
277 <complexType name="EntitiesDescriptorType">
278   <sequence>
279     <element ref="ds:Signature" minOccurs="0"/>
280     <element ref="md:Extensions" minOccurs="0"/>
281     <choice minOccurs="1" maxOccurs="unbounded">
282       <element ref="md:EntityDescriptor"/>
283       <element ref="md:EntitiesDescriptor"/>
284     </choice>
285   </sequence>
286   <attribute name="validUntil" type="dateTime" use="optional"/>
287   <attribute name="cacheDuration" type="duration" use="optional"/>
288   <attribute name="ID" type="ID" use="optional"/>
289   <attribute name="Name" type="string" use="optional"/>
290 </complexType>
291 <element name="Extensions" type="md:ExtensionsType"/>
292 <complexType final="#all" name="ExtensionsType">
293   <sequence>
294     <any namespace="##other" processContents="lax"
295     maxOccurs="unbounded"/>
296   </sequence>
297 </complexType>

```

298 2.3.2 Element <EntityDescriptor>

299 The <EntityDescriptor> element specifies metadata for a single SAML entity. A single entity may act

300 in many different roles in the support of multiple profiles. This specification directly supports the following

301 concrete roles as well as the abstract <RoleDescriptor> element for extensibility (see subsequent

302 sections for more details):

- 303 • SSO Identity Provider
- 304 • SSO Service Provider
- 305 • Authentication Authority
- 306 • Attribute Authority

- 307 • Attribute Requester
- 308 • Policy Decision Point
- 309 • Affiliation

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

311 `entityID` [Required]

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

314 `ID` [Optional]

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

316 `validUntil` [Optional]

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

319 `cacheDuration` [Optional]

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

322 `<ds:Signature>` [Optional]

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

325 `<Extensions>` [Optional]

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

329 `<RoleDescriptor>`, `<IDPSSODescriptor>`, `<SPSSODescriptor>`,
330 `<AuthnAuthorityDescriptor>`, `<AttributeAuthorityDescriptor>`,
331 `<AttributeConsumerDescriptor>`, `<PDPDescriptor>`, or any element from a non-SAML
332 namespace [One or More]

333 **OR**

334 `<AffiliationDescriptor>` [Required]

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

337 `<Organization>` [Optional]

338 Optional element identifying the organization responsible for the SAML entity described by the
339 element.

340 `<ContactPerson>` [Zero or More]

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

342 `<AdditionalMetadataLocation>` [Zero or More]

343 Optional sequence of namespace-qualified locations where additional metadata exists for the
344 SAML entity. This may include metadata in alternate formats or describing adherence to other
345 non-SAML specifications.

346 Arbitrary qualified attributes from non-SAML namespaces may also be included.

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

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

```
352 <element name="EntityDescriptor" type="md:EntityDescriptorType"/>
353 <complexType name="EntityDescriptorType">
354   <sequence>
355     <element ref="ds:Signature" minOccurs="0"/>
356     <element ref="md:Extensions" minOccurs="0"/>
357     <choice>
358       <choice maxOccurs="unbounded">
359         <element ref="md:RoleDescriptor"/>
360         <element ref="md:IDPSSODescriptor"/>
361         <element ref="md:SPSSODescriptor"/>
362         <element ref="md:AuthnAuthorityDescriptor"/>
363         <element
364 ref="md:AttributeAuthorityDescriptor"/>
365         <element
366 ref="md:AttributeConsumerDescriptor"/>
367         <element ref="md:PDPDescriptor"/>
368         <any namespace="##other"
369 processContents="lax"/>
370       </choice>
371       <element ref="md:AffiliationDescriptor"/>
372     </choice>
373     <element ref="md:Organization" minOccurs="0"/>
374     <element ref="md:ContactPerson" minOccurs="0"
375 maxOccurs="unbounded"/>
376     <element ref="md:AdditionalMetadataLocation" minOccurs="0"
377 maxOccurs="unbounded"/>
378   </sequence>
379   <attribute name="entityID" type="md:entityIDType" use="required"/>
380   <attribute name="validUntil" type="dateTime" use="optional"/>
381   <attribute name="cacheDuration" type="duration" use="optional"/>
382   <attribute name="ID" type="ID" use="optional"/>
383   <anyAttribute namespace="##other" processContents="lax"/>
384 </complexType>
```

385 2.3.2.1 Element `<Organization>`

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

390 `<Extensions>` [Optional]

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

394 `<OrganizationName>` [One or More]

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

396 `<OrganizationDisplayName>` [One or More]

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

398 `<OrganizationURL>` [One or More]

399 One or more language-qualified URIs that specify a location to which to direct a principal for
400 additional information. Note that the language qualifier refers to the content of the material at the

401 specified location.

402 Arbitrary qualified attributes from non-SAML namespaces may also be included.

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

