



---

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

5 **OASIS Standard, 15 March 2005**

6 **Document identifier:**

7 saml-metadata-2.0-os

8 **Location:**

9 <http://docs.oasis-open.org/security/saml/v2.0/>

10 **Editors:**

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

15 **SAML V2.0 Contributors:**

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

44 Greg Whitehead, Trustgenix

45 **Abstract:**

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

52 **Status:**

53 This is an **OASIS Standard** document produced by the Security Services Technical Committee. It  
54 was approved by the OASIS membership on 1 March 2005.

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

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

# Table of Contents

65	1 Introduction.....	5
66	1.1 Notation.....	5
67	2 Metadata for SAML V2.0.....	6
68	2.1 Namespaces .....	6
69	2.2 Common Types.....	7
70	2.2.1 Simple Type entityIDType.....	7
71	2.2.2 Complex Type EndpointType.....	7
72	2.2.3 Complex Type IndexedEndpointType.....	8
73	2.2.4 Complex Type localizedNameType.....	8
74	2.2.5 Complex Type localizedURIType.....	9
75	2.3 Root Elements.....	9
76	2.3.1 Element <EntitiesDescriptor>.....	9
77	2.3.2 Element <EntityDescriptor>.....	10
78	2.3.2.1 Element <Organization>.....	12
79	2.3.2.2 Element <ContactPerson>.....	12
80	2.3.2.3 Element <AdditionalMetadataLocation>.....	14
81	2.4 Role Descriptor Elements.....	14
82	2.4.1 Element <RoleDescriptor>.....	14
83	2.4.1.1 Element <KeyDescriptor>.....	15
84	2.4.2 Complex Type SSODescriptorType.....	16
85	2.4.3 Element <IDPSSODescriptor>.....	17
86	2.4.4 Element <SPSSODescriptor>.....	18
87	2.4.4.1 Element <AttributeConsumingService>.....	19
88	2.4.4.2 Element <RequestedAttribute>.....	19
89	2.4.5 Element <AuthnAuthorityDescriptor>.....	20
90	2.4.6 Element <PDPDescriptor>.....	20
91	2.4.7 Element <AttributeAuthorityDescriptor>.....	21
92	2.5 Element <AffiliationDescriptor>.....	22
93	2.6 Examples.....	23
94	3 Signature Processing.....	27
95	3.1 XML Signature Profile.....	27
96	3.1.1 Signing Formats and Algorithms.....	27
97	3.1.2 References.....	27
98	3.1.3 Canonicalization Method.....	27
99	3.1.4 Transforms.....	28
100	3.1.5 KeyInfo.....	28
101	4 Metadata Publication and Resolution.....	29
102	4.1 Publication and Resolution via Well-Known Location.....	29
103	4.1.1 Publication.....	29
104	4.1.2 Resolution.....	29
105	4.2 Publishing and Resolution via DNS.....	29
106	4.2.1 Publication.....	30
107	4.2.1.1 First Well Known Rule.....	30
108	4.2.1.2 The Order Field.....	30
109	4.2.1.3 The Preference Field.....	30
110	4.2.1.4 The Flag Field.....	31
111	4.2.1.5 The Service Field.....	31

112	4.2.1.6 The Regex and Replacement Fields.....	31
113	4.2.2 NAPTR Examples.....	32
114	4.2.2.1 Entity Metadata NAPTR Examples.....	32
115	4.2.2.2 Name Identifier Examples.....	32
116	4.2.3 Resolution.....	32
117	4.2.3.1 Parsing the Unique Identifier.....	32
118	4.2.3.2 Obtaining Metadata via the DNS.....	33
119	4.2.4 Metadata Location Caching.....	33
120	4.3 Post-Processing of Metadata.....	33
121	4.3.1 Metadata Instance Caching.....	33
122	4.3.2 Handling of HTTPS Redirects.....	33
123	4.3.3 Processing of XML Signatures and General Trust Processing.....	33
124	4.3.3.1 Processing Signed DNS Zones.....	34
125	4.3.3.2 Processing Signed Documents and Fragments.....	34
126	4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL.....	34
127	5 References.....	35
128	Appendix A.Registration of MIME media type application/samlmetadata+xml.....	37
129	Appendix B. Acknowledgments.....	41
130	Appendix C. Notices.....	43

---

# 1 Introduction

131

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

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

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

## 1.1 Notation

142

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

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

147

148 `Example code listings appear like this.`

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

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

Prefix	XML Namespace	Comments
saml:	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace [SAMLCore]. The prefix is generally elided in mentions of SAML assertion-related elements in text.
samlp:	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace [SAMLCore]. The prefix is generally elided in mentions of XML protocol-related elements in text.
md:	urn:oasis:names:tc:SAML:2.0:metadata	This is the SAML V2.0 metadata namespace, defined in a schema [SAMLMeta-xsd].
ds:	http://www.w3.org/2000/09/xmldsig#	This is the XML Signature namespace [XMLSig].
xenc:	http://www.w3.org/2001/04/xmlenc#	This is the XML Encryption namespace [XMLEnc].
xs:	http://www.w3.org/2001/XMLSchema	This namespace is defined in the W3C XML Schema specification [Schema1]. In schema listings, this is the default namespace and no prefix is shown. For clarity, the prefix is generally shown in specification text when XML Schema-related constructs are mentioned.

---

## 2 Metadata for SAML V2.0

152

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

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

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

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

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

### 2.1 Namespaces

173

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

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

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

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

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

```
201 Document identifier: saml-schema-metadata-2.0
202 Location: http://docs.oasis-open.org/security/saml/v2.0/
203 Revision history:
204     V2.0 (March, 2005):
205         Schema for SAML metadata, first published in SAML 2.0.
206     </documentation>
207 </annotation>
208 ...
209 </schema>
```

## 210 2.2 Common Types

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

### 213 2.2.1 Simple Type `entityIDType`

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

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

```
220 <simpleType name="entityIDType">
221     <restriction base="anyURI">
222         <maxLength value="1024"/>
223     </restriction>
224 </simpleType>
```

### 225 2.2.2 Complex Type `EndpointType`

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

229 `Binding` [Required]

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

232 `Location` [Required]

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

235 `ResponseLocation` [Optional]

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

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

243 In most contexts, elements of this type appear in unbounded sequences in the schema. This is to permit a  
244 protocol or profile to be offered by an entity at multiple endpoints, usually with different protocol bindings,  
245 allowing the metadata consumer to choose an appropriate endpoint for its needs. Multiple endpoints might

246 also offer "client-side" load-balancing or failover, particular in the case of a synchronous protocol binding.  
247 This element also permits the use of arbitrary elements and attributes defined in a non-SAML namespace.  
248 Any such content **MUST** be namespace-qualified.

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

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

### 260 2.2.3 Complex Type IndexedEndpointType

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

264 **index** [Required]

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

269 **isDefault** [Optional]

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

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

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

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

### 285 2.2.4 Complex Type localizedNameType

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

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



## 295 2.2.5 Complex Type localizedURIType

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

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

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

## 306 2.3 Root Elements

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

### 310 2.3.1 Element `<EntitiesDescriptor>`

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

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 Name [Optional]

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

324 `<ds:Signature>` [Optional]

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

327 `<Extensions>` [Optional]

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

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

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

333 When used as the root element of a metadata instance, this element **MUST** contain either a `validUntil`  
334 or `cacheDuration` attribute. It is **RECOMMENDED** that only the root element of a metadata instance  
335 contain either attribute.

