

---

# Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0 – Errata Composite

Working Draft, ~~1212 August 2006~~ February 2007

## Document identifier:

sstc-saml-metadata-errata-2.0-wd-034

## Location:

[http://www.oasis-open.org/committees/documents.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/documents.php?wg_abbrev=security)

## Editors:

Scott Cantor, Internet2  
Jahan Moreh, Sigaba  
Rob Philpott, RSA Security  
~~Jahan Moreh, Sigaba (errata document editor)~~  
Eve Maler, Sun Microsystems (errata ~~composite document~~ editor)

## Contributors to the Errata:

[Nick Ragouzis, Enosis Group](#)  
[Thomas Wisniewski, Entrust](#)  
[Greg Whitehead, HP](#)  
[Heather Hinton, IBM](#)  
[Connor P. Cahill, Intel](#)  
[Scott Cantor, Internet2](#)  
[Eric Tiffany, Liberty Alliance](#)  
[Tom Scavo, NCSA/University of Illinois](#)  
[Jeff Hodges, Neustar](#)  
[Ari Kermaier, Oracle](#)  
[Prateek Mishra, Oracle](#)  
[Brian Campbell, Ping Identity](#)  
[Jim Lien, RSA Security](#)  
[Rob Philpott, RSA Security](#)  
[Jahan Moreh, Sigaba](#)  
[Emily Xu, Sun Microsystems](#)  
[David Staggs, Veteran's Health Administration](#)

## SAML V2.0 Contributors:

Conor P. Cahill, AOL  
John Hughes, Atos Origin  
Hal Lockhart, BEA Systems  
Michael Beach, Boeing  
Rebekah Metz, Booz Allen Hamilton  
Rick Randall, Booz Allen Hamilton  
Thomas Wisniewski, Entrust  
Irving Reid, Hewlett-Packard  
Paula Austel, IBM  
Maryann Hondo, IBM  
Michael McIntosh, IBM  
Tony Nadalin, IBM  
Nick Ragouzis, Individual

48 Scott Cantor, Internet2  
49 RL 'Bob' Morgan, Internet2  
50 Peter C Davis, Neustar  
51 Jeff Hodges, Neustar  
52 Frederick Hirsch, Nokia  
53 John Kemp, Nokia  
54 Paul Madsen, NTT  
55 Steve Anderson, OpenNetwork  
56 Prateek Mishra, Principal Identity  
57 John Linn, RSA Security  
58 Rob Philpott, RSA Security  
59 Jahan Moreh, Sigaba  
60 Anne Anderson, Sun Microsystems  
61 Eve Maler, Sun Microsystems  
62 Ron Monzillo, Sun Microsystems  
63 Greg Whitehead, Trustgenix

64 **Abstract:**

65 SAML profiles require agreements between system entities regarding identifiers, binding support  
66 and endpoints, certificates and keys, and so forth. A metadata specification is useful for  
67 describing this information in a standardized way. The SAML V2.0 Metadata document defines an  
68 extensible metadata format for SAML system entities, organized by roles that reflect SAML  
69 profiles. Such roles include that of Identity Provider, Service Provider, Affiliation, Attribute  
70 Authority, Attribute Consumer, and Policy Decision Point. This document, known as an “errata  
71 composite”, combines corrections to reported errata with the original specification text. By design,  
72 the corrections are limited to clarifications of ambiguous or conflicting specification text. This  
73 document shows deletions from the original specification as struck-through text, and additions as  
74 **bluecolored** underlined text. The “[PE~~nn~~” designations embedded in the text refer to particular  
75 errata and their dispositions.

76 **Status:**

77 This errata composite document is a **working draft** based on the **original** OASIS Standard  
78 document that had been produced by the Security Services Technical Committee and approved  
79 by the OASIS membership on 1 March 2005. While the errata corrections appearing here are  
80 non-normative, they reflect ~~the consensus of the TC about how to interpret the specification and~~  
81 ~~are likely to be incorporated into any future standards-track revision of the SAML-~~  
82 ~~specifications-changes specified by the Approved Errata document (currently at Working Draft~~  
83 ~~revision 02), which is on an OASIS standardization track. In case of any discrepancy between this~~  
84 ~~document and the Approved Errata, the latter has precedence. See also the Errata Working~~  
85 ~~Document (currently at revision 39), which provides background on the changes specified here.~~

86 This document includes ~~errata~~ corrections ~~through revision 33 of the~~for errata ~~document,~~  
87 ~~including PE7, PE33, PE34, PE37, and PE41, and E62.~~

88 Committee members should submit comments and potential errata to the [security-](mailto:security-services@lists.oasis-open.org)  
89 [services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should submit them by following the instructions at  
90 [http://www.oasis-open.org/committees/comments/form.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security).

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

# Table of Contents

96	1 Introduction.....	5
97	1.1 Notation.....	5
98	2 Metadata for SAML V2.0.....	6
99	2.1 Namespaces .....	6
100	2.2 Common Types.....	7
101	2.2.1 Simple Type entityIDType.....	7
102	2.2.2 Complex Type EndpointType.....	7
103	2.2.3 Complex Type IndexedEndpointType.....	8
104	2.2.4 Complex Type localizedNameType.....	8
105	2.2.5 Complex Type localizedURIType.....	9
106	2.3 Root Elements.....	9
107	2.3.1 Element <EntitiesDescriptor>.....	9
108	2.3.2 Element <EntityDescriptor>.....	10
109	2.3.2.1 Element <Organization>.....	12
110	2.3.2.2 Element <ContactPerson>.....	13
111	2.3.2.3 Element <AdditionalMetadataLocation>.....	14
112	2.4 Role Descriptor Elements.....	14
113	2.4.1 Element <RoleDescriptor>.....	14
114	2.4.1.1 Element <KeyDescriptor>.....	15
115	2.4.2 Complex Type SSODescriptorType.....	16
116	2.4.3 Element <IDPSSODescriptor>.....	17
117	2.4.4 Element <SPSSODescriptor>.....	18
118	2.4.4.1 Element <AttributeConsumingService>.....	19
119	2.4.4.1.1 [E34]Element <RequestedAttribute>.....	20
120	2.4.5 Element <AuthnAuthorityDescriptor>.....	20
121	2.4.6 Element <PDPDescriptor>.....	21
122	2.4.7 Element <AttributeAuthorityDescriptor>.....	22
123	2.5 Element <AffiliationDescriptor>.....	23
124	2.6 Examples.....	24
125	3 Signature Processing.....	27
126	3.1 XML Signature Profile.....	27
127	3.1.1 Signing Formats and Algorithms.....	27
128	3.1.2 References.....	27
129	3.1.3 Canonicalization Method.....	27
130	3.1.4 Transforms.....	28
131	3.1.5 KeyInfo.....	28
132	4 Metadata Publication and Resolution.....	29
133	4.1 Publication and Resolution via Well-Known Location.....	29
134	4.1.1 Publication.....	29
135	4.1.2 Resolution.....	29
136	4.2 Publishing and Resolution via DNS.....	29
137	4.2.1 Publication.....	30
138	4.2.1.1 First Well Known Rule.....	30
139	4.2.1.2 The Order Field.....	30
140	4.2.1.3 The Preference Field.....	30
141	4.2.1.4 The Flag Field.....	31
142	4.2.1.5 The Service Field.....	31

143	4.2.1.6 The Regex and Replacement Fields.....	31
144	4.2.2 NAPTR Examples.....	32
145	4.2.2.1 Entity Metadata NAPTR Examples.....	32
146	4.2.2.2 Name Identifier Examples.....	32
147	4.2.3 Resolution.....	32
148	4.2.3.1 Parsing the Unique Identifier.....	32
149	4.2.3.2 Obtaining Metadata via the DNS.....	33
150	4.2.4 Metadata Location Caching.....	33
151	4.3 Post-Processing of Metadata.....	33
152	4.3.1 Metadata Instance Caching.....	33
153	4.3.2 Handling of HTTPS Redirects.....	33
154	4.3.3 Processing of XML Signatures and General Trust Processing.....	33
155	4.3.3.1 Processing Signed DNS Zones.....	34
156	4.3.3.2 Processing Signed Documents and Fragments.....	34
157	4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL.....	34
158	5 References.....	35
159	Appendix A.Registration of MIME media type application/samlmetadata+xml.....	37
160	Appendix B. Acknowledgments.....	41
161	Appendix C. Notices.....	43

---

# 1 Introduction

162

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

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

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

## 1.1 Notation

173

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

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

178

179 `Example code listings appear like this.`

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

181 Conventional XML namespace prefixes are used throughout this specification to stand for their respective  
182 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

183

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

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

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

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

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

### 2.1 Namespaces

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

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

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

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

```
209 <schema
210   targetNamespace="urn:oasis:names:tc:SAML:2.0:metadata"
211   xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
212   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
213   xmlns:xenc="http://www.w3.org/2001/04/xmenc#"
214   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
215   xmlns="http://www.w3.org/2001/XMLSchema"
216   elementFormDefault="unqualified"
217   attributeFormDefault="unqualified"
218   blockDefault="substitution"
219   version="2.0">
220   <import namespace="http://www.w3.org/2000/09/xmldsig#"
221     schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
222 20020212/xmldsig-core-schema.xsd"/>
223   <import namespace="http://www.w3.org/2001/04/xmenc#"
224     schemaLocation="http://www.w3.org/TR/2002/REC-xmenc-core-
225 20021210/xenc-schema.xsd"/>
226   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
227     schemaLocation="saml-schema-assertion-2.0.xsd"/>
228   <import namespace="http://www.w3.org/XML/1998/namespace"
229     schemaLocation="http://www.w3.org/2001/xml.xsd"/>
230   <annotation>
```

```

231     <documentation>
232         Document identifier: saml-schema-metadata-2.0
233         Location: http://docs.oasis-open.org/security/saml/v2.0/
234         Revision history:
235             V2.0 (March, 2005):
236             Schema for SAML metadata, first published in SAML 2.0.
237     </documentation>
238 </annotation>
239 ...
240 </schema>

```

## 241 2.2 Common Types

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

### 244 2.2.1 Simple Type `entityIDType`

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

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

```

251 <simpleType name="entityIDType">
252     <restriction base="anyURI">
253         <maxLength value="1024"/>
254     </restriction>
255 </simpleType>

```

### 256 2.2.2 Complex Type `EndpointType`

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

#### 260 `Binding` [Required]

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

#### 263 `Location` [Required]

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

#### 266 `ResponseLocation` [Optional]

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

269 The `ResponseLocation` attribute is used to enable different endpoints to be specified for receiving  
 270 request and response messages associated with a protocol or profile, not as a means of load-balancing or  
 271 redundancy (multiple elements of this type can be included for this purpose). When a role contains an  
 272 element of this type pertaining to a protocol or profile for which only a single type of message (request or  
 273 response) is applicable, then the `ResponseLocation` attribute is unused. [\[E41\]](#) If the  
 274 `ResponseLocation` attribute is omitted, any response messages associated with a protocol or profile  
 275 may be assumed to be handled at the URI indicated by the `Location` attribute.