```
405 <element name="Organization" type="md:OrganizationType"/>
406 <complexType name="OrganizationType">
407   <sequence>
408     <element ref="md:Extensions" minOccurs="0"/>
409     <element ref="md:OrganizationName" maxOccurs="unbounded"/>
410     <element ref="md:OrganizationDisplayName"
411 maxOccurs="unbounded"/>
412     <element ref="md:OrganizationURL" maxOccurs="unbounded"/>
413   </sequence>
414   <anyAttribute namespace="##other" processContents="lax"/>
415 </complexType>
416 <element name="OrganizationName" type="md:localizedNameType"/>
417 <element name="OrganizationDisplayName" type="md:localizedNameType"/>
418 <element name="OrganizationURL" type="md:localizedURIType"/>
```

419 2.3.2.2 Element <ContactPerson>

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

424 **contactType** [Required]

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

427 <Extensions> [Optional]

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

431 <Company> [Optional]

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

433 <GivenName> [Optional]

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

435 <SurName> [Optional]

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

437 <EmailAddress> [Zero or More]

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

440 <TelephoneNumber> [Zero or More]

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

442 Arbitrary qualified attributes from non-SAML namespaces may also be included.

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

```
445 <element name="ContactPerson" type="md:ContactType"/>
```

```

446 <complexType name="ContactType">
447   <sequence>
448     <element ref="md:Extensions" minOccurs="0"/>
449     <element ref="md:Company" minOccurs="0"/>
450     <element ref="md:GivenName" minOccurs="0"/>
451     <element ref="md:SurName" minOccurs="0"/>
452     <element ref="md:EmailAddress" minOccurs="0"
453 maxOccurs="unbounded"/>
454     <element ref="md:TelephoneNumber" minOccurs="0"
455 maxOccurs="unbounded"/>
456   </sequence>
457   <attribute name="contactType" type="md:ContactTypeType"
458 use="required"/>
459   <anyAttribute namespace="##other" processContents="lax"/>
460 </complexType>
461 <element name="Company" type="string"/>
462 <element name="GivenName" type="string"/>
463 <element name="SurName" type="string"/>
464 <element name="EmailAddress" type="anyURI"/>
465 <element name="TelephoneNumber" type="string"/>
466 <simpleType name="ContactTypeType">
467   <restriction base="string">
468     <enumeration value="technical"/>
469     <enumeration value="support"/>
470     <enumeration value="administrative"/>
471     <enumeration value="billing"/>
472     <enumeration value="other"/>
473   </restriction>
474 </simpleType>

```

475 2.3.2.3 Element <AdditionalMetadataLocation>

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

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

```

483 <element name="AdditionalMetadataLocation"
484 type="md:AdditionalMetadataLocationType"/>
485 <complexType name="AdditionalMetadataLocationType">
486   <simpleContent>
487     <extension base="anyURI">
488       <attribute name="namespace" type="anyURI"
489 use="required"/>
490     </extension>
491   </simpleContent>
492 </complexType>

```

493 2.4 Role Descriptor Elements

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

497 2.4.1 Element <RoleDescriptor>

498 The <RoleDescriptor> element is an abstract extension point that contains common descriptive
499 information intended to provide processing commonality across different roles. New roles can be defined

500 by extending its abstract **RoleDescriptorType** complex type, which contains the following elements and
501 attributes:

502 ID [Optional]
503 A document-unique identifier for the element, typically used as a reference point when signing.

504 validUntil [Optional]
505 Optional attribute indicates the expiration time of the metadata contained in the element and any
506 contained elements.

507 cacheDuration [Optional]
508 Optional attribute indicates the maximum length of time a consumer should cache the metadata
509 contained in the element and any contained elements.

510 protocolSupportEnumeration [Required]
511 A space-delimited set of URIs that identify the set of protocol specifications supported by the role
512 element. For SAML 2.0 entities, this set MUST include the SAML protocol namespace URI,
513 urn:oasis:names:tc:SAML:2.0:protocol.

514 errorURL [Optional]
515 Optional URI attribute that specifies a location to direct a user for problem resolution and
516 additional support related to this role.

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

520 <Extensions> [Optional]
521 This contains optional metadata extensions that are agreed upon between a metadata publisher
522 and consumer. Extensions MUST NOT include local (non-namespace-qualified) elements or
523 elements qualified by a SAML-defined namespace within this element.

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

527 <Organization> [Optional]
528 Optional element specifies the organization associated with this role. Identical to the element used
529 within the <EntityDescriptor> element.

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

533 Arbitrary qualified attributes from non-SAML namespaces may also be included.

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

```

536 <element name="RoleDescriptor" type="md:RoleDescriptorType"/>
537 <complexType name="RoleDescriptorType" abstract="true">
538   <sequence>
539     <element ref="ds:Signature" minOccurs="0"/>
540     <element ref="md:Extensions" minOccurs="0"/>
541     <element ref="md:KeyDescriptor" minOccurs="0"
542     maxOccurs="unbounded"/>
543     <element ref="md:Organization" minOccurs="0"/>

```