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

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

### 359 **2.3.2 Element <EntityDescriptor>**

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

- 364 • SSO Identity Provider
- 365 • SSO Service Provider
- 366 • Authentication Authority
- 367 • Attribute Authority
- 368 • Policy Decision Point
- 369 • Affiliation

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

371 **entityID** [Required]

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

374 **ID** [Optional]

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

376 **validUntil** [Optional]

377 Optional attribute indicates the expiration time of the metadata contained in the element and any  
378 contained elements.

379 **cacheDuration** [Optional]

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

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

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

389 <RoleDescriptor>, <IDPSSODescriptor>, <SPSSODescriptor>,  
 390 <AuthnAuthorityDescriptor>, <AttributeAuthorityDescriptor>, <PDPDescriptor> [One  
 391 or More]  
 392 **OR**

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

396 <Organization> [Optional]  
 397         Optional element identifying the organization responsible for the SAML entity described by the  
 398         element.

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

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

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

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

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

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

```

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

```

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

```

### 443 2.3.2.1 Element <Organization>

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

448 <Extensions> [Optional]

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

452 <OrganizationName> [One or More]

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

454 <OrganizationDisplayName> [One or More]

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

456 <OrganizationURL> [One or More]

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

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

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

```

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

```

### 476 2.3.2.2 Element <ContactPerson>

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

480 consists of the following elements and attributes:

481 `contactType` [Required]

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

484 `<Extensions>` [Optional]

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

488 `<Company>` [Optional]

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

490 `<GivenName>` [Optional]

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

492 `<SurName>` [Optional]

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

494 `<EmailAddress>` [Zero or More]

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

497 `<TelephoneNumber>` [Zero or More]

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

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

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

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

### 529 2.3.2.3 Element <AdditionalMetadataLocation>

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

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

```
537 <element name="AdditionalMetadataLocation"  
538 type="md:AdditionalMetadataLocationType"/>  
539 <complexType name="AdditionalMetadataLocationType">  
540   <simpleContent>  
541     <extension base="anyURI">  
542       <attribute name="namespace" type="anyURI" use="required"/>  
543     </extension>  
544   </simpleContent>  
545 </complexType>
```

## 546 2.4 Role Descriptor Elements

547 The elements in this section make up the bulk of the operational support component of the metadata.  
548 Each element (save for the abstract one) defines a specific collection of operational behaviors in support  
549 of SAML profiles defined in [SAMLProf].

### 550 2.4.1 Element <RoleDescriptor>

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

555 ID [Optional]

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

557 validUntil [Optional]

558 Optional attribute indicates the expiration time of the metadata contained in the element and any  
559 contained elements.

560 cacheDuration [Optional]

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

563 protocolSupportEnumeration [Required]

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

569 errorURL [Optional]

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

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

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

```

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

```

#### 611 **2.4.1.1 Element `<KeyDescriptor>`**

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

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

620 the use of this element.

621 <EncryptionMethod> [Zero or More]

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

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

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

## 643 2.4.2 Complex Type SSODescriptorType

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

648 <ArtifactResolutionService> [Zero or More]

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

652 <SingleLogoutService> [Zero or More]

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

655 <ManageNameIDService> [Zero or More]

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

658 <NameIDFormat> [Zero or More]

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

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

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

```



```

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

```

### 683 2.4.3 Element <IDPSSODescriptor>

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

687 WantAuthnRequestsSigned [Optional]

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

690 <SingleSignOnService> [One or More]

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

694 <NameIDMappingService> [Zero or More]

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

698 <AssertionIDRequestService> [Zero or More]

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

702 <AttributeProfile> [Zero or More]

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

705 <saml:Attribute> [Zero or More]

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

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

```

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

```

```

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

```

## 736 2.4.4 Element <SPSSODescriptor>

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

740 AuthnRequestsSigned [Optional]

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

743 WantAssertionsSigned [Optional]

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

748 <AssertionConsumerService> [One or More]

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

752 <AttributeConsumingService> [Zero or More]

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

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

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

```

759 <element name="SPSSODescriptor" type="md:SPSSODescriptorType"/>
760 <complexType name="SPSSODescriptorType">
761     <complexContent>
762         <extension base="md:SSODescriptorType">
763             <sequence>
764                 <element ref="md:AssertionConsumerService"
765 maxOccurs="unbounded"/>
766                 <element ref="md:AttributeConsumingService" minOccurs="0"
767 maxOccurs="unbounded"/>
768             </sequence>

```

```

769         <attribute name="AuthnRequestsSigned" type="boolean"
770 use="optional"/>
771         <attribute name="WantAssertionsSigned" type="boolean"
772 use="optional"/>
773     </extension>
774 </complexContent>
775 </complexType>
776 <element name="AssertionConsumerService" type="md:IndexedEndpointType"/>

```

#### 777 2.4.4.1 Element <AttributeConsumingService>

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

781 index [Required]

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

784 isDefault [Optional]

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

787 <ServiceName> [One or More]

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

789 <ServiceDescription> [Zero or More]

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

791 <RequestedAttribute> [One or More]

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

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

```

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

```

#### 809 2.4.4.2 Element <RequestedAttribute>

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

813 isRequired [Optional]

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

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

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

```
820 <element name="RequestedAttribute" type="md:RequestedAttributeType"/>
821 <complexType name="RequestedAttributeType">
822   <complexContent>
823     <extension base="saml:AttributeType">
824       <attribute name="isRequired" type="boolean" use="optional"/>
825     </extension>
826   </complexContent>
827 </complexType>
```

## 828 2.4.5 Element `<AuthnAuthorityDescriptor>`

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

832 `<AuthnQueryService>` [One or More]

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

836 `<AssertionIDRequestService>` [Zero or More]

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

840 `<NameIDFormat>` [Zero or More]

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

843 The following schema fragment defines the `<AuthnAuthorityDescriptor>` element and its  
844 **AuthnAuthorityDescriptorType** complex type:

```
845 <element name="AuthnAuthorityDescriptor"
846 type="md:AuthnAuthorityDescriptorType"/>
847 <complexType name="AuthnAuthorityDescriptorType">
848   <complexContent>
849     <extension base="md:RoleDescriptorType">
850       <sequence>
851         <element ref="md:AuthnQueryService" maxOccurs="unbounded"/>
852         <element ref="md:AssertionIDRequestService" minOccurs="0"
853 maxOccurs="unbounded"/>
854         <element ref="md:NameIDFormat" minOccurs="0"
855 maxOccurs="unbounded"/>
856       </sequence>
857     </extension>
858   </complexContent>
859 </complexType>
860 <element name="AuthnQueryService" type="md:EndpointType"/>
```

## 861 2.4.6 Element `<PDPDescriptor>`

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

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

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

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

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

```
878 <element name="PDPDescriptor" type="md:PDPDescriptorType"/>  
879 <complexType name="PDPDescriptorType">  
880 <complexContent>  
881 <extension base="md:RoleDescriptorType">  
882 <sequence>  
883 <element ref="md:AuthzService" maxOccurs="unbounded"/>  
884 <element ref="md:AssertionIDRequestService" minOccurs="0"  
885 maxOccurs="unbounded"/>  
886 <element ref="md:NameIDFormat" minOccurs="0"  
887 maxOccurs="unbounded"/>  
888 </sequence>  
889 </extension>  
890 </complexContent>  
891 </complexType>  
892 <element name="AuthzService" type="md:EndpointType"/>
```

## 893 2.4.7 Element <AttributeAuthorityDescriptor>

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

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

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

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

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

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

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

```
918 <element name="AttributeAuthorityDescriptor"  
919 type="md:AttributeAuthorityDescriptorType"/>  
920 <complexType name="AttributeAuthorityDescriptorType">  
921 <complexContent>  
922 <extension base="md:RoleDescriptorType">  
923 <sequence>  
924 <element ref="md:AttributeService" maxOccurs="unbounded"/>  
925 <element ref="md:AssertionIDRequestService" minOccurs="0"  
926 maxOccurs="unbounded"/>  
927 <element ref="md:NameIDFormat" minOccurs="0"  
928 maxOccurs="unbounded"/>  
929 <element ref="md:AttributeProfile" minOccurs="0"  
930 maxOccurs="unbounded"/>  
931 <element ref="saml:Attribute" minOccurs="0"  
932 maxOccurs="unbounded"/>  
933 </sequence>  
934 </extension>  
935 </complexContent>  
936 </complexType>  
937 <element name="AttributeService" type="md:EndpointType"/>
```

## 938 2.5 Element <AffiliationDescriptor>

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

945 affiliationOwnerID [Required]

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

949 ID [Optional]

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

951 validUntil [Optional]

952 Optional attribute indicates the expiration time of the metadata contained in the element and any  
953 contained elements.

