

---

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

Working Draft 05, 8 September 2015

**Document identifier:**

sstc-saml-metadata-errata-2.0-wd-05

**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  
Eve Maler, Sun Microsystems (errata editor)  
Eric Goodman, Individual (errata editor)

**Contributors to the Errata:**

Rob Philpott, EMC Corporation  
Nick Ragouzis, Enosis Group  
Thomas Wisniewski, Entrust  
Greg Whitehead, HP  
Heather Hinton, IBM  
Connor P. Cahill, Intel  
Scott Cantor, Internet2  
Nate Klingenstein, Internet2  
RL 'Bob' Morgan, Internet2  
John Bradley, Individual  
Jeff Hodges, Individual  
Joni Brennan, Liberty Alliance  
Eric Tiffany, Liberty Alliance  
Thomas Hardjono, M.I.T.  
Tom Scavo, NCSA  
Peter Davis, NeuStar, Inc.  
Frederick Hirsch, Nokia Corporation  
Paul Madsen, NTT Corporation  
Ari Kermaier, Oracle Corporation  
Hal Lockhart, Oracle Corporation  
Prateek Mishra, Oracle Corporation  
Brian Campbell, Ping Identity  
Anil Saldhana, Red Hat Inc.  
Jim Lien, RSA Security  
Jahan Moreh, Sigaba  
Kent Spaulding, Skyworth TTG Holdings Limited  
Emily Xu, Sun Microsystems  
David Staggs, Veteran's Health Administration

**SAML V2.0 Contributors:**

Conor P. Cahill, AOL  
John Hughes, Atos Origin

48 Hal Lockhart, BEA Systems  
49 Michael Beach, Boeing  
50 Rebekah Metz, Booz Allen Hamilton  
51 Rick Randall, Booz Allen Hamilton  
52 Thomas Wisniewski, Entrust  
53 Irving Reid, Hewlett-Packard  
54 Paula Austel, IBM  
55 Maryann Hondo, IBM  
56 Michael McIntosh, IBM  
57 Tony Nadalin, IBM  
58 Nick Ragouzis, Individual  
59 Scott Cantor, Internet2  
60 RL 'Bob' Morgan, Internet2  
61 Peter C Davis, Neustar  
62 Jeff Hodges, Neustar  
63 Frederick Hirsch, Nokia  
64 John Kemp, Nokia  
65 Paul Madsen, NTT  
66 Steve Anderson, OpenNetwork  
67 Prateek Mishra, Principal Identity  
68 John Linn, RSA Security  
69 Rob Philpott, RSA Security  
70 Jahan Moreh, Sigaba  
71 Anne Anderson, Sun Microsystems  
72 Eve Maler, Sun Microsystems  
73 Ron Monzillo, Sun Microsystems  
74 Greg Whitehead, Trustgenix

75 **Abstract:**

76 SAML profiles require agreements between system entities regarding identifiers, binding support  
77 and endpoints, certificates and keys, and so forth. A metadata specification is useful for  
78 describing this information in a standardized way. The SAML V2.0 Metadata document defines an  
79 extensible metadata format for SAML system entities, organized by roles that reflect SAML  
80 profiles. Such roles include that of Identity Provider, Service Provider, Affiliation, Attribute  
81 Authority, Attribute Consumer, and Policy Decision Point. This document, known as an “errata  
82 composite”, combines corrections to reported errata with the original specification text. By design,  
83 the corrections are limited to clarifications of ambiguous or conflicting specification text. This  
84 document shows deletions from the original specification as struck-through text, and additions as  
85 colored underlined text. The “[*Enn*]” designations embedded in the text refer to particular errata  
86 and their dispositions.

87 **Status:**

88 This errata composite document is a **working draft** based on the [original](#) OASIS Standard  
89 document that had been produced by the Security Services Technical Committee and approved  
90 by the OASIS membership on 1 March 2005. While the errata corrections appearing here are  
91 non-normative, they reflect changes specified by the Approved Errata document (currently at  
92 Working Draft revision 02), which is on an OASIS standardization track. In case of any  
93 discrepancy between this document and the Approved Errata, the latter has precedence.

94 This document includes corrections for errata E7, E33, E34, E37, E41, E62, E66, E68, E69, E74,  
95 E76, E77, E81, E82, E83, E87, E88, E89, E91, and E94.

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

99 For information on whether any patents have been disclosed that may be essential to  
100 implementing this specification, and any offers of patent licensing terms, please refer to the  
101 Intellectual Property Rights web page for the Security Services TC ([http://www.oasis-  
open.org/committees/security/ipr.php](http://www.oasis-<br/>102 open.org/committees/security/ipr.php)).

---

# 103 Table of Contents

104	1 Introduction.....	6
105	1.1 Notation.....	6
106	2 Metadata for SAML V2.0.....	7
107	2.1 Namespaces .....	7
108	2.2 Common Types.....	8
109	2.2.1 Simple Type entityIDType.....	8
110	2.2.2 Complex Type EndpointType.....	8
111	2.2.3 Complex Type IndexedEndpointType.....	9
112	2.2.4 Complex Type localizedNameType.....	9
113	2.2.5 Complex Type localizedURIType.....	10
114	2.3 Root Elements.....	10
115	2.3.1 Element <EntitiesDescriptor>.....	10
116	2.3.2 Element <EntityDescriptor>.....	11
117	2.3.2.1 Element <Organization>.....	13
118	2.3.2.2 Element <ContactPerson>.....	14
119	2.3.2.3 Element <AdditionalMetadataLocation>.....	15
120	2.4 Role Descriptor Elements.....	15
121	2.4.1 Element <RoleDescriptor>.....	15
122	2.4.1.1 Element <KeyDescriptor>.....	17
123	2.4.2 Complex Type SSODescriptorType.....	18
124	2.4.3 Element <IDPSSODescriptor>.....	18
125	2.4.4 Element <SPSSODescriptor>.....	20
126	2.4.4.1 Element <AttributeConsumingService>.....	21
127	2.4.4.1.1 [E34]Element <RequestedAttribute>.....	21
128	2.4.5 Element <AuthnAuthorityDescriptor>.....	22
129	2.4.6 Element <PDPDescriptor>.....	23
130	2.4.7 Element <AttributeAuthorityDescriptor>.....	23
131	2.5 Element <AffiliationDescriptor>.....	24
132	2.6 Examples.....	25
133	3 Signature Processing.....	29
134	3.1 XML Signature Profile.....	29
135	3.1.1 Signing Formats and Algorithms.....	29
136	3.1.2 References.....	29
137	3.1.3 Canonicalization Method.....	30
138	3.1.4 Transforms.....	30
139	3.1.5 [E91] Object.....	30
140	3.1.6 KeyInfo.....	30
141	4 Metadata Publication and Resolution.....	31
142	4.1 Publication and Resolution via Well-Known Location.....	31
143	4.1.1 Publication.....	31
144	4.1.2 Resolution.....	31
145	4.2 Publishing and Resolution via DNS.....	31
146	4.2.1 Publication.....	32
147	4.2.1.1 First Well Known Rule.....	32
148	4.2.1.2 The Order Field.....	32
149	4.2.1.3 The Preference Field.....	32
150	4.2.1.4 The Flag Field.....	33

151	4.2.1.5 The Service Field.....	33
152	4.2.1.6 The Regex and Replacement Fields.....	33
153	4.2.2 NAPTR Examples.....	34
154	4.2.2.1 Entity Metadata NAPTR Examples.....	34
155	4.2.2.2 Name Identifier Examples.....	34
156	4.2.3 Resolution.....	34
157	4.2.3.1 Parsing the Unique Identifier.....	34
158	4.2.3.2 Obtaining Metadata via the DNS.....	35
159	4.2.4 Metadata Location Caching.....	35
160	4.3 Post-Processing of Metadata.....	35
161	4.3.1 Metadata Instance Caching.....	35
162	4.3.2 [E94] Metadata Instance Validity .....	35
163	4.3.3 Handling of HTTPS Redirects.....	36
164	4.3.4 Processing of XML Signatures and General Trust Processing.....	36
165	4.3.4.1 Processing Signed DNS Zones.....	36
166	4.3.4.2 Processing Signed Documents and Fragments.....	36
167	4.3.4.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL.....	36
168	5 References.....	37
169	Appendix A.Registration of MIME media type application/samlmetadata+xml.....	39
170	Appendix B. Acknowledgments.....	43
171	Appendix C. Notices.....	45

---

# 1 Introduction

172

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

179 [E77]A variety of extension points are also included to allow for the use of SAML metadata in non-SAML  
180 specifications, profiles, and deployments, and such use is encouraged.

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

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

## 1.1 Notation

185

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

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

190

191 `Example code listings appear like this.`

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

193 Conventional XML namespace prefixes are used throughout this specification to stand for their respective  
194 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

195

196 SAML metadata is organized around an extensible collection of roles representing common combinations  
197 of SAML [E77](and potentially non-SAML) protocols and profiles supported by system entities. Each role  
198 is described by an element derived from the extensible base type of `RoleDescriptor`. Such descriptors  
199 are in turn collected into the `<EntityDescriptor>` container element, the primary unit of SAML  
200 metadata. An entity might alternatively represent an affiliation of other entities, such as an affiliation of  
201 service providers. The `<AffiliationDescriptor>` is provided for this purpose.