```

544         <element ref="md:ContactPerson" minOccurs="0"
545 maxOccurs="unbounded"/>
546     </sequence>
547     <attribute name="ID" type="ID" use="optional"/>
548     <attribute name="validUntil" type="dateTime" use="optional"/>
549     <attribute name="cacheDuration" type="duration" use="optional"/>
550     <attribute name="protocolSupportEnumeration" type="NMTOKENS"
551 use="required"/>
552     <attribute name="errorURL" type="anyURI" use="optional"/>
553     <anyAttribute namespace="##other" processContents="lax"/>
554 </complexType>

```

555 2.4.1.1 Element <KeyDescriptor>

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

559 use [Optional]

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

562 <ds:KeyInfo> [Required]

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

565 <xenc:EncryptionMethod> [Zero or More]

566 Optional element specifying an algorithm and algorithm-specific settings supported by the entity.
567 The exact content varies based on the algorithm supported.

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

```

570 <element name="KeyDescriptor" type="md:KeyDescriptorType"/>
571 <complexType name="KeyDescriptorType">
572     <sequence>
573         <element ref="ds:KeyInfo"/>
574         <element ref="xenc:EncryptionMethod minOccurs="0"
575 maxOccurs="unbounded"/>
576     </sequence>
577     <attribute name="use" type="md:KeyTypes" use="optional"/>
578 </complexType>
579 <simpleType name="KeyTypes">
580     <restriction base="string">
581         <enumeration value="encryption"/>
582         <enumeration value="signing"/>
583     </restriction>
584 </simpleType>

```

585 2.4.2 Complex Type SSODescriptorType

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

590 <ArtifactResolutionService> [Zero or More]

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

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

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

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

```
601 <complexType name="SSODescriptorType" abstract="true">  
602   <complexContent>  
603     <extension base="md:RoleDescriptorType">  
604       <sequence>  
605         <element ref="md:ArtifactResolutionService"  
606         minOccurs="0" maxOccurs="unbounded"/>  
607         <element ref="md:SingleLogoutService"  
608         minOccurs="0" maxOccurs="unbounded"/>  
609         <element ref="md:ManageNameIDService"  
610         minOccurs="0" maxOccurs="unbounded"/>  
611       </sequence>  
612     </extension>  
613   </complexContent>  
614 </complexType>  
615 <element name="ArtifactResolutionService" type="md:IndexedEndpointType"/>  
616 <element name="SingleLogoutService" type="md:EndpointType"/>  
617 <element name="ManageNameIDService" type="md:EndpointType"/>
```

618 2.4.3 Element <IDPSSODescriptor>

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

622 WantAuthnRequestsSigned [Optional]
623 Optional attribute that indicates a requirement for the <samlp:AuthnRequest> messages
624 received by this identity provider to be signed. If omitted, the value is assumed to be false.

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

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

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

```
635 <element name="IDPSSODescriptor" type="md:IDPSSODescriptorType"/>  
636 <complexType name="IDPSSODescriptorType">  
637   <complexContent>  
638     <extension base="md:SSODescriptorType">  
639       <sequence>  
640         <element ref="md:SingleSignOnService"  
641         maxOccurs="unbounded"/>  
642         <element ref="md:NameIDMappingService"  
643         minOccurs="0" maxOccurs="unbounded"/>
```

```

644         </sequence>
645         <attribute name="WantAuthnRequestsSigned"
646 type="boolean" use="optional"/>
647     </extension>
648 </complexContent>
649 </complexType>
650 <element name="SingleSignOnService" type="md:EndpointType"/>
651 <element name="NameIDMappingService" type="md:EndpointType"/>

```

652 2.4.4 Element <SPSSODescriptor>

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

656 AuthnRequestsSigned [Optional]

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

659 WantAssertionsSigned [Optional]

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

664 <AssertionConsumerService> [One or More]

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

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

```

670 <element name="SPSSODescriptor" type="md:SPSSODescriptorType"/>
671 <complexType name="SPSSODescriptorType">
672     <complexContent>
673         <extension base="md:SSODescriptorType">
674             <sequence>
675                 <element ref="md:AssertionConsumerService"
676 maxOccurs="unbounded"/>
677             </sequence>
678             <attribute name="AuthnRequestsSigned" type="boolean"
679 use="optional"/>
680             <attribute name="WantAssertionsSigned" type="boolean"
681 use="optional"/>
682         </extension>
683     </complexContent>
684 </complexType>
685 <element name="AssertionConsumerService"
686 type="md:AssertionConsumerServiceType"/>

```

687 2.4.5 Element <AuthnAuthorityDescriptor>

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

691 <AuthnQueryService> [One or More]

692 One or more elements of type **EndpointType** that describe endpoints that support the profile of
693 the Authentication Query protocol defined in [SAMLProf]. All authentication authorities support at

694 least one such endpoint, by definition.

695 <AssertionIDRequestService> [Zero or More]

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

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

