



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

Committee Draft 01, 18 August 2004

Document identifier:

sstc-saml-metadata-2.0-cd-01

Location:

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

Editors:

Scott Cantor, Internet2
Jahan Moreh, Sigaba
Rob Philpott, RSA Security
Eve Maler, Sun Microsystems

Contributors:

Steve Anderson, OpenNetwork (secretary)

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 Consumer, and Policy Decision Point.

Status:

This is a **Committee Draft** approved by the Security Services Technical Committee on 17 August 2004.

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

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 web page for the Security Services TC (<http://www.oasis-open.org/committees/security/ipr.php>).

36 Table of Contents

37	1 Introduction.....	4
38	1.1 Notation.....	4
39	2 Metadata for SAML V2.0.....	5
40	2.1 Namespaces	5
41	2.2 Common Types.....	6
42	2.2.1 Simple Type entityIDType.....	6
43	2.2.2 Complex Type EndpointType.....	7
44	2.2.3 Complex Type IndexedEndpointType.....	7
45	2.2.4 Complex Type localizedNameType.....	8
46	2.2.5 Complex Type localizedURIType.....	8
47	2.3 Root Elements.....	8
48	2.3.1 Element <EntitiesDescriptor>.....	8
49	2.3.2 Element <EntityDescriptor>.....	9
50	2.3.2.1 Element <Organization>.....	11
51	2.3.2.2 Element <ContactPerson>.....	12
52	2.3.2.3 Element <AdditionalMetadataLocation>.....	13
53	2.4 Role Descriptor Elements.....	13
54	2.4.1 Element <RoleDescriptor>.....	13
55	2.4.1.1 Element <KeyDescriptor>.....	15
56	2.4.2 Complex Type SSODescriptorType.....	15
57	2.4.3 Element <IDPSSODescriptor>.....	16
58	2.4.4 Element <SPSSODescriptor>.....	17
59	2.4.5 Element <AuthnAuthorityDescriptor>.....	17
60	2.4.6 Element <PDPDescriptor>.....	18
61	2.4.7 Element <AttributeAuthorityDescriptor>.....	19
62	2.4.8 Element <AttributeConsumerDescriptor>.....	20
63	2.4.8.1 Element <AttributeConsumingService>.....	20
64	2.4.8.2 Element <RequestedAttribute>.....	21
65	2.5 Element <AffiliationDescriptor>.....	21
66	2.6 Examples.....	23
67	3 Signature Processing.....	26
68	3.1 XML Signature Profile.....	26
69	3.1.1 Signing Formats and Algorithms.....	26
70	3.1.2 References.....	26
71	3.1.3 Canonicalization Method.....	26
72	3.1.4 Transforms.....	27
73	3.1.5 KeyInfo.....	27
74	4 Metadata Publication and Resolution.....	28
75	4.1 Publication and Resolution via Well-Known Location.....	28
76	4.1.1 Publication.....	28
77	4.1.2 Resolution.....	28
78	4.2 Publishing and Resolution via DNS.....	28
79	4.2.1 Publication.....	29
80	4.2.1.1 First Well Known Rule.....	29
81	4.2.1.2 The Order Field.....	29
82	4.2.1.3 The Preference Field.....	29
83	4.2.1.4 The Flag Field.....	30

84	4.2.1.5 The Service Field.....	30
85	4.2.1.6 The Regex and Replacement Fields.....	30
86	4.2.2 NAPTR Examples.....	31
87	4.2.2.1 Entity Metadata NAPTR Examples.....	31
88	4.2.2.2 Name Identifier Examples.....	31
89	4.2.3 Resolution.....	31
90	4.2.3.1 Parsing the Unique Identifier.....	31
91	4.2.3.2 Obtaining Metadata via the DNS.....	32
92	4.2.4 Metadata Location Caching.....	32
93	4.3 Post-Processing of Metadata.....	32
94	4.3.1 Metadata Instance Caching.....	32
95	4.3.2 Handling of HTTPS Redirects.....	32
96	4.3.3 Processing of XML Signatures and General Trust Processing.....	32
97	4.3.3.1 Processing Signed DNS Zones.....	33
98	4.3.3.2 Processing Signed Documents and Fragments.....	33
99	4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL.....	33
100	5 References.....	34
101	Appendix A. Acknowledgments.....	35
102	Appendix B. Notices.....	37

1 Introduction

103

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

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

1.1 Notation

112

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

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

117

118 `Example code listings appear like this.`

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

120 Conventional XML namespace prefixes are used throughout this specification to stand for their respective
121 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

122

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

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

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

2.1 Namespaces

133

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

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

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

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

138

```
139 <schema
140   targetNamespace="urn:oasis:names:tc:SAML:2.0:metadata"
141   xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
142   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
143   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
144   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
145   xmlns="http://www.w3.org/2001/XMLSchema"
146   elementFormDefault="unqualified"
147   attributeFormDefault="unqualified"
148   blockDefault="substitution"
149   version="2.0">
150   <import namespace="http://www.w3.org/2000/09/xmldsig#"
151     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
152     schema.xsd"/>
153   <import namespace="http://www.w3.org/2001/04/xmlenc#"
154     schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-
155     20021210/xenc-schema.xsd"/>
156   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
157     schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
158   <import namespace="http://www.w3.org/XML/1998/namespace"
159     schemaLocation="http://www.w3.org/2001/xml.xsd"/>
160   <annotation>
161     <documentation>
162       Document identifier: sstc-saml-schema-metadata-2.0
163       Location: http://www.oasis-
164       open.org/committees/documents.php?wg_abbrev=security
165       Revision history:
166       V2.0 (August, 2004):
167         Schema for SAML metadata, first published in SAML 2.0.
168     </documentation>
169   </annotation>
170 </schema>
```

171 2.2 Common Types

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

174 2.2.1 Simple Type **entityIDType**

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

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

```
181 <simpleType name="entityIDType">
182   <restriction base="anyURI">
183     <maxLength value="1024"/>
184   </restriction>
185 </simpleType>
```

186 2.2.2 Complex Type EndpointType

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

190 **Binding** [Required]

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

193 **Location** [Required]

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

196 **ResponseLocation** [Optional]

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

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

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

```
202 <complexType name="EndpointType">
203   <sequence>
204     <any namespace="##other" processContents="lax" minOccurs="0"
205     maxOccurs="unbounded"/>
206   </sequence>
207   <attribute name="Binding" type="anyURI" use="required"/>
208   <attribute name="Location" type="anyURI" use="required"/>
209   <attribute name="ResponseLocation" type="anyURI" use="optional"/>
210   <anyAttribute namespace="##other" processContents="lax"/>
211 </complexType>
```

212 2.2.3 Complex Type IndexedEndpointType

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

216 **index** [Required]

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

219 **isDefault** [Optional]

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

222 In any such sequence of like endpoints based on this type, the default endpoint is the first such endpoint
223 with the *isDefault* attribute set to *true*. If no such endpoints exist, the default endpoint is the first such
224 endpoint without the *isDefault* attribute set to *false*. If no such endpoints exist, the default endpoint is

225 the first element in the sequence.

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

```
227 <complexType name="IndexedEndpointType">
228   <complexContent>
229     <extension base="md:EndpointType">
230       <attribute name="index" type="unsignedShort" use="required"/>
231       <attribute name="isDefault" type="boolean" use="optional"/>
232     </extension>
233   </complexContent>
234 </complexType>
```

235 2.2.4 Complex Type localizedNameType

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

```
238 <complexType name="localizedNameType">
239   <simpleContent>
240     <extension base="string">
241       <attribute ref="xml:lang" use="required"/>
242     </extension>
243   </simpleContent>
244 </complexType>
```

245 2.2.5 Complex Type localizedURIType

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

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

```
249 <complexType name="localizedURIType">
250   <simpleContent>
251     <extension base="anyURI">
252       <attribute ref="xml:lang" use="required"/>
253     </extension>
254   </simpleContent>
255 </complexType>
```

256 2.3 Root Elements

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

260 2.3.1 Element `<EntitiesDescriptor>`

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

264 ID [Optional]

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

266 validUntil [Optional]

267 Optional attribute indicates the expiration time of the metadata contained in the element and any
268 contained elements.