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

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

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

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

### 2.1 Namespaces

216

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

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

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

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

```
221 <schema
222   targetNamespace="urn:oasis:names:tc:SAML:2.0:metadata"
223   xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
224   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
225   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
226   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
227   xmlns="http://www.w3.org/2001/XMLSchema"
228   elementFormDefault="unqualified"
229   attributeFormDefault="unqualified"
230   blockDefault="substitution"
231   version="2.0">
232   <import namespace="http://www.w3.org/2000/09/xmldsig#"
233     schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
234 20020212/xmldsig-core-schema.xsd"/>
235   <import namespace="http://www.w3.org/2001/04/xmlenc#"
236     schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-
237 20021210/xenc-schema.xsd"/>
238   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
239     schemaLocation="saml-schema-assertion-2.0.xsd"/>
240   <import namespace="http://www.w3.org/XML/1998/namespace"
241     schemaLocation="http://www.w3.org/2001/xml.xsd"/>
```

```

242 <annotation>
243   <documentation>
244     Document identifier: saml-schema-metadata-2.0
245     Location: http://docs.oasis-open.org/security/saml/v2.0/
246     Revision history:
247       V2.0 (March, 2005):
248         Schema for SAML metadata, first published in SAML 2.0.
249   </documentation>
250 </annotation>
251 ...
252 </schema>

```

## 253 2.2 Common Types

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

### 256 2.2.1 Simple Type `entityIDType`

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

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

```

263 <simpleType name="entityIDType">
264   <restriction base="anyURI">
265     <maxLength value="1024"/>
266   </restriction>
267 </simpleType>

```

### 268 2.2.2 Complex Type `EndpointType`

269 The complex type **EndpointType** describes a [E77] protocol binding endpoint at which an [E77] entity can  
 270 be sent protocol messages. Various protocol or profile-specific metadata elements are bound to this type.  
 271 It consists of the following attributes:

272 **Binding** [Required]

273 A required attribute that specifies the [E77] binding supported by the endpoint. Each binding is  
 274 assigned a URI to identify it.

275 **Location** [Required]

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

278 **ResponseLocation** [Optional]

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

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

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

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

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

```
296 <complexType name="EndpointType">  
297   <sequence>  
298     <any namespace="##other" processContents="lax" minOccurs="0"  
299     maxOccurs="unbounded"/>  
300   </sequence>  
301   <attribute name="Binding" type="anyURI" use="required"/>  
302   <attribute name="Location" type="anyURI" use="required"/>  
303   <attribute name="ResponseLocation" type="anyURI" use="optional"/>  
304   <anyAttribute namespace="##other" processContents="lax"/>  
305 </complexType>
```

### 306 2.2.3 Complex Type IndexedEndpointType

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

310 **index** [Required]

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

315 **isDefault** [Optional]

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

318 In any such sequence of [E37]indexed endpoints that share a common element name and namespace  
319 (i.e. all instances of `<md:AssertionConsumerService>` within a role), the default endpoint is the first  
320 such endpoint with the **isDefault** attribute set to *true*. If no such endpoints exist, the default endpoint  
321 is the first such endpoint without the **isDefault** attribute set to *false*. If no such endpoints exist, the  
322 default endpoint is the first element in the sequence.

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

```
324 <complexType name="IndexedEndpointType">  
325   <complexContent>  
326     <extension base="md:EndpointType">  
327       <attribute name="index" type="unsignedShort" use="required"/>  
328       <attribute name="isDefault" type="boolean" use="optional"/>  
329     </extension>  
330   </complexContent>  
331 </complexType>
```

### 332 2.2.4 Complex Type LocalizedNameType

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

```
335 <complexType name="localizedNameType">
```

```
336     <simpleContent>
337         <extension base="string">
338             <attribute ref="xml:lang" use="required"/>
339         </extension>
340     </simpleContent>
341 </complexType>
```

## 342 2.2.5 Complex Type localizedURIType

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

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

```
346 <complexType name="localizedURIType">
347     <simpleContent>
348         <extension base="anyURI">
349             <attribute ref="xml:lang" use="required"/>
350         </extension>
351     </simpleContent>
352 </complexType>
```

## 353 2.3 Root Elements

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

### 357 2.3.1 Element `<EntitiesDescriptor>`

358 The `<EntitiesDescriptor>` element contains the metadata for an optionally named group of[E77]  
359 entities. Its **EntitiesDescriptorType** complex type contains a sequence of `<EntityDescriptor>`  
360 elements, `<EntitiesDescriptor>` elements, or both:

361 ID [Optional]

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

363 validUntil [Optional]

364 Optional attribute indicates the expiration time of the metadata contained in the element and any  
365 contained elements.

366 cacheDuration [Optional]

367 Optional attribute indicates the maximum length of time a consumer should cache the metadata  
368 contained in the element and any contained elements [E94] before attempting to refresh it.

369 Name [Optional]

370 A string name that identifies a group of[E77] entities in the context of some deployment.

371 `<ds:Signature>` [Optional]

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

374 `<Extensions>` [Optional]

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

378 <EntitiesDescriptor> or <EntityDescriptor> [One or More]  
 379       Contains the metadata for one or more[E77] entities, or a nested group of additional metadata.  
 380 When used as the root element of a metadata instance, this element MUST contain either a `validUntil`  
 381 or `cacheDuration` attribute. It is RECOMMENDED that only the root element of a metadata instance  
 382 contain either attribute.  
 383 [E76]When not used as the root element of a metadata instance, a `validUntil` or `cacheDuration`  
 384 attribute MAY be used to impose a shorter expiration or cache duration than that of the parent or root  
 385 element, but never a longer one; the smaller value takes precedence.

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

```

388 <element name="EntitiesDescriptor" type="md:EntitiesDescriptorType"/>
389 <complexType name="EntitiesDescriptorType">
390   <sequence>
391     <element ref="ds:Signature" minOccurs="0"/>
392     <element ref="md:Extensions" minOccurs="0"/>
393     <choice minOccurs="1" maxOccurs="unbounded">
394       <element ref="md:EntityDescriptor"/>
395       <element ref="md:EntitiesDescriptor"/>
396     </choice>
397   </sequence>
398   <attribute name="validUntil" type="dateTime" use="optional"/>
399   <attribute name="cacheDuration" type="duration" use="optional"/>
400   <attribute name="ID" type="ID" use="optional"/>
401   <attribute name="Name" type="string" use="optional"/>
402 </complexType>
403 <element name="Extensions" type="md:ExtensionsType"/>
404 <complexType final="#all" name="ExtensionsType">
405   <sequence>
406     <any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
407   </sequence>
408 </complexType>
  
```

### 409 2.3.2 Element <EntityDescriptor>

410 The <EntityDescriptor> element specifies metadata for a single[E77] entity. A single entity may act  
 411 in many different roles in the support of multiple profiles. This specification directly supports the following  
 412 concrete roles as well as the abstract <RoleDescriptor> element for extensibility (see subsequent  
 413 sections for more details):

- 414 • SSO Identity Provider
- 415 • SSO Service Provider
- 416 • Authentication Authority
- 417 • Attribute Authority
- 418 • Policy Decision Point
- 419 • Affiliation

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

421 `entityID` [Required]  
 422       Specifies the unique identifier of the[E77] entity whose metadata is described by the element's  
 423 contents.

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

426 validUntil [Optional]  
427 Optional attribute indicates the expiration time of the metadata contained in the element and any  
428 contained elements.

429 cacheDuration [Optional]  
430 Optional attribute indicates the maximum length of time a consumer should cache the metadata  
431 contained in the element and any contained elements [E94] before attempting to refresh it.

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

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

439 <RoleDescriptor>, <IDPSSODescriptor>, <SPSSODescriptor>,  
440 <AuthnAuthorityDescriptor>, <AttributeAuthorityDescriptor>, <PDPDescriptor> [One  
441 or More]  
442 **OR**  
443 <AffiliationDescriptor> [Required]  
444 The primary content of the element is either a sequence of one or more role descriptor elements,  
445 or a specialized descriptor that defines an affiliation.

446 <Organization> [Optional]  
447 Optional element identifying the organization responsible for the[E77] entity described by the  
448 element.

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

451 <AdditionalMetadataLocation> [Zero or More]  
452 Optional sequence of namespace-qualified locations where additional metadata exists for  
453 the[E77] entity. This may include metadata in alternate formats or describing adherence to other  
454 non-SAML specifications.

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

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

459 [E76]When not used as the root element of a metadata instance, a validUntil or cacheDuration  
460 attribute MAY be used to impose a shorter expiration or cache duration than that of the parent or root  
461 element, but never a longer one; the smaller value takes precedence.

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

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

```
469 <element name="EntityDescriptor" type="md:EntityDescriptorType"/>
470 <complexType name="EntityDescriptorType">
471   <sequence>
472     <element ref="ds:Signature" minOccurs="0"/>
473     <element ref="md:Extensions" minOccurs="0"/>
474     <choice>
475       <choice maxOccurs="unbounded">
476         <element ref="md:RoleDescriptor"/>
477         <element ref="md:IDPSSODescriptor"/>
478         <element ref="md:SPSSODescriptor"/>
479         <element ref="md:AuthnAuthorityDescriptor"/>
480         <element ref="md:AttributeAuthorityDescriptor"/>
481         <element ref="md:PDPDescriptor"/>
482       </choice>
483       <element ref="md:AffiliationDescriptor"/>
484     </choice>
485     <element ref="md:Organization" minOccurs="0"/>
486     <element ref="md:ContactPerson" minOccurs="0" maxOccurs="unbounded"/>
487     <element ref="md:AdditionalMetadataLocation" minOccurs="0"
488 maxOccurs="unbounded"/>
489   </sequence>
490   <attribute name="entityID" type="md:entityIDType" use="required"/>
491   <attribute name="validUntil" type="dateTime" use="optional"/>
492   <attribute name="cacheDuration" type="duration" use="optional"/>
493   <attribute name="ID" type="ID" use="optional"/>
494   <anyAttribute namespace="##other" processContents="lax"/>
495 </complexType>
```

### 496 2.3.2.1 Element <Organization>

497 The <Organization> element specifies basic information about an organization responsible for an[E77]  
498 entity or role. The use of this element is always optional. Its content is informative in nature and does not  
499 directly map to any core SAML elements or attributes. Its **OrganizationType** complex type consists of the  
500 following elements:

501 <Extensions> [Optional]

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

505 <OrganizationName> [One or More]

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

507 <OrganizationDisplayName> [One or More]

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

509 <OrganizationURL> [One or More]