276 In most contexts, elements of this type appear in unbounded sequences in the schema. This is to permit a  
277 protocol or profile to be offered by an entity at multiple endpoints, usually with different protocol bindings,  
278 allowing the metadata consumer to choose an appropriate endpoint for its needs. Multiple endpoints might  
279 also offer "client-side" load-balancing or failover, particular in the case of a synchronous protocol binding.

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

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

```
283 <complexType name="EndpointType">  
284   <sequence>  
285     <any namespace="##other" processContents="lax" minOccurs="0"  
286     maxOccurs="unbounded"/>  
287   </sequence>  
288   <attribute name="Binding" type="anyURI" use="required"/>  
289   <attribute name="Location" type="anyURI" use="required"/>  
290   <attribute name="ResponseLocation" type="anyURI" use="optional"/>  
291   <anyAttribute namespace="##other" processContents="lax"/>  
292 </complexType>
```

### 293 2.2.3 Complex Type IndexedEndpointType

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

297 **index** [Required]

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

302 **isDefault** [Optional]

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

305 In any such sequence of [\[E37\]#keindexed](#) endpoints ~~based on this type that share a common element~~  
306 ~~name and namespace (i.e. all instances of <md:AssertionConsumerService> within a role)~~, the  
307 default endpoint is the first such endpoint with the **isDefault** attribute set to *true*. If no such endpoints  
308 exist, the default endpoint is the first such endpoint without the **isDefault** attribute set to *false*. If no  
309 such endpoints exist, the default endpoint is the first element in the sequence.

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

```
311 <complexType name="IndexedEndpointType">  
312   <complexContent>  
313     <extension base="md:EndpointType">  
314       <attribute name="index" type="unsignedShort" use="required"/>  
315       <attribute name="isDefault" type="boolean" use="optional"/>  
316     </extension>  
317   </complexContent>  
318 </complexType>
```

### 319 2.2.4 Complex Type localizedNameType

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

```
322 <complexType name="localizedNameType">  
323   <simpleContent>  
324     <extension base="string">
```

```
325     <attribute ref="xml:lang" use="required"/>
326   </extension>
327 </simpleContent>
328 </complexType>
```

## 329 2.2.5 Complex Type localizedURIType

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

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

```
333 <complexType name="localizedURIType">
334   <simpleContent>
335     <extension base="anyURI">
336       <attribute ref="xml:lang" use="required"/>
337     </extension>
338   </simpleContent>
339 </complexType>
```

## 340 2.3 Root Elements

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

### 344 2.3.1 Element `<EntitiesDescriptor>`

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

348 ID [Optional]

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

350 validUntil [Optional]

351 Optional attribute indicates the expiration time of the metadata contained in the element and any  
352 contained elements.

353 cacheDuration [Optional]

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

356 Name [Optional]

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

358 `<ds:Signature>` [Optional]

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

361 `<Extensions>` [Optional]

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

365 <EntitiesDescriptor> or <EntityDescriptor> [One or More]  
366       Contains the metadata for one or more SAML entities, or a nested group of additional metadata.  
367 When used as the root element of a metadata instance, this element MUST contain either a `validUntil`  
368 or `cacheDuration` attribute. It is RECOMMENDED that only the root element of a metadata instance  
369 contain either attribute.

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

```
372 <element name="EntitiesDescriptor" type="md:EntitiesDescriptorType"/>  
373 <complexType name="EntitiesDescriptorType">  
374   <sequence>  
375     <element ref="ds:Signature" minOccurs="0"/>  
376     <element ref="md:Extensions" minOccurs="0"/>  
377     <choice minOccurs="1" maxOccurs="unbounded">  
378       <element ref="md:EntityDescriptor"/>  
379       <element ref="md:EntitiesDescriptor"/>  
380     </choice>  
381   </sequence>  
382   <attribute name="validUntil" type="dateTime" use="optional"/>  
383   <attribute name="cacheDuration" type="duration" use="optional"/>  
384   <attribute name="ID" type="ID" use="optional"/>  
385   <attribute name="Name" type="string" use="optional"/>  
386 </complexType>  
387 <element name="Extensions" type="md:ExtensionsType"/>  
388 <complexType final="#all" name="ExtensionsType">  
389   <sequence>  
390     <any namespace="##other" processContents="lax" maxOccurs="unbounded"/>  
391   </sequence>  
392 </complexType>
```

### 393 **2.3.2 Element <EntityDescriptor>**

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

- 398     • SSO Identity Provider
- 399     • SSO Service Provider
- 400     • Authentication Authority
- 401     • Attribute Authority
- 402     • Policy Decision Point
- 403     • Affiliation

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

405 `entityID` [Required]

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

408 `ID` [Optional]

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

410 `validUntil` [Optional]

411       Optional attribute indicates the expiration time of the metadata contained in the element and any

412 contained elements.

413 `cacheDuration` [Optional]

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

416 `<ds:Signature>` [Optional]

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

419 `<Extensions>` [Optional]

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

423 `<RoleDescriptor>`, `<IDPSSODescriptor>`, `<SPSSODescriptor>`,  
424 `<AuthnAuthorityDescriptor>`, `<AttributeAuthorityDescriptor>`, `<PDPDescriptor>` [One  
425 or More]

426 **OR**

427 `<AffiliationDescriptor>` [Required]

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

430 `<Organization>` [Optional]

431 Optional element identifying the organization responsible for the SAML entity described by the  
432 element.

433 `<ContactPerson>` [Zero or More]

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

435 `<AdditionalMetadataLocation>` [Zero or More]

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

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

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

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

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

```

450 <element name="EntityDescriptor" type="md:EntityDescriptorType"/>
451 <complexType name="EntityDescriptorType">
452   <sequence>
453     <element ref="ds:Signature" minOccurs="0"/>
454     <element ref="md:Extensions" minOccurs="0"/>
455     <choice>
456       <choice maxOccurs="unbounded">

```

```

457         <element ref="md:RoleDescriptor"/>
458         <element ref="md:IDPSSODescriptor"/>
459         <element ref="md:SPSSODescriptor"/>
460         <element ref="md:AuthnAuthorityDescriptor"/>
461         <element ref="md:AttributeAuthorityDescriptor"/>
462         <element ref="md:PDPDescriptor"/>
463     </choice>
464     <element ref="md:AffiliationDescriptor"/>
465 </choice>
466 <element ref="md:Organization" minOccurs="0"/>
467 <element ref="md:ContactPerson" minOccurs="0" maxOccurs="unbounded"/>
468 <element ref="md:AdditionalMetadataLocation" minOccurs="0"
469 maxOccurs="unbounded"/>
470 </sequence>
471 <attribute name="entityID" type="md:entityIDType" use="required"/>
472 <attribute name="validUntil" type="dateTime" use="optional"/>
473 <attribute name="cacheDuration" type="duration" use="optional"/>
474 <attribute name="ID" type="ID" use="optional"/>
475 <anyAttribute namespace="##other" processContents="lax"/>
476 </complexType>

```

### 477 2.3.2.1 Element <Organization>

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

482 <Extensions> [Optional]

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

486 <OrganizationName> [One or More]

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

488 <OrganizationDisplayName> [One or More]

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

490 <OrganizationURL> [One or More]

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

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

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

```

497 <element name="Organization" type="md:OrganizationType"/>
498 <complexType name="OrganizationType">
499     <sequence>
500         <element ref="md:Extensions" minOccurs="0"/>
501         <element ref="md:OrganizationName" maxOccurs="unbounded"/>
502         <element ref="md:OrganizationDisplayName" maxOccurs="unbounded"/>
503         <element ref="md:OrganizationURL" maxOccurs="unbounded"/>
504     </sequence>
505     <anyAttribute namespace="##other" processContents="lax"/>
506 </complexType>
507 <element name="OrganizationName" type="md:localizedNameType"/>
508 <element name="OrganizationDisplayName" type="md:localizedNameType"/>

```

509 `<element name="OrganizationURL" type="md:localizedURIType"/>`

### 510 2.3.2.2 Element <ContactPerson>

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

515 **contactType** [Required]

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

518 <Extensions> [Optional]

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

522 <Company> [Optional]

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

524 <GivenName> [Optional]

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

526 <SurName> [Optional]

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

528 <EmailAddress> [Zero or More]

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

531 <TelephoneNumber> [Zero or More]

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

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

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

```
536 <element name="ContactPerson" type="md:ContactType"/>
537 <complexType name="ContactType">
538   <sequence>
539     <element ref="md:Extensions" minOccurs="0"/>
540     <element ref="md:Company" minOccurs="0"/>
541     <element ref="md:GivenName" minOccurs="0"/>
542     <element ref="md:SurName" minOccurs="0"/>
543     <element ref="md:EmailAddress" minOccurs="0" maxOccurs="unbounded"/>
544     <element ref="md:TelephoneNumber" minOccurs="0" maxOccurs="unbounded"/>
545   </sequence>
546   <attribute name="contactType" type="md:ContactTypeType" use="required"/>
547   <anyAttribute namespace="##other" processContents="lax"/>
548 </complexType>
549 <element name="Company" type="string"/>
550 <element name="GivenName" type="string"/>
551 <element name="SurName" type="string"/>
552 <element name="EmailAddress" type="anyURI"/>
553 <element name="TelephoneNumber" type="string"/>
554 <simpleType name="ContactTypeType">
555   <restriction base="string">
```

```

556     <enumeration value="technical"/>
557     <enumeration value="support"/>
558     <enumeration value="administrative"/>
559     <enumeration value="billing"/>
560     <enumeration value="other"/>
561   </restriction>
562 </simpleType>

```

### 563 2.3.2.3 Element <AdditionalMetadataLocation>

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

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

```

571 <element name="AdditionalMetadataLocation"
572   type="md:AdditionalMetadataLocationType"/>
573 <complexType name="AdditionalMetadataLocationType">
574   <simpleContent>
575     <extension base="anyURI">
576       <attribute name="namespace" type="anyURI" use="required"/>
577     </extension>
578   </simpleContent>
579 </complexType>

```

## 580 2.4 Role Descriptor Elements

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

### 584 2.4.1 Element <RoleDescriptor>

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

589 ID [Optional]

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

591 validUntil [Optional]

592 Optional attribute indicates the expiration time of the metadata contained in the element and any  
593 contained elements.