269 `cacheDuration` [Optional]
 270 Optional attribute indicates the maximum length of time a consumer should cache the metadata
 271 contained in the element and any contained elements.

272 `Name` [Optional]
 273 A string name that identifies a group of SAML entities in the context of some deployment.

274 `<ds:Signature>` [Optional]
 275 An XML signature that authenticates the containing element and its contents, as described in
 276 Section 3.

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

281 `<EntitiesDescriptor>` or `<EntityDescriptor>` [One or More]
 282 Contains the metadata for one or more SAML entities, or a nested group of additional metadata.

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

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

```

288     <element name="EntitiesDescriptor" type="md:EntitiesDescriptorType"/>
289     <complexType name="EntitiesDescriptorType">
290       <sequence>
291         <element ref="ds:Signature" minOccurs="0"/>
292         <element ref="md:Extensions" minOccurs="0"/>
293         <choice minOccurs="1" maxOccurs="unbounded">
294           <element ref="md:EntityDescriptor"/>
295           <element ref="md:EntitiesDescriptor"/>
296         </choice>
297       </sequence>
298       <attribute name="validUntil" type="dateTime" use="optional"/>
299       <attribute name="cacheDuration" type="duration" use="optional"/>
300       <attribute name="ID" type="ID" use="optional"/>
301       <attribute name="Name" type="string" use="optional"/>
302     </complexType>
303     <element name="Extensions" type="md:ExtensionsType"/>
304     <complexType final="#all" name="ExtensionsType">
305       <sequence>
306         <any namespace="##other" processContents="lax"
307         maxOccurs="unbounded"/>
308       </sequence>
309     </complexType>
  
```

310 **2.3.2 Element `<EntityDescriptor>`**

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

- 315 • SSO Identity Provider
- 316 • SSO Service Provider
- 317 • Authentication Authority

- 318 • Attribute Authority
- 319 • Attribute Consumer
- 320 • Policy Decision Point
- 321 • Affiliation

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

323 `entityID` [Required]

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

326 `ID` [Optional]

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

328 `validUntil` [Optional]

329 Optional attribute indicates the expiration time of the metadata contained in the element and any
330 contained elements.

331 `cacheDuration` [Optional]

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

334 `<ds:Signature>` [Optional]

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

337 `<Extensions>` [Optional]

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

341 `<RoleDescriptor>`, `<IDPSSODescriptor>`, `<SPSSODescriptor>`,
342 `<AuthnAuthorityDescriptor>`, `<AttributeAuthorityDescriptor>`,
343 `<AttributeConsumerDescriptor>`, `<PDPDescriptor>`, or any element from a non-SAML
344 namespace [One or More]

345 **OR**

346 `<AffiliationDescriptor>` [Required]

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

349 `<Organization>` [Optional]

350 Optional element identifying the organization responsible for the SAML entity described by the
351 element.

352 `<ContactPerson>` [Zero or More]

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

354 `<AdditionalMetadataLocation>` [Zero or More]

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

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

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

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

```
364 <element name="EntityDescriptor" type="md:EntityDescriptorType"/>
365 <complexType name="EntityDescriptorType">
366   <sequence>
367     <element ref="ds:Signature" minOccurs="0"/>
368     <element ref="md:Extensions" minOccurs="0"/>
369     <choice>
370       <choice maxOccurs="unbounded">
371         <element ref="md:RoleDescriptor"/>
372         <element ref="md:IDPSSODescriptor"/>
373         <element ref="md:SPSSODescriptor"/>
374         <element ref="md:AuthnAuthorityDescriptor"/>
375         <element ref="md:AttributeAuthorityDescriptor"/>
376         <element ref="md:AttributeConsumerDescriptor"/>
377         <element ref="md:PDPDescriptor"/>
378         <any namespace="##other" processContents="lax"/>
379       </choice>
380       <element ref="md:AffiliationDescriptor"/>
381     </choice>
382     <element ref="md:Organization" minOccurs="0"/>
383     <element ref="md:ContactPerson" minOccurs="0"
384     maxOccurs="unbounded"/>
385     <element ref="md:AdditionalMetadataLocation" minOccurs="0"
386     maxOccurs="unbounded"/>
387   </sequence>
388   <attribute name="entityID" type="md:entityIDType" use="required"/>
389   <attribute name="validUntil" type="dateTime" use="optional"/>
390   <attribute name="cacheDuration" type="duration" use="optional"/>
391   <attribute name="ID" type="ID" use="optional"/>
392   <anyAttribute namespace="##other" processContents="lax"/>
393 </complexType>
```

394 2.3.2.1 Element `<Organization>`

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

399 `<Extensions>` [Optional]

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

403 `<OrganizationName>` [One or More]

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

405 `<OrganizationDisplayName>` [One or More]

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

407 `<OrganizationURL>` [One or More]

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

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

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

```
414 <element name="Organization" type="md:OrganizationType"/>
415 <complexType name="OrganizationType">
416   <sequence>
417     <element ref="md:Extensions" minOccurs="0"/>
418     <element ref="md:OrganizationName" maxOccurs="unbounded"/>
419     <element ref="md:OrganizationDisplayName" maxOccurs="unbounded"/>
420     <element ref="md:OrganizationURL" maxOccurs="unbounded"/>
421   </sequence>
422   <anyAttribute namespace="##other" processContents="lax"/>
423 </complexType>
424 <element name="OrganizationName" type="md:localizedNameType"/>
425 <element name="OrganizationDisplayName" type="md:localizedNameType"/>
426 <element name="OrganizationURL" type="md:localizedURIType"/>
```

427 2.3.2.2 Element <ContactPerson>

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

432 contactType [Required]

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

435 <Extensions> [Optional]

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

439 <Company> [Optional]

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

441 <GivenName> [Optional]

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

443 <SurName> [Optional]

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

445 <EmailAddress> [Zero or More]

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

448 <TelephoneNumber> [Zero or More]

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

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

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

```
453 <element name="ContactPerson" type="md:ContactType"/>
454 <complexType name="ContactType">
455   <sequence>
456     <element ref="md:Extensions" minOccurs="0"/>
457     <element ref="md:Company" minOccurs="0"/>
458     <element ref="md:GivenName" minOccurs="0"/>
```

```

459     <element ref="md:SurName" minOccurs="0"/>
460     <element ref="md:EmailAddress" minOccurs="0"
461 maxOccurs="unbounded"/>
462     <element ref="md:TelephoneNumber" minOccurs="0"
463 maxOccurs="unbounded"/>
464   </sequence>
465   <attribute name="contactType" type="md:ContactTypeType"
466 use="required"/>
467   <anyAttribute namespace="##other" processContents="lax"/>
468 </complexType>
469 <element name="Company" type="string"/>
470 <element name="GivenName" type="string"/>
471 <element name="SurName" type="string"/>
472 <element name="EmailAddress" type="anyURI"/>
473 <element name="TelephoneNumber" type="string"/>
474 <simpleType name="ContactTypeType">
475   <restriction base="string">
476     <enumeration value="technical"/>
477     <enumeration value="support"/>
478     <enumeration value="administrative"/>
479     <enumeration value="billing"/>
480     <enumeration value="other"/>
481   </restriction>
482 </simpleType>

```

483 2.3.2.3 Element <AdditionalMetadataLocation>

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

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

```

491 <element name="AdditionalMetadataLocation"
492 type="md:AdditionalMetadataLocationType"/>
493 <complexType name="AdditionalMetadataLocationType">
494   <simpleContent>
495     <extension base="anyURI">
496       <attribute name="namespace" type="anyURI" use="required"/>
497     </extension>
498   </simpleContent>
499 </complexType>

```

500 2.4 Role Descriptor Elements

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

504 2.4.1 Element <RoleDescriptor>

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

509 ID [Optional]

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

511 validUntil [Optional]
512 Optional attribute indicates the expiration time of the metadata contained in the element and any
513 contained elements.