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

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

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

```
516 <element name="Organization" type="md:OrganizationType"/>
```

```

517 <complexType name="OrganizationType">
518   <sequence>
519     <element ref="md:Extensions" minOccurs="0"/>
520     <element ref="md:OrganizationName" maxOccurs="unbounded"/>
521     <element ref="md:OrganizationDisplayName" maxOccurs="unbounded"/>
522     <element ref="md:OrganizationURL" maxOccurs="unbounded"/>
523   </sequence>
524   <anyAttribute namespace="##other" processContents="lax"/>
525 </complexType>
526 <element name="OrganizationName" type="md:localizedNameType"/>
527 <element name="OrganizationDisplayName" type="md:localizedNameType"/>
528 <element name="OrganizationURL" type="md:localizedURIType"/>

```

### 529 **2.3.2.2 Element <ContactPerson>**

530 The <ContactPerson> element specifies basic contact information about a person responsible in some  
531 capacity for an[E77] entity or role. The use of this element is always optional. Its content is informative in  
532 nature and does not directly map to any core SAML elements or attributes. Its **ContactType** complex type  
533 consists of the following elements and attributes:

534 **contactType** [Required]

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

537 **<Extensions>** [Optional]

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

541 **<Company>** [Optional]

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

543 **<GivenName>** [Optional]

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

545 **<SurName>** [Optional]

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

547 **<EmailAddress>** [Zero or More]

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

550 **<TelephoneNumber>** [Zero or More]

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

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

553 [E82] At least one child element SHOULD be present in a <ContactPerson> element..

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

```

556 <element name="ContactPerson" type="md:ContactType"/>
557 <complexType name="ContactType">
558   <sequence>
559     <element ref="md:Extensions" minOccurs="0"/>
560     <element ref="md:Company" minOccurs="0"/>
561     <element ref="md:GivenName" minOccurs="0"/>

```

```

562     <element ref="md:SurName" minOccurs="0"/>
563     <element ref="md:EmailAddress" minOccurs="0" maxOccurs="unbounded"/>
564     <element ref="md:TelephoneNumber" minOccurs="0" maxOccurs="unbounded"/>
565 </sequence>
566 <attribute name="contactType" type="md:ContactTypeType" use="required"/>
567 <anyAttribute namespace="##other" processContents="lax"/>
568 </complexType>
569 <element name="Company" type="string"/>
570 <element name="GivenName" type="string"/>
571 <element name="SurName" type="string"/>
572 <element name="EmailAddress" type="anyURI"/>
573 <element name="TelephoneNumber" type="string"/>
574 <simpleType name="ContactTypeType">
575   <restriction base="string">
576     <enumeration value="technical"/>
577     <enumeration value="support"/>
578     <enumeration value="administrative"/>
579     <enumeration value="billing"/>
580     <enumeration value="other"/>
581   </restriction>
582 </simpleType>

```

### 583 2.3.2.3 Element <AdditionalMetadataLocation>

584 The <AdditionalMetadataLocation> element is a namespace-qualified URI that specifies where  
585 additional XML-based metadata may exist for an[E77] entity. Its **AdditionalMetadataLocationType**  
586 complex type extends the **anyURI** type with a **namespace** attribute (also of type **anyURI**). This required  
587 attribute **MUST** contain the XML namespace of the root element of the instance document found at the  
588 specified location.

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

```

591 <element name="AdditionalMetadataLocation"
592 type="md:AdditionalMetadataLocationType"/>
593 <complexType name="AdditionalMetadataLocationType">
594   <simpleContent>
595     <extension base="anyURI">
596       <attribute name="namespace" type="anyURI" use="required"/>
597     </extension>
598   </simpleContent>
599 </complexType>

```

## 600 2.4 Role Descriptor Elements

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

### 604 2.4.1 Element <RoleDescriptor>

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

609 ID [Optional]

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

611 `validUntil` [Optional]  
 612         Optional attribute indicates the expiration time of the metadata contained in the element and any  
 613         contained elements.

614 `cacheDuration` [Optional]  
 615         Optional attribute indicates the maximum length of time a consumer should cache the metadata  
 616         contained in the element and any contained elements [E94] before attempting to refresh it

617 `protocolSupportEnumeration` [Required]  
 618         A whitespace-delimited set of URIs that identify the set of protocol specifications supported by the  
 619         role element. For SAML V2.0 entities, this set MUST include the SAML protocol namespace URI,  
 620         `urn:oasis:names:tc:SAML:2.0:protocol`. Note that future SAML specifications might  
 621         share the same namespace URI, but SHOULD provide alternate "protocol support" identifiers to  
 622         ensure discrimination when necessary.

623 `errorURL` [Optional]  
 624         Optional URI attribute that specifies a location to direct a user for problem resolution and  
 625         additional support related to this role.

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

629 `<Extensions>` [Optional]  
 630         This contains optional metadata extensions that are agreed upon between a metadata publisher  
 631         and consumer. Extension elements MUST be namespace-qualified by a non-SAML-defined  
 632         namespace.

633 `<KeyDescriptor>` [Zero or More]  
 634         Optional sequence of elements that provides information about the cryptographic keys that the  
 635         entity uses when acting in this role.

636 `<Organization>` [Optional]  
 637         Optional element specifies the organization associated with this role. Identical to the element  
 638         used within the `<EntityDescriptor>` element.

639 `<ContactPerson>` [Zero or More]  
 640         Optional sequence of elements specifying contacts associated with this role. Identical to the  
 641         element used within the `<EntityDescriptor>` element.

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

643 [E76]A `validUntil` or `cacheDuration` attribute MAY be used to impose a shorter expiration or cache  
 644 duration than that of the parent or root element, but never a longer one; the smaller value takes  
 645 precedence.

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

```

648 <element name="RoleDescriptor" type="md:RoleDescriptorType"/>
649 <complexType name="RoleDescriptorType" abstract="true">
650   <sequence>
651     <element ref="ds:Signature" minOccurs="0"/>
652     <element ref="md:Extensions" minOccurs="0"/>
653     <element ref="md:KeyDescriptor" minOccurs="0" maxOccurs="unbounded"/>
654     <element ref="md:Organization" minOccurs="0"/>
  
```

```

655     <element ref="md:ContactPerson" minOccurs="0" maxOccurs="unbounded"/>
656   </sequence>
657   <attribute name="ID" type="ID" use="optional"/>
658   <attribute name="validUntil" type="dateTime" use="optional"/>
659   <attribute name="cacheDuration" type="duration" use="optional"/>
660   <attribute name="protocolSupportEnumeration" type="md:anyURLListType"
661 use="required"/>
662   <attribute name="errorURL" type="anyURI" use="optional"/>
663   <anyAttribute namespace="##other" processContents="lax"/>
664 </complexType>
665 <simpleType name="anyURLListType">
666   <list itemType="anyURI"/>
667 </simpleType>

```

#### 668 2.4.1.1 Element <KeyDescriptor>

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

672 use [Optional]

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

675 <ds:KeyInfo> [Required]

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

678 <EncryptionMethod> [Zero or More]

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

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

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

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

687 [E68]The inclusion of multiple <KeyDescriptor> elements with the same use attribute (or no such  
688 attribute) indicates that any of the included keys may be used by the containing role or affiliation. A relying  
689 party SHOULD allow for the use of any of the included keys. When possible the signing or encrypting  
690 party SHOULD indicate as specifically as possible which key it used to enable more efficient processing.

691 [E69]The <ds:KeyInfo> element is a highly generic and extensible means of communicating key  
692 material. This specification takes no position on the allowable or suggested content of this element, nor  
693 on its meaning to a relying party. As a concrete example, no implications of including an X.509 certificate  
694 by value or reference are to be assumed. Its validity period, extensions, revocation status, and other  
695 relevant content may or may not be enforced, at the discretion of the relying party. The details of such  
696 processing, and their security implications, are out of scope; they may, however, be addressed by other  
697 SAML profiles.

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

```

700 <element name="KeyDescriptor" type="md:KeyDescriptorType"/>
701 <complexType name="KeyDescriptorType">
702   <sequence>

```

```

703         <element ref="ds:KeyInfo"/>
704         <element ref="md:EncryptionMethod" minOccurs="0"
705 maxOccurs="unbounded"/>
706     </sequence>
707     <attribute name="use" type="md:KeyTypes" use="optional"/>
708 </complexType>
709 <simpleType name="KeyTypes">
710     <restriction base="string">
711         <enumeration value="encryption"/>
712         <enumeration value="signing"/>
713     </restriction>
714 </simpleType>
715 <element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>

```

## 716 2.4.2 Complex Type SSODescriptorType

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

721 <ArtifactResolutionService> [Zero or More]

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

725 <SingleLogoutService> [Zero or More]

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

728 <ManageNameIDService> [Zero or More]

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

731 <NameIDFormat> [Zero or More]

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

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

```

736 <complexType name="SSODescriptorType" abstract="true">
737     <complexContent>
738         <extension base="md:RoleDescriptorType">
739             <sequence>
740                 <element ref="md:ArtifactResolutionService" minOccurs="0"
741 maxOccurs="unbounded"/>
742                 <element ref="md:SingleLogoutService" minOccurs="0"
743 maxOccurs="unbounded"/>
744                 <element ref="md:ManageNameIDService" minOccurs="0"
745 maxOccurs="unbounded"/>
746                 <element ref="md:NameIDFormat" minOccurs="0"
747 maxOccurs="unbounded"/>
748             </sequence>
749         </extension>
750     </complexContent>
751 </complexType>
752 <element name="ArtifactResolutionService" type="md:IndexedEndpointType"/>
753 <element name="SingleLogoutService" type="md:EndpointType"/>
754 <element name="ManageNameIDService" type="md:EndpointType"/>

```

```
755 <element name="NameIDFormat" type="anyURI"/>
```

### 756 2.4.3 Element <IDPSSODescriptor>

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

760 **WantAuthnRequestsSigned** [Optional]

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

763 <SingleSignOnService> [One or More]

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

767 <NameIDMappingService> [Zero or More]

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

771 <AssertionIDRequestService> [Zero or More]

772 Zero or more elements of type **EndpointType** that describe endpoints that support the profile of  
773 the Assertion [E33]Query/Request protocol defined in [SAMLProf] or the special URI binding for  
774 assertion requests defined in [SAMLBind].