954 cacheDuration [Optional]

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

957 <ds:Signature> [Optional]

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



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

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

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

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

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

```

974 <element name="AffiliationDescriptor" type="md:AffiliationDescriptorType"/>
975 <complexType name="AffiliationDescriptorType">
976   <sequence>
977     <element ref="ds:Signature" minOccurs="0"/>
978     <element ref="md:Extensions" minOccurs="0"/>
979     <element ref="md:AffiliateMember" maxOccurs="unbounded"/>
980     <element ref="md:KeyDescriptor" minOccurs="0" maxOccurs="unbounded"/>
981   </sequence>
982   <attribute name="affiliationOwnerID" type="md:entityIDType"
983 use="required"/>
984   <attribute name="validUntil" type="dateTime" use="optional"/>
985   <attribute name="cacheDuration" type="duration" use="optional"/>
986   <attribute name="ID" type="ID" use="optional"/>
987   <anyAttribute namespace="##other" processContents="lax"/>
988 </complexType>
989 <element name="AffiliateMember" type="md:entityIDType"/>

```

## 990 2.6 Examples

991 The following is an example of metadata for a SAML system entity acting as an identity provider and an  
 992 attribute authority. A signature is shown as a placeholder, without the actual content.  
 993

```

994 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
995   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
996   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
997   entityID="https://IdentityProvider.com/SAML">
998   <ds:Signature>...</ds:Signature>
999   <IDPSSODescriptor WantAuthnRequestsSigned="true"
1000     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1001     <KeyDescriptor use="signing">
1002       <ds:KeyInfo>
1003         <ds:KeyName>IdentityProvider.com SSO Key</ds:KeyName>
1004       </ds:KeyInfo>
1005     </KeyDescriptor>
1006     <ArtifactResolutionService isDefault="true" index="0"
1007       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1008       Location="https://IdentityProvider.com/SAML/Artifact"/>
1009     <SingleLogoutService
1010       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1011       Location="https://IdentityProvider.com/SAML/SLO/SOAP"/>
1012     <SingleLogoutService
1013       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1014       Location="https://IdentityProvider.com/SAML/SLO/Browser"
1015       ResponseLocation="https://IdentityProvider.com/SAML/SLO/Response"/>

```

```

1016 <NameIDFormat>
1017   urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1018 </NameIDFormat>
1019 <NameIDFormat>
1020   urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1021 </NameIDFormat>
1022 <NameIDFormat>
1023   urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1024 </NameIDFormat>
1025 <SingleSignOnService
1026   Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1027   Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1028 </SingleSignOnService
1029   Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1030   Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1031 <saml:Attribute
1032   NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1033   Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1034   FriendlyName="eduPersonPrincipalName">
1035 </saml:Attribute>
1036 <saml:Attribute
1037   NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1038   Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1039   FriendlyName="eduPersonAffiliation">
1040   <saml:AttributeValue>member</saml:AttributeValue>
1041   <saml:AttributeValue>student</saml:AttributeValue>
1042   <saml:AttributeValue>faculty</saml:AttributeValue>
1043   <saml:AttributeValue>employee</saml:AttributeValue>
1044   <saml:AttributeValue>staff</saml:AttributeValue>
1045 </saml:Attribute>
1046 </IDPSSODescriptor>
1047 <AttributeAuthorityDescriptor
1048   protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1049   <KeyDescriptor use="signing">
1050     <ds:KeyInfo>
1051       <ds:KeyName>IdentityProvider.com AA Key</ds:KeyName>
1052     </ds:KeyInfo>
1053   </KeyDescriptor>
1054   <AttributeService
1055     Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1056     Location="https://IdentityProvider.com/SAML/AA/SOAP"/>
1057   <AssertionIDRequestService
1058     Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
1059     Location="https://IdentityProvider.com/SAML/AA/URI"/>
1060   <NameIDFormat>
1061     urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1062   </NameIDFormat>
1063   <NameIDFormat>
1064     urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1065   </NameIDFormat>
1066   <NameIDFormat>
1067     urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1068   </NameIDFormat>
1069   <saml:Attribute
1070     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1071     Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1072     FriendlyName="eduPersonPrincipalName">
1073   </saml:Attribute>
1074   <saml:Attribute
1075     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1076     Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1077     FriendlyName="eduPersonAffiliation">
1078     <saml:AttributeValue>member</saml:AttributeValue>
1079     <saml:AttributeValue>student</saml:AttributeValue>
1080     <saml:AttributeValue>faculty</saml:AttributeValue>
1081     <saml:AttributeValue>employee</saml:AttributeValue>
1082     <saml:AttributeValue>staff</saml:AttributeValue>

```



```

1083     </saml:Attribute>
1084   </AttributeAuthorityDescriptor>
1085   <Organization>
1086     <OrganizationName xml:lang="en">Identity Providers R
1087   US</OrganizationName>
1088     <OrganizationDisplayName xml:lang="en">
1089       Identity Providers R US, a Division of Lerxst Corp.
1090     </OrganizationDisplayName>
1091     <OrganizationURL
1092   xml:lang="en">https://IdentityProvider.com</OrganizationURL>
1093     </Organization>
1094   </EntityDescriptor>
1095

```

1096 The following is an example of metadata for a SAML system entity acting as a service provider. A  
 1097 signature is shown as a placeholder, without the actual content. For illustrative purposes, the service is  
 1098 one that does not require users to uniquely identify themselves, but rather authorizes access on the basis  
 1099 of a role-like attribute.  
 1100

```

1101 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1102   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1103   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1104   entityID="https://ServiceProvider.com/SAML">
1105   <ds:Signature>...</ds:Signature>
1106   <SPSSODescriptor AuthnRequestsSigned="true"
1107     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1108     <KeyDescriptor use="signing">
1109       <ds:KeyInfo>
1110         <ds:KeyName>ServiceProvider.com SSO Key</ds:KeyName>
1111       </ds:KeyInfo>
1112     </KeyDescriptor>
1113     <KeyDescriptor use="encryption">
1114       <ds:KeyInfo>
1115         <ds:KeyName>ServiceProvider.com Encrypt Key</ds:KeyName>
1116       </ds:KeyInfo>
1117       <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmenc#rsa-
1118   1_5"/>
1119     </KeyDescriptor>
1120     <SingleLogoutService
1121       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1122       Location="https://ServiceProvider.com/SAML/SLO/SOAP"/>
1123     <SingleLogoutService
1124       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1125       Location="https://ServiceProvider.com/SAML/SLO/Browser"
1126       ResponseLocation="https://ServiceProvider.com/SAML/SLO/Response"/>
1127     <NameIDFormat>
1128       urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1129     </NameIDFormat>
1130     <AssertionConsumerService isDefault="true" index="0"
1131       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
1132       Location="https://ServiceProvider.com/SAML/SSO/Artifact"/>
1133     <AssertionConsumerService index="1"
1134       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1135       Location="https://ServiceProvider.com/SAML/SSO/POST"/>
1136     <AttributeConsumingService index="0">
1137       <ServiceName xml:lang="en">Academic Journals R US</ServiceName>
1138       <RequestedAttribute
1139         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1140         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7"
1141         FriendlyName="eduPersonEntitlement">
1142         <saml:AttributeValue>
1143           https://ServiceProvider.com/entitlements/123456789
1144         </saml:AttributeValue>
1145       </RequestedAttribute>
1146     </AttributeConsumingService>
1147   </SPSSODescriptor>
1148 </Organization>