514 cacheDuration [Optional]
515 Optional attribute indicates the maximum length of time a consumer should cache the metadata
516 contained in the element and any contained elements.

517 protocolSupportEnumeration [Required]
518 A space-delimited set of URIs that identify the set of protocol specifications supported by the role
519 element. For SAML V2.0 entities, this set MUST include the SAML protocol namespace URI,
520 urn:oasis:names:tc:SAML:2.0:protocol.

521 errorURL [Optional]
522 Optional URI attribute that specifies a location to direct a user for problem resolution and
523 additional support related to this role.

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

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

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

534 <Organization> [Optional]
535 Optional element specifies the organization associated with this role. Identical to the element used
536 within the <EntityDescriptor> element.

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

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

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

```

543 <element name="RoleDescriptor" type="md:RoleDescriptorType"/>
544 <complexType name="RoleDescriptorType" abstract="true">
545   <sequence>
546     <element ref="ds:Signature" minOccurs="0"/>
547     <element ref="md:Extensions" minOccurs="0"/>
548     <element ref="md:KeyDescriptor" minOccurs="0"
549 maxOccurs="unbounded"/>
550     <element ref="md:Organization" minOccurs="0"/>
551     <element ref="md:ContactPerson" minOccurs="0"
552 maxOccurs="unbounded"/>
553   </sequence>
554   <attribute name="ID" type="ID" use="optional"/>
555   <attribute name="validUntil" type="dateTime" use="optional"/>
556   <attribute name="cacheDuration" type="duration" use="optional"/>

```

```

557     <attribute name="protocolSupportEnumeration" type="NMTOKENS"
558     use="required"/>
559     <attribute name="errorURL" type="anyURI" use="optional"/>
560     <anyAttribute namespace="##other" processContents="lax"/>
561 </complexType>

```

562 2.4.1.1 Element <KeyDescriptor>

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

566 use [Optional]

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

569 <ds:KeyInfo> [Required]

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

572 <EncryptionMethod> [Zero or More]

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

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

```

578 <element name="KeyDescriptor" type="md:KeyDescriptorType"/>
579 <complexType name="KeyDescriptorType">
580   <sequence>
581     <element ref="ds:KeyInfo"/>
582     <element ref="md:EncryptionMethod minOccurs="0"
583 maxOccurs="unbounded"/>
584   </sequence>
585   <attribute name="use" type="md:KeyTypes" use="optional"/>
586 </complexType>
587 <simpleType name="KeyTypes">
588   <restriction base="string">
589     <enumeration value="encryption"/>
590     <enumeration value="signing"/>
591   </restriction>
592 </simpleType>
593 <element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>

```

594 2.4.2 Complex Type SSODescriptorType

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

599 <ArtifactResolutionService> [Zero or More]

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

603 <SingleLogoutService> [Zero or More]

604 Zero or more elements of type **EndpointType** that describe endpoints that support the Single

605 Logout profiles defined in [SAMLProf].

606 <ManageNameIDService> [Zero or More]

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

609 <NameIDFormat> [Zero or More]

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

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

```
614 <complexType name="SSODescriptorType" abstract="true">
615   <complexContent>
616     <extension base="md:RoleDescriptorType">
617       <sequence>
618         <element ref="md:ArtifactResolutionService" minOccurs="0"
619 maxOccurs="unbounded"/>
620         <element ref="md:SingleLogoutService" minOccurs="0"
621 maxOccurs="unbounded"/>
622         <element ref="md:ManageNameIDService" minOccurs="0"
623 maxOccurs="unbounded"/>
624         <element ref="md:NameIDFormat" minOccurs="0"
625 maxOccurs="unbounded"/>
626       </sequence>
627     </extension>
628   </complexContent>
629 </complexType>
630 <element name="ArtifactResolutionService" type="md:IndexedEndpointType"/>
631 <element name="SingleLogoutService" type="md:EndpointType"/>
632 <element name="ManageNameIDService" type="md:EndpointType"/>
633 <element name="NameIDFormat" type="anyURI"/>
```

634 2.4.3 Element <IDPSSODescriptor>

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

638 WantAuthnRequestsSigned [Optional]

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

641 <SingleSignOnService> [One or More]

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

645 <NameIDMappingService> [Zero or More]

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

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

```
651 <element name="IDPSSODescriptor" type="md:IDPSSODescriptorType"/>
652 <complexType name="IDPSSODescriptorType">
653   <complexContent>
```



```

654         <extension base="md:SSODescriptorType">
655             <sequence>
656                 <element ref="md:SingleSignOnService"
657 maxOccurs="unbounded"/>
658                 <element ref="md:NameIDMappingService" minOccurs="0"
659 maxOccurs="unbounded"/>
660             </sequence>
661             <attribute name="WantAuthnRequestsSigned" type="boolean"
662 use="optional"/>
663         </extension>
664     </complexType>
665 </complexType>
666 <element name="SingleSignOnService" type="md:EndpointType"/>
667 <element name="NameIDMappingService" type="md:EndpointType"/>

```

668 2.4.4 Element <SPSSODescriptor>

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

672 AuthnRequestsSigned [Optional]

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

675 WantAssertionsSigned [Optional]

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

680 <AssertionConsumerService> [One or More]

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

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

```

686 <element name="SPSSODescriptor" type="md:SPSSODescriptorType"/>
687 <complexType name="SPSSODescriptorType">
688     <complexContent>
689         <extension base="md:SSODescriptorType">
690             <sequence>
691                 <element ref="md:AssertionConsumerService"
692 maxOccurs="unbounded"/>
693             </sequence>
694             <attribute name="AuthnRequestsSigned" type="boolean"
695 use="optional"/>
696             <attribute name="WantAssertionsSigned" type="boolean"
697 use="optional"/>
698         </extension>
699     </complexContent>
700 </complexType>
701 <element name="AssertionConsumerService"
702 type="md:AssertionConsumerServiceType"/>

```

703 2.4.5 Element <AuthnAuthorityDescriptor>

704 The <AuthnAuthorityDescriptor> element extends **RoleDescriptorType** with content reflecting
705 profiles specific to authentication authorities, SAML authorities that respond to <samlp:AuthnQuery>

706 messages. Its **AuthnAuthorityDescriptorType** complex type contains the following additional element:

707 <AuthnQueryService> [One or More]

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

711 <AssertionIDRequestService> [Zero or More]

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

715 <NameIDFormat> [Zero or More]

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

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

```
720 <element name="AuthnAuthorityDescriptor"  
721 type="md:AuthnAuthorityDescriptorType"/>  
722 <complexType name="AuthnAuthorityDescriptorType">  
723 <complexContent>  
724 <extension base="md:RoleDescriptorType">  
725 <sequence>  
726 <element ref="md:AuthnQueryService" maxOccurs="unbounded"/>  
727 <element ref="md:AssertionIDRequestService" minOccurs="0"  
728 maxOccurs="unbounded"/>  
729 <element ref="md:NameIDFormat" minOccurs="0"  
730 maxOccurs="unbounded"/>  
731 </sequence>  
732 </extension>  
733 </complexContent>  
734 </complexType>  
735 <element name="AuthnQueryService" type="md:EndpointType"/>  
736 <element name="AssertionIDRequestService" type="md:EndpointType"/>
```