775 <AttributeProfile> [Zero or More]

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

778 <saml:Attribute> [Zero or More]

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

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

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

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

```
795 <element name="IDPSSODescriptor" type="md:IDPSSODescriptorType"/>  
796 <complexType name="IDPSSODescriptorType">  
797   <complexContent>  
798     <extension base="md:SSODescriptorType">  
799       <sequence>
```

```

800         <element ref="md:SingleSignOnService" maxOccurs="unbounded"/>
801         <element ref="md:NameIDMappingService" minOccurs="0"
802 maxOccurs="unbounded"/>
803         <element ref="md:AssertionIDRequestService" minOccurs="0"
804 maxOccurs="unbounded"/>
805         <element ref="md:AttributeProfile" minOccurs="0"
806 maxOccurs="unbounded"/>
807         <element ref="saml:Attribute" minOccurs="0"
808 maxOccurs="unbounded"/>
809         </sequence>
810         <attribute name="WantAuthnRequestsSigned" type="boolean"
811 use="optional"/>
812     </extension>
813 </complexContent>
814 </complexType>
815 <element name="SingleSignOnService" type="md:EndpointType"/>
816 <element name="NameIDMappingService" type="md:EndpointType"/>
817 <element name="AssertionIDRequestService" type="md:EndpointType"/>
818 <element name="AttributeProfile" type="anyURI"/>

```

#### 819 2.4.4 Element <SPSSODescriptor>

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

##### 823 AuthnRequestsSigned [Optional]

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

##### 831 WantAssertionsSigned [Optional]

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

##### 838 <AssertionConsumerService> [One or More]

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

##### 842 <AttributeConsumingService> [Zero or More]

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

845 At most one <AttributeConsumingService> element can have the attribute *isDefault* set to  
846 *true*. [E87] The default element is the first element with the *isDefault* attribute set to *true*. If no such  
847 elements exist, the default element is the first element without the *isDefault* attribute set to *false*. If no  
848 such elements exist, the default element is the first element in the sequence.

849 The following schema fragment defines the <SPSSODescriptor> element and its

850 **SPSSODescriptorType** complex type:

```
851 <element name="SPSSODescriptor" type="md:SPSSODescriptorType"/>
852 <complexType name="SPSSODescriptorType">
853   <complexContent>
854     <extension base="md:SSODescriptorType">
855       <sequence>
856         <element ref="md:AssertionConsumerService"
857 maxOccurs="unbounded"/>
858         <element ref="md:AttributeConsumingService" minOccurs="0"
859 maxOccurs="unbounded"/>
860       </sequence>
861       <attribute name="AuthnRequestsSigned" type="boolean"
862 use="optional"/>
863       <attribute name="WantAssertionsSigned" type="boolean"
864 use="optional"/>
865     </extension>
866   </complexContent>
867 </complexType>
868 <element name="AssertionConsumerService" type="md:IndexedEndpointType"/>
```

#### 869 **2.4.4.1 Element <AttributeConsumingService>**

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

873 **index** [Required]

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

876 **isDefault** [Optional]

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

879 **<ServiceName>** [One or More]

880 One or more language-qualified names for the service [E88] that are suitable for human  
881 consumption.

882 **<ServiceDescription>** [Zero or More]

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

884 **<RequestedAttribute>** [One or More]

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

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

```
888 <element name="AttributeConsumingService"
889 type="md:AttributeConsumingServiceType"/>
890 <complexType name="AttributeConsumingServiceType">
891   <sequence>
892     <element ref="md:ServiceName" maxOccurs="unbounded"/>
893     <element ref="md:ServiceDescription" minOccurs="0"
894 maxOccurs="unbounded"/>
895     <element ref="md:RequestedAttribute" maxOccurs="unbounded"/>
896   </sequence>
897   <attribute name="index" type="unsignedShort" use="required"/>
898   <attribute name="isDefault" type="boolean" use="optional"/>
899 </complexType>
```

```
900 <element name="ServiceName" type="md:localizedNameType"/>
901 <element name="ServiceDescription" type="md:localizedNameType"/>
```

#### 902 2.4.4.1.1 [E34]Element <RequestedAttribute>

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

906 **isRequired** [Optional]

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

909 [E89] If no NameFormat value is provided, the identifier  
910 urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified (see Section 8.2.1 of  
911 [SAMLv2Core]) is in effect

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

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

```
916 <element name="RequestedAttribute" type="md:RequestedAttributeType"/>
917 <complexType name="RequestedAttributeType">
918   <complexContent>
919     <extension base="saml:AttributeType">
920       <attribute name="isRequired" type="boolean" use="optional"/>
921     </extension>
922   </complexContent>
923 </complexType>
```

#### 924 2.4.5 Element <AuthnAuthorityDescriptor>

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

928 <AuthnQueryService> [One or More]

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

932 <AssertionIDRequestService> [Zero or More]

933 Zero or more elements of type **EndpointType** that describe endpoints that support the profile of  
934 the Assertion [E33]Query/Request protocol defined in [SAMLProf] or the special URI binding for  
935 assertion requests defined in [SAMLBind].

936 <NameIDFormat> [Zero or More]

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

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

```
941 <element name="AuthnAuthorityDescriptor"
942   type="md:AuthnAuthorityDescriptorType"/>
943 <complexType name="AuthnAuthorityDescriptorType">
944   <complexContent>
```

```

945     <extension base="md:RoleDescriptorType">
946         <sequence>
947             <element ref="md:AuthnQueryService" maxOccurs="unbounded"/>
948             <element ref="md:AssertionIDRequestService" minOccurs="0"
949 maxOccurs="unbounded"/>
950             <element ref="md:NameIDFormat" minOccurs="0"
951 maxOccurs="unbounded"/>
952         </sequence>
953     </extension>
954 </complexContent>
955 </complexType>
956 <element name="AuthnQueryService" type="md:EndpointType"/>

```

## 957 2.4.6 Element <PDPDescriptor>

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

961 <AuthzService> [One or More]

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

965 <AssertionIDRequestService> [Zero or More]

966 Zero or more elements of type **EndpointType** that describe endpoints that support the profile of  
967 the Assertion [E33]Query/Request protocol defined in [SAMLProf] or the special URI binding for  
968 assertion requests defined in [SAMLBind].

969 <NameIDFormat> [Zero or More]

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

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

```

974 <element name="PDPDescriptor" type="md:PDPDescriptorType"/>
975 <complexType name="PDPDescriptorType">
976     <complexContent>
977         <extension base="md:RoleDescriptorType">
978             <sequence>
979                 <element ref="md:AuthzService" maxOccurs="unbounded"/>
980                 <element ref="md:AssertionIDRequestService" minOccurs="0"
981 maxOccurs="unbounded"/>
982                 <element ref="md:NameIDFormat" minOccurs="0"
983 maxOccurs="unbounded"/>
984             </sequence>
985         </extension>
986     </complexContent>
987 </complexType>
988 <element name="AuthzService" type="md:EndpointType"/>

```

## 989 2.4.7 Element <AttributeAuthorityDescriptor>

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

- 994 <AttributeService> [One or More]  
 995 One or more elements of type **EndpointType** that describe endpoints that support the profile of  
 996 the Attribute Query protocol defined in [SAMLProf]. All attribute authorities support at least one  
 997 such endpoint, by definition.
- 998 <AssertionIDRequestService> [Zero or More]  
 999 Zero or more elements of type **EndpointType** that describe endpoints that support the profile of  
 1000 the Assertion [E33]Query/Request protocol defined in [SAMLProf] or the special URI binding for  
 1001 assertion requests defined in [SAMLBind].
- 1002 <NameIDFormat> [Zero or More]  
 1003 Zero or more elements of type **anyURI** that enumerate the name identifier formats supported by  
 1004 this authority. See Section 8.3 of [SAMLCore] for some possible values for this element.
- 1005 <AttributeProfile> [Zero or More]  
 1006 Zero or more elements of type **anyURI** that enumerate the attribute profiles supported by this  
 1007 authority. See [SAMLProf] for some possible values for this element.
- 1008 <saml:Attribute> [Zero or More]  
 1009 Zero or more elements that identify the SAML attributes supported by the authority. Specific  
 1010 values MAY optionally be included, indicating that only certain values permitted by the attribute's  
 1011 definition are supported.

1012 The following schema fragment defines the <AttributeAuthorityDescriptor> element and its  
 1013 **AttributeAuthorityDescriptorType** complex type:

```

1014 <element name="AttributeAuthorityDescriptor"
1015 type="md:AttributeAuthorityDescriptorType"/>
1016 <complexType name="AttributeAuthorityDescriptorType">
1017   <complexContent>
1018     <extension base="md:RoleDescriptorType">
1019       <sequence>
1020         <element ref="md:AttributeService" maxOccurs="unbounded"/>
1021         <element ref="md:AssertionIDRequestService" minOccurs="0"
1022 maxOccurs="unbounded"/>
1023         <element ref="md:NameIDFormat" minOccurs="0"
1024 maxOccurs="unbounded"/>
1025         <element ref="md:AttributeProfile" minOccurs="0"
1026 maxOccurs="unbounded"/>
1027         <element ref="saml:Attribute" minOccurs="0"
1028 maxOccurs="unbounded"/>
1029       </sequence>
1030     </extension>
1031   </complexContent>
1032 </complexType>
1033 <element name="AttributeService" type="md:EndpointType"/>

```

1034 **2.5 Element <AffiliationDescriptor>**

1035 The <AffiliationDescriptor> element is an alternative to the sequence of role descriptors  
 1036 described in Section 2.4 that is used when an <EntityDescriptor> describes an affiliation of[E77]  
 1037 entities (typically service providers) rather than a single entity. The <AffiliationDescriptor>  
 1038 element provides a summary of the individual entities that make up the affiliation along with general  
 1039 information about the affiliation itself. Its **AffiliationDescriptorType** complex type contains the following  
 1040 elements and attributes:

- 1041 affiliationOwnerID [Required]  
 1042 Specifies the unique identifier of the entity responsible for the affiliation. The owner is NOT

1043 presumed to be a member of the affiliation; if it is a member, its identifier MUST also appear in an  
1044 <AffiliateMember> element.