594 cacheDuration [Optional]

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

597 protocolSupportEnumeration [Required]

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

602 ensure discrimination when necessary.

603 `errorURL` [Optional]

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

606 `<ds:Signature>` [Optional]

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

609 `<Extensions>` [Optional]

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

613 `<KeyDescriptor>` [Zero or More]

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

616 `<Organization>` [Optional]

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

619 `<ContactPerson>` [Zero or More]

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

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

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

```

625 <element name="RoleDescriptor" type="md:RoleDescriptorType"/>
626 <complexType name="RoleDescriptorType" abstract="true">
627   <sequence>
628     <element ref="ds:Signature" minOccurs="0"/>
629     <element ref="md:Extensions" minOccurs="0"/>
630     <element ref="md:KeyDescriptor" minOccurs="0" maxOccurs="unbounded"/>
631     <element ref="md:Organization" minOccurs="0"/>
632     <element ref="md:ContactPerson" minOccurs="0" maxOccurs="unbounded"/>
633   </sequence>
634   <attribute name="ID" type="ID" use="optional"/>
635   <attribute name="validUntil" type="dateTime" use="optional"/>
636   <attribute name="cacheDuration" type="duration" use="optional"/>
637   <attribute name="protocolSupportEnumeration" type="md:anyURIListType"
638 use="required"/>
639   <attribute name="errorURL" type="anyURI" use="optional"/>
640   <anyAttribute namespace="##other" processContents="lax"/>
641 </complexType>
642 <simpleType name="anyURIListType">
643   <list itemType="anyURI"/>
644 </simpleType>

```

#### 645 **2.4.1.1 Element `<KeyDescriptor>`**

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

649 use [Optional]

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

652 `<ds:KeyInfo>` [Required]

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

655 `<EncryptionMethod>` [Zero or More]

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

659 [E62]A use value of "signing" means that the contained key information is applicable to both signing  
660 and TLS/SSL operations performed by the entity when acting in the enclosing role.

661 A use value of "encryption" means that the contained key information is suitable for use in wrapping  
662 encryption keys for use by the entity when acting in the enclosing role.

663 If the use attribute is omitted, then the contained key information is applicable to both of the above uses.

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

```
666 <element name="KeyDescriptor" type="md:KeyDescriptorType"/>
667 <complexType name="KeyDescriptorType">
668   <sequence>
669     <element ref="ds:KeyInfo"/>
670     <element ref="md:EncryptionMethod" minOccurs="0"
671     maxOccurs="unbounded"/>
672   </sequence>
673   <attribute name="use" type="md:KeyTypes" use="optional"/>
674 </complexType>
675 <simpleType name="KeyTypes">
676   <restriction base="string">
677     <enumeration value="encryption"/>
678     <enumeration value="signing"/>
679   </restriction>
680 </simpleType>
681 <element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>
```

## 682 2.4.2 Complex Type **SSODescriptorType**

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

687 `<ArtifactResolutionService>` [Zero or More]

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

691 `<SingleLogoutService>` [Zero or More]

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

694 `<ManageNameIDService>` [Zero or More]

695 Zero or more elements of type **EndpointType** that describe endpoints that support the Name

696 Identifier Management profiles defined in [SAMLProf].

697 <NameIDFormat> [Zero or More]

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

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

```
702 <complexType name="SSODescriptorType" abstract="true">  
703   <complexContent>  
704     <extension base="md:RoleDescriptorType">  
705       <sequence>  
706         <element ref="md:ArtifactResolutionService" minOccurs="0"  
707 maxOccurs="unbounded"/>  
708         <element ref="md:SingleLogoutService" minOccurs="0"  
709 maxOccurs="unbounded"/>  
710         <element ref="md:ManageNameIDService" minOccurs="0"  
711 maxOccurs="unbounded"/>  
712         <element ref="md:NameIDFormat" minOccurs="0"  
713 maxOccurs="unbounded"/>  
714       </sequence>  
715     </extension>  
716   </complexContent>  
717 </complexType>  
718 <element name="ArtifactResolutionService" type="md:IndexedEndpointType"/>  
719 <element name="SingleLogoutService" type="md:EndpointType"/>  
720 <element name="ManageNameIDService" type="md:EndpointType"/>  
721 <element name="NameIDFormat" type="anyURI"/>
```

### 722 2.4.3 Element <IDPSSODescriptor>

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

726 WantAuthnRequestsSigned [Optional]

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

729 <SingleSignOnService> [One or More]

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

733 <NameIDMappingService> [Zero or More]

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

737 <AssertionIDRequestService> [Zero or More]

738 Zero or more elements of type **EndpointType** that describe endpoints that support the profile of  
739 the Assertion [\[E33\]Query/Request](#) protocol defined in [SAMLProf] or the special URI binding for  
740 assertion requests defined in [SAMLBind].

741 <AttributeProfile> [Zero or More]

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

744 <saml:Attribute> [Zero or More]

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

749 [E7]The WantAuthnRequestsSigned attribute is intended to indicate to service providers whether or  
750 not they can expect an unsigned <AuthnRequest> message to be accepted by the identity provider. The  
751 identity provider is not obligated to reject unsigned requests nor is a service provider obligated to sign its  
752 requests, although it might reasonably expect an unsigned request will be rejected. In some cases, a  
753 service provider may not even know which identity provider will ultimately receive and respond to its  
754 requests, so the use of this attribute in such a case cannot be strictly defined.

755 Furthermore, note that the specific method of signing that would be expected is binding dependent. The  
756 HTTP Redirect binding (see [SAMLBind]) requires that the signature be applied to the URL-encoded value  
757 rather than placed within the XML message, while other bindings generally permit the signature to be  
758 within the message in the usual fashion.

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

```
761 <element name="IDPSSODescriptor" type="md:IDPSSODescriptorType"/>
762 <complexType name="IDPSSODescriptorType">
763   <complexContent>
764     <extension base="md:SSODescriptorType">
765       <sequence>
766         <element ref="md:SingleSignOnService" maxOccurs="unbounded"/>
767         <element ref="md:NameIDMappingService" minOccurs="0"
768 maxOccurs="unbounded"/>
769         <element ref="md:AssertionIDRequestService" minOccurs="0"
770 maxOccurs="unbounded"/>
771         <element ref="md:AttributeProfile" minOccurs="0"
772 maxOccurs="unbounded"/>
773         <element ref="saml:Attribute" minOccurs="0"
774 maxOccurs="unbounded"/>
775       </sequence>
776       <attribute name="WantAuthnRequestsSigned" type="boolean"
777 use="optional"/>
778     </extension>
779   </complexContent>
780 </complexType>
781 <element name="SingleSignOnService" type="md:EndpointType"/>
782 <element name="NameIDMappingService" type="md:EndpointType"/>
783 <element name="AssertionIDRequestService" type="md:EndpointType"/>
784 <element name="AttributeProfile" type="anyURI"/>
```

## 785 2.4.4 Element <SPSSODescriptor>

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

789 AuthnRequestsSigned [Optional]

790 Optional attribute that indicates whether the <samlp:AuthnRequest> messages sent by this  
791 service provider will be signed. If omitted, the value is assumed to be false. [E7]A value of  
792 false (or omission of this attribute) does not imply that the service provider will never sign its  
793 requests or that a signed request should be considered an error. However, an identity provider  
794 that receives an unsigned <samlp:AuthnRequest> message from a service provider whose  
795 metadata contains this attribute with a value of true MUST return a SAML error response and  
796 MUST NOT fulfill the request.

797 WantAssertionsSigned [Optional]

798 Optional attribute that indicates a requirement for the <saml:Assertion> elements received by  
799 this service provider to be signed. If omitted, the value is assumed to be false. This requirement  
800 is in addition to any requirement for signing derived from the use of a particular profile/binding  
801 combination. [E7]Note that an enclosing signature at the SAML binding or protocol layer does not  
802 suffice to meet this requirement, for example signing a <samlp:Response> containing the  
803 assertion(s) or a TLS connection.

804 <AssertionConsumerService> [One or More]

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

808 <AttributeConsumingService> [Zero or More]

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

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

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

```
815 <element name="SPSSODescriptor" type="md:SPSSODescriptorType"/>
816 <complexType name="SPSSODescriptorType">
817   <complexContent>
818     <extension base="md:SSODescriptorType">
819       <sequence>
820         <element ref="md:AssertionConsumerService"
821 maxOccurs="unbounded"/>
822         <element ref="md:AttributeConsumingService" minOccurs="0"
823 maxOccurs="unbounded"/>
824       </sequence>
825       <attribute name="AuthnRequestsSigned" type="boolean"
826 use="optional"/>
827       <attribute name="WantAssertionsSigned" type="boolean"
828 use="optional"/>
829     </extension>
830   </complexContent>
831 </complexType>
832 <element name="AssertionConsumerService" type="md:IndexedEndpointType"/>
```

#### 833 2.4.4.1 Element <AttributeConsumingService>

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

837 index [Required]

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

840 isDefault [Optional]

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

843 <ServiceName> [One or More]

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

845 <ServiceDescription> [Zero or More]  
846 Zero or more language-qualified strings that describe the service.

847 <RequestedAttribute> [One or More]  
848 One or more elements specifying attributes required or desired by this service.

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

```
851 <element name="AttributeConsumingService"  
852 type="md:AttributeConsumingServiceType"/>  
853 <complexType name="AttributeConsumingServiceType">  
854 <sequence>  
855 <element ref="md:ServiceName" maxOccurs="unbounded"/>  
856 <element ref="md:ServiceDescription" minOccurs="0"  
857 maxOccurs="unbounded"/>  
858 <element ref="md:RequestedAttribute" maxOccurs="unbounded"/>  
859 </sequence>  
860 <attribute name="index" type="unsignedShort" use="required"/>  
861 <attribute name="isDefault" type="boolean" use="optional"/>  
862 </complexType>  
863 <element name="ServiceName" type="md:localizedNameType"/>  
864 <element name="ServiceDescription" type="md:localizedNameType"/>
```

#### 865 | 2.4.4.1.1 [E34] Element <RequestedAttribute>

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

869 **isRequired** [Optional]  
870 Optional XML attribute indicates if the service requires the corresponding SAML attribute in order  
871 to function at all (as opposed to merely finding an attribute useful or desirable).