737 2.4.6 Element <PDPDescriptor>

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

741 <AuthzService> [One or More]

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

745 <AssertionIDRequestService> [Zero or More]

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

749 <NameIDFormat> [Zero or More]

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

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

```

754 <element name="PDPDescriptor" type="md:PDPDescriptorType"/>
755 <complexType name="PDPDescriptorType">
756   <complexContent>
757     <extension base="md:RoleDescriptorType">
758       <sequence>
759         <element ref="md:AuthzService" maxOccurs="unbounded"/>
760         <element ref="md:AssertionIDRequestService" minOccurs="0"
761 maxOccurs="unbounded"/>
762         <element ref="md:NameIDFormat" minOccurs="0"
763 maxOccurs="unbounded"/>
764       </sequence>
765     </extension>
766   </complexContent>
767 </complexType>
768 <element name="AuthzService" type="md:EndpointType"/>

```

769 2.4.7 Element <AttributeAuthorityDescriptor>

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

774 <AttributeService> [One or More]

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

778 <AssertionIDRequestService> [Zero or More]

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

782 <saml:Attribute> [Zero or More]

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

786 <NameIDFormat> [Zero or More]

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

789 <AttributeProfile> [Zero or More]

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

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

```

794 <element name="AttributeAuthorityDescriptor"
795 type="md:AttributeAuthorityDescriptorType"/>
796 <complexType name="AttributeAuthorityDescriptorType">
797   <complexContent>
798     <extension base="md:RoleDescriptorType">
799       <sequence>
800         <element ref="md:AttributeService" maxOccurs="unbounded"/>
801         <element ref="md:AssertionIDRequestService" minOccurs="0"
802 maxOccurs="unbounded"/>
803         <element ref="saml:Attribute" minOccurs="0"
804 maxOccurs="unbounded"/>

```

```

805         <element ref="md:NameIDFormat" minOccurs="0"
806 maxOccurs="unbounded"/>
807         <element ref="md:AttributeProfile" minOccurs="0"
808 maxOccurs="unbounded"/>
809     </sequence>
810 </extension>
811 </complexContent>
812 </complexType>
813 <element name="AttributeService" type="md:EndpointType"/>
814 <element name="AttributeProfile" type="anyURI"/>

```

815 2.4.8 Element <AttributeConsumerDescriptor>

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

819 <AttributeConsumingService> [One or More]

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

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

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

```

828 <element name="AttributeConsumerDescriptor"
829 type="md:AttributeConsumerDescriptorType"/>
830 <complexType name="AttributeConsumerDescriptorType">
831 <complexContent>
832 <extension base="md:RoleDescriptorType">
833 <sequence>
834 <element ref="md:AttributeConsumingService"
835 maxOccurs="unbounded"/>
836 </sequence>
837 </extension>
838 </complexContent>
839 </complexType>

```

840 2.4.8.1 Element <AttributeConsumingService>

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

844 `index` [Required]

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

847 `isDefault` [Optional]

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

850 `WantAssertionsSigned` [Optional]

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

854 combination.

855 <ServiceName> [One or More]

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

857 <ServiceDescription> [Zero or More]

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

859 <RequestedAttribute> [One or More]

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

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

```
863 <element name="AttributeConsumingService"  
864 type="md:AttributeConsumingServiceType"/>  
865 <complexType name="AttributeConsumingServiceType">  
866 <sequence>  
867 <element ref="md:ServiceName" maxOccurs="unbounded"/>  
868 <element ref="md:ServiceDescription" minOccurs="0"  
869 maxOccurs="unbounded"/>  
870 <element ref="md:RequestedAttribute" maxOccurs="unbounded"/>  
871 </sequence>  
872 <attribute name="index" type="unsignedShort" use="required"/>  
873 <attribute name="isDefault" type="boolean" use="optional"/>  
874 <attribute name="WantAssertionsSigned" type="boolean" use="optional"/>  
875 </complexType>  
876 <element name="ServiceName" type="md:localizedNameType"/>  
877 <element name="ServiceDescription" type="md:localizedNameType"/>
```

878 2.4.8.2 Element <RequestedAttribute>

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

882 **isRequired** [Optional]

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

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

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

```
889 <element name="RequestedAttribute" type="md:RequestedAttributeType"/>  
890 <complexType name="RequestedAttributeType">  
891 <complexContent>  
892 <extension base="saml:AttributeType">  
893 <attribute name="isRequired" type="boolean" use="optional"/>  
894 </extension>  
895 </complexContent>  
896 </complexType>
```

897 2.5 Element <AffiliationDescriptor>

898 The <AffiliationDescriptor> element is an alternative to the sequence of role descriptors
899 described in Section 2.4 that is used when an <EntityDescriptor> describes an affiliation of SAML
900 entities (typically service providers) rather than a single entity. The <AffiliationDescriptor>

901 element provides a summary of the individual entities that make up the affiliation along with general
902 information about the affiliation itself. Its **AffiliationDescriptorType** complex type contains the following
903 elements and attributes:

904 affiliationOwnerID [Required]

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

908 ID [Optional]

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

910 validUntil [Optional]

911 Optional attribute indicates the expiration time of the metadata contained in the element and any
912 contained elements.

913 cacheDuration [Optional]

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

916 <ds:Signature> [Optional]

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

919 <Extensions> [Optional]

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

923 <AffiliateMember> [One or More]

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

926 <KeyDescriptor> [Zero or More]

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

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

```
932 <element name="AffiliationDescriptor"  
933 type="md:AffiliationDescriptorType"/>  
934 <complexType name="AffiliationDescriptorType">  
935 <sequence>  
936 <element ref="ds:Signature" minOccurs="0"/>  
937 <element ref="md:Extensions" minOccurs="0"/>  
938 <element ref="md:AffiliateMember" maxOccurs="unbounded"/>  
939 <element ref="md:KeyDescriptor" minOccurs="0"  
940 maxOccurs="unbounded"/>  
941 </sequence>  
942 <attribute name="affiliationOwnerID" type="md:entityIDType"  
943 use="required"/>  
944 <attribute name="validUntil" type="dateTime" use="optional"/>  
945 <attribute name="cacheDuration" type="duration" use="optional"/>  
946 <attribute name="ID" type="ID" use="optional"/>  
947 <anyAttribute namespace="##other" processContents="lax"/>  
948 </complexType>  
949 <element name="AffiliateMember" type="md:entityIDType"/>
```