1045 ID [Optional]

1046 A document-unique identifier for the element, typically used as a reference point when signing.

1047 validUntil [Optional]

1048 Optional attribute indicates the expiration time of the metadata contained in the element and any  
1049 contained elements.

1050 cacheDuration [Optional]

1051 Optional attribute indicates the maximum length of time a consumer should cache the metadata  
1052 contained in the element and any contained elements [E94] before attempting to refresh it.

1053 <ds:Signature> [Optional]

1054 An XML signature that authenticates the containing element and its contents, as described in  
1055 Section 3.

1056 <Extensions> [Optional]

1057 This contains optional metadata extensions that are agreed upon between a metadata publisher  
1058 and consumer. Extension elements MUST be namespace-qualified by a non-SAML-defined  
1059 namespace.

1060 <AffiliateMember> [One or More]

1061 One or more elements enumerating the members of the affiliation by specifying each member's  
1062 unique identifier. See also Section 8.3.6 of [SAMLCore].

1063 <KeyDescriptor> [Zero or More]

1064 Optional sequence of elements that provides information about the cryptographic keys that the  
1065 affiliation uses as a whole, as distinct from keys used by individual members of the affiliation,  
1066 which are published in the metadata for those entities.

1067 Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

1068 [E76]A validUntil or cacheDuration attribute MAY be used to impose a shorter expiration or cache  
1069 duration than that of the parent or root element, but never a longer one; the smaller value takes  
1070 precedence.

1071 The following schema fragment defines the <AffiliationDescriptor> element and its  
1072 **AffiliationDescriptorType** complex type:

```
1073 <element name="AffiliationDescriptor" type="md:AffiliationDescriptorType"/>
1074 <complexType name="AffiliationDescriptorType">
1075   <sequence>
1076     <element ref="ds:Signature" minOccurs="0"/>
1077     <element ref="md:Extensions" minOccurs="0"/>
1078     <element ref="md:AffiliateMember" maxOccurs="unbounded"/>
1079     <element ref="md:KeyDescriptor" minOccurs="0" maxOccurs="unbounded"/>
1080   </sequence>
1081   <attribute name="affiliationOwnerID" type="md:entityIDType"
1082 use="required"/>
1083   <attribute name="validUntil" type="dateTime" use="optional"/>
1084   <attribute name="cacheDuration" type="duration" use="optional"/>
1085   <attribute name="ID" type="ID" use="optional"/>
1086   <anyAttribute namespace="##other" processContents="lax"/>
1087 </complexType>
1088 <element name="AffiliateMember" type="md:entityIDType"/>
```

## 2.6 Examples

1090 The following is an example of metadata for a SAML system entity acting as an identity provider and an  
 1091 attribute authority. A signature is shown as a placeholder, without the actual content.  
 1092

```

1093 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1094   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1095   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1096   entityID="https://IdentityProvider.com/SAML">
1097   <ds:Signature>...</ds:Signature>
1098   <IDPSSODescriptor WantAuthnRequestsSigned="true"
1099     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1100     <KeyDescriptor use="signing">
1101       <ds:KeyInfo>
1102         <ds:KeyName>IdentityProvider.com SSO Key</ds:KeyName>
1103       </ds:KeyInfo>
1104     </KeyDescriptor>
1105     <ArtifactResolutionService isDefault="true" index="0"
1106       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1107       Location="https://IdentityProvider.com/SAML/Artifact"/>
1108     <SingleLogoutService
1109       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1110       Location="https://IdentityProvider.com/SAML/SLO/SOAP"/>
1111     <SingleLogoutService
1112       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1113       Location="https://IdentityProvider.com/SAML/SLO/Browser"
1114       ResponseLocation="https://IdentityProvider.com/SAML/SLO/Response"/>
1115     <NameIDFormat>
1116       urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1117     </NameIDFormat>
1118     <NameIDFormat>
1119       urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1120     </NameIDFormat>
1121     <NameIDFormat>
1122       urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1123     </NameIDFormat>
1124     <SingleSignOnService
1125       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1126       Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1127     <SingleSignOnService
1128       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1129       Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1130     <saml:Attribute
1131       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1132       Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1133       FriendlyName="eduPersonPrincipalName">
1134     </saml:Attribute>
1135     <saml:Attribute
1136       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1137       Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1138       FriendlyName="eduPersonAffiliation">
1139       <saml:AttributeValue>member</saml:AttributeValue>
1140       <saml:AttributeValue>student</saml:AttributeValue>
1141       <saml:AttributeValue>faculty</saml:AttributeValue>
1142       <saml:AttributeValue>employee</saml:AttributeValue>
1143       <saml:AttributeValue>staff</saml:AttributeValue>
1144     </saml:Attribute>
1145   </IDPSSODescriptor>
1146   <AttributeAuthorityDescriptor
1147     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1148     <KeyDescriptor use="signing">
1149       <ds:KeyInfo>
1150         <ds:KeyName>IdentityProvider.com AA Key</ds:KeyName>
1151       </ds:KeyInfo>

```

```

1152     </KeyDescriptor>
1153     <AttributeService
1154         Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1155         Location="https://IdentityProvider.com/SAML/AA/SOAP"/>
1156     <AssertionIDRequestService
1157         Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
1158         Location="https://IdentityProvider.com/SAML/AA/URI"/>
1159     <NameIDFormat>
1160         urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1161     </NameIDFormat>
1162     <NameIDFormat>
1163         urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1164     </NameIDFormat>
1165     <NameIDFormat>
1166         urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1167     </NameIDFormat>
1168     <saml:Attribute
1169         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1170         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1171         FriendlyName="eduPersonPrincipalName">
1172     </saml:Attribute>
1173     <saml:Attribute
1174         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1175         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1176         FriendlyName="eduPersonAffiliation">
1177         <saml:AttributeValue>member</saml:AttributeValue>
1178         <saml:AttributeValue>student</saml:AttributeValue>
1179         <saml:AttributeValue>faculty</saml:AttributeValue>
1180         <saml:AttributeValue>employee</saml:AttributeValue>
1181         <saml:AttributeValue>staff</saml:AttributeValue>
1182     </saml:Attribute>
1183     </AttributeAuthorityDescriptor>
1184     <Organization>
1185         <OrganizationName xml:lang="en">Identity Providers R
1186 US</OrganizationName>
1187         <OrganizationDisplayName xml:lang="en">
1188             Identity Providers R US, a Division of Lerxst Corp.
1189         </OrganizationDisplayName>
1190         <OrganizationURL
1191 xml:lang="en">https://IdentityProvider.com</OrganizationURL>
1192         </Organization>
1193     </EntityDescriptor>
1194

```

1195 The following is an example of metadata for a SAML system entity acting as a service provider. A  
1196 signature is shown as a placeholder, without the actual content. For illustrative purposes, the service is  
1197 one that does not require users to uniquely identify themselves, but rather authorizes access on the basis  
1198 of a role-like attribute.  
1199

```

1200 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1201     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1202     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1203     entityID="https://ServiceProvider.com/SAML">
1204     <ds:Signature>...</ds:Signature>
1205     <SPSSODescriptor AuthnRequestsSigned="true"
1206         protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1207         <KeyDescriptor use="signing">
1208             <ds:KeyInfo>
1209                 <ds:KeyName>ServiceProvider.com SSO Key</ds:KeyName>
1210             </ds:KeyInfo>
1211         </KeyDescriptor>
1212         <KeyDescriptor use="encryption">
1213             <ds:KeyInfo>
1214                 <ds:KeyName>ServiceProvider.com Encrypt Key</ds:KeyName>
1215             </ds:KeyInfo>

```

```

1216         <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
1217 1_5"/>
1218     </KeyDescriptor>
1219     <SingleLogoutService
1220         Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1221         Location="https://ServiceProvider.com/SAML/SLO/SOAP"/>
1222     <SingleLogoutService
1223         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1224         Location="https://ServiceProvider.com/SAML/SLO/Browser"
1225         ResponseLocation="https://ServiceProvider.com/SAML/SLO/Response"/>
1226     <NameIDFormat>
1227         urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1228     </NameIDFormat>
1229     <AssertionConsumerService isDefault="true" index="0"
1230         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
1231         Location="https://ServiceProvider.com/SAML/SSO/Artifact"/>
1232     <AssertionConsumerService index="1"
1233         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1234         Location="https://ServiceProvider.com/SAML/SSO/POST"/>
1235     <AttributeConsumingService index="0">
1236         <ServiceName xml:lang="en">Academic Journals R US</ServiceName>
1237         <RequestedAttribute
1238             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1239             Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7"
1240             FriendlyName="eduPersonEntitlement">
1241             <saml:AttributeValue>
1242                 https://ServiceProvider.com/entitlements/123456789
1243             </saml:AttributeValue>
1244         </RequestedAttribute>
1245     </AttributeConsumingService>
1246 </SPSSODescriptor>
1247 <Organization>
1248     <OrganizationName xml:lang="en">Academic Journals R
1249 US</OrganizationName>
1250     <OrganizationDisplayName xml:lang="en">
1251         Academic Journals R US, a Division of Dirk Corp.
1252     </OrganizationDisplayName>
1253     <OrganizationURL
1254 xml:lang="en">https://ServiceProvider.com</OrganizationURL>
1255     </Organization>
1256 </EntityDescriptor>

```

---

## 1257 3 Signature Processing

1258 Various elements in a metadata instance can be digitally signed (as indicated by the element's inclusion  
1259 of a `<ds:Signature>` element), with the following benefits:

- 1260 • Metadata integrity
- 1261 • Authentication of the metadata by a trusted signer

1262 A digital signature is not always required, for example if the relying party obtains the information directly  
1263 from the publishing entity directly (with no intermediaries) through a secure channel, with the entity having  
1264 authenticated to the relying party by some means other than a digital signature.