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

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

```
876 <element name="RequestedAttribute" type="md:RequestedAttributeType"/>  
877 <complexType name="RequestedAttributeType">  
878 <complexContent>  
879 <extension base="saml:AttributeType">  
880 <attribute name="isRequired" type="boolean" use="optional"/>  
881 </extension>  
882 </complexContent>  
883 </complexType>
```

#### 884 | 2.4.5 Element <AuthnAuthorityDescriptor>

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

888 <AuthnQueryService> [One or More]  
889 One or more elements of type **EndpointType** that describe endpoints that support the profile of  
890 the Authentication Query protocol defined in [SAMLProf]. All authentication authorities support at  
891 least one such endpoint, by definition.

892 <AssertionIDRequestService> [Zero or More]  
893 Zero or more elements of type **EndpointType** that describe endpoints that support the profile of  
894 the Assertion [\[E33\]Query/Request](#) protocol defined in [SAMLProf] or the special URI binding for  
895 assertion requests defined in [SAMLBind].

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

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

```
901 <element name="AuthnAuthorityDescriptor"  
902 type="md:AuthnAuthorityDescriptorType"/>  
903 <complexType name="AuthnAuthorityDescriptorType">  
904 <complexContent>  
905 <extension base="md:RoleDescriptorType">  
906 <sequence>  
907 <element ref="md:AuthnQueryService" maxOccurs="unbounded"/>  
908 <element ref="md:AssertionIDRequestService" minOccurs="0"  
909 maxOccurs="unbounded"/>  
910 <element ref="md:NameIDFormat" minOccurs="0"  
911 maxOccurs="unbounded"/>  
912 </sequence>  
913 </extension>  
914 </complexContent>  
915 </complexType>  
916 <element name="AuthnQueryService" type="md:EndpointType"/>
```

## 917 2.4.6 Element <PDPDescriptor>

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

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

925 <AssertionIDRequestService> [Zero or More]  
926 Zero or more elements of type **EndpointType** that describe endpoints that support the profile of  
927 the Assertion [\[E33\]Query/Request](#) protocol defined in [SAMLProf] or the special URI binding for  
928 assertion requests defined in [SAMLBind].

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

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

```
934 <element name="PDPDescriptor" type="md:PDPDescriptorType"/>  
935 <complexType name="PDPDescriptorType">  
936 <complexContent>  
937 <extension base="md:RoleDescriptorType">  
938 <sequence>  
939 <element ref="md:AuthzService" maxOccurs="unbounded"/>  
940 <element ref="md:AssertionIDRequestService" minOccurs="0"  
941 maxOccurs="unbounded"/>
```

```

942         <element ref="md:NameIDFormat" minOccurs="0"
943 maxOccurs="unbounded"/>
944     </sequence>
945 </extension>
946 </complexContent>
947 </complexType>
948 <element name="AuthzService" type="md:EndpointType"/>

```

## 949 2.4.7 Element <AttributeAuthorityDescriptor>

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

954 <AttributeService> [One or More]

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

958 <AssertionIDRequestService> [Zero or More]

959 Zero or more elements of type **EndpointType** that describe endpoints that support the profile of  
960 the Assertion [\[E33\]Query/Request](#) protocol defined in [SAMLProf] or the special URI binding for  
961 assertion requests defined in [SAMLBind].

962 <NameIDFormat> [Zero or More]

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

965 <AttributeProfile> [Zero or More]

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

968 <saml:Attribute> [Zero or More]

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

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

```

974 <element name="AttributeAuthorityDescriptor"
975 type="md:AttributeAuthorityDescriptorType"/>
976 <complexType name="AttributeAuthorityDescriptorType">
977     <complexContent>
978         <extension base="md:RoleDescriptorType">
979             <sequence>
980                 <element ref="md:AttributeService" maxOccurs="unbounded"/>
981                 <element ref="md:AssertionIDRequestService" minOccurs="0"
982 maxOccurs="unbounded"/>
983                 <element ref="md:NameIDFormat" minOccurs="0"
984 maxOccurs="unbounded"/>
985                 <element ref="md:AttributeProfile" minOccurs="0"
986 maxOccurs="unbounded"/>
987                 <element ref="saml:Attribute" minOccurs="0"
988 maxOccurs="unbounded"/>
989             </sequence>
990         </extension>
991     </complexContent>
992 </complexType>

```

```
993 <element name="AttributeService" type="md:EndpointType"/>
```

## 994 2.5 Element <AffiliationDescriptor>

995 The <AffiliationDescriptor> element is an alternative to the sequence of role descriptors  
996 described in Section 2.4 that is used when an <EntityDescriptor> describes an affiliation of SAML  
997 entities (typically service providers) rather than a single entity. The <AffiliationDescriptor>  
998 element provides a summary of the individual entities that make up the affiliation along with general  
999 information about the affiliation itself. Its **AffiliationDescriptorType** complex type contains the following  
1000 elements and attributes:

1001 affiliationOwnerID [Required]

1002 Specifies the unique identifier of the entity responsible for the affiliation. The owner is NOT  
1003 presumed to be a member of the affiliation; if it is a member, its identifier MUST also appear in an  
1004 <AffiliateMember> element.

1005 ID [Optional]

1006 A document-unique identifier for the element, typically used as a reference point when signing.

1007 validUntil [Optional]

1008 Optional attribute indicates the expiration time of the metadata contained in the element and any  
1009 contained elements.

1010 cacheDuration [Optional]

1011 Optional attribute indicates the maximum length of time a consumer should cache the metadata  
1012 contained in the element and any contained elements.

1013 <ds:Signature> [Optional]

1014 An XML signature that authenticates the containing element and its contents, as described in  
1015 Section 3.

1016 <Extensions> [Optional]

1017 This contains optional metadata extensions that are agreed upon between a metadata publisher  
1018 and consumer. Extension elements MUST be namespace-qualified by a non-SAML-defined  
1019 namespace.

1020 <AffiliateMember> [One or More]

1021 One or more elements enumerating the members of the affiliation by specifying each member's  
1022 unique identifier. See also Section 8.3.6 of [SAMLCore].

1023 <KeyDescriptor> [Zero or More]

1024 Optional sequence of elements that provides information about the cryptographic keys that the  
1025 affiliation uses as a whole, as distinct from keys used by individual members of the affiliation,  
1026 which are published in the metadata for those entities.

1027 Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

1028 The following schema fragment defines the <AffiliationDescriptor> element and its  
1029 **AffiliationDescriptorType** complex type:

```
1030 <element name="AffiliationDescriptor" type="md:AffiliationDescriptorType"/>  
1031 <complexType name="AffiliationDescriptorType">  
1032   <sequence>  
1033     <element ref="ds:Signature" minOccurs="0"/>  
1034     <element ref="md:Extensions" minOccurs="0"/>  
1035     <element ref="md:AffiliateMember" maxOccurs="unbounded"/>  
1036     <element ref="md:KeyDescriptor" minOccurs="0" maxOccurs="unbounded"/>
```

```

1037 </sequence>
1038 <attribute name="affiliationOwnerID" type="md:entityIDType"
1039 use="required"/>
1040 <attribute name="validUntil" type="dateTime" use="optional"/>
1041 <attribute name="cacheDuration" type="duration" use="optional"/>
1042 <attribute name="ID" type="ID" use="optional"/>
1043 <anyAttribute namespace="##other" processContents="lax"/>
1044 </complexType>
1045 <element name="AffiliateMember" type="md:entityIDType"/>

```

## 1046 2.6 Examples

1047 The following is an example of metadata for a SAML system entity acting as an identity provider and an  
1048 attribute authority. A signature is shown as a placeholder, without the actual content.

```

1049
1050 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1051 xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1052 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1053 entityID="https://IdentityProvider.com/SAML"
1054 <ds:Signature>...</ds:Signature>
1055 <IDPSSODescriptor WantAuthnRequestsSigned="true"
1056 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1057 <KeyDescriptor use="signing">
1058 <ds:KeyInfo>
1059 <ds:KeyName>IdentityProvider.com SSO Key</ds:KeyName>
1060 </ds:KeyInfo>
1061 </KeyDescriptor>
1062 <ArtifactResolutionService isDefault="true" index="0"
1063 Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1064 Location="https://IdentityProvider.com/SAML/Artifact"/>
1065 <SingleLogoutService
1066 Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1067 Location="https://IdentityProvider.com/SAML/SLO/SOAP"/>
1068 <SingleLogoutService
1069 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1070 Location="https://IdentityProvider.com/SAML/SLO/Browser"
1071 ResponseLocation="https://IdentityProvider.com/SAML/SLO/Response"/>
1072 <NameIDFormat>
1073 urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1074 </NameIDFormat>
1075 <NameIDFormat>
1076 urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1077 </NameIDFormat>
1078 <NameIDFormat>
1079 urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1080 </NameIDFormat>
1081 <SingleSignOnService
1082 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1083 Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1084 <SingleSignOnService
1085 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1086 Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1087 <saml:Attribute
1088 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1089 Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1090 FriendlyName="eduPersonPrincipalName">
1091 </saml:Attribute>
1092 <saml:Attribute
1093 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1094 Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1095 FriendlyName="eduPersonAffiliation">
1096 <saml:AttributeValue>member</saml:AttributeValue>
1097 <saml:AttributeValue>student</saml:AttributeValue>
1098 <saml:AttributeValue>faculty</saml:AttributeValue>
1099 <saml:AttributeValue>employee</saml:AttributeValue>
1100 <saml:AttributeValue>staff</saml:AttributeValue>