```
701 <element name="AuthnAuthorityDescriptor"  
702 type="md:AuthnAuthorityDescriptorType"/>  
703 <complexType name="AuthnAuthorityDescriptorType">  
704 <complexContent>  
705 <extension base="md:RoleDescriptorType">  
706 <sequence>  
707 <element ref="md:AuthnQueryService"  
708 maxOccurs="unbounded"/>  
709 <element ref="md:AssertionIDRequestService"  
710 minOccurs="0" maxOccurs="unbounded"/>  
711 </sequence>  
712 </extension>  
713 </complexContent>  
714 </complexType>  
715 <element name="AuthnQueryService" type="md:EndpointType"/>  
716 <element name="AssertionIDRequestService" type="md:EndpointType"/>
```

717 2.4.6 Element <PDPDescriptor>

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

721 <AuthzService> [One or More]

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

725 <AssertionIDRequestService> [Zero or More]

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

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

```
731 <element name="PDPDescriptor" type="md:PDPDescriptorType"/>  
732 <complexType name="PDPDescriptorType">  
733 <complexContent>  
734 <extension base="md:RoleDescriptorType">  
735 <sequence>  
736 <element ref="md:AuthzService"  
737 maxOccurs="unbounded"/>  
738 <element ref="md:AssertionIDRequestService"  
739 minOccurs="0" maxOccurs="unbounded"/>  
740 </sequence>  
741 </extension>  
742 </complexContent>  
743 </complexType>  
744 <element name="AuthzService" type="md:EndpointType"/>
```

745 **2.4.7 Element <AttributeAuthorityDescriptor>**

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

750 <AttributeService> [One or More]

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

754 <AssertionIDRequestService> [Zero or More]

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

758 <saml:AttributeDesignator> [Zero or More]

759 Zero or more attribute designators that identify attributes supported by the authority.

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

```
762 <element name="AttributeAuthorityDescriptor"  
763 type="md:AttributeAuthorityDescriptorType"/>  
764 <complexType name="AttributeAuthorityDescriptorType">  
765 <complexContent>  
766 <extension base="md:RoleDescriptorType">  
767 <sequence>  
768 <element ref="md:AttributeService"  
769 maxOccurs="unbounded"/>  
770 <element ref="md:AssertionIDRequestService"  
771 minOccurs="0" maxOccurs="unbounded"/>  
772 <element ref="saml:AttributeDesignator"  
773 minOccurs="0" maxOccurs="unbounded"/>  
774 </sequence>  
775 </extension>  
776 </complexContent>  
777 </complexType>  
778 <element name="AttributeService" type="md:EndpointType"/>
```

779 **2.4.8 Element <AttributeConsumerDescriptor>**

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

783 <AttributeConsumingService> [One or More]

784 One or more elements that describe a service provided by the entity that requires or desires the
785 use of SAML attributes.

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

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

```

792 <element name="AttributeConsumerDescriptor"
793 type="md:AttributeConsumerDescriptorType"/>
794 <complexType name="AttributeConsumerDescriptorType">
795   <complexContent>
796     <extension base="md:RoleDescriptorType">
797       <sequence>
798         <element ref="md:AttributeConsumingService"
799 maxOccurs="unbounded"/>
800       </sequence>
801     </extension>
802   </complexContent>
803 </complexType>

```

804 **2.4.8.1 Element <AttributeConsumingService>**

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

808 index [Required]

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

811 isDefault [Optional]

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

814 WantAssertionsSigned [Optional]

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

819 <ServiceName> [One or More]

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

821 <ServiceDescription> [Zero or More]

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

823 <RequestedAttribute> [One or More]

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

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

```

827 <element name="AttributeConsumingService"
828 type="md:AttributeConsumingServiceType"/>
829 <complexType name="AttributeConsumingServiceType">
830   <sequence>
831     <element ref="md:ServiceName" maxOccurs="unbounded"/>
832     <element ref="md:ServiceDescription" minOccurs="0"
833 maxOccurs="unbounded"/>
834     <element ref="md:RequestedAttribute" maxOccurs="unbounded"/>
835   </sequence>
836   <attribute name="index" type="unsignedShort" use="required"/>
837   <attribute name="isDefault" type="boolean" use="optional"/>
838   <attribute name="WantAssertionsSigned" type="boolean"
839 use="optional"/>
840 </complexType>
841 <element name="ServiceName" type="md:localizedNameType"/>

```

```
842 <element name="ServiceDescription" type="md:localizedNameType"/>
```

843 2.4.8.2 Element <RequestedAttribute>

844 The <RequestedAttribute> element specifies an attribute consumer's interest in a specific SAML attribute.
845 Its **RequestedAttributeType** complex type extends the **saml:AttributeDesignatorType** with the following
846 attribute:

847 **isRequired** [Optional]

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

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