1265 Many different techniques are available for "direct" authentication and secure channel establishment  
1266 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,  
1267 the applicable security requirements depend on the communicating applications.

1268 Additionally, elements can inherit signatures on enclosing parent elements that are themselves signed.

1269 In the absence of such context, it is RECOMMENDED that at least the root element of a metadata  
1270 instance be signed.

### 1271 3.1 XML Signature Profile

1272 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility  
1273 and many choices. This section details the constraints on these facilities so that metadata processors do  
1274 not have to deal with the full generality of XML Signature processing. This usage makes specific use of  
1275 the `xs:ID`-typed attributes optionally present on the elements to which signatures can apply. These  
1276 attributes are collectively referred to in this section as the identifier attributes.

#### 1277 3.1.1 Signing Formats and Algorithms

1278 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and  
1279 detached.

1280 SAML metadata MUST use enveloped signatures when signing the elements defined in this specification.  
1281 [E81] Any algorithm defined for use with the XML Signature specification MAY be used.

#### 1282 3.1.2 References

1283 Signed metadata elements MUST supply a value for the identifier attribute on the signed element. The  
1284 element may or may not be the root element of the actual XML document containing the signed metadata  
1285 element.

1286 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute  
1287 value of the metadata element being signed. For example, if the identifier attribute value is "foo", then the  
1288 URI attribute in the `<ds:Reference>` element MUST be "#foo".

1289 As a consequence, a metadata element's signature MUST apply to the content of the signed element and  
1290 any child elements it contains.

#### 1291 3.1.3 Canonicalization Method

1292 SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the  
1293 `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a `<ds:Transform>`

1294 algorithm. [E83] Use of Exclusive Canonicalization facilitates the verification of signatures created over  
1295 SAML messages when placed into a different XML context than present during signing.

1296 Note that use of this algorithm alone does not guarantee that a particular signed object can be moved  
1297 from one context to another safely, nor is that a requirement of signed SAML objects in general, though it  
1298 MAY be required by particular profiles.

### 1299 **3.1.4 Transforms**

1300 Signatures in SAML metadata SHOULD NOT contain transforms other than the enveloped signature  
1301 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive  
1302 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or  
1303 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

1304 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do  
1305 not, verifiers MUST ensure that no content of the signed metadata element is excluded from the  
1306 signature. This can be accomplished by establishing out-of-band agreement as to what transforms are  
1307 acceptable, or by applying the transforms manually to the content and reverifying the result as consisting  
1308 of the same SAML metadata.

### 1309 **3.1.5 [E91] Object**

1310 The `<ds:Object>` element is not defined for use with SAML metadata signatures, and SHOULD NOT be  
1311 present. Since it can be used in service of an attacker by carrying unsigned data, verifiers SHOULD reject  
1312 signatures that contain a `<ds:Object>` element.

### 1313 **3.1.6 KeyInfo**

1314 XML Signature [XMLSig] defines usage of the `<ds:KeyInfo>` element. SAML does not require the  
1315 use of `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY  
1316 be absent.

---

## 1317 4 Metadata Publication and Resolution

1318 Two mechanisms are provided for an entity to publish (and for a consumer to resolve the location of)  
1319 metadata documents: via a "well-known-location" by directly dereferencing the entity's unique identifier (a  
1320 URI variously referred to as an *entityID* or *providerID*), or indirectly by publishing the location of metadata  
1321 in the DNS. Other out-of-band mechanisms are of course also permitted. A consumer that supports both  
1322 approaches defined in this document MUST attempt resolution via DNS before using the "well-known-  
1323 location" mechanism.

1324 When retrieval requires network transport of the document, the transport SHOULD be protected with  
1325 mechanisms providing server authentication and integrity protection. For example, HTTP-based  
1326 resolution SHOULD be protected with TLS/SSL [RFC2246] as amended by [RFC3546].

1327 Various mechanisms are described in this section to aid in establishing trust in the accuracy and  
1328 legitimacy of metadata, including use of XML signatures, SSL/TLS server authentication, and DNS  
1329 signatures. Regardless of the mechanism(s) used, relying parties SHOULD have some means by which  
1330 to establish trust in metadata information before relying on it.

### 1331 4.1 Publication and Resolution via Well-Known Location

1332 The following sections describe publication and resolution of metadata by means of a well-known  
1333 location.

#### 1334 4.1.1 Publication

1335 Entities MAY publish their metadata documents at a well known location by placing the document at the  
1336 location denoted by its unique identifier, which MUST be in the form of a URL (rather than a URN). See  
1337 Section 8.3.6 of [SAMLCore] for more information about such identifiers. It is STRONGLY  
1338 RECOMMENDED that `https` URLs be used for this purpose. An indirection mechanism supported by the  
1339 URL scheme (such as an HTTP 1.1 302 redirect) MAY be used if the document is not placed directly at  
1340 the location. If the publishing protocol permits MIME-based identification of content types, the content  
1341 type of the metadata instance MUST be `application/samlmetadata+xml`.

1342 The XML document provided at the well-known location MUST describe the metadata only for the entity  
1343 represented by the unique identifier (that is, the root element MUST be an `<EntityDescriptor>` with  
1344 an `entityID` matching the location). If other entities need to be described, the  
1345 `<AdditionalMetadataLocation>` element MUST be used. Thus the `<EntitiesDescriptor>`  
1346 element MUST NOT be used in documents published using this mechanism, since a group of entities are  
1347 not defined by such an identifier.

#### 1348 4.1.2 Resolution

1349 If an entity's unique identifier is a URL, metadata consumers MAY attempt to resolve an entity's unique  
1350 identifier directly, in a scheme-specific manner, by dereferencing the identifier.

### 1351 4.2 Publishing and Resolution via DNS

1352 To improve the accessibility of metadata documents and provide additional indirection between an entity's  
1353 unique identifier and the location of metadata, entities MAY publish their metadata document locations in  
1354 a zone of their corresponding DNS [RFC1034]. The entity's unique identifier (a URI) is used as the input  
1355 to the process. Since URIs are flexible identifiers, location publication methods and the resolution process  
1356 are determined by the URI's scheme and fully-qualified name. URI locations for metadata are

1357 subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in [RFC2915]  
1358 and [RFC3403].

1359 It is RECOMMENDED that entities publish their resource records in signed zone files using [E66]  
1360 [RFC4035] such that relying parties may establish the validity of the published location and authority of  
1361 the zone, and integrity of the DNS response. If DNS zone signatures are present, relying parties MUST  
1362 properly validate the signature.

## 1363 **4.2.1 Publication**

1364 This specification makes use of the NAPTR resource record described in [RFC2915] and [RFC3403].  
1365 Familiarity with these documents is encouraged.

1366 Dynamic Delegation Discovery System (DDDS) [RFC3401] is a general purpose system for the retrieval of  
1367 information based on an application-specific input string and the application of well known rules to  
1368 transform that string until a terminal condition is reached requiring a look-up into an application-specific  
1369 defined database or resolution of a URL based on the rules defined by the application. DDDS defines a  
1370 specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS  
1371 necessary to apply DDDS rules.

1372 Entities MAY publish separate URLs when multiple metadata documents need to be distributed, or when  
1373 different metadata documents are required due to multiple trust relationships that require separate keying  
1374 material, or when service interfaces require separate metadata declarations. This may be accomplished  
1375 through the use of the optional `<AdditionalMetadataLocation>` element, or through the regexp  
1376 facility and multiple service definition fields in the NAPTR resource record itself.

1377 If the publishing protocol permits MIME-based identification of content types, the content type of the  
1378 metadata instance MUST be `application/samlmetadata+xml`.

1379 If the entity's unique identifier is a URN, publication of the corresponding metadata location proceeds as  
1380 specified in [RFC3404]. Otherwise, the resolution of the metadata location proceeds as specified below.

1381 The following is the application-specific profile of DDDS for SAML metadata resolution.

### 1382 **4.2.1.1 First Well Known Rule**

1383 The "first well-known-rule" for processing SAML metadata resolution is to parse the entity's unique  
1384 identifier and extract the fully-qualified domain name (subexpression 3) as described in Section 4.2.3.1.

### 1385 **4.2.1.2 The Order Field**

1386 The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY  
1387 provide multiple NAPTR resource records which MUST be processed by the resolver application in the  
1388 order indicated by this field.

### 1389 **4.2.1.3 The Preference Field**

1390 For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving  
1391 application. The resolving application MAY ignore this order, in cases where the service field value does  
1392 not meet the resolver's requirements (e.g.: the resource record returns a protocol the application does not  
1393 support).

#### 1394 4.2.1.4 The Flag Field

1395 SAML metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying  
1396 additional resource records are to be processed). The "U" flag indicates that the output of the rule is a  
1397 URI.