950 2.6 Examples

951 The following is an example of metadata for a SAML system entity acting as an identity provider and an
952 attribute authority. A signature is shown as a placeholder, without the actual content.
953

```
954 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"  
955   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
956   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
957   entityID="https://IdentityProvider.com/SAML">  
958   <ds:Signature>...</ds:Signature>  
959   <IDPSSODescriptor WantAuthnRequestsSigned="true"  
960     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">  
961     <KeyDescriptor use="signing">  
962       <ds:KeyInfo>  
963         <ds:KeyName>IdentityProvider.com SSO Key</ds:KeyName>  
964       </ds:KeyInfo>  
965     </KeyDescriptor>  
966     <ArtifactResolutionService isDefault="true" index="0"  
967       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"  
968       Location="https://IdentityProvider.com/SAML/Artifact"/>  
969     <SingleLogoutService  
970       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"  
971       Location="https://IdentityProvider.com/SAML/SLO/SOAP"/>  
972     <SingleLogoutService  
973       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"  
974       Location="https://IdentityProvider.com/SAML/SLO/Browser"  
975     ResponseLocation="https://IdentityProvider.com/SAML/SLO/Response"/>  
976     <NameIDFormat>  
977       urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName  
978     </NameIDFormat>  
979     <NameIDFormat>  
980       urn:oasis:names:tc:SAML:2.0:nameid-format:persistent  
981     </NameIDFormat>  
982     <NameIDFormat>  
983       urn:oasis:names:tc:SAML:2.0:nameid-format:transient  
984     </NameIDFormat>  
985     <SingleSignOnService  
986       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"  
987       Location="https://IdentityProvider.com/SAML/SSO/Browser"/>  
988     <SingleSignOnService  
989       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"  
990       Location="https://IdentityProvider.com/SAML/SSO/Browser"/>  
991   </IDPSSODescriptor>  
992   <AttributeAuthorityDescriptor  
993     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">  
994     <KeyDescriptor use="signing">  
995       <ds:KeyInfo>  
996         <ds:KeyName>IdentityProvider.com AA Key</ds:KeyName>  
997       </ds:KeyInfo>  
998     </KeyDescriptor>  
999     <AttributeService  
1000       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"  
1001       Location="https://IdentityProvider.com/SAML/AA/SOAP"/>  
1002     <AssertionIDRequestService  
1003       Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"  
1004       Location="https://IdentityProvider.com/SAML/AA/URI"/>  
1005     <saml:Attribute  
1006       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1007       Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"  
1008       FriendlyName="eduPersonPrincipalName">  
1009     </saml:Attribute>  
1010     <saml:Attribute  
1011       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1012       Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"  
1013       FriendlyName="eduPersonAffiliation">  
1014       <saml:AttributeValue>member</saml:AttributeValue>  
1015       <saml:AttributeValue>student</saml:AttributeValue>
```

```

1016         <saml:AttributeValue>faculty</saml:AttributeValue>
1017         <saml:AttributeValue>employee</saml:AttributeValue>
1018         <saml:AttributeValue>staff</saml:AttributeValue>
1019     </saml:Attribute>
1020     <NameIDFormat>
1021     urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1022 </NameIDFormat>
1023 <NameIDFormat>
1024 urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1025 </NameIDFormat>
1026 <NameIDFormat>
1027 urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1028 </NameIDFormat>
1029 </AttributeAuthorityDescriptor>
1030 <Organization>
1031     <OrganizationName xml:lang="en">
1032     Identity Providers R US
1033 </OrganizationName>
1034 <OrganizationDisplayName xml:lang="en">
1035     Identity Providers R US, a Division of Lerxst Corp.
1036 </OrganizationDisplayName>
1037 <OrganizationURL xml:lang="en">
1038     https://IdentityProvider.com
1039 </OrganizationURL>
1040 </Organization>
1041 </EntityDescriptor>
1042

```

1043 The following is an example of metadata for a SAML system entity acting as a service provider and an
1044 attribute consumer. A signature is shown as a placeholder, without the actual content. For illustrative
1045 purposes, the service is one that does not require users to uniquely identify themselves, but rather
1046 authorizes access on the basis of a role-like attribute.
1047

```

1048 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1049     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1050     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1051     entityID="https://ServiceProvider.com/SAML">
1052 <ds:Signature>...</ds:Signature>
1053 <SPSSODescriptor AuthnRequestsSigned="true"
1054     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1055     <KeyDescriptor use="signing">
1056         <ds:KeyInfo>
1057             <ds:KeyName>ServiceProvider.com SSO Key</ds:KeyName>
1058         </ds:KeyInfo>
1059     </KeyDescriptor>
1060     <SingleLogoutService
1061     Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1062     Location="https://ServiceProvider.com/SAML/SLO/SOAP"/>
1063     <SingleLogoutService
1064     Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1065     Location="https://ServiceProvider.com/SAML/SLO/Browser"
1066     ResponseLocation="https://ServiceProvider.com/SAML/SLO/Response"/>
1067     <NameIDFormat>
1068     urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1069 </NameIDFormat>
1070     <AssertionConsumerService isDefault="true" index="0"
1071     Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
1072     Location="https://ServiceProvider.com/SAML/SSO/Artifact"/>
1073     <AssertionConsumerService index="1"
1074     Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1075     Location="https://ServiceProvider.com/SAML/SSO/POST"/>
1076 </SPSSODescriptor>
1077 <AttributeConsumerDescriptor
1078     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1079     <KeyDescriptor use="encryption">
1080         <ds:KeyInfo>
1081             <ds:KeyName>ServiceProvider.com Encrypt Key</ds:KeyName>
1082         </ds:KeyInfo>

```



```

1083         <EncryptionMethod
1084 Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
1085     </KeyDescriptor>
1086     <AttributeConsumingService index="0">
1087         <ServiceName xml:lang="en">
1088             Academic Journals R US
1089         </ServiceName>
1090         <RequestedAttribute
1091 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1092 Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7"
1093 FriendlyName="eduPersonEntitlement">
1094             <saml:AttributeValue>
1095                 https://ServiceProvider.com/entitlements/123456789
1096             </saml:AttributeValue>
1097         </RequestedAttribute>
1098     </AttributeConsumingService>
1099 </AttributeConsumerDescriptor>
1100 <Organization>
1101     <OrganizationName xml:lang="en">
1102         Academic Journals R US
1103     </OrganizationName>
1104     <OrganizationDisplayName xml:lang="en">
1105         Academic Journals R US, a Division of Dirk Corp.
1106     </OrganizationDisplayName>
1107     <OrganizationURL xml:lang="en">
1108         https://ServiceProvider.com
1109     </OrganizationURL>
1110 </Organization>
1111 </EntityDescriptor>

```

1112 3 Signature Processing

1113 Various elements in a metadata instance can be digitally signed (as indicated by the element's inclusion of
1114 a `<ds:Signature>` element), with the following benefits:

- 1115 • Metadata integrity
- 1116 • Authentication of the metadata by a trusted signer

1117 A digital signature is not always required, for example if the relying party obtains the information directly
1118 from the publishing entity directly (with no intermediaries) through a secure channel, with the entity having
1119 authenticated to the relying party by some means other than a digital signature.

1120 Many different techniques are available for "direct" authentication and secure channel establishment
1121 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,
1122 the applicable security requirements depend on the communicating applications.

1123 Additionally, elements can inherit signatures on enclosing parent elements that are themselves signed.

1124 In the absence of such context, it is RECOMMENDED that at least the root element of a metadata
1125 instance be signed.

1126 3.1 XML Signature Profile

1127 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
1128 and many choices. This section details the constraints on these facilities so that metadata processors do
1129 not have to deal with the full generality of XML Signature processing. This usage makes specific use of
1130 the `xs:ID`-typed attributes optionally present on the elements to which signatures can apply. These
1131 attributes are collectively referred to in this section as the identifier attributes.

1132 3.1.1 Signing Formats and Algorithms

1133 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
1134 detached.

1135 SAML metadata MUST use enveloped signatures when signing the elements defined in this specification.
1136 SAML processors SHOULD support the use of RSA signing and verification for public key operations in
1137 accordance with the algorithm identified by <http://www.w3.org/2000/09/xmlsig#rsa-sha1>.

1138 3.1.2 References

1139 Signed metadata elements MUST supply a value for the identifier attribute on the signed element. The
1140 element may or may not be the root element of the actual XML document containing the signed metadata
1141 element.

1142 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute
1143 value of the metadata element being signed. For example, if the identifier attribute value is "foo", then the
1144 URI attribute in the `<ds:Reference>` element MUST be "#foo".

1145 As a consequence, a metadata element's signature MUST apply to the content of the signed element and
1146 any child elements it contains.

1147 3.1.3 Canonicalization Method

1148 SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the
1149 `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a `<ds:Transform>`
1150 algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML metadata

1151 embedded in an XML context can be verified independent of that context.