```

```

1101     </saml:Attribute>
1102 </IDPSSODescriptor>
1103 <AttributeAuthorityDescriptor
1104   protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1105   <KeyDescriptor use="signing">
1106     <ds:KeyInfo>
1107       <ds:KeyName>IdentityProvider.com AA Key</ds:KeyName>
1108     </ds:KeyInfo>
1109   </KeyDescriptor>
1110   <AttributeService
1111     Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1112     Location="https://IdentityProvider.com/SAML/AA/SOAP"/>
1113   <AssertionIDRequestService
1114     Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
1115     Location="https://IdentityProvider.com/SAML/AA/URI"/>
1116   <NameIDFormat>
1117     urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1118   </NameIDFormat>
1119   <NameIDFormat>
1120     urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1121   </NameIDFormat>
1122   <NameIDFormat>
1123     urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1124   </NameIDFormat>
1125   <saml:Attribute
1126     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1127     Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1128     FriendlyName="eduPersonPrincipalName">
1129   </saml:Attribute>
1130   <saml:Attribute
1131     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1132     Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1133     FriendlyName="eduPersonAffiliation">
1134     <saml:AttributeValue>member</saml:AttributeValue>
1135     <saml:AttributeValue>student</saml:AttributeValue>
1136     <saml:AttributeValue>faculty</saml:AttributeValue>
1137     <saml:AttributeValue>employee</saml:AttributeValue>
1138     <saml:AttributeValue>staff</saml:AttributeValue>
1139   </saml:Attribute>
1140   </AttributeAuthorityDescriptor>
1141   <Organization>
1142     <OrganizationName xml:lang="en">Identity Providers R
1143 US</OrganizationName>
1144     <OrganizationDisplayName xml:lang="en">
1145 Identity Providers R US, a Division of Lerxst Corp.
1146 </OrganizationDisplayName>
1147     <OrganizationURL
1148 xml:lang="en">https://IdentityProvider.com</OrganizationURL>
1149   </Organization>
1150 </EntityDescriptor>
1151

```

1152 The following is an example of metadata for a SAML system entity acting as a service provider. A  
1153 signature is shown as a placeholder, without the actual content. For illustrative purposes, the service is  
1154 one that does not require users to uniquely identify themselves, but rather authorizes access on the basis  
1155 of a role-like attribute.

```

1157 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1158   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1159   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1160   entityID="https://ServiceProvider.com/SAML">
1161   <ds:Signature>...</ds:Signature>
1162   <SPSSODescriptor AuthnRequestsSigned="true"
1163     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1164     <KeyDescriptor use="signing">
1165       <ds:KeyInfo>
1166         <ds:KeyName>ServiceProvider.com SSO Key</ds:KeyName>
1167       </ds:KeyInfo>

```

```

1168     </KeyDescriptor>
1169     <KeyDescriptor use="encryption">
1170         <ds:KeyInfo>
1171             <ds:KeyName>ServiceProvider.com Encrypt Key</ds:KeyName>
1172         </ds:KeyInfo>
1173         <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
1174 1_5"/>
1175     </KeyDescriptor>
1176     <SingleLogoutService
1177         Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1178         Location="https://ServiceProvider.com/SAML/SLO/SOAP"/>
1179     <SingleLogoutService
1180         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1181         Location="https://ServiceProvider.com/SAML/SLO/Browser"
1182         ResponseLocation="https://ServiceProvider.com/SAML/SLO/Response"/>
1183     <NameIDFormat>
1184         urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1185     </NameIDFormat>
1186     <AssertionConsumerService isDefault="true" index="0"
1187         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
1188         Location="https://ServiceProvider.com/SAML/SSO/Artifact"/>
1189     <AssertionConsumerService index="1"
1190         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1191         Location="https://ServiceProvider.com/SAML/SSO/POST"/>
1192     <AttributeConsumingService index="0">
1193         <ServiceName xml:lang="en">Academic Journals R US</ServiceName>
1194         <RequestedAttribute
1195             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1196             Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7"
1197             FriendlyName="eduPersonEntitlement">
1198             <saml:AttributeValue>
1199                 https://ServiceProvider.com/entitlements/123456789
1200             </saml:AttributeValue>
1201         </RequestedAttribute>
1202     </AttributeConsumingService>
1203 </SPSSODescriptor>
1204 <Organization>
1205     <OrganizationName xml:lang="en">Academic Journals R
1206 US</OrganizationName>
1207     <OrganizationDisplayName xml:lang="en">
1208         Academic Journals R US, a Division of Dirk Corp.
1209     </OrganizationDisplayName>
1210     <OrganizationURL
1211 xml:lang="en">https://ServiceProvider.com</OrganizationURL>
1212 </Organization>
1213 </EntityDescriptor>

```

---

## 1214 3 Signature Processing

1215 Various elements in a metadata instance can be digitally signed (as indicated by the element's inclusion of  
1216 a `<ds:Signature>` element), with the following benefits:

- 1217 • Metadata integrity
- 1218 • Authentication of the metadata by a trusted signer

1219 A digital signature is not always required, for example if the relying party obtains the information directly  
1220 from the publishing entity directly (with no intermediaries) through a secure channel, with the entity having  
1221 authenticated to the relying party by some means other than a digital signature.

1222 Many different techniques are available for "direct" authentication and secure channel establishment  
1223 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,  
1224 the applicable security requirements depend on the communicating applications.

1225 Additionally, elements can inherit signatures on enclosing parent elements that are themselves signed.

1226 In the absence of such context, it is RECOMMENDED that at least the root element of a metadata  
1227 instance be signed.

### 1228 3.1 XML Signature Profile

1229 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility  
1230 and many choices. This section details the constraints on these facilities so that metadata processors do  
1231 not have to deal with the full generality of XML Signature processing. This usage makes specific use of  
1232 the **xs:ID**-typed attributes optionally present on the elements to which signatures can apply. These  
1233 attributes are collectively referred to in this section as the identifier attributes.

#### 1234 3.1.1 Signing Formats and Algorithms

1235 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and  
1236 detached.

1237 SAML metadata MUST use enveloped signatures when signing the elements defined in this specification.  
1238 SAML processors SHOULD support the use of RSA signing and verification for public key operations in  
1239 accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

#### 1240 3.1.2 References

1241 Signed metadata elements MUST supply a value for the identifier attribute on the signed element. The  
1242 element may or may not be the root element of the actual XML document containing the signed metadata  
1243 element.

1244 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute  
1245 value of the metadata element being signed. For example, if the identifier attribute value is "foo", then the  
1246 URI attribute in the `<ds:Reference>` element MUST be "#foo".

1247 As a consequence, a metadata element's signature MUST apply to the content of the signed element and  
1248 any child elements it contains.

#### 1249 3.1.3 Canonicalization Method

1250 SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the

1251 <ds:CanonicalizationMethod> element of <ds:SignedInfo>, and as a <ds:Transform>  
1252 algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML metadata  
1253 embedded in an XML context can be verified independent of that context.

### 1254 **3.1.4 Transforms**

1255 Signatures in SAML metadata SHOULD NOT contain transforms other than the enveloped signature  
1256 transform (with the identifier <http://www.w3.org/2000/09/xmlsig#enveloped-signature>) or the exclusive  
1257 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or  
1258 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

1259 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do  
1260 not, verifiers MUST ensure that no content of the signed metadata element is excluded from the  
1261 signature. This can be accomplished by establishing out-of-band agreement as to what transforms are  
1262 acceptable, or by applying the transforms manually to the content and reverifying the result as consisting  
1263 of the same SAML metadata.

### 1264 **3.1.5 KeyInfo**

1265 XML Signature [XMLSig] defines usage of the <ds:KeyInfo> element. SAML does not require the  
1266 use of <ds:KeyInfo> nor does it impose any restrictions on its use. Therefore, <ds:KeyInfo> MAY  
1267 be absent.

1268

## 4 Metadata Publication and Resolution

1269 Two mechanisms are provided for an entity to publish (and for a consumer to resolve the location of)  
1270 metadata documents: via a "well-known-location" by directly dereferencing the entity's unique identifier (a  
1271 URI variously referred to as an *entityID* or *providerID*), or indirectly by publishing the location of metadata  
1272 in the DNS. Other out-of-band mechanisms are of course also permitted. A consumer that supports both  
1273 approaches defined in this document MUST attempt resolution via DNS before using the "well-known-  
1274 location" mechanism.

1275 When retrieval requires network transport of the document, the transport SHOULD be protected with  
1276 mechanisms providing server authentication and integrity protection. For example, HTTP-based resolution  
1277 SHOULD be protected with TLS/SSL [RFC 2246] as amended by [RFC3546].

1278 Various mechanisms are described in this section to aid in establishing trust in the accuracy and  
1279 legitimacy of metadata, including use of XML signatures, SSL/TLS server authentication, and DNS  
1280 signatures. Regardless of the mechanism(s) used, relying parties SHOULD have some means by which to  
1281 establish trust in metadata information before relying on it.

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

1282 The following sections describe publication and resolution of metadata by means of a well-known location.

#### 4.1.1 Publication

1285 Entities MAY publish their metadata documents at a well known location by placing the document at the  
1286 location denoted by its unique identifier, which MUST be in the form of a URL (rather than a URN). See  
1287 Section 8.3.6 of [SAMLCore] for more information about such identifiers. It is STRONGLY  
1288 RECOMMENDED that `https` URLs be used for this purpose. An indirection mechanism supported by the  
1289 URL scheme (such as an HTTP 1.1 302 redirect) MAY be used if the document is not placed directly at  
1290 the location. If the publishing protocol permits MIME-based identification of content types, the content type  
1291 of the metadata instance MUST be `application/samlmetadata+xml`.

1292 The XML document provided at the well-known location MUST describe the metadata only for the entity  
1293 represented by the unique identifier (that is, the root element MUST be an `<EntityDescriptor>` with  
1294 an `entityID` matching the location). If other entities need to be described, the  
1295 `<AdditionalMetadataLocation>` element MUST be used. Thus the `<EntitiesDescriptor>`  
1296 element MUST NOT be used in documents published using this mechanism, since a group of entities are  
1297 not defined by such an identifier.

#### 4.1.2 Resolution

1299 If an entity's unique identifier is a URL, metadata consumers MAY attempt to resolve an entity's unique  
1300 identifier directly, in a scheme-specific manner, by dereferencing the identifier.

### 4.2 Publishing and Resolution via DNS

1302 To improve the accessibility of metadata documents and provide additional indirection between an entity's  
1303 unique identifier and the location of metadata, entities MAY publish their metadata document locations in a  
1304 zone of their corresponding DNS [RFC1034]. The entity's unique identifier (a URI) is used as the input to  
1305 the process. Since URIs are flexible identifiers, location publication methods and the resolution process  
1306 are determined by the URI's scheme and fully-qualified name. URI locations for metadata are

1307 subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in [RFC2915]  
1308 and [RFC3403].