```

```
1149         <OrganizationName xml:lang="en">Academic Journals R
1150 US</OrganizationName>
1151         <OrganizationDisplayName xml:lang="en">
1152           Academic Journals R US, a Division of Dirk Corp.
1153         </OrganizationDisplayName>
1154         <OrganizationURL
1155 xml:lang="en">https://ServiceProvider.com</OrganizationURL>
1156         </Organization>
1157 </EntityDescriptor>
```

---

## 1158 3 Signature Processing

1159 Various elements in a metadata instance can be digitally signed (as indicated by the element's inclusion of  
1160 a `<ds:Signature>` element), with the following benefits:

- 1161 • Metadata integrity
- 1162 • Authentication of the metadata by a trusted signer

1163 A digital signature is not always required, for example if the relying party obtains the information directly  
1164 from the publishing entity directly (with no intermediaries) through a secure channel, with the entity having  
1165 authenticated to the relying party by some means other than a digital signature.

1166 Many different techniques are available for "direct" authentication and secure channel establishment  
1167 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,  
1168 the applicable security requirements depend on the communicating applications.

1169 Additionally, elements can inherit signatures on enclosing parent elements that are themselves signed.

1170 In the absence of such context, it is RECOMMENDED that at least the root element of a metadata  
1171 instance be signed.

### 1172 3.1 XML Signature Profile

1173 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility  
1174 and many choices. This section details the constraints on these facilities so that metadata processors do  
1175 not have to deal with the full generality of XML Signature processing. This usage makes specific use of  
1176 the **xs:ID**-typed attributes optionally present on the elements to which signatures can apply. These  
1177 attributes are collectively referred to in this section as the identifier attributes.

#### 1178 3.1.1 Signing Formats and Algorithms

1179 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and  
1180 detached.

1181 SAML metadata MUST use enveloped signatures when signing the elements defined in this specification.  
1182 SAML processors SHOULD support the use of RSA signing and verification for public key operations in  
1183 accordance with the algorithm identified by <http://www.w3.org/2000/09/xmlsig#rsa-sha1>.

#### 1184 3.1.2 References

1185 Signed metadata elements MUST supply a value for the identifier attribute on the signed element. The  
1186 element may or may not be the root element of the actual XML document containing the signed metadata  
1187 element.

1188 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute  
1189 value of the metadata element being signed. For example, if the identifier attribute value is "foo", then the  
1190 URI attribute in the `<ds:Reference>` element MUST be "#foo".

1191 As a consequence, a metadata element's signature MUST apply to the content of the signed element and  
1192 any child elements it contains.

#### 1193 3.1.3 Canonicalization Method

1194 SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the  
1195 `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a `<ds:Transform>`

1197 embedded in an XML context can be verified independent of that context.

### 1198 **3.1.4 Transforms**

1199 Signatures in SAML metadata SHOULD NOT contain transforms other than the enveloped signature  
1200 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive  
1201 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or  
1202 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

1203 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do  
1204 not, verifiers MUST ensure that no content of the signed metadata element is excluded from the  
1205 signature. This can be accomplished by establishing out-of-band agreement as to what transforms are  
1206 acceptable, or by applying the transforms manually to the content and reverifying the result as consisting  
1207 of the same SAML metadata.

### 1208 **3.1.5 KeyInfo**

1209 XML Signature [XMLSig] defines usage of the `<ds:KeyInfo>` element. SAML does not require the  
1210 use of `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY  
1211 be absent.

---

## 1212 4 Metadata Publication and Resolution

1213 Two mechanisms are provided for an entity to publish (and for a consumer to resolve the location of)  
1214 metadata documents: via a "well-known-location" by directly dereferencing the entity's unique identifier (a  
1215 URI variously referred to as an *entityID* or *providerID*), or indirectly by publishing the location of metadata  
1216 in the DNS. Other out-of-band mechanisms are of course also permitted. A consumer that supports both  
1217 approaches defined in this document MUST attempt resolution via DNS before using the "well-known-  
1218 location" mechanism.

1219 When retrieval requires network transport of the document, the transport SHOULD be protected with  
1220 mechanisms providing server authentication and integrity protection. For example, HTTP-based resolution  
1221 SHOULD be protected with TLS/SSL [RFC 2246] as amended by [RFC3546].

1222 Various mechanisms are described in this section to aid in establishing trust in the accuracy and  
1223 legitimacy of metadata, including use of XML signatures, SSL/TLS server authentication, and DNS  
1224 signatures. Regardless of the mechanism(s) used, relying parties SHOULD have some means by which to  
1225 establish trust in metadata information before relying on it.

### 1226 4.1 Publication and Resolution via Well-Known Location

1227 The following sections describe publication and resolution of metadata by means of a well-known location.

#### 1228 4.1.1 Publication

1229 Entities MAY publish their metadata documents at a well known location by placing the document at the  
1230 location denoted by its unique identifier, which MUST be in the form of a URL (rather than a URN). See  
1231 Section 8.3.6 of [SAMLCore] for more information about such identifiers. It is STRONGLY  
1232 RECOMMENDED that `https` URLs be used for this purpose. An indirection mechanism supported by the  
1233 URL scheme (such as an HTTP 1.1 302 redirect) MAY be used if the document is not placed directly at  
1234 the location. If the publishing protocol permits MIME-based identification of content types, the content type  
1235 of the metadata instance MUST be `application/samlmetadata+xml`.

1236 The XML document provided at the well-known location MUST describe the metadata only for the entity  
1237 represented by the unique identifier (that is, the root element MUST be an `<EntityDescriptor>` with  
1238 an `entityID` matching the location). If other entities need to be described, the  
1239 `<AdditionalMetadataLocation>` element MUST be used. Thus the `<EntitiesDescriptor>`  
1240 element MUST NOT be used in documents published using this mechanism, since a group of entities are  
1241 not defined by such an identifier.

#### 1242 4.1.2 Resolution

1243 If an entity's unique identifier is a URL, metadata consumers MAY attempt to resolve an entity's unique  
1244 identifier directly, in a scheme-specific manner, by dereferencing the identifier.

### 1245 4.2 Publishing and Resolution via DNS

1246 To improve the accessibility of metadata documents and provide additional indirection between an entity's  
1247 unique identifier and the location of metadata, entities MAY publish their metadata document locations in a  
1248 zone of their corresponding DNS [RFC1034]. The entity's unique identifier (a URI) is used as the input to  
1249 the process. Since URIs are flexible identifiers, location publication methods and the resolution process  
1250 are determined by the URI's scheme and fully-qualified name. URI locations for metadata are

1251 subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in [RFC2915]  
1252 and [RFC3403].

1253 It is RECOMMENDED that entities publish their resource records in signed zone files using [RFC2535]  
1254 such that relying parties may establish the validity of the published location and authority of the zone, and  
1255 integrity of the DNS response. If DNS zone signatures are present, relying parties MUST properly validate  
1256 the signature.