1152 **3.1.4 Transforms**

1153 Signatures in SAML metadata SHOULD NOT contain transforms other than the enveloped signature
1154 transform (with the identifier <http://www.w3.org/2000/09/xmlsig#enveloped-signature>) or the exclusive
1155 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
1156 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

1157 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
1158 not, verifiers MUST ensure that no content of the signed metadata element is excluded from the
1159 signature. This can be accomplished by establishing out-of-band agreement as to what transforms are
1160 acceptable, or by applying the transforms manually to the content and reverifying the result as consisting
1161 of the same SAML metadata.

1162 **3.1.5 KeyInfo**

1163 XML Signature [XMLSig] defines usage of the `<ds:KeyInfo>` element. SAML does not require the
1164 use of `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY
1165 be absent.

1166 4 Metadata Publication and Resolution

1167 Two mechanisms are provided for an entity to publish (and for a consumer to resolve the location of)
1168 metadata documents: via a "well-known-location" by directly dereferencing the entity's unique identifier (a
1169 URI variously referred to as an *entityID* or *providerID*), or indirectly by publishing the location of metadata
1170 in the DNS. Other out-of-band mechanisms are of course also permitted. A consumer that supports both
1171 approaches defined in this document MUST attempt resolution via DNS before using the "well-known-
1172 location" mechanism.

1173 When retrieval requires network transport of the document, the transport SHOULD be protected with
1174 mechanisms providing server authentication and integrity protection. For example, HTTP-based resolution
1175 SHOULD be protected with TLS/SSL [RFC2246] as amended by [RFC3546].

1176 Various mechanisms are described in this section to aid in establishing trust in the accuracy and
1177 legitimacy of metadata, including use of XML signatures, SSL/TLS server authentication, and DNS
1178 signatures. Regardless of the mechanism(s) used, relying parties SHOULD have some means by which to
1179 establish trust in metadata information before relying on it.

1180 4.1 Publication and Resolution via Well-Known Location

1181 The following sections describe publication and resolution of metadata by means of a well-known location.

1182 4.1.1 Publication

1183 Entities MAY publish their metadata documents at a well known location by placing the document at the
1184 location denoted by its unique identifier, which MUST be in the form of a URL (rather than a URN). See
1185 Section 8.3.6 of [SAMLCore] for more information about such identifiers. It is STRONGLY
1186 RECOMMENDED that `https` URLs be used for this purpose. An indirection mechanism supported by the
1187 URL scheme (such as an HTTP 1.1 302 redirect) MAY be used if the document is not placed directly at
1188 the location. If the publishing protocol permits MIME-based identification of content types, the content type
1189 of the metadata instance MUST be `application/samlmetadata+xml`.

1190 The XML document provided at the well-known location MUST describe the metadata only for the entity
1191 represented by the unique identifier (that is, the root element MUST be an `<EntityDescriptor>` with
1192 an `entityID` matching the location). If other entities need to be described, the
1193 `<AdditionalMetaLocation>` element MUST be used. Thus the `<EntitiesDescriptor>` element
1194 MUST NOT be used in documents published using this mechanism, since a group of entities are not
1195 defined by such an identifier.

1196 4.1.2 Resolution

1197 If an entity's unique identifier is a URL, metadata consumers MAY attempt to resolve an entity's unique
1198 identifier directly, in a scheme-specific manner, by dereferencing the identifier.

1199 4.2 Publishing and Resolution via DNS

1200 To improve the accessibility of metadata documents and provide additional indirection between an entity's
1201 unique identifier and the location of metadata, entities MAY publish their metadata document locations in a
1202 zone of their corresponding DNS [RFC1034]. The entity's unique identifier (a URI) is used as the input to
1203 the process. Since URIs are flexible identifiers, location publication methods and the resolution process
1204 are determined by the URI's scheme and fully-qualified name. URI locations for metadata are

1205 subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in [\[RFC2915\]](#)
1206 and [\[RFC3403\]](#).

1207 It is RECOMMENDED that entities publish their resource records in signed zone files using [\[RFC2535\]](#)
1208 such that relying parties may establish the validity of the published location and authority of the zone, and
1209 integrity of the DNS response. If DNS zone signatures are present, relying parties MUST properly validate
1210 the signature.

1211 **4.2.1 Publication**

1212 This specification makes use of the NAPTR resource record described in [\[RFC2915\]](#) and [\[RFC3403\]](#).
1213 Familiarity with these documents is encouraged.

1214 Dynamic Delegation Discovery System (DDDS) [\[RFC3401\]](#) is a general purpose system for the retrieval of
1215 information based on an application-specific input string and the application of well known rules to
1216 transform that string until a terminal condition is reached requiring a look-up into an application-specific
1217 defined database or resolution of a URL based on the rules defined by the application. DDDS defines a
1218 specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS
1219 necessary to apply DDDS rules.

1220 Entities MAY publish separate URLs when multiple metadata documents need to be distributed, or when
1221 different metadata documents are required due to multiple trust relationships that require separate keying
1222 material, or when service interfaces require separate metadata declarations. This may be accomplished
1223 through the use of the optional `<AdditionalMetaLocation>` element, or through the regexp facility and
1224 multiple service definition fields in the NAPTR resource record itself.

1225 If the publishing protocol permits MIME-based identification of content types, the content type of the
1226 metadata instance MUST be `application/samlmetadata+xml`.

1227 If the entity's unique identifier is a URN, publication of the corresponding metadata location proceeds as
1228 specified in [\[RFC3404\]](#). Otherwise, the resolution of the metadata location proceeds as specified below.

1229 The following is the application-specific profile of DDDS for SAML metadata resolution.

1230 **4.2.1.1 First Well Known Rule**

1231 The "first well-known-rule" for processing SAML metadata resolution is to parse the entity's unique
1232 identifier and extract the fully-qualified domain name (subexpression 3) as described in Section "[Parsing](#)
1233 [the providerID](#)".

1234 **4.2.1.2 The Order Field**

1235 The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY
1236 provide multiple NAPTR resource records which MUST be processed by the resolver application in the
1237 order indicated by this field.

1238 **4.2.1.3 The Preference Field**

1239 For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving
1240 application. The resolving application MAY ignore this order, in cases where the service field value does
1241 not meet the resolver's requirements (e.g.: the resource record returns a protocol the application does not
1242 support).

1243 4.2.1.4 The Flag Field

1244 SAML metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying
1245 additional resource records are to be processed). The "U" flag indicates that the output of the rule is a
1246 URI.

1247 4.2.1.5 The Service Field

1248 The SAML-specific service field, as described in the following BNF, declares the modes by which instance
1249 document(s) shall be made available:

```
1250 servicefield = 1("PID2U" / "NID2U") "+" proto [*(":" class) *(":" servicetype)]
1251 proto = 1("https" / "uddi")
1252 class = 1[ "entity" / "entitygroup" ]
1253 servicetype = 1(si / "spssso" / "idpssso" / "authn" / "authnauth" / "pdp" / "attrauth" /
1254 "attrcons" / alphanum )
1255 si = "si" [ ":" alphanum ] [ ":" endpoint" ]
1256 alphanum = 1*32 (ALPHA / DIGIT)
```

1257 where:

- 1258 • servicefield PID2U resolves an entity's unique identifier to metadata URL.
- 1259 • servicefield NID2U resolves a principal's <NameIdentifier> into a metadata URL.
- 1260 • proto describes the retrieval protocol (`https` or `uddi`). In the case of UDDI, the URL will be an
1261 http(s) URL referencing a WSDL document.
- 1262 • class identifies whether the referenced metadata document describes a single entity, or multiple.
1263 In the latter case, the referenced document MUST contain the entity defined by the original unique
1264 identifier as a member of a group of entities within the document itself such as an
1265 <AffiliationDescriptor> or <EntitiesDescriptor>.
- 1266 • servicetype allows an entity to publish metadata for distinct roles and services as separate
1267 documents. Resolvers who encounter multiple servicetype declarations will dereference the
1268 appropriate URI, depending on which service is required for an operation (e.g.: an entity operating
1269 both as an identity provider and a service provider can publish metadata for each role at different
1270 locations). The `authn` service type represents a <SingleSignOnService> endpoint.
- 1271 • si (with optional endpoint component) allows the publisher to either directly publish the metadata
1272 for a service instance, or by articulating a SOAP endpoint (using `endpoint`).