1309 It is RECOMMENDED that entities publish their resource records in signed zone files using [RFC2535]  
1310 such that relying parties may establish the validity of the published location and authority of the zone, and  
1311 integrity of the DNS response. If DNS zone signatures are present, relying parties MUST properly validate  
1312 the signature.

## 1313 **4.2.1 Publication**

1314 This specification makes use of the NAPTR resource record described in [RFC2915] and [RFC3403].  
1315 Familiarity with these documents is encouraged.

1316 Dynamic Delegation Discovery System (DDDS) [RFC3401] is a general purpose system for the retrieval of  
1317 information based on an application-specific input string and the application of well known rules to  
1318 transform that string until a terminal condition is reached requiring a look-up into an application-specific  
1319 defined database or resolution of a URL based on the rules defined by the application. DDDS defines a  
1320 specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS  
1321 necessary to apply DDDS rules.

1322 Entities MAY publish separate URLs when multiple metadata documents need to be distributed, or when  
1323 different metadata documents are required due to multiple trust relationships that require separate keying  
1324 material, or when service interfaces require separate metadata declarations. This may be accomplished  
1325 through the use of the optional `<AdditionalMetadataLocation>` element, or through the regexp  
1326 facility and multiple service definition fields in the NAPTR resource record itself.

1327 If the publishing protocol permits MIME-based identification of content types, the content type of the  
1328 metadata instance MUST be `application/samlmetadata+xml`.

1329 If the entity's unique identifier is a URN, publication of the corresponding metadata location proceeds as  
1330 specified in [RFC3404]. Otherwise, the resolution of the metadata location proceeds as specified below.

1331 The following is the application-specific profile of DDDS for SAML metadata resolution.

### 1332 **4.2.1.1 First Well Known Rule**

1333 The "first well-known-rule" for processing SAML metadata resolution is to parse the entity's unique  
1334 identifier and extract the fully-qualified domain name (subexpression 3) as described in Section 4.2.3.1.

### 1335 **4.2.1.2 The Order Field**

1336 The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY  
1337 provide multiple NAPTR resource records which MUST be processed by the resolver application in the  
1338 order indicated by this field.

### 1339 **4.2.1.3 The Preference Field**

1340 For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving  
1341 application. The resolving application MAY ignore this order, in cases where the service field value does  
1342 not meet the resolver's requirements (e.g.: the resource record returns a protocol the application does not  
1343 support).

#### 1344 4.2.1.4 The Flag Field

1345 SAML metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying  
1346 additional resource records are to be processed). The "U" flag indicates that the output of the rule is a  
1347 URI.

#### 1348 4.2.1.5 The Service Field

1349 The SAML-specific service field, as described in the following BNF, declares the modes by which instance  
1350 document(s) shall be made available:

```
1351 servicefield = 1("PID2U" / "NID2U") "+" proto [*(":" class) *(":" servicetype)]  
1352 proto = 1("https" / "uddi")  
1353 class = 1[ "entity" / "entitygroup" ]  
1354 servicetype = 1(si / "spssso" / "idpssso" / "authn" / "authnauth" / "pdp" / "attrauth" /  
1355 alphanum )  
1356 si = "si" [ ":" alphanum ] [ ":" endpoint ]  
1357 alphanum = 1*32 (ALPHA / DIGIT)
```

1358 where:

- 1359 • servicefield PID2U resolves an entity's unique identifier to metadata URL.
- 1360 • servicefield NID2U resolves a principal's <NameID> into a metadata URL.
- 1361 • proto describes the retrieval protocol (`https` or `uddi`). In the case of UDDI, the URL will be an  
1362 http(s) URL referencing a WSDL document.
- 1363 • class identifies whether the referenced metadata document describes a single entity, or multiple.  
1364 In the latter case, the referenced document MUST contain the entity defined by the original unique  
1365 identifier as a member of a group of entities within the document itself such as an  
1366 <AffiliationDescriptor> or <EntitiesDescriptor>.
- 1367 • servicetype allows an entity to publish metadata for distinct roles and services as separate  
1368 documents. Resolvers who encounter multiple servicetype declarations will dereference the  
1369 appropriate URI, depending on which service is required for an operation (e.g.: an entity operating  
1370 both as an identity provider and a service provider can publish metadata for each role at different  
1371 locations). The authn service type represents a <SingleSignOnService> endpoint.
- 1372 • si (with optional endpoint component) allows the publisher to either directly publish the metadata  
1373 for a service instance, or by articulating a SOAP endpoint (using endpoint).

1374 For example:

- 1375 • PID2U+https:entity - represents the entity's complete metadata document available via the  
1376 https protocol
- 1377 • PID2U+uddi:entity:si:foo - represents the WSDL document location that describes a service  
1378 instance "foo"
- 1379 • PID2U+https:entitygroup:idpssso - represents the metadata for a group of entities acting as  
1380 SSO identity providers, of which the original entity is a member.
- 1381 • NID2U+https:idp - represents the metadata for the SSO identity provider of a principal

#### 1382 4.2.1.6 The Regex and Replacement Fields

1383 The expected output after processing the input string through the regex MUST be a valid https URL or  
1384 UDDI node (WSDL document) address.

1385 **4.2.2 NAPTR Examples**

1386 **4.2.2.1 Entity Metadata NAPTR Examples**

1387 Entities publish metadata URLs in the following manner:

```

1388 $ORIGIN provider.biz
1389
1390 ;; order pref f service regexp or replacement
1391
1392 IN NAPTR 100 10 "U" PID2U+https:entity
1393     "!^.*$!https://host.provider.biz/some/directory/trust.xml!" ""
1394 IN NAPTR 110 10 "U" PID2U+https: entity:trust
1395     "!^.*$!https://foo.provider.biz:1443/mdtrust.xml!" ""
1396 IN NAPTR 125 10 "U" PID2U+https:"
1397 IN NAPTR 110 10 "U" PID2U+uddi:entity
1398     "!^.*$!https://this.uddi.node.provider.biz/libmd.wsdl" ""

```

1399 **4.2.2.2 Name Identifier Examples**

1400 A principal's employer `example.int` operates an identity provider which may be used by an office supply  
1401 company to authenticate authorized buyers. The supplier takes a users' email address  
1402 `buyer@example.int` as input to the resolution process, and parses the email address to extract the  
1403 FQDN (`example.int`). The employer publishes the following NAPTR record in the `example.int` DNS:

```

1404 $ORIGIN example.int
1405
1406 IN NAPTR 100 10 "U" NID2U+https:authn
1407     "!^([\^@]+)@(.*)$!https://serv.example.int:8000/cgi-bin/getmd?\1!" ""
1408 IN NAPTR 100 10 "U" NID2U+https:idp
1409     "!^([\^@]+)@(.*)$!https://auth.example.int/app/auth?\1" ""

```

1410 **4.2.3 Resolution**

1411 When resolving metadata for an entity via the DNS, the unique identifier of the entity is used as the initial  
1412 input into the resolution process, rather than as an actual location Proceed as follows:

- 1413 • If the unique identifier is a URN, proceed with the resolution steps as defined in [RFC3404].
- 1414 • Otherwise, parse the identifier to obtain the fully-qualified domain name.
- 1415 • Query the DNS for NAPTR resource records of the domain iteratively until a terminal resource  
1416 record is returned.
- 1417 • Identify which resource record to use based on the service fields, then order fields, then preference  
1418 fields of the result set.
- 1419 • Obtain the document(s) at the provided location(s) as required by the application.

1420 **4.2.3.1 Parsing the Unique Identifier**

1421 To initiate the resolution of the location of the metadata information, it will be necessary in some cases to  
1422 decompose the entity's unique identifier (expressed as a URI) into one or more atomic elements.

1423 The following regular expression should be used when initiating the decomposition process:

```

1424 ^([\^:/?#]+)?/*([\^:/?#]*@)?(((\^/?#*\.)*((\^/?#:\.)(\^/?#:\.)))?(:\d+
1425 )?([\^?#]*)(\#[\^#]*)?(\#.*)?$
1426     1           2           34           56           7           8
1427     9           10          11

```

1428 Subexpression 3 MUST result in a Fully-Qualified Domain Name (FQDN), which will be the basis for  
1429 retrieving metadata locations from this zone.

#### 1430 **4.2.3.2 Obtaining Metadata via the DNS**

1431 Upon completion of the parsing of the identifier, the application then performs a DNS query for the resulting  
1432 domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses.  
1433 Applications MAY exclude from the result set any service definitions that do not concern the present  
1434 request operations.

1435 Resolving applications MUST subsequently order the result set according to the order field, and MAY  
1436 order the result set based on the preference set. Resolvers are NOT REQUIRED to follow the ordering of  
1437 the preferences field. The resulting NAPTR resource record(s) are operated on iteratively (based on the  
1438 order flag) until a terminal NAPTR resource record is reached.

1439 The result will be a well-formed, absolute URL, which is then used to retrieve the metadata document.

#### 1440 **4.2.4 Metadata Location Caching**

1441 Location caching MUST NOT exceed the TTL of the DNS zone from which the location was derived.  
1442 Resolvers MUST obtain a fresh copy of the metadata location upon reaching the expiration of the TTL of  
1443 the zone.

1444 Publishers of metadata documents should carefully consider the TTL of the zone when making changes  
1445 to metadata document locations. Should such a location change occur, a publisher MUST either keep the  
1446 document at both the old and new location until all conforming resolvers are certain to have the updated  
1447 location (e.g.: time of zone change + TTL), or provide an HTTP Redirect [RFC2616] response at the old  
1448 location specifying the new location.

### 1449 **4.3 Post-Processing of Metadata**

1450 The following sections describe the post-processing of metadata.

#### 1451 **4.3.1 Metadata Instance Caching**

1452 Document caching MUST NOT exceed the `validUntil` or `cacheDuration` attribute of the subject  
1453 element(s). If metadata elements have parent elements which contain caching policies, the parent  
1454 element takes precedence.

1455 To properly process the `cacheDuration` attribute, consumers MUST retain the date and time when the  
1456 document was retrieved.

1457 When a document or element has expired, the consumer MUST retrieve a fresh copy, which may require  
1458 a refresh of the document location(s). Consumers SHOULD process document cache processing  
1459 according to [RFC2616] Section 13, and MAY request the Last-Modified date and time from the HTTP  
1460 server. Publishers SHOULD ensure acceptable cache processing as described in [RFC2616] (Section  
1461 10.3.5 304 Not Modified).