#### 1398 4.2.1.5 The Service Field

1399 The SAML-specific service field, as described in the following BNF, declares the modes by which instance  
1400 document(s) shall be made available:

```
1401 servicefield = 1("PID2U" / "NID2U") "+" proto [*(":" class) *(":" servicetype)]  
1402 proto = 1("https" / "uddi")  
1403 class = 1[ "entity" / "entitygroup" ]  
1404 servicetype = 1(si / "spsso" / "idpsso" / "authn" / "authnauth" / "pdp" / "attrauth" /  
1405 alphanum )  
1406 si = "si" [ ":" alphanum ] [ ":" endpoint ]  
1407 alphanum = 1*32 (ALPHA / DIGIT)
```

1408 where:

- 1409 • servicefield PID2U resolves an entity's unique identifier to metadata URL.
- 1410 • servicefield NID2U resolves a principal's <NameID> into a metadata URL.
- 1411 • proto describes the retrieval protocol (https or uddi). In the case of UDDI, the URL will be an  
1412 http(s) URL referencing a WSDL document.
- 1413 • class identifies whether the referenced metadata document describes a single entity, or multiple.  
1414 In the latter case, the referenced document MUST contain the entity defined by the original unique  
1415 identifier as a member of a group of entities within the document itself such as an  
1416 <AffiliationDescriptor> or <EntitiesDescriptor>.
- 1417 • servicetype allows an entity to publish metadata for distinct roles and services as separate  
1418 documents. Resolvers who encounter multiple servicetype declarations will dereference the  
1419 appropriate URI, depending on which service is required for an operation (e.g.: an entity operating  
1420 both as an identity provider and a service provider can publish metadata for each role at different  
1421 locations). The authn service type represents a <SingleSignOnService> endpoint.
- 1422 • si (with optional endpoint component) allows the publisher to either directly publish the metadata  
1423 for a service instance, or by articulating a SOAP endpoint (using endpoint).

1424 For example:

- 1425 • PID2U+https:entity - represents the entity's complete metadata document available via the  
1426 https protocol
- 1427 • PID2U+uddi:entity:si:foo - represents the WSDL document location that describes a service  
1428 instance "foo"
- 1429 • PID2U+https:entitygroup:idpsso - represents the metadata for a group of entities acting as  
1430 SSO identity providers, of which the original entity is a member.
- 1431 • NID2U+https:idp - represents the metadata for the SSO identity provider of a principal

#### 1432 4.2.1.6 The Regex and Replacement Fields

1433 The expected output after processing the input string through the regex MUST be a valid https URL or  
1434 UDDI node (WSDL document) address.

## 1435 4.2.2 NAPTR Examples

### 1436 4.2.2.1 Entity Metadata NAPTR Examples

1437 Entities publish metadata URLs in the following manner:

```
1438 $ORIGIN provider.biz
1439
1440 ;; order pref f service regexp or replacement
1441
1442 IN NAPTR 100 10 "U" PID2U+https:entity
1443     "!^.*$!https://host.provider.biz/some/directory/trust.xml!" ""
1444 IN NAPTR 110 10 "U" PID2U+https: entity:trust
1445     "!^.*$!https://foo.provider.biz:1443/mdtrust.xml!" ""
1446 IN NAPTR 125 10 "U" PID2U+https:"
1447 IN NAPTR 110 10 "U" PID2U+uddi:entity
1448     "!^.*$!https://this.uddi.node.provider.biz/libmd.wsdl" ""
```

### 1449 4.2.2.2 Name Identifier Examples

1450 A principal's employer `example.int` operates an identity provider which may be used by an office  
1451 supply company to authenticate authorized buyers. The supplier takes a users' email address  
1452 `buyer@example.int` as input to the resolution process, and parses the email address to extract the  
1453 FQDN (`example.int`). The employer publishes the following NAPTR record in the `example.int` DNS:

```
1454 $ORIGIN example.int
1455
1456 IN NAPTR 100 10 "U" NID2U+https:authn
1457     "!^([\^@]+)@(.*)$!https://serv.example.int:8000/cgi-bin/getmd?\1!" ""
1458 IN NAPTR 100 10 "U" NID2U+https:idp
1459     "!^([\^@]+)@(.*)$!https://auth.example.int/app/auth?\1" ""
```

## 1460 4.2.3 Resolution

1461 When resolving metadata for an entity via the DNS, the unique identifier of the entity is used as the initial  
1462 input into the resolution process, rather than as an actual location Proceed as follows:

- 1463 • If the unique identifier is a URN, proceed with the resolution steps as defined in [RFC3404].
- 1464 • Otherwise, parse the identifier to obtain the fully-qualified domain name.
- 1465 • Query the DNS for NAPTR resource records of the domain iteratively until a terminal resource  
1466 record is returned.
- 1467 • Identify which resource record to use based on the service fields, then order fields, then preference  
1468 fields of the result set.
- 1469 • Obtain the document(s) at the provided location(s) as required by the application.

### 1470 4.2.3.1 Parsing the Unique Identifier

1471 To initiate the resolution of the location of the metadata information, it will be necessary in some cases to  
1472 decompose the entity's unique identifier (expressed as a URI) into one or more atomic elements.

1473 The following regular expression should be used when initiating the decomposition process:

```
1474 ^([\^:/?#]+)?/*([\^:/?#]*@)?((([\^/?:#]*\.)?((([\^/?#:\.]+)\.([\^/?#:\.]+)))
1475 (: \d+)?([\^?#]*)(\?[\^#]*)?(\#.*)?$
1476     1           2           3         4           5           6           7           8
1477     9           10          11
```

1478 Subexpression 3 MUST result in a Fully-Qualified Domain Name (FQDN), which will be the basis for  
1479 retrieving metadata locations from this zone.

#### 1480 **4.2.3.2 Obtaining Metadata via the DNS**

1481 Upon completion of the parsing of the identifier, the application then performs a DNS query for the  
1482 resulting domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses.  
1483 Applications MAY exclude from the result set any service definitions that do not concern the present  
1484 request operations.

1485 Resolving applications MUST subsequently order the result set according to the order field, and MAY  
1486 order the result set based on the preference set. Resolvers are NOT REQUIRED to follow the ordering of  
1487 the preferences field. The resulting NAPTR resource record(s) are operated on iteratively (based on the  
1488 order flag) until a terminal NAPTR resource record is reached.

1489 The result will be a well-formed, absolute URL, which is then used to retrieve the metadata document.

#### 1490 **4.2.4 Metadata Location Caching**

1491 Location caching MUST NOT exceed the TTL of the DNS zone from which the location was derived.  
1492 Resolvers MUST obtain a fresh copy of the metadata location upon reaching the expiration of the TTL of  
1493 the zone.

1494 Publishers of metadata documents should carefully consider the TTL of the zone when making changes  
1495 to metadata document locations. Should such a location change occur, a publisher MUST either keep the  
1496 document at both the old and new location until all conforming resolvers are certain to have the updated  
1497 location (e.g.: time of zone change + TTL), or provide an HTTP Redirect [RFC2616] response at the old  
1498 location specifying the new location.

### 1499 **4.3 Post-Processing of Metadata**

1500 The following sections describe the post-processing of metadata.

#### 1501 **4.3.1 Metadata Instance Caching**

1502 [E94] Document caching MUST be based on the duration indicated by the `cacheDuration` attribute of  
1503 the subject element(s). If metadata elements have parent elements which contain caching policies, the  
1504 parent element takes precedence. To properly process the `cacheDuration` attribute, consumers must  
1505 retain the date and time when an instance was obtained.

1506 Note that cache expiration does not imply a lack of validity in the absence of a `validUntil` attribute or  
1507 other information; failure to update a cached instance (e.g., due to network failure) need not render  
1508 metadata invalid, although implementations may offer such controls to deployers.

1509 When a document or element has expired, the consumer MUST retrieve a fresh copy, which may require  
1510 a refresh of the document location(s). Consumers SHOULD process document cache processing  
1511 according to [RFC2616] Section 13, and MAY request the Last-Modified date and time from the HTTP  
1512 server. Publishers SHOULD ensure acceptable cache processing as described in [RFC2616] (Section  
1513 10.3.5 304 Not Modified).

#### 1515 **4.3.2 [E94] Metadata Instance Validity**

1516 Metadata MUST be considered invalid upon reaching the time specified in a `validUntil` attribute of the  
1517 subject element(s). The effective expiration may be adjusted downward by parent element(s) with earlier  
1518 expirations. Invalid metadata MUST NOT be used. This contrasts with "stale" metadata that may be  
1519 beyond its optimum cache duration but is not explicitly invalid. Such metadata remains valid and MAY be  
1520 used at the discretion of the implementation.

### 1521 **4.3.3 Handling of HTTPS Redirects**

1522 Publishers MAY issue an HTTP Redirect (301 Moved Permanently, 302 or 307 Temporary Redirect)  
1523 [RFC2616], and user agents MUST follow the specified URL in the Redirect response. Redirects  
1524 SHOULD be of the same protocol as the initial request.

### 1525 **4.3.4 Processing of XML Signatures and General Trust Processing**

1526 Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and  
1527 for the trust ascribed to the entity described by such metadata:

- 1528 • Trust derived from the signature of the DNS zone from which the metadata location URL was  
1529 resolved, ensuring accuracy of the metadata document location(s)
- 1530 • Trust derived from signature processing of the metadata document itself, ensuring the integrity of  
1531 the XML document
- 1532 • Trust derived from the SSL/TLS server authentication of the metadata location URL, ensuring the  
1533 identity of the publisher of the metadata

1534 Post-processing of the metadata document MUST include signature processing at the XML-document  
1535 level and MAY include one of the other two processes. Specifically, the relying party MAY choose to trust  
1536 any of the cited authorities in the resolution and parsing process. Publishers of metadata MUST employ a  
1537 document-integrity mechanism and MAY employ any of the other two processing profiles to establish trust  
1538 in the metadata document, governed by implementation policies.

#### 1539 **4.3.4.1 Processing Signed DNS Zones**

1540 Verification of DNS zone signature SHOULD be processed, if present, as described in [E66][RFC4035].

#### 1541 **4.3.4.2 Processing Signed Documents and Fragments**

1542 Published metadata documents SHOULD be signed, as described in Section 3, either by a certificate  
1543 issued to the subject of the document, or by another trusted party. Publishers MAY consider signatures of  
1544 other parties as a means of trust conveyance.

1545 Metadata consumers MUST validate signatures, when present, on the metadata document as described  
1546 by Section 3.