## 1257 **4.2.1 Publication**

1258 This specification makes use of the NAPTR resource record described in [RFC2915] and [RFC3403].  
1259 Familiarity with these documents is encouraged.

1260 Dynamic Delegation Discovery System (DDDS) [RFC3401] is a general purpose system for the retrieval of  
1261 information based on an application-specific input string and the application of well known rules to  
1262 transform that string until a terminal condition is reached requiring a look-up into an application-specific  
1263 defined database or resolution of a URL based on the rules defined by the application. DDDS defines a  
1264 specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS  
1265 necessary to apply DDDS rules.

1266 Entities MAY publish separate URLs when multiple metadata documents need to be distributed, or when  
1267 different metadata documents are required due to multiple trust relationships that require separate keying  
1268 material, or when service interfaces require separate metadata declarations. This may be accomplished  
1269 through the use of the optional `<AdditionalMetadataLocation>` element, or through the regexp  
1270 facility and multiple service definition fields in the NAPTR resource record itself.

1271 If the publishing protocol permits MIME-based identification of content types, the content type of the  
1272 metadata instance MUST be `application/samlmetadata+xml`.

1273 If the entity's unique identifier is a URN, publication of the corresponding metadata location proceeds as  
1274 specified in [RFC3404]. Otherwise, the resolution of the metadata location proceeds as specified below.

1275 The following is the application-specific profile of DDDS for SAML metadata resolution.

### 1276 **4.2.1.1 First Well Known Rule**

1277 The "first well-known-rule" for processing SAML metadata resolution is to parse the entity's unique  
1278 identifier and extract the fully-qualified domain name (subexpression 3) as described in Section 4.2.3.1.

### 1279 **4.2.1.2 The Order Field**

1280 The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY  
1281 provide multiple NAPTR resource records which MUST be processed by the resolver application in the  
1282 order indicated by this field.

### 1283 **4.2.1.3 The Preference Field**

1284 For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving  
1285 application. The resolving application MAY ignore this order, in cases where the service field value does  
1286 not meet the resolver's requirements (e.g.: the resource record returns a protocol the application does not  
1287 support).

#### 1288 4.2.1.4 The Flag Field

1289 SAML metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying  
1290 additional resource records are to be processed). The "U" flag indicates that the output of the rule is a  
1291 URI.

#### 1292 4.2.1.5 The Service Field

1293 The SAML-specific service field, as described in the following BNF, declares the modes by which instance  
1294 document(s) shall be made available:

```
1295 servicefield = 1("PID2U" / "NID2U") "+" proto [*(":" class) *(":" servicetype)]
1296 proto = 1("https" / "uddi")
1297 class = 1[ "entity" / "entitygroup" ]
1298 servicetype = 1(si / "spssso" / "idpssso" / "authn" / "authnauth" / "pdp" / "attrauth" /
1299 alphanum )
1300 si = "si" [ ":" alphanum ] [ ":" endpoint ]
1301 alphanum = 1*32( ALPHA / DIGIT )
```

1302 where:

- 1303 • servicefield PID2U resolves an entity's unique identifier to metadata URL.
- 1304 • servicefield NID2U resolves a principal's <NameID> into a metadata URL.
- 1305 • proto describes the retrieval protocol (`https` or `uddi`). In the case of UDDI, the URL will be an  
1306 http(s) URL referencing a WSDL document.
- 1307 • class identifies whether the referenced metadata document describes a single entity, or multiple.  
1308 In the latter case, the referenced document MUST contain the entity defined by the original unique  
1309 identifier as a member of a group of entities within the document itself such as an  
1310 <AffiliationDescriptor> or <EntitiesDescriptor>.
- 1311 • servicetype allows an entity to publish metadata for distinct roles and services as separate  
1312 documents. Resolvers who encounter multiple servicetype declarations will dereference the  
1313 appropriate URI, depending on which service is required for an operation (e.g.: an entity operating  
1314 both as an identity provider and a service provider can publish metadata for each role at different  
1315 locations). The `authn` service type represents a <SingleSignOnService> endpoint.
- 1316 • si (with optional endpoint component) allows the publisher to either directly publish the metadata  
1317 for a service instance, or by articulating a SOAP endpoint (using `endpoint`).

1318 For example:

- 1319 • PID2U+https:entity - represents the entity's complete metadata document available via the  
1320 https protocol
- 1321 • PID2U+uddi:entity:si:foo - represents the WSDL document location that describes a service  
1322 instance "foo"
- 1323 • PID2U+https:entitygroup:idpssso - represents the metadata for a group of entities acting as  
1324 SSO identity providers, of which the original entity is a member.
- 1325 • NID2U+https:idp - represents the metadata for the SSO identity provider of a principal

#### 1326 4.2.1.6 The Regex and Replacement Fields

1327 The expected output after processing the input string through the regex MUST be a valid `https` URL or  
1328 UDDI node (WSDL document) address.

## 1329 4.2.2 NAPTR Examples

### 1330 4.2.2.1 Entity Metadata NAPTR Examples

1331 Entities publish metadata URLs in the following manner:

```
1332 $ORIGIN provider.biz
1333
1334 ;; order pref f service regexp or replacement
1335
1336 IN NAPTR 100 10 "U" PID2U+https:entity
1337     "!^.*$!https://host.provider.biz/some/directory/trust.xml!" ""
1338 IN NAPTR 110 10 "U" PID2U+https: entity:trust
1339     "!^.*$!https://foo.provider.biz:1443/mdtrust.xml!" ""
1340 IN NAPTR 125 10 "U" PID2U+https:"
1341 IN NAPTR 110 10 "U" PID2U+uddi:entity
1342     "!^.*$!https://this.uddi.node.provider.biz/libmd.wsdl" ""
```

### 1343 4.2.2.2 Name Identifier Examples

1344 A principal's employer `example.int` operates an identity provider which may be used by an office supply  
1345 company to authenticate authorized buyers. The supplier takes a users' email address  
1346 `buyer@example.int` as input to the resolution process, and parses the email address to extract the  
1347 FQDN (`example.int`). The employer publishes the following NAPTR record in the `example.int` DNS:

```
1348 $ORIGIN example.int
1349
1350 IN NAPTR 100 10 "U" NID2U+https:authn
1351     "!^([\^@]+)@(.*)$!https://serv.example.int:8000/cgi-bin/getmd?\1!" ""
1352 IN NAPTR 100 10 "U" NID2U+https:idp
1353     "!^([\^@]+)@(.*)$!https://auth.example.int/app/auth?\1" ""
```

## 1354 4.2.3 Resolution

1355 When resolving metadata for an entity via the DNS, the unique identifier of the entity is used as the initial  
1356 input into the resolution process, rather than as an actual location Proceed as follows:

- 1357 • If the unique identifier is a URN, proceed with the resolution steps as defined in [RFC3404].
- 1358 • Otherwise, parse the identifier to obtain the fully-qualified domain name.
- 1359 • Query the DNS for NAPTR resource records of the domain iteratively until a terminal resource  
1360 record is returned.
- 1361 • Identify which resource record to use based on the service fields, then order fields, then preference  
1362 fields of the result set.
- 1363 • Obtain the document(s) at the provided location(s) as required by the application.

### 1364 4.2.3.1 Parsing the Unique Identifier

1365 To initiate the resolution of the location of the metadata information, it will be necessary in some cases to  
1366 decompose the entity's unique identifier (expressed as a URI) into one or more atomic elements.