#### 1462 **4.3.2 Handling of HTTPS Redirects**

1463 Publishers MAY issue an HTTP Redirect (301 Moved Permanently, 302 or 307 Temporary Redirect)  
1464 [RFC2616], and user agents MUST follow the specified URL in the Redirect response. Redirects  
1465 SHOULD be of the same protocol as the initial request.

#### 1466 **4.3.3 Processing of XML Signatures and General Trust Processing**

1467 Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and  
1468 for the trust ascribed to the entity described by such metadata:

- 1469 • Trust derived from the signature of the DNS zone from which the metadata location URL was

1470 resolved, ensuring accuracy of the metadata document location(s)

- 1471 • Trust derived from signature processing of the metadata document itself, ensuring the integrity of
- 1472 the XML document
- 1473 • Trust derived from the SSL/TLS server authentication of the metadata location URL, ensuring the
- 1474 identity of the publisher of the metadata

1475 Post-processing of the metadata document MUST include signature processing at the XML-document

1476 level and MAY include one of the other two processes. Specifically, the relying party MAY choose to trust

1477 any of the cited authorities in the resolution and parsing process. Publishers of metadata MUST employ a

1478 document-integrity mechanism and MAY employ any of the other two processing profiles to establish trust

1479 in the metadata document, governed by implementation policies.

#### 1480 **4.3.3.1 Processing Signed DNS Zones**

1481 Verification of DNS zone signature SHOULD be processed, if present, as described in [RFC2535].

#### 1482 **4.3.3.2 Processing Signed Documents and Fragments**

1483 Published metadata documents SHOULD be signed, as described in Section 3, either by a certificate

1484 issued to the subject of the document, or by another trusted party. Publishers MAY consider signatures of

1485 other parties as a means of trust conveyance.

1486 Metadata consumers MUST validate signatures, when present, on the metadata document as described

1487 by Section 3.

#### 1488 **4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL**

1489 It is STRONGLY RECOMMENDED that publishers implement TLS/SSL URLs; therefore, consumers

1490 SHOULD consider the trust inherited from the issuer of the TLS/SSL certificate. Publication URLs may not

1491 always be located in the domain of the subject of the metadata document; therefore, consumers SHOULD

1492 NOT presume certificates whose subject is the entity in question, as it may be hosted by another trusted

1493 party.

1494 As the basis of this trust may not be available against a cached document, other mechanisms SHOULD

1495 be used under such circumstances.

---

## 5 References

1496

- 1497 [RFC1034] P. Mockapetris. *Domain Names – Concepts and Facilities*. IETF RFC 1034,  
1498 November 1987. See <http://www.ietf.org/rfc/rfc1034.txt>.
- 1499 [RFC2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*,  
1500 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1501 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January  
1502 1999. See <http://www.ietf.org/rfc/rfc2246.txt>.
- 1503 [RFC2535] D. Eastlake. *Domain Name System Security Extensions*. IETF RFC  
1504 2535, March 1999. See <http://www.ietf.org/rfc/rfc2535.txt>.
- 1505 [RFC2616] R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. IETF RFC  
1506 2616, June 1999. See <http://www.ietf.org/rfc/rfc2616.txt>.
- 1507 [RFC2915] M. Mealling. *The Naming Authority Pointer (NAPTR) DNS Resource  
1508 Record*. IETF RFC 2915, September 2000. See <http://www.ietf.org/rfc/rfc2915.txt>.
- 1509 [RFC3401] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part One:  
1510 The Comprehensive DDDS*. IETF RFC 3401, October 2002. See  
1511 <http://www.ietf.org/rfc/rfc3401.txt>.
- 1512 [RFC3403] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Three:  
1513 The Domain Name System (DNS) Database*. IETF RFC 3403, October 2002. See  
1514 <http://www.ietf.org/rfc/rfc3403.txt>.
- 1515 [RFC3404] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Four:  
1516 The Uniform Resource Identifiers (URI) Resolution Application*. IETF RFC 3404,  
1517 October 2002. See <http://www.ietf.org/rfc/rfc3404.txt>.
- 1518 [RFC3546] S. Blake-Wilson et al. *Transport Layer Security (TLS) Extensions*. IETF  
1519 RFC 3546, June 2003. See <http://www.ietf.org/rfc/rfc3546.txt>.
- 1520 [SAMLBind] S. Cantor et al. *Bindings for the OASIS Security Assertion Markup Language  
1521 (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os.  
1522 See <http://www.oasis-open.org/committees/security/>.
- 1523 [SAMLConform] P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion  
1524 Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-  
1525 conformance-2.0-os. <http://www.oasis-open.org/committees/security/>.
- 1526 [SAMLCore] S. Cantor et al. *Assertions and Protocols for the OASIS Security Assertion  
1527 Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-  
1528 core-2.0-os. See <http://www.oasis-open.org/committees/security/>.
- 1529 [SAMLMeta-xsd] S. Cantor et al. *SAML metadata schema*. OASIS SSTC, March 2005. Document  
1530 ID saml-schema-metadata-2.0. See [http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>1531 open.org/committees/security/).
- 1532 [SAMLProf] S. Cantor et al. *Profiles for the OASIS Security Assertion Markup Language  
1533 (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-profiles-2.0-os. See  
1534 <http://www.oasis-open.org/committees/security/>.
- 1535 [SAMLSec] F. Hirsch et al. *Security and Privacy Considerations for the OASIS Security  
1536 Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document  
1537 ID saml-sec-consider-2.0-os. See [http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>1538 open.org/committees/security/).
- 1539 [Schema1] H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web  
1540 Consortium Recommendation, May 2001. See [http://www.w3.org/TR/xmlschema-  
1/](http://www.w3.org/TR/xmlschema-<br/>1541 1/). Note that this specification normatively references [Schema2], listed below.
- 1542 [Schema2] P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web

1543 Consortium Recommendation, May 2001. See <http://www.w3.org/TR/xmlschema->  
1544 **[XMLEnc]** D. Eastlake et al. XML-Encryption Syntax and Processing,  
1545 <http://www.w3.org/TR/xmlenc-core/>, World Wide Web Consortium.  
1546 **[XMLSig]** D. Eastlake et al. XML-Signature Syntax and Processing,  
1547 <http://www.w3.org/TR/xmlsig-core/>, World Wide Web Consortium.

---

1548 **Appendix A.Registration of MIME media type**  
1549 **application/samlmetadata+xml**

1550 **Introduction**

1551 This document defines a MIME media type -- `application/samlmetadata+xml` -- for use  
1552 with the XML serialization of Security Assertion Markup Language metadata.  
1553

1554 SAML is a work product of the OASIS Security Services Technical Committee [SSTC]. The SAML  
1555 specifications define XML-based constructs with which one may make, and convey, security  
1556 assertions. Using SAML, one can assert that an authentication event pertaining to some subject  
1557 has occurred and convey said assertion to a relying party, for example.  
1558

1559 SAML profiles require agreements between system entities regarding identifiers, binding support,  
1560 endpoints, certificates, keys, and so forth. Such information is treated as metadata by SAML v2.0.  
1561 [SAMLv2Meta] specifies this metadata, as well as specifying metadata publication and resolution  
1562 mechanisms. If the publishing protocol permits MIME-based identification of content types, then  
1563 use of the `application/samlmetadata+xml` MIME media type is required.

1564 **MIME media type name**

1565 `application`

1566 **MIME subtype name**

1567 `samlmetadata+xml`

1568 **Required parameters**

1569 None

1570 **Optional parameters**

1571 `charset`

1572 Same as `charset` parameter of `application/xml` [RFC3023].

1573 **Encoding considerations**

1574 Same as for `application/xml` [RFC3023].

1575 **Security considerations**

1576 Per their specification, `samlmetadata+xml` typed objects do not contain executable content.  
1577 However, these objects are XML-based [XML], and thus they have all of the general security  
1578 considerations presented in Section10 of [RFC3023].

1579 SAML metadata [SAMLv2Meta] contains information whose integrity and authenticity is important  
1580 – identity provider and service provider public keys and endpoint addresses, for example.

1581 To counter potential issues, the publisher may sign `samlmetadata+xml` typed objects. Any such  
1582 signature should be verified by the recipient of the data - both as a valid signature, and as being  
1583 the signature of the publisher.

1584 Additionally, various of the publication protocols, e.g. HTTP-over-TLS/SSL, offer means for  
1585 ensuring the authenticity of the publishing party and for protecting the metadata in transit.

1586 [SAMLv2Meta] also defines prescriptive metadata caching directives, as well as guidance on  
1587 handling HTTPS redirects, trust processing, server authentication, and related items.  
1588 For a more detailed discussion of SAML v2.0 metadata and its security considerations, please  
1589 see [SAMLv2Meta]. For a discussion of overall SAML v2.0 security considerations and specific  
1590 security-related design features, please refer to the SAML v2.0 specifications listed in the below  
1591 bibliography. The specifications containing security-specific information are explicitly listed.

## 1592 **Interoperability considerations**

1593 SAML v2.0 metadata explicitly supports identifying the protocols and versions supported by the  
1594 identified entities. For example, an identity provider entity can be denoted as supporting SAML  
1595 v2.0 [SAMLv2.0], SAML v1.1 [SAMLv1.1], Liberty ID-FF 1.2 [LAPFF], or even other protocols if  
1596 they are unambiguously identifiable via URI [RFC2396]. This protocol support information is  
1597 conveyed via the `protocolSupportEnumeration` attribute of metadata objects of the  
1598 **RoleDescriptorType**.

## 1599 **Published specification**

1600 [SAMLv2Meta] explicitly specifies use of the `application/samlmetadata+xml` MIME media  
1601 type.

## 1602 **Applications which use this media type**

1603 Potentially any application implementing SAML v2.0, as well as those applications implementing  
1604 specifications based on SAML, e.g. those available from the Liberty Alliance [LAP].