```
852 <element name="RequestedAttribute" type="md:RequestedAttributeType"/>  
853 <complexType name="RequestedAttributeType">  
854   <complexContent>  
855     <extension base="saml:AttributeDesignatorType">  
856       <attribute name="isRequired" type="boolean"  
857       use="optional"/>  
858     </extension>  
859   </complexContent>  
860 </complexType>
```

861 2.5 Element <AffiliationDescriptor>

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

868 **affiliationOwnerID** [Required]

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

872 **ID** [Optional]

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

874 **validUntil** [Optional]

875 Optional attribute indicates the expiration time of the metadata contained in the element and any
876 contained elements.

877 **cacheDuration** [Optional]

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

880 <ds:Signature> [Optional]

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

883 <Extensions> [Optional]

884 This contains optional metadata extensions that are agreed upon between a metadata publisher
885 and consumer. Extensions MUST NOT include local (non-namespace-qualified) elements or

886 elements qualified by a SAML-defined namespace within this element.

887 <AffiliateMember> [One or More]

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

890 <KeyDescriptor> [Zero or More]

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

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

```
896 <element name="AffiliationDescriptor"  
897 type="md:AffiliationDescriptorType"/>  
898 <complexType name="AffiliationDescriptorType">  
899   <sequence>  
900     <element ref="ds:Signature" minOccurs="0"/>  
901     <element ref="md:Extensions" minOccurs="0"/>  
902     <element ref="md:AffiliateMember" maxOccurs="unbounded"/>  
903     <element ref="md:KeyDescriptor" minOccurs="0"  
904 maxOccurs="unbounded"/>  
905   </sequence>  
906   <attribute name="affiliationOwnerID" type="md:entityIDType"  
907 use="required"/>  
908   <attribute name="validUntil" type="dateTime" use="optional"/>  
909   <attribute name="cacheDuration" type="duration" use="optional"/>  
910   <attribute name="ID" type="ID" use="optional"/>  
911   <anyAttribute namespace="##other" processContents="lax"/>  
912 </complexType>  
913 <element name="AffiliateMember" type="md:entityIDType"/>
```

914 2.6 Examples

915 TODO

916 3 Signature Processing

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

- 919 • Metadata integrity
- 920 • Authentication of the metadata by a trusted signer

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

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

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

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

930 3.1 XML Signature Profile

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

936 3.1.1 Signing Formats and Algorithms

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

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

942 3.1.2 References

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

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

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

951 3.1.3 Canonicalization Method

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

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

956 **3.1.4 Transforms**

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

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

966 **3.1.5 KeyInfo**

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

970 4 Metadata Publication and Resolution

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

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

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

984 4.1 Publication and Resolution via Well-Known Location

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

986 4.1.1 Publication

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

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

1000 4.1.2 Resolution

1001 If an entity's unique identifier is a URL, metadata consumers MAY attempt to resolve an entity's unique
1002 identifier directly, in a scheme-specific manner, by dereferencing the identifier.

1003 4.2 Publishing and Resolution via DNS

1004 To improve the accessibility of metadata documents and provide additional indirection between an entity's
1005 unique identifier and the location of metadata, entities MAY publish their metadata document locations in a
1006 zone of their corresponding DNS [RFC1034]. The entity's unique identifier (a URI) is used as the input to
1007 the process. Since URIs are flexible identifiers, location publication methods and the resolution process
1008 are determined by the URI's scheme and fully-qualified name. URI locations for metadata are
1009 subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in [RFC2915]

1010 and [\[RFC3403\]](#).

1011 It is RECOMMENDED that entities publish their resource records in signed zone files using [\[RFC2535\]](#)
1012 such that relying parties may establish the validity of the published location and authority of the zone, and
1013 integrity of the DNS response. If DNS zone signatures are present, relying parties MUST properly validate
1014 the signature.

1015 **4.2.1 Publication**

1016 This specification makes use of the NAPTR resource record described in [\[RFC2915\]](#) and [\[RFC3403\]](#).
1017 Familiarity with these documents is encouraged.

1018 Dynamic Delegation Discovery System (DDDS) [\[RFC3401\]](#) is a general purpose system for the retrieval of
1019 information based on an application-specific input string and the application of well known rules to
1020 transform that string until a terminal condition is reached requiring a look-up into an application-specific
1021 defined database or resolution of a URL based on the rules defined by the application. DDDS defines a
1022 specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS
1023 necessary to apply DDDS rules.

1024 Entities MAY publish separate URLs when multiple metadata documents need to be distributed, or when
1025 different metadata documents are required due to multiple trust relationships that require separate keying
1026 material, or when service interfaces require separate metadata declarations. This may be accomplished
1027 through the use of the optional `<AdditionalMetaLocation>` element, or through the regexp facility and
1028 multiple service definition fields in the NAPTR resource record itself.

1029 If the entity's unique identifier is a URN, publication of the corresponding metadata location proceeds as
1030 specified in [\[RFC3404\]](#). Otherwise, the resolution of the metadata location proceeds as specified below.

1031 The following is the application-specific profile of DDDS for SAML metadata resolution.

1032 **4.2.1.1 First Well Known Rule**

1033 The "first well-known-rule" for processing SAML metadata resolution is to parse the entity's unique
1034 identifier and extract the fully-qualified domain name (subexpression 3) as described in Section "[Parsing](#)
1035 [the providerID](#)".

1036 **4.2.1.2 The Order Field**

1037 The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY
1038 provide multiple NAPTR resource records which MUST be processed by the resolver application in the
1039 order indicated by this field.

1040 **4.2.1.3 The Preference Field**

1041 For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving
1042 application. The resolving application MAY ignore this order, in cases where the service field value does
1043 not meet the resolver's requirements (e.g.: the resource record returns a protocol the application does not
1044 support).

1045 **4.2.1.4 The Flag Field**

1046 SAML metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying
1047 additional resource records are to be processed). The "U" flag indicates that the output of the rule is a
1048 URI.

1049 4.2.1.5 The Service Field

1050 The SAML-specific service field, as described in the following BNF, declares the modes by which instance
1051 document(s) shall be made available:

```
1052 servicefield = 1("PID2U" / "NID2U") "+" proto [*(":" class) *(":" servicetype)]
1053 proto = 1("https" / "uddi")
1054 class = 1[ "entity" / "entitygroup" ]
1055 servicetype = 1(si / "spssso" / "idpssso" / "authn" / "authnauth" / "pdp" / "attrauth" /
1056 "attrcons" / alphanum )
1057 si = "si" [ ":" alphanum ] [ ":" endpoint ]
1058 alphanum = 1*32(ALPHA / DIGIT)
```

1059 where:

- 1060 • servicefield PID2U resolves an entity's unique identifier to metadata URL.
- 1061 • servicefield NID2U resolves a principal's <NameIdentifier> into a metadata URL.
- 1062 • proto describes the retrieval protocol (`https` or `uddi`). In the case of UDDI, the URL will be an
1063 http(s) URL referencing a WSDL document.
- 1064 • class identifies whether the referenced metadata document describes a single entity, or multiple.
1065 In the latter case, the referenced document MUST contain the entity defined by the original unique
1066 identifier as a member of a group of entities within the document itself such as an
1067 <AffiliationDescriptor> or <EntitiesDescriptor>.
- 1068 • servicetype allows an entity to publish metadata for distinct roles and services as separate
1069 documents. Resolvers who encounter multiple servicetype declarations will dereference the
1070 appropriate URI, depending on which service is required for an operation (e.g.: an entity operating
1071 both as an identity provider and a service provider can publish metadata for each role at different
1072 locations). The authn service type represents a <SingleSignOnService> endpoint.
- 1073 • si (with optional endpoint component) allows the publisher to either directly publish the metadata
1074 for a service instance, or by articulating a SOAP endpoint (using endpoint).

1075 For example:

- 1076 • PID2U+https:entity - represents the entity's complete metadata document available via the
1077 https protocol
- 1078 • PID2U+uddi:entity:si:foo - represents the WSDL document location that describes a service
1079 instance "foo"
- 1080 • PID2U+https:entitygroup:idpssso - represents the metadata for a group of entities acting as
1081 SSO identity providers, of which the original entity is a member.
- 1082 • NID2U+https:idp - represents the metadata for the SSO identity provider of a principal

1083 4.2.1.6 The Regex and Replacement Fields

1084 The expected output after processing the input string through the regex MUST be a valid https URL or
1085 UDDI node (WSDL document) address.

1086 4.2.2 NAPTR Examples

1087 4.2.2.1 Entity Metadata NAPTR Examples

1088 Entities publish metadata URLs in the following manner:

```
1089 $ORIGIN provider.biz
1090
```

```

1091      ; order pref f service regexp or replacement
1092
1093      IN NAPTR 100 10 "U" PID2U+https:entity
1094          "!^.*$!https://host.provider.biz/some/directory/trust.xml!" ""
1095      IN NAPTR 110 10 "U" PID2U+https: entity:trust
1096          "!^.*$!https://foo.provider.biz:1443/mdtrust.xml!" ""
1097      IN NAPTR 125 10 "U" PID2U+https:"
1098      IN NAPTR 110 10 "U" PID2U+uddi:entity
1099          "!^.*$!https://this.uddi.node.provider.biz/libmd.wsdl" ""

```

1100 4.2.2.2 Name Identifier Examples

1101 A principal's employer `example.int` operates an identity provider which may be used by an office supply
 1102 company to authenticate authorized buyers. The supplier takes a users' email address
 1103 `buyer@example.int` as input to the resolution process, and parses the email address to extract the
 1104 FQDN (`example.int`). The employer publishes the following NAPTR record in the `example.int` DNS:

```

1105      $ORIGIN example.int
1106
1107      IN NAPTR 100 10 "U" NID2U+https:authn
1108          "!^([\^@]+)@(.*)$!https://serv.example.int:8000/cgi-bin/getmd?\1!"
1109      ""
1110      IN NAPTR 100 10 "U" NID2U+https:idp
1111          "!^([\^@]+)@(.*)$!https://auth.example.int/app/auth?\1" ""

```

1112 4.2.3 Resolution

1113 When resolving metadata for an entity via the DNS, the unique identifier of the entity is used as the initial
 1114 input into the resolution process, rather than as an actual location Proceed as follows:

- 1115 • If the unique identifier is a URN, proceed with the resolution steps as defined in [\[RFC3404\]](#).
- 1116 • Otherwise, parse the identifier to obtain the fully-qualified domain name.
- 1117 • Query the DNS for NAPTR resource records of the domain iteratively until a terminal resource
 1118 record is returned.
- 1119 • Identify which resource record to use based on the service fields, then order fields, then preference
 1120 fields of the result set.
- 1121 • Obtain the document(s) at the provided location(s) as required by the application.

1122 4.2.3.1 Parsing the Unique Identifier

1123 To initiate the resolution of the location of the metadata information, it will be necessary in some cases to
 1124 decompose the entity's unique identifier (expressed as a URI) into one or more atomic elements.

1125 The following regular expression should be used when initiating the decomposition process:

```

1126      ^([\^:/?#]+)?/*([\^:/?#]*@)?(((\^/?:#*\.)*((\^/?#:\.])\.([\^/?#:\.]+))
1127      (: \d+)?([\^?#]*)(\?[\^#]*)?(#[.]*)?$
1128      1           2           34           56           7
1129      8           9           10          11

```

1130 Subexpression 3 MUST result in a Fully-Qualified Domain Name (FQDN), which will be the basis for
 1131 retrieving metadata locations from this zone.

1132 4.2.3.2 Obtaining Metadata via the DNS

1133 Upon completion of the parsing of the identifier, the application then performs a DNS query for the resulting
 1134 domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses.
 1135 Applications MAY exclude from the result set any service definitions that do not concern the present
 1136 request operations.

1137 Resolving applications MUST subsequently order the result set according to the order field, and MAY
1138 order the result set based on the preference set. Resolvers are NOT REQUIRED to follow the ordering of
1139 the preferences field. The resulting NAPTR resource record(s) are operated on iteratively (based on the
1140 order flag) until a terminal NAPTR resource record is reached.

1141 The result will be a well-formed, absolute URL, which is then used to retrieve the metadata document.

1142 **4.2.4 Metadata Location Caching**

1143 Location caching MUST NOT exceed the TTL of the DNS zone from which the location was derived.
1144 Resolvers MUST obtain a fresh copy of the metadata location upon reaching the expiration of the TTL of
1145 the zone.

1146 Publishers of metadata documents should carefully consider the TTL of the zone when making changes
1147 to metadata document locations. Should such a location change occur, a publisher MUST either keep the
1148 document at both the old and new location until all conforming resolvers are certain to have the updated
1149 location (e.g.: time of zone change + TTL), or provide an HTTP Redirect [RFC2616] response at the old
1150 location specifying the new location.

1151 **4.3 Post-Processing of Metadata**

1152 The following sections describe the post-processing of metadata.

1153 **4.3.1 Metadata Instance Caching**

1154 Document caching MUST NOT exceed the `validUntil` or `cacheDuration` attribute of the subject
1155 element(s). If metadata elements have parent elements which contain caching policies, the parent
1156 element takes precedence.

1157 To properly process the `cacheDuration` attribute, consumers MUST retain the date and time when the
1158 document was retrieved.

1159 When a document or element has expired, the consumer MUST retrieve a fresh copy, which may require
1160 a refresh of the document location(s). Consumers SHOULD process document cache processing
1161 according to [RFC2616] Section 13, and MAY request the Last-Modified date and time from the HTTP
1162 server. Publishers SHOULD ensure acceptable cache processing as described in [RFC2616] (Section
1163 10.3.5 304 Not Modified).

1164 **4.3.2 Handling of HTTPS Redirects**

1165 Publishers MAY issue an HTTP Redirect (301 Moved Permanently, 302 or 307 Temporary Redirect)
1166 [RFC2616], and user agents MUST follow the specified URL in the Redirect response. Redirects
1167 SHOULD be of the same protocol as the initial request.

1168 **4.3.3 Processing of XML Signatures and General Trust Processing**

1169 Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and
1170 for the trust ascribed to the entity described by such metadata:

- 1171 • Trust derived from the signature of the DNS zone from which the metadata location URL was
1172 resolved, ensuring accuracy of the metadata document location(s)
- 1173 • Trust derived from signature processing of the metadata document itself, ensuring the integrity of
1174 the XML document
- 1175 • Trust derived from the SSL/TLS server authentication of the metadata location URL, ensuring the
1176 identity of the publisher of the metadata

1177 Post-processing of the metadata document MUST include signature processing at the XML-document
1178 level and MAY include one of the other two processes. Specifically, the relying party MAY choose to trust
1179 any of the cited authorities in the resolution and parsing process. Publishers of metadata MUST employ a
1180 document-integrity mechanism and MAY employ any of the other two processing profiles to establish trust
1181 in the metadata document, governed by implementation policies.

1182 **4.3.3.1 Processing Signed DNS Zones**

1183 Verification of DNS zone signature SHOULD be processed, if present, as described in [\[RFC2535\]](#).

1184 **4.3.3.2 Processing Signed Documents and Fragments**

1185 Published metadata documents SHOULD be signed, as described in Section 3, either by a certificate
1186 issued to the subject of the document, or by another trusted party. Publishers MAY consider signatures of
1187 other parties as a means of trust conveyance.

1188 Metadata consumers MUST validate signatures, when present, on the metadata document as described
1189 by Section 3.

1190 **4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL**

1191 It is STRONGLY RECOMMENDED that publishers implement TLS/SSL URLs; therefore, consumers
1192 SHOULD consider the trust inherited from the issuer of the TLS/SSL certificate. Publication URLs may not
1193 always be located in the domain of the subject of the metadata document; therefore, consumers SHOULD
1194 NOT presume certificates whose subject is the entity in question, as it may be hosted by another trusted
1195 party.

1196 As the basis of this trust may not be available against a cached document, other mechanisms SHOULD
1197 be used under such circumstances.

1198

5 References

- 1199 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
1200 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1201 **[SAMLCore]** P. Hallam-Baker, P., and E. Maler, (Editors), *Assertions and Protocol for the*
1202 *OASIS Security Assertion Markup Language (SAML)*, Committee Specification
1203 01, available from <http://www.oasis-open.org/committees/security>, OASIS, May
1204 2002.
- 1205 **[SAMLBind]** E. Maler et al. *Bindings for the OASIS Security Assertion Markup Language*
1206 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-bindings-2.0.
1207 <http://www.oasis-open.org/committees/security/>.
- 1208 **[SAMLProf]** E. Maler et al. *Profiles for the OASIS Security Assertion Markup Language*
1209 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-profiles-2.0.
1210 <http://www.oasis-open.org/committees/security/>.
- 1211 **[SAMLSecure]** Security and Privacy Considerations for the OASIS Security Assertion Markup
1212 Language (SAML), [http://www.oasis-open.org/committees/security/docs/cs-sstc-](http://www.oasis-open.org/committees/security/docs/cs-sstc-sec-consider-01.doc)
1213 sec-consider-01.doc.
- 1214 **[XMLSig]** D. Eastlake et al., *XML-Signature Syntax and Processing*,
1215 <http://www.w3.org/TR/xmlsig-core/>, World Wide Web Consortium.
- 1216 **[XMLEnc]** D. Eastlake et al., *XML-Encryption Syntax and Processing*,
1217 <http://www.w3.org/TR/xmlenc-core/>, World Wide Web Consortium.

Appendix A.Revision History

Rev	Date	By Whom	What
0	15 Sep 2003	Jahan Moreh	Initial draft based on Draft 07 of SAML 1.1 Metadata specification.
1	15 Mar 2004	Jahan Moreh	Major rewrite based on schema defined by Scott Cantor
2	30 Mar 2004	Jahan Moreh	Completed all sections based on schema defined by Scott Cantor.
3	31 Mar 2004	Jahan Moreh	Minor modifications based on author's review
4	24 May 2004	Jahan Moreh	Rewrite based on version 2.0 of metadata schema by Scott Cantor
5	26 May 2004	Peter Davis	Addition of Resolution and post-processing.
6	8 Jun 2004	Scott Cantor	Restructured content to match style of SAML core spec, schema changes to align with core-14.
7	4 Jul 2004	Scott Cantor	Filled in some missing elements, incorporated F2F feedback, reworked encryption metadata, reworked section 4, added signature profile.
8	13 Jul 2004	Eve Maler	Final cleanup in preparation for last-call working draft publication.

1220

Appendix B. Notices

1221 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
1222 might be claimed to pertain to the implementation or use of the technology described in this document or
1223 the extent to which any license under such rights might or might not be available; neither does it represent
1224 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
1225 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
1226 available for publication and any assurances of licenses to be made available, or the result of an attempt
1227 made to obtain a general license or permission for the use of such proprietary rights by implementors or
1228 users of this specification, can be obtained from the OASIS Executive Director.
1229 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
1230 other proprietary rights which may cover technology that may be required to implement this specification.
1231 Please address the information to the OASIS Executive Director.

1232 **Copyright © OASIS Open 2004. All Rights Reserved.**

1233 This document and translations of it may be copied and furnished to others, and derivative works that
1234 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
1235 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
1236 this paragraph are included on all such copies and derivative works. However, this document itself does
1237 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
1238 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
1239 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
1240 into languages other than English.

1241 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
1242 or assigns.

1243 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1244 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1245 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1246 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

1247