1367 The following regular expression should be used when initiating the decomposition process:

```
1368 ^([\^:/?#]+)?/*([\^:/?#]*@)?(((\^/?:#*\.)*((\^/?#:\.)(\^/?#:\.]+)))
1369 (:\.d+)?([\^?#]*)(\?[\^#]*)?(#[.]*)?$
1370     1           2           34           56           7           8
1371     9           10          11
```

1372 Subexpression 3 MUST result in a Fully-Qualified Domain Name (FQDN), which will be the basis for  
1373 retrieving metadata locations from this zone.



#### 1374 **4.2.3.2 Obtaining Metadata via the DNS**

1375 Upon completion of the parsing of the identifier, the application then performs a DNS query for the resulting  
1376 domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses.  
1377 Applications MAY exclude from the result set any service definitions that do not concern the present  
1378 request operations.

1379 Resolving applications MUST subsequently order the result set according to the order field, and MAY  
1380 order the result set based on the preference set. Resolvers are NOT REQUIRED to follow the ordering of  
1381 the preferences field. The resulting NAPTR resource record(s) are operated on iteratively (based on the  
1382 order flag) until a terminal NAPTR resource record is reached.

1383 The result will be a well-formed, absolute URL, which is then used to retrieve the metadata document.

#### 1384 **4.2.4 Metadata Location Caching**

1385 Location caching MUST NOT exceed the TTL of the DNS zone from which the location was derived.  
1386 Resolvers MUST obtain a fresh copy of the metadata location upon reaching the expiration of the TTL of  
1387 the zone.

1388 Publishers of metadata documents should carefully consider the TTL of the zone when making changes  
1389 to metadata document locations. Should such a location change occur, a publisher MUST either keep the  
1390 document at both the old and new location until all conforming resolvers are certain to have the updated  
1391 location (e.g.: time of zone change + TTL), or provide an HTTP Redirect [RFC2616] response at the old  
1392 location specifying the new location.

### 1393 **4.3 Post-Processing of Metadata**

1394 The following sections describe the post-processing of metadata.

#### 1395 **4.3.1 Metadata Instance Caching**

1396 Document caching MUST NOT exceed the `validUntil` or `cacheDuration` attribute of the subject  
1397 element(s). If metadata elements have parent elements which contain caching policies, the parent  
1398 element takes precedence.

1399 To properly process the `cacheDuration` attribute, consumers MUST retain the date and time when the  
1400 document was retrieved.

1401 When a document or element has expired, the consumer MUST retrieve a fresh copy, which may require  
1402 a refresh of the document location(s). Consumers SHOULD process document cache processing  
1403 according to [RFC2616] Section 13, and MAY request the Last-Modified date and time from the HTTP  
1404 server. Publishers SHOULD ensure acceptable cache processing as described in [RFC2616] (Section  
1405 10.3.5 304 Not Modified).

#### 1406 **4.3.2 Handling of HTTPS Redirects**

1407 Publishers MAY issue an HTTP Redirect (301 Moved Permanently, 302 or 307 Temporary Redirect)  
1408 [RFC2616], and user agents MUST follow the specified URL in the Redirect response. Redirects  
1409 SHOULD be of the same protocol as the initial request.

#### 1410 **4.3.3 Processing of XML Signatures and General Trust Processing**

1411 Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and  
1412 for the trust ascribed to the entity described by such metadata:

- 1413 • Trust derived from the signature of the DNS zone from which the metadata location URL was

1414 resolved, ensuring accuracy of the metadata document location(s)

- 1415 • Trust derived from signature processing of the metadata document itself, ensuring the integrity of
- 1416 the XML document
- 1417 • Trust derived from the SSL/TLS server authentication of the metadata location URL, ensuring the
- 1418 identity of the publisher of the metadata

1419 Post-processing of the metadata document MUST include signature processing at the XML-document

1420 level and MAY include one of the other two processes. Specifically, the relying party MAY choose to trust

1421 any of the cited authorities in the resolution and parsing process. Publishers of metadata MUST employ a

1422 document-integrity mechanism and MAY employ any of the other two processing profiles to establish trust

1423 in the metadata document, governed by implementation policies.

#### 1424 **4.3.3.1 Processing Signed DNS Zones**

1425 Verification of DNS zone signature SHOULD be processed, if present, as described in [RFC2535].

#### 1426 **4.3.3.2 Processing Signed Documents and Fragments**

1427 Published metadata documents SHOULD be signed, as described in Section 3, either by a certificate

1428 issued to the subject of the document, or by another trusted party. Publishers MAY consider signatures of

1429 other parties as a means of trust conveyance.

1430 Metadata consumers MUST validate signatures, when present, on the metadata document as described

1431 by Section 3.

#### 1432 **4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL**

1433 It is STRONGLY RECOMMENDED that publishers implement TLS/SSL URLs; therefore, consumers

1434 SHOULD consider the trust inherited from the issuer of the TLS/SSL certificate. Publication URLs may not

1435 always be located in the domain of the subject of the metadata document; therefore, consumers SHOULD

1436 NOT presume certificates whose subject is the entity in question, as it may be hosted by another trusted

1437 party.

1438 As the basis of this trust may not be available against a cached document, other mechanisms SHOULD

1439 be used under such circumstances.

---

## 5 References

1440

- 1441 [RFC1034] P. Mockapetris. *Domain Names – Concepts and Facilities*. IETF RFC 1034,  
1442 November 1987. See <http://www.ietf.org/rfc/rfc1034.txt>.
- 1443 [RFC2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*,  
1444 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1445 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999.  
1446 See <http://www.ietf.org/rfc/rfc2246.txt>.
- 1447 [RFC2535] D. Eastlake. *Domain Name System Security Extensions*. IETF RFC 2535, March  
1448 1999. See <http://www.ietf.org/rfc/rfc2535.txt>.
- 1449 [RFC2616] R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. IETF RFC 2616, June  
1450 1999. See <http://www.ietf.org/rfc/rfc2616.txt>.
- 1451 [RFC2915] M. Mealling. *The Naming Authority Pointer (NAPTR) DNS Resource Record*.  
1452 IETF RFC 2915, September 2000. See <http://www.ietf.org/rfc/rfc2915.txt>.
- 1453 [RFC3401] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part One: The  
1454 Comprehensive DDDS*. IETF RFC 3401, October 2002. See  
1455 <http://www.ietf.org/rfc/rfc3401.txt>.
- 1456 [RFC3403] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Three: The  
1457 Domain Name System (DNS) Database*. IETF RFC 3403, October 2002. See  
1458 <http://www.ietf.org/rfc/rfc3403.txt>.
- 1459 [RFC3404] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Four: The  
1460 Uniform Resource Identifiers (URI) Resolution Application*. IETF RFC 3404,  
1461 October 2002. See <http://www.ietf.org/rfc/rfc3404.txt>.
- 1462 [RFC3546] S. Blake-Wilson et al. *Transport Layer Security (TLS) Extensions*. IETF RFC  
1463 3546, June 2003. See <http://www.ietf.org/rfc/rfc3546.txt>.
- 1464 [SAMLBind] S. Cantor et al. *Bindings for the OASIS Security Assertion Markup Language  
1465 (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os.  
1466 See <http://www.oasis-open.org/committees/security/>.
- 1467 [SAMLConform] P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion  
1468 Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-  
1469 conformance-2.0-os. <http://www.oasis-open.org/committees/security/>.
- 1470 [SAMLCore] S. Cantor et al. *Assertions and Protocols for the OASIS Security Assertion  
1471 Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-  
1472 core-2.0-os. See <http://www.oasis-open.org/committees/security/>.
- 1473 [SAMLMeta-xsd] S. Cantor et al. *SAML metadata schema*. OASIS SSTC, March 2005. Document  
1474 ID saml-schema-metadata-2.0. See [http://www.oasis-  
1475 open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1476 [SAMLProf] S. Cantor et al. *Profiles for the OASIS Security Assertion Markup Language  
1477 (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-profiles-2.0-os. See  
1478 <http://www.oasis-open.org/committees/security/>.
- 1479 [SAMLSec] F. Hirsch et al. *Security and Privacy Considerations for the OASIS Security  
1480 Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document  
1481 ID saml-sec-consider-2.0-os. See [http://www.oasis-  
1482 open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1483 [Schema1] H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web  
1484 Consortium Recommendation, May 2001. See [http://www.w3.org/TR/xmlschema-  
1485 1/](http://www.w3.org/TR/xmlschema-1/). Note that this specification normatively references [Schema2], listed below.
- 1486 [Schema2] P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium  
1487 Recommendation, May 2001. See <http://www.w3.org/TR/xmlschema->

1489

<http://www.w3.org/TR/xmlenc-core/>, World Wide Web Consortium.