## 1605 **Additional information**

### 1606 **Magic number(s)**

1607 In general, the same as for `application/xml` [RFC3023]. In particular, the XML root element of  
1608 the returned object will have a namespace-qualified name with:  
1609

- 1610 – a local name of: `EntityDescriptor`, or  
1611 `AffiliationDescriptor`, or  
1612 `EntitiesDescriptor`
- 1614 – a namespace URI of: `urn:oasis:names:tc:SAML:2.0:metadata`  
1615 (the SAMLv2.0 metadata namespace)

### 1616 **File extension(s)**

1617 None

### 1618 **Macintosh File Type Code(s)**

1619 None

## 1620 **Person & email address to contact for further information**

1621 This registration is made on behalf of the OASIS Security Services Technical Committee (SSTC)  
1622 Please refer to the SSTC website for current information on committee chairperson(s) and their  
1623 contact addresses: <http://www.oasis-open.org/committees/security/>. Committee members should  
1624 submit comments and potential errata to the [securityservices@lists.oasis-open.org](mailto:securityservices@lists.oasis-open.org) list. Others  
1625 should submit them by filling out the web form located at [http://www.oasis-](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security)  
1626 [open.org/committees/comments/form.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security).

1627  
1628 Additionally, the SAML developer community email distribution list, [saml-dev@lists.oasis-](mailto:saml-dev@lists.oasis-open.org)  
1629 [open.org](http://lists.oasis-open.org), may be employed to discuss usage of the `application/samlmetadata+xml` MIME  
1630 media type. The "saml-dev" mailing list is publicly archived here: [http://lists.oasis-](http://lists.oasis-open.org/archives/saml-dev/)  
1631 [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  
1632 subscribe, send a message with the single word "subscribe" in the message body, to: [saml-dev-](mailto:saml-dev-request@lists.oasis-open.org)  
1633 [request@lists.oasis-open.org](mailto:saml-dev-request@lists.oasis-open.org).

## 1634 Intended usage

1635 COMMON

## 1636 Author/Change controller

1637 The SAML specification sets are a work product of the OASIS Security Services Technical  
1638 Committee (SSTC). OASIS and the SSTC have change control over the SAML specification sets.

## 1639 Bibliography

- 1640 [LAP] “*Liberty Alliance Project*”. See <http://www.projectliberty.org/>
- 1641 [LAPFF] “*Liberty Alliance Project: Federation Framework*”. See  
1642 <http://www.projectliberty.org/resources/specifications.php#box1>
- 1643 [OASIS] “*Organization for the Advancement of Structured Information Systems*”.  
1644 See <http://www.oasis-open.org/>
- 1645 [RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifiers*  
1646 *(URI): Generic Syntax*. IETF RFC 2396, August 1998. Available at  
1647 <http://www.ietf.org/rfc/rfc2396.txt>
- 1648 [RFC3023] M. Murata, S. St.Laurent, D. Kohn, “*XML Media Types*”, IETF Request for  
1649 Comments 3023, January 2001. Available as [http://www.rfc-](http://www.rfc-editor.org/rfc/rfc3023.txt)  
1650 [editor.org/rfc/rfc3023.txt](http://www.rfc-editor.org/rfc/rfc3023.txt)
- 1651 [SAMLv1.1] OASIS Security Services Technical Committee, “*Security Assertion*  
1652 *Markup Language (SAML) Version 1.1 Specification Set*”. OASIS  
1653 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)  
1654 [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)
- 1655 [SAMLv2.0] OASIS Security Services Technical Committee, “*Security Assertion*  
1656 *Markup Language (SAML) Version 2.0 Specification Set*”. OASIS  
1657 Standard, 15-Mar-2005. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip)  
1658 [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)
- 1659 [SAMLv2Bind] S. Cantor et al., “*Bindings for the OASIS Security Assertion Markup*  
1660 *Language (SAML) V2.0*”. OASIS, March 2005. Document ID `saml-`  
1661 `bindings-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf)  
1662 [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)
- 1663 [SAMLv2Core] S. Cantor et al., “*Assertions and Protocols for the OASIS Security*  
1664 *Assertion Markup Language (SAML) V2.0*”. OASIS, March 2005.  
1665 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)  
1666 [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)
- 1667 [SAMLv2Meta] S. Cantor et al., “*Metadata for the OASIS Security Assertion Markup*  
1668 *Language (SAML) V2.0*”. OASIS SSTC, August 2004. Document ID `saml-`  
1669 `metadata-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf)  
1670 [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)
- 1671 [SAMLv2Prof] S. Cantor et al., “*Profiles for the OASIS Security Assertion Markup*  
1672 *Language (SAML) V2.0*”. OASIS, March 2005. Document ID `saml-`  
1673 `profiles-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf)  
1674 [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)

1675	[SAMLv2Sec]	F. Hirsch et al., "Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0". OASIS, March 2005. Document ID saml-sec-consider-2.0-os. Available at: <a href="http://docs.oasis-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</a>
1676		
1677		
1678		
1679	[SSTC]	"OASIS Security Services Technical Committee". See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a>
1680		
1681	[XML]	Bray, T., Paoli, J., Sperberg-McQueen, C.M. and E. Maler, François Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium Recommendation REC-xml, Feb 2004, Available as <a href="http://www.w3.org/TR/REC-xml/">http://www.w3.org/TR/REC-xml/</a>
1682		
1683		
1684		
1685		

---

## 1686 Appendix B. Acknowledgments

1687 The editors would like to acknowledge the contributions of the OASIS Security Services Technical  
1688 Committee, whose voting members at the time of publication were:

- 1689 • Conor Cahill, AOL
- 1690 • John Hughes, Atos Origin
- 1691 • Hal Lockhart, BEA Systems
- 1692 • Mike Beach, Boeing
- 1693 • Rebekah Metz, Booz Allen Hamilton
- 1694 • Rick Randall, Booz Allen Hamilton
- 1695 • Ronald Jacobson, Computer Associates
- 1696 • Gavenraj Sodhi, Computer Associates
- 1697 • Thomas Wisniewski, Entrust
- 1698 • Carolina Canales-Valenzuela, Ericsson
- 1699 • Dana Kaufman, Forum Systems
- 1700 • Irving Reid, Hewlett-Packard
- 1701 • Guy Denton, IBM
- 1702 • Heather Hinton, IBM
- 1703 • Maryann Hondo, IBM
- 1704 • Michael McIntosh, IBM
- 1705 • Anthony Nadalin, IBM
- 1706 • Nick Ragouzis, Individual
- 1707 • Scott Cantor, Internet2
- 1708 • Bob Morgan, Internet2
- 1709 • Peter Davis, Neustar
- 1710 • Jeff Hodges, Neustar
- 1711 • Frederick Hirsch, Nokia
- 1712 • Senthil Sengodan, Nokia
- 1713 • Abbie Barbir, Nortel Networks
- 1714 • Scott Kiestler, Novell
- 1715 • Cameron Morris, Novell
- 1716 • Paul Madsen, NTT
- 1717 • Steve Anderson, OpenNetwork
- 1718 • Ari Kermaier, Oracle
- 1719 • Vamsi Motukuru, Oracle
- 1720 • Darren Platt, Ping Identity
- 1721 • Prateek Mishra, Principal Identity
- 1722 • Jim Lien, RSA Security
- 1723 • John Linn, RSA Security
- 1724 • Rob Philpott, RSA Security
- 1725 • Dipak Chopra, SAP
- 1726 • Jahan Moreh, Sigaba
- 1727 • Bhavna Bhatnagar, Sun Microsystems
- 1728 • Eve Maler, Sun Microsystems

- 1729 • Ronald Monzillo, Sun Microsystems
- 1730 • Emily Xu, Sun Microsystems
- 1731 • Greg Whitehead, Trustgenix
- 1732

1733 The editors also would like to acknowledge the following former SSTC members for their contributions to  
1734 this or previous versions of the OASIS Security Assertions Markup Language Standard:

- 1735 • Stephen Farrell, Baltimore Technologies
- 1736 • David Orchard, BEA Systems
- 1737 • Krishna Sankar, Cisco Systems
- 1738 • Zahid Ahmed, CommerceOne
- 1739 • Tim Alsop, CyberSafe Limited
- 1740 • Carlisle Adams, Entrust
- 1741 • Tim Moses, Entrust
- 1742 • Nigel Edwards, Hewlett-Packard
- 1743 • Joe Pato, Hewlett-Packard
- 1744 • Bob Blakley, IBM
- 1745 • Marlena Erdos, IBM
- 1746 • Marc Chanliau, Netegrity
- 1747 • Chris McLaren, Netegrity
- 1748 • Lynne Rosenthal, NIST
- 1749 • Mark Skall, NIST
- 1750 • Charles Knouse, Oblix
- 1751 • Simon Godik, Overxeer
- 1752 • Charles Norwood, SAIC
- 1753 • Evan Prodromou, Securant
- 1754 • Robert Griffin, RSA Security (former editor)
- 1755 • Sai Allarvarpu, Sun Microsystems
- 1756 • Gary Ellison, Sun Microsystems
- 1757 • Chris Ferris, Sun Microsystems
- 1758 • Mike Myers, Traceroute Security
- 1759 • Phillip Hallam-Baker, VeriSign (former editor)
- 1760 • James Vanderbeek, Vodafone
- 1761 • Mark O'Neill, Vordel
- 1762 • Tony Palmer, Vordel

1763  
1764 Finally, the editors wish to acknowledge the following people for their contributions of material used as  
1765 input to the OASIS Security Assertions Markup Language specifications:

- 1766 • Thomas Gross, IBM
- 1767 • Birgit Pfitzmann, IBM

1768 The editors also would like to gratefully acknowledge Jahan Moreh of Sigaba, who during his tenure on  
1769 the SSTC was the primary editor of the errata working document and who made major substantive  
1770 contributions to all of the errata materials.

---

## 1771 Appendix C. Notices

1772 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
1773 might be claimed to pertain to the implementation or use of the technology described in this document or  
1774 the extent to which any license under such rights might or might not be available; neither does it represent  
1775 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to  
1776 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made  
1777 available for publication and any assurances of licenses to be made available, or the result of an attempt  
1778 made to obtain a general license or permission for the use of such proprietary rights by implementors or  
1779 users of this specification, can be obtained from the OASIS Executive Director.

1780 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or  
1781 other proprietary rights which may cover technology that may be required to implement this specification.  
1782 Please address the information to the OASIS Executive Director.

1783 **Copyright © OASIS Open 2005. All Rights Reserved.**

1784 This document and translations of it may be copied and furnished to others, and derivative works that  
1785 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and  
1786 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and  
1787 this paragraph are included on all such copies and derivative works. However, this document itself may  
1788 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as  
1789 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights  
1790 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it  
1791 into languages other than English.

1792 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
1793 or assigns.

1794 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
1795 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
1796 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR  
1797 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.