1273 For example:

- 1274 • PID2U+https:entity - represents the entity's complete metadata document available via the
1275 https protocol
- 1276 • PID2U+uddi:entity:si:foo - represents the WSDL document location that describes a service
1277 instance "foo"
- 1278 • PID2U+https:entitygroup:idpssso - represents the metadata for a group of entities acting as
1279 SSO identity providers, of which the original entity is a member.
- 1280 • NID2U+https:idp - represents the metadata for the SSO identity provider of a principal

1281 4.2.1.6 The Regexp and Replacement Fields

1282 The expected output after processing the input string through the regex MUST be a valid `https` URL or
1283 UDDI node (WSDL document) address.

1284 4.2.2 NAPTR Examples

1285 4.2.2.1 Entity Metadata NAPTR Examples

1286 Entities publish metadata URLs in the following manner:

```
1287 $ORIGIN provider.biz
1288
1289 ;; order pref f service regexp or replacement
1290
1291 IN NAPTR 100 10 "U" PID2U+https:entity
1292 "!.*!https://host.provider.biz/some/directory/trust.xml!" ""
1293 IN NAPTR 110 10 "U" PID2U+https: entity:trust
1294 "!.*!https://foo.provider.biz:1443/mdtrust.xml!" ""
1295 IN NAPTR 125 10 "U" PID2U+https:"
1296 IN NAPTR 110 10 "U" PID2U+uddi:entity
1297 "!.*!https://this.uddi.node.provider.biz/libmd.wsd1" ""
```

1298 4.2.2.2 Name Identifier Examples

1299 A principal's employer `example.int` operates an identity provider which may be used by an office supply
1300 company to authenticate authorized buyers. The supplier takes a users' email address
1301 `buyer@example.int` as input to the resolution process, and parses the email address to extract the
1302 FQDN (`example.int`). The employer publishes the following NAPTR record in the `example.int` DNS:

```
1303 $ORIGIN example.int
1304
1305 IN NAPTR 100 10 "U" NID2U+https:authn
1306 "!.^([\^@]+)@(.*)!https://serv.example.int:8000/cgi-bin/getmd?\1!" ""
1307 IN NAPTR 100 10 "U" NID2U+https:idp
1308 "!.^([\^@]+)@(.*)!https://auth.example.int/app/auth?\1" ""
```

1309 4.2.3 Resolution

1310 When resolving metadata for an entity via the DNS, the unique identifier of the entity is used as the initial
1311 input into the resolution process, rather than as an actual location Proceed as follows:

- 1312 • If the unique identifier is a URN, proceed with the resolution steps as defined in [\[RFC3404\]](#).
- 1313 • Otherwise, parse the identifier to obtain the fully-qualified domain name.
- 1314 • Query the DNS for NAPTR resource records of the domain iteratively until a terminal resource
1315 record is returned.
- 1316 • Identify which resource record to use based on the service fields, then order fields, then preference
1317 fields of the result set.
- 1318 • Obtain the document(s) at the provided location(s) as required by the application.

1319 4.2.3.1 Parsing the Unique Identifier

1320 To initiate the resolution of the location of the metadata information, it will be necessary in some cases to
1321 decompose the entity's unique identifier (expressed as a URI) into one or more atomic elements.

1322 The following regular expression should be used when initiating the decomposition process:

```
1323 ^([\^:/?#+:)?/*([\^:/?#+:)?((([\^/?:#]*\.)*)(([\^/?#:\.]+)\.([\^/?#:\.]+))
1324 (: \d+)?([\^?#]*)(\?[\^#]*)?(#[.]*)?$
1325      1           2           3         4           5           6           7
1326      8           9          10          11
```

1327 Subexpression 3 MUST result in a Fully-Qualified Domain Name (FQDN), which will be the basis for
1328 retrieving metadata locations from this zone.

1329 **4.2.3.2 Obtaining Metadata via the DNS**

1330 Upon completion of the parsing of the identifier, the application then performs a DNS query for the resulting
1331 domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses.
1332 Applications MAY exclude from the result set any service definitions that do not concern the present
1333 request operations.

1334 Resolving applications MUST subsequently order the result set according to the order field, and MAY
1335 order the result set based on the preference set. Resolvers are NOT REQUIRED to follow the ordering of
1336 the preferences field. The resulting NAPTR resource record(s) are operated on iteratively (based on the
1337 order flag) until a terminal NAPTR resource record is reached.

1338 The result will be a well-formed, absolute URL, which is then used to retrieve the metadata document.

1339 **4.2.4 Metadata Location Caching**

1340 Location caching MUST NOT exceed the TTL of the DNS zone from which the location was derived.
1341 Resolvers MUST obtain a fresh copy of the metadata location upon reaching the expiration of the TTL of
1342 the zone.

1343 Publishers of metadata documents should carefully consider the TTL of the zone when making changes
1344 to metadata document locations. Should such a location change occur, a publisher MUST either keep the
1345 document at both the old and new location until all conforming resolvers are certain to have the updated
1346 location (e.g.: time of zone change + TTL), or provide an HTTP Redirect [RFC2616] response at the old
1347 location specifying the new location.

1348 **4.3 Post-Processing of Metadata**

1349 The following sections describe the post-processing of metadata.

1350 **4.3.1 Metadata Instance Caching**

1351 Document caching MUST NOT exceed the `validUntil` or `cacheDuration` attribute of the subject
1352 element(s). If metadata elements have parent elements which contain caching policies, the parent
1353 element takes precedence.

1354 To properly process the `cacheDuration` attribute, consumers MUST retain the date and time when the
1355 document was retrieved.

1356 When a document or element has expired, the consumer MUST retrieve a fresh copy, which may require
1357 a refresh of the document location(s). Consumers SHOULD process document cache processing
1358 according to [RFC2616] Section 13, and MAY request the Last-Modified date and time from the HTTP
1359 server. Publishers SHOULD ensure acceptable cache processing as described in [RFC2616] (Section
1360 10.3.5 304 Not Modified).

1361 **4.3.2 Handling of HTTPS Redirects**

1362 Publishers MAY issue an HTTP Redirect (301 Moved Permanently, 302 or 307 Temporary Redirect)
1363 [RFC2616], and user agents MUST follow the specified URL in the Redirect response. Redirects
1364 SHOULD be of the same protocol as the initial request.

1365 **4.3.3 Processing of XML Signatures and General Trust Processing**

1366 Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and
1367 for the trust ascribed to the entity described by such metadata:

- 1368 • Trust derived from the signature of the DNS zone from which the metadata location URL was

- 1369 resolved, ensuring accuracy of the metadata document location(s)
- 1370 • Trust derived from signature processing of the metadata document itself, ensuring the integrity of
1371 the XML document
- 1372 • Trust derived from the SSL/TLS server authentication of the metadata location URL, ensuring the
1373 identity of the publisher of the metadata
- 1374 Post-processing of the metadata document MUST include signature processing at the XML-document
1375 level and MAY include one of the other two processes. Specifically, the relying party MAY choose to trust
1376 any of the cited authorities in the resolution and parsing process. Publishers of metadata MUST employ a
1377 document-integrity mechanism and MAY employ any of the other two processing profiles to establish trust
1378 in the metadata document, governed by implementation policies.