1490

**[XMLSig]**

D. Eastlake et al. XML-Signature Syntax and Processing,

1491

<http://www.w3.org/TR/xmldsig-core/>, World Wide Web Consortium.

---

1492 **Appendix A.Registration of MIME media type**  
1493 **application/samlmetadata+xml**

1494 **Introduction**

1495 This document defines a MIME media type -- `application/samlmetadata+xml` -- for use  
1496 with the XML serialization of Security Assertion Markup Language metadata.  
1497

1498 SAML is a work product of the OASIS Security Services Technical Committee [SSTC]. The SAML  
1499 specifications define XML-based constructs with which one may make, and convey, security  
1500 assertions. Using SAML, one can assert that an authentication event pertaining to some subject  
1501 has occurred and convey said assertion to a relying party, for example.  
1502

1503 SAML profiles require agreements between system entities regarding identifiers, binding support,  
1504 endpoints, certificates, keys, and so forth. Such information is treated as metadata by SAML v2.0.  
1505 [SAMLv2Meta] specifies this metadata, as well as specifying metadata publication and resolution  
1506 mechanisms. If the publishing protocol permits MIME-based identification of content types, then  
1507 use of the `application/samlmetadata+xml` MIME media type is required.

1508 **MIME media type name**

1509 `application`

1510 **MIME subtype name**

1511 `samlmetadata+xml`

1512 **Required parameters**

1513 `None`

1514 **Optional parameters**

1515 `charset`

1516 Same as `charset` parameter of `application/xml` [RFC3023].

1517 **Encoding considerations**

1518 Same as for `application/xml` [RFC3023].

1519 **Security considerations**

1520 Per their specification, `samlmetadata+xml` typed objects do not contain executable content.  
1521 However, these objects are XML-based [XML], and thus they have all of the general security  
1522 considerations presented in Section10 of [RFC3023].

1523 SAML metadata [SAMLv2Meta] contains information whose integrity and authenticity is important  
1524 – identity provider and service provider public keys and endpoint addresses, for example.

1525 To counter potential issues, the publisher may sign `samlmetadata+xml` typed objects. Any such  
1526 signature should be verified by the recipient of the data - both as a valid signature, and as being  
1527 the signature of the publisher.

1528 Additionally, various of the publication protocols, e.g. HTTP-over-TLS/SSL, offer means for  
1529 ensuring the authenticity of the publishing party and for protecting the metadata in transit.  
1530 [SAMLv2Meta] also defines prescriptive metadata caching directives, as well as guidance on

1531 handling HTTPS redirects, trust processing, server authentication, and related items.  
1532 For a more detailed discussion of SAML v2.0 metadata and its security considerations, please  
1533 see [SAMLv2Meta]. For a discussion of overall SAML v2.0 security considerations and specific  
1534 security-related design features, please refer to the SAML v2.0 specifications listed in the below  
1535 bibliography. The specifications containing security-specific information are explicitly listed.

## 1536 Interoperability considerations

1537 SAML v2.0 metadata explicitly supports identifying the protocols and versions supported by the  
1538 identified entities. For example, an identity provider entity can be denoted as supporting SAML  
1539 v2.0 [SAMLv2.0], SAML v1.1 [SAMLv1.1], Liberty ID-FF 1.2 [LAPFF], or even other protocols if  
1540 they are unambiguously identifiable via URI [RFC2396]. This protocol support information is  
1541 conveyed via the `protocolSupportEnumeration` attribute of metadata objects of the  
1542 **RoleDescriptorType**.

## 1543 Published specification

1544 [SAMLv2Meta] explicitly specifies use of the `application/samlmetadata+xml` MIME media  
1545 type.

## 1546 Applications which use this media type

1547 Potentially any application implementing SAML v2.0, as well as those applications implementing  
1548 specifications based on SAML, e.g. those available from the Liberty Alliance [LAP].

## 1549 Additional information

### 1550 Magic number(s)

1551 In general, the same as for `application/xml` [RFC3023]. In particular, the XML root element of  
1552 the returned object will have a namespace-qualified name with:

- 1553
- 1554 – a local name of: `EntityDescriptor`, or  
1555 `AffiliationDescriptor`, or  
1556 `EntitiesDescriptor`
  
  - 1558 – a namespace URI of: `urn:oasis:names:tc:SAML:2.0:metadata`  
1559 (the SAMLv2.0 metadata namespace)

### 1560 File extension(s)

1561 None

### 1562 Macintosh File Type Code(s)

1563 None

### 1564 Person & email address to contact for further information

1565 This registration is made on behalf of the OASIS Security Services Technical Committee (SSTC)  
1566 Please refer to the SSTC website for current information on committee chairperson(s) and their  
1567 contact addresses: <http://www.oasis-open.org/committees/security/>. Committee members should  
1568 submit comments and potential errata to the [securityservices@lists.oasis-open.org](mailto:securityservices@lists.oasis-open.org) list. Others  
1569 should submit them by filling out the web form located at [http://www.oasis-  
1570 open.org/committees/comments/form.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security).  
1571

1572 Additionally, the SAML developer community email distribution list, [saml-dev@lists.oasis-](mailto:saml-dev@lists.oasis-open.org)  
1573 [open.org](http://lists.oasis-open.org), may be employed to discuss usage of the `application/samlmetadata+xml` MIME  
1574 media type. The "saml-dev" mailing list is publicly archived here: [http://lists.oasis-](http://lists.oasis-open.org/archives/saml-dev/)  
1575 [open.org/archives/saml-dev/](http://lists.oasis-open.org/archives/saml-dev/). To post to the "saml-dev" mailing list, one must subscribe to it. To  
1576 subscribe, send a message with the single word "subscribe" in the message body, to: [saml-dev-](mailto:saml-dev-request@lists.oasis-open.org)  
1577 [request@lists.oasis-open.org](mailto:saml-dev-request@lists.oasis-open.org).

## 1578 Intended usage

1579 COMMON

## 1580 Author/Change controller

1581 The SAML specification sets are a work product of the OASIS Security Services Technical  
1582 Committee (SSTC). OASIS and the SSTC have change control over the SAML specification sets.

## 1583 Bibliography

- 1584 [LAP] “*Liberty Alliance Project*”. See <http://www.projectliberty.org/>
- 1585 [LAPFF] “*Liberty Alliance Project: Federation Framework*”. See  
1586 <http://www.projectliberty.org/resources/specifications.php#box1>
- 1587 [OASIS] “*Organization for the Advancement of Structured Information Systems*”.  
1588 See <http://www.oasis-open.org/>
- 1589 [RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifiers*  
1590 *(URI): Generic Syntax*. IETF RFC 2396, August 1998. Available at  
1591 <http://www.ietf.org/rfc/rfc2396.txt>
- 1592 [RFC3023] M. Murata, S. St.Laurent, D. Kohn, “*XML Media Types*”, IETF Request for  
1593 Comments 3023, January 2001. Available as [http://www.rfc-](http://www.rfc-editor.org/rfc/rfc3023.txt)  
1594 [editor.org/rfc/rfc3023.txt](http://www.rfc-editor.org/rfc/rfc3023.txt)
- 1595 [SAMLv1.1] OASIS Security Services Technical Committee, “*Security Assertion*  
1596 *Markup Language (SAML) Version 1.1 Specification Set*”. OASIS  
1597 Standard 200308, August 2003. Available as [http://www.oasis-](http://www.oasis-open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip)  
1598 [open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip](http://www.oasis-open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip)
- 1599 [SAMLv2.0] OASIS Security Services Technical Committee, “*Security Assertion*  
1600 *Markup Language (SAML) Version 2.0 Specification Set*”. OASIS  
1601 Standard, 15-Mar-2005. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip)  
1602 [open.org/security/saml/v2.0/saml-2.0-os.zip](http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip)
- 1603 [SAMLv2Bind] S. Cantor et al., “*Bindings for the OASIS Security Assertion Markup*  
1604 *Language (SAML) V2.0*”. OASIS, March 2005. Document ID `saml-`  
1605 `bindings-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf)  
1606 [open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf](http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf)
- 1607 [SAMLv2Core] S. Cantor et al., “*Assertions and Protocols for the OASIS Security*  
1608 *Assertion Markup Language (SAML) V2.0*”. OASIS, March 2005.  
1609 Document ID `saml-core-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf)  
1610 [open.org/security/saml/v2.0/saml-core-2.0-os.pdf](http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf)
- 1611 [SAMLv2Meta] S. Cantor et al., “*Metadata for the OASIS Security Assertion Markup*  
1612 *Language (SAML) V2.0*”. OASIS SSTC, August 2004. Document ID `saml-`  
1613 `metadata-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf)  
1614 [open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf](http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf)
- 1615 [SAMLv2Prof] S. Cantor et al., “*Profiles for the OASIS Security Assertion Markup*  
1616 *Language (SAML) V2.0*”. OASIS, March 2005. Document ID `saml-`  
1617 `profiles-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf)  
1618 [open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf](http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf)
- 1619 [SAMLv2Sec] F. Hirsch et al., “*Security and Privacy Considerations for the OASIS*







---

## 1630 Appendix B. Acknowledgments

1631 The editors would like to acknowledge the contributions of the OASIS Security Services Technical  
1632 Committee, whose voting members at the time of publication were:

- 1633 • Conor Cahill, AOL
- 1634 • John Hughes, Atos Origin
- 1635 • Hal Lockhart, BEA Systems
- 1636 • Mike Beach, Boeing
- 1637 • Rebekah Metz, Booz Allen Hamilton
- 1638 • Rick Randall, Booz Allen Hamilton
- 1639 • Ronald Jacobson, Computer Associates
- 1640 • Gavenraj Sodhi, Computer Associates
- 1641 • Thomas Wisniewski, Entrust
- 1642 • Carolina Canales-Valenzuela, Ericsson
- 1643 • Dana Kaufman, Forum Systems
- 1644 • Irving Reid, Hewlett-Packard
- 1645 • Guy Denton, IBM
- 1646 • Heather Hinton, IBM
- 1647 • Maryann Hondo, IBM
- 1648 • Michael McIntosh, IBM
- 1649 • Anthony Nadalin, IBM
- 1650 • Nick Ragouzis, Individual
- 1651 • Scott Cantor, Internet2
- 1652 • Bob Morgan, Internet2
- 1653 • Peter Davis, Neustar
- 1654 • Jeff Hodges, Neustar
- 1655 • Frederick Hirsch, Nokia
- 1656 • Senthil Sengodan, Nokia
- 1657 • Abbie Barbir, Nortel Networks
- 1658 • Scott Kiester, Novell
- 1659 • Cameron Morris, Novell
- 1660 • Paul Madsen, NTT
- 1661 • Steve Anderson, OpenNetwork
- 1662 • Ari Kermaier, Oracle
- 1663 • Vamsi Motukuru, Oracle
- 1664 • Darren Platt, Ping Identity
- 1665 • Prateek Mishra, Principal Identity
- 1666 • Jim Lien, RSA Security
- 1667 • John Linn, RSA Security
- 1668 • Rob Philpott, RSA Security
- 1669 • Dipak Chopra, SAP
- 1670 • Jahan Moreh, Sigaba
- 1671 • Bhavna Bhatnagar, Sun Microsystems
- 1672 • Eve Maler, Sun Microsystems
- 1673 • Ronald Monzillo, Sun Microsystems

1674 • Emily Xu, Sun Microsystems

1675 • Greg Whitehead, Trustgenix

1676

1677 The editors also would like to acknowledge the following former SSTC members for their contributions to  
1678 this or previous versions of the OASIS Security Assertions Markup Language Standard:

1679 • Stephen Farrell, Baltimore Technologies

1680 • David Orchard, BEA Systems

1681 • Krishna Sankar, Cisco Systems

1682 • Zahid Ahmed, CommerceOne

1683 • Tim Alsop, CyberSafe Limited

1684 • Carlisle Adams, Entrust

1685 • Tim Moses, Entrust

1686 • Nigel Edwards, Hewlett-Packard

1687 • Joe Pato, Hewlett-Packard

1688 • Bob Blakley, IBM

1689 • Marlena Erdos, IBM

1690 • Marc Chanliau, Netegrity

1691 • Chris McLaren, Netegrity

1692 • Lynne Rosenthal, NIST

1693 • Mark Skall, NIST

1694 • Charles Knouse, Oblix

1695 • Simon Godik, Overxeer

1696 • Charles Norwood, SAIC

1697 • Evan Prodromou, Securant

1698 • Robert Griffin, RSA Security (former editor)

1699 • Sai Allarvarpu, Sun Microsystems

1700 • Gary Ellison, Sun Microsystems

1701 • Chris Ferris, Sun Microsystems

1702 • Mike Myers, Traceroute Security

1703 • Phillip Hallam-Baker, VeriSign (former editor)

1704 • James Vanderbeek, Vodafone

1705 • Mark O'Neill, Vordel

1706 • Tony Palmer, Vordel

1707

1708 Finally, the editors wish to acknowledge the following people for their contributions of material used as  
1709 input to the OASIS Security Assertions Markup Language specifications:

1710 • Thomas Gross, IBM

1711 • Birgit Pfitzmann, IBM

1712

---

## Appendix C. Notices

1713 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
1714 might be claimed to pertain to the implementation or use of the technology described in this document or  
1715 the extent to which any license under such rights might or might not be available; neither does it represent  
1716 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to  
1717 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made  
1718 available for publication and any assurances of licenses to be made available, or the result of an attempt  
1719 made to obtain a general license or permission for the use of such proprietary rights by implementors or  
1720 users of this specification, can be obtained from the OASIS Executive Director.

1721 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or  
1722 other proprietary rights which may cover technology that may be required to implement this specification.  
1723 Please address the information to the OASIS Executive Director.

1724 **Copyright © OASIS Open 2005. All Rights Reserved.**

1725 This document and translations of it may be copied and furnished to others, and derivative works that  
1726 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and  
1727 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and  
1728 this paragraph are included on all such copies and derivative works. However, this document itself may  
1729 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as  
1730 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights  
1731 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it  
1732 into languages other than English.

1733 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
1734 or assigns.

1735 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
1736 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
1737 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR  
1738 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.