1379 **4.3.3.1 Processing Signed DNS Zones**

1380 Verification of DNS zone signature SHOULD be processed, if present, as described in [\[RFC2535\]](#).

1381 **4.3.3.2 Processing Signed Documents and Fragments**

1382 Published metadata documents SHOULD be signed, as described in Section 3, either by a certificate
1383 issued to the subject of the document, or by another trusted party. Publishers MAY consider signatures of
1384 other parties as a means of trust conveyance.

1385 Metadata consumers MUST validate signatures, when present, on the metadata document as described
1386 by Section 3.

1387 **4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL**

1388 It is STRONGLY RECOMMENDED that publishers implement TLS/SSL URLs; therefore, consumers
1389 SHOULD consider the trust inherited from the issuer of the TLS/SSL certificate. Publication URLs may not
1390 always be located in the domain of the subject of the metadata document; therefore, consumers SHOULD
1391 NOT presume certificates whose subject is the entity in question, as it may be hosted by another trusted
1392 party.

1393 As the basis of this trust may not be available against a cached document, other mechanisms SHOULD
1394 be used under such circumstances.

5 References

1395

- 1396 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
1397 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1398 **[SAMLBind]** S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language*
1399 *(SAML) V2.0*. OASIS SSTC, August 2004. Document ID sstc-saml-bindings-2.0-
1400 cd-01. See <http://www.oasis-open.org/committees/security/>.
- 1401 **[SAMLCore]** S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion*
1402 *Markup Language (SAML) V2.0*. OASIS SSTC, August 2004. Document ID sstc-
1403 saml-core-2.0-cd-01. See <http://www.oasis-open.org/committees/security/>.
- 1404 **[SAMLMeta-xsd]** S. Cantor et al., SAML metadata schema. OASIS SSTC, August 2004. Document
1405 ID sstc-saml-schema-metadata-2.0. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1406 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1407 **[SAMLProf]** S. Cantor et al., *Profiles for the OASIS Security Assertion Markup Language*
1408 *(SAML) V2.0*. OASIS SSTC, August 2004. Document ID sstc-saml-profiles-2.0-
1409 cd-01. See <http://www.oasis-open.org/committees/security/>.
- 1410 **[SAMLSec]** F. Hirsch et al., *Security and Privacy Considerations for the OASIS Security*
1411 *Assertion Markup Language (SAML) V2.0*. OASIS SSTC, August 2004.
1412 Document ID sstc-saml-sec-consider-2.0-cd-01. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1413 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1414 **[XMLEnc]** D. Eastlake et al., XML-Encryption Syntax and Processing,
1415 <http://www.w3.org/TR/xmlenc-core/>, World Wide Web Consortium.
- 1416 **[XMLSig]** D. Eastlake et al., XML-Signature Syntax and Processing,
1417 <http://www.w3.org/TR/xmlsig-core/>, World Wide Web Consortium.

1418 Appendix A. Acknowledgments

1419 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
1420 Committee, whose voting members at the time of publication were:

- 1421 • Conor Cahill, AOL
- 1422 • Hal Lockhart, BEA Systems
- 1423 • Rick Randall, Booz Allen Hamilton
- 1424 • Ronald Jacobson, Computer Associates
- 1425 • Gavenraj Sodhi, Computer Associates
- 1426 • Tim Alsop, CyberSafe Limited
- 1427 • Paul Madsen, Entrust
- 1428 • Carolina Canales-Valenzuela, Ericsson
- 1429 • Dana Kaufman, Forum Systems
- 1430 • Irving Reid, Hewlett-Packard
- 1431 • Paula Austel, IBM
- 1432 • Maryann Hondo, IBM
- 1433 • Michael McIntosh, IBM
- 1434 • Anthony Nadalin, IBM
- 1435 • Nick Ragouzis, Individual
- 1436 • Scott Cantor, Internet2
- 1437 • Bob Morgan, Internet2
- 1438 • Prateek Mishra, Netegrity
- 1439 • Forest Yin, Netegrity
- 1440 • Peter Davis, Neustar
- 1441 • Frederick Hirsch, Nokia
- 1442 • John Kemp, Nokia
- 1443 • Senthil Sengodan, Nokia
- 1444 • Scott Kiestler, Novell
- 1445 • Steve Anderson, OpenNetwork
- 1446 • Ari Kermaier, Oracle
- 1447 • Vamsi Motukuru, Oracle
- 1448 • Darren Platt, Ping Identity
- 1449 • Jim Lien, RSA Security
- 1450 • John Linn, RSA Security
- 1451 • Rob Philpott, RSA Security
- 1452 • Dipak Chopra, SAP
- 1453 • Jahan Moreh, Sigaba
- 1454 • Bhavna Bhatnagar, Sun Microsystems
- 1455 • Jeff Hodges, Sun Microsystems
- 1456 • Eve Maler, Sun Microsystems
- 1457 • Ronald Monzillo, Sun Microsystems
- 1458 • Emily Xu, Sun Microsystems
- 1459 • Mike Beach, Boeing

- 1460 • Greg Whitehead, Trustgenix
- 1461 • James Vanderbeek, Vodafone
- 1462

1463 The editors also would like to acknowledge the following people for their contributions to previous versions
1464 of the OASIS Security Assertions Markup Language Standard:

- 1465 • Stephen Farrell, Baltimore Technologies
- 1466 • David Orchard, BEA Systems
- 1467 • Krishna Sankar, Cisco Systems
- 1468 • Zahid Ahmed, CommerceOne
- 1469 • Carlisle Adams, Entrust
- 1470 • Tim Moses, Entrust
- 1471 • Nigel Edwards, Hewlett-Packard
- 1472 • Joe Pato, Hewlett-Packard
- 1473 • Bob Blakley, IBM
- 1474 • Marlena Erdos, IBM
- 1475 • Marc Chanliau, Netegrity
- 1476 • Chris McLaren, Netegrity
- 1477 • Lynne Rosenthal, NIST
- 1478 • Mark Skall, NIST
- 1479 • Simon Godik, Overxeer
- 1480 • Charles Norwood, SAIC
- 1481 • Evan Prodromou, Securant
- 1482 • Robert Griffin, RSA Security (former editor)
- 1483 • Sai Allarvarpu, Sun Microsystems
- 1484 • Chris Ferris, Sun Microsystems
- 1485 • Emily Xu, Sun Microsystems
- 1486 • Mike Myers, Traceroute Security
- 1487 • Phillip Hallam-Baker, VeriSign (former editor)
- 1488 • James Vanderbeek, Vodafone
- 1489 • Mark O'Neill, Vordel
- 1490 • Tony Palmer, Vordel

1491
1492 Finally, the editors wish to acknowledge the following people for their contributions of material used as
1493 input to the OASIS Security Assertions Markup Language specifications:

- 1494 • Thomas Gross, IBM
- 1495 • Birgit Pfitzmann, IBM

Appendix B. Notices

1497 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
1498 might be claimed to pertain to the implementation or use of the technology described in this document or
1499 the extent to which any license under such rights might or might not be available; neither does it represent
1500 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
1501 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
1502 available for publication and any assurances of licenses to be made available, or the result of an attempt
1503 made to obtain a general license or permission for the use of such proprietary rights by implementors or
1504 users of this specification, can be obtained from the OASIS Executive Director.

1505 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
1506 other proprietary rights which may cover technology that may be required to implement this specification.
1507 Please address the information to the OASIS Executive Director.

1508 **Copyright © OASIS Open 2004. All Rights Reserved.**

1509 This document and translations of it may be copied and furnished to others, and derivative works that
1510 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
1511 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
1512 this paragraph are included on all such copies and derivative works. However, this document itself may
1513 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
1514 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
1515 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
1516 into languages other than English.

1517 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
1518 or assigns.

1519 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1520 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1521 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1522 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.