#### 1547 **4.3.4.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL**

1548 It is STRONGLY RECOMMENDED that publishers implement TLS/SSL URLs; therefore, consumers  
1549 SHOULD consider the trust inherited from the issuer of the TLS/SSL certificate. Publication URLs may not  
1550 always be located in the domain of the subject of the metadata document; therefore, consumers  
1551 SHOULD NOT presume certificates whose subject is the entity in question, as it may be hosted by  
1552 another trusted party.

1553 As the basis of this trust may not be available against a cached document, other mechanisms SHOULD  
1554 be used under such circumstances.

---

## 5 References

1555

- 1556 [RFC1034] P. Mockapetris. *Domain Names – Concepts and Facilities*. IETF RFC 1034,  
1557 November 1987. See <http://www.ietf.org/rfc/rfc1034.txt>.
- 1558 [RFC2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*,  
1559 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1560 [RFC2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999.  
1561 See <http://www.ietf.org/rfc/rfc2246.txt>.
- 1562 [E66][RFC2616] R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. IETF RFC 2616, June  
1563 1999. See <http://www.ietf.org/rfc/rfc2616.txt>.
- 1564 [RFC2915] M. Mealling. *The Naming Authority Pointer (NAPTR) DNS Resource Record*.  
1565 IETF RFC 2915, September 2000. See <http://www.ietf.org/rfc/rfc2915.txt>.
- 1566 [RFC3401] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part One: The  
1567 Comprehensive DDDS*. IETF RFC 3401, October 2002. See  
1568 <http://www.ietf.org/rfc/rfc3401.txt>.
- 1569 [RFC3403] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Three: The  
1570 Domain Name System (DNS) Database*. IETF RFC 3403, October 2002. See  
1571 <http://www.ietf.org/rfc/rfc3403.txt>.
- 1572 [RFC3404] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Four: The  
1573 Uniform Resource Identifiers (URI) Resolution Application*. IETF RFC 3404,  
1574 October 2002. See <http://www.ietf.org/rfc/rfc3404.txt>.
- 1575 [RFC3546] S. Blake-Wilson et al. *Transport Layer Security (TLS) Extensions*. IETF RFC  
1576 3546, June 2003. See <http://www.ietf.org/rfc/rfc3546.txt>.
- 1577 [E66][RFC4035] R. Arends et al. *Protocol Modifications for the DNS Security Extensions*. IETF  
1578 RFC 4035, March 2005. See <http://www.ietf.org/rfc/rfc4035.txt>.
- 1579 [SAMLBind] S. Cantor et al. *Bindings for the OASIS Security Assertion Markup Language  
1580 (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os.  
1581 See <http://www.oasis-open.org/committees/security/>.
- 1582 [SAMLConform] P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion  
1583 Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-  
1584 conformance-2.0-os. <http://www.oasis-open.org/committees/security/>.
- 1585 [SAMLCore] S. Cantor et al. *Assertions and Protocols for the OASIS Security Assertion  
1586 Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-  
1587 core-2.0-os. See <http://www.oasis-open.org/committees/security/>.
- 1588 [SAMLMeta-xsd] S. Cantor et al. *SAML metadata schema*. OASIS SSTC, March 2005. Document  
1589 ID saml-schema-metadata-2.0. See [http://www.oasis-  
1590 open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1591 [SAMLProf] S. Cantor et al. *Profiles for the OASIS Security Assertion Markup Language  
1592 (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-profiles-2.0-os. See  
1593 <http://www.oasis-open.org/committees/security/>.
- 1594 [SAMLSec] F. Hirsch et al. *Security and Privacy Considerations for the OASIS Security  
1595 Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005.  
1596 Document ID saml-sec-consider-2.0-os. See [http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>1597 open.org/committees/security/).
- 1598 [Schema1] H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web  
1599 Consortium Recommendation, May 2001. See [http://www.w3.org/TR/xmlschema-  
1600 1/](http://www.w3.org/TR/xmlschema-1/). Note that this specification normatively references [Schema2], listed below.
- 1601 [Schema2] P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium  
1602 Recommendation, May 2001. See <http://www.w3.org/TR/xmlschema->

1603	<b>[XMLEnc]</b>	D. Eastlake et al. <i>XML-Encryption Syntax and Processing</i> , <a href="http://www.w3.org/TR/xmlenc-core/">http://www.w3.org/TR/xmlenc-core/</a> , World Wide Web Consortium.
1604		
1605	<b>[XMLSig]</b>	D. Eastlake et al. <i>XML-Signature Syntax and Processing, [E74] Second Edition</i> . World Wide Web Consortium, June 2008. See <a href="http://www.w3.org/TR/xmldsig-core/">http://www.w3.org/TR/xmldsig-</a> core/.
1606		
1607		

---

## 1608 Appendix A.Registration of MIME media type 1609 application/samlmetadata+xml

### 1610 Introduction

1611 This document defines a MIME media type -- `application/samlmetadata+xml` -- for use  
1612 with the XML serialization of Security Assertion Markup Language metadata.  
1613

1614 SAML is a work product of the OASIS Security Services Technical Committee [SSTC]. The SAML  
1615 specifications define XML-based constructs with which one may make, and convey, security  
1616 assertions. Using SAML, one can assert that an authentication event pertaining to some subject  
1617 has occurred and convey said assertion to a relying party, for example.  
1618

1619 SAML profiles require agreements between system entities regarding identifiers, binding support,  
1620 endpoints, certificates, keys, and so forth. Such information is treated as metadata by SAML v2.0.  
1621 [SAMLv2Meta] specifies this metadata, as well as specifying metadata publication and resolution  
1622 mechanisms. If the publishing protocol permits MIME-based identification of content types, then  
1623 use of the `application/samlmetadata+xml` MIME media type is required.

### 1624 MIME media type name

1625 `application`

### 1626 MIME subtype name

1627 `samlmetadata+xml`

### 1628 Required parameters

1629 None

### 1630 Optional parameters

1631 `charset`

1632 Same as `charset` parameter of `application/xml` [RFC3023].

### 1633 Encoding considerations

1634 Same as for `application/xml` [RFC3023].

### 1635 Security considerations

1636 Per their specification, `samlmetadata+xml` typed objects do not contain executable content.  
1637 However, these objects are XML-based [XML], and thus they have all of the general security  
1638 considerations presented in Section 10 of [RFC3023].

1639 SAML metadata [SAMLv2Meta] contains information whose integrity and authenticity is important  
1640 – identity provider and service provider public keys and endpoint addresses, for example.

1641 To counter potential issues, the publisher may sign `samlmetadata+xml` typed objects. Any such  
1642 signature should be verified by the recipient of the data - both as a valid signature, and as being  
1643 the signature of the publisher.

1644 Additionally, various of the publication protocols, e.g. HTTP-over-TLS/SSL, offer means for  
1645 ensuring the authenticity of the publishing party and for protecting the metadata in transit.

1646 [SAMLv2Meta] also defines prescriptive metadata caching directives, as well as guidance on  
1647 handling HTTPS redirects, trust processing, server authentication, and related items.  
1648 For a more detailed discussion of SAML v2.0 metadata and its security considerations, please  
1649 see [SAMLv2Meta]. For a discussion of overall SAML v2.0 security considerations and specific  
1650 security-related design features, please refer to the SAML v2.0 specifications listed in the below  
1651 bibliography. The specifications containing security-specific information are explicitly listed.

## 1652 **Interoperability considerations**

1653 SAML v2.0 metadata explicitly supports identifying the protocols and versions supported by the  
1654 identified entities. For example, an identity provider entity can be denoted as supporting SAML  
1655 v2.0 [SAMLv2.0], SAML v1.1 [SAMLv1.1], Liberty ID-FF 1.2 [LAPFF], or even other protocols if  
1656 they are unambiguously identifiable via URI [RFC2396]. This protocol support information is  
1657 conveyed via the `protocolSupportEnumeration` attribute of metadata objects of the  
1658 **RoleDescriptorType**.

## 1659 **Published specification**

1660 [SAMLv2Meta] explicitly specifies use of the `application/samlmetadata+xml` MIME media  
1661 type.

## 1662 **Applications which use this media type**

1663 Potentially any application implementing SAML v2.0, as well as those applications implementing  
1664 specifications based on SAML, e.g. those available from the Liberty Alliance [LAP].

## 1665 **Additional information**

### 1666 **Magic number(s)**

1667 In general, the same as for `application/xml` [RFC3023]. In particular, the XML root element of  
1668 the returned object will have a namespace-qualified name with:  
1669

1670 – a local name of: `EntityDescriptor`, or  
1671 `AffiliationDescriptor`, or  
1672 `EntitiesDescriptor`

1674 – a namespace URI of: `urn:oasis:names:tc:SAML:2.0:metadata`  
1675 (the SAMLv2.0 metadata namespace)

### 1676 **File extension(s)**

1677 None

### 1678 **Macintosh File Type Code(s)**

1679 None

## 1680 **Person & email address to contact for further information**

1681 This registration is made on behalf of the OASIS Security Services Technical Committee (SSTC)  
1682 Please refer to the SSTC website for current information on committee chairperson(s) and their  
1683 contact addresses: <http://www.oasis-open.org/committees/security/>. Committee members should  
1684 submit comments and potential errata to the [securityservices@lists.oasis-open.org](mailto:securityservices@lists.oasis-open.org) list. Others  
1685 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)  
1686 [open.org/committees/comments/form.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security).

1687  
1688 Additionally, the SAML developer community email distribution list, [saml-dev@lists.oasis-](mailto:saml-dev@lists.oasis-open.org)  
1689 [open.org](http://lists.oasis-open.org), may be employed to discuss usage of the `application/samlmetadata+xml` MIME  
1690 media type. The "saml-dev" mailing list is publicly archived here: [http://lists.oasis-](http://lists.oasis-open.org/archives/saml-dev/)  
1691 [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  
1692 subscribe, send a message with the single word "subscribe" in the message body, to: [saml-dev-](mailto:saml-dev-request@lists.oasis-open.org)  
1693 [request@lists.oasis-open.org](mailto:saml-dev-request@lists.oasis-open.org).

## 1694 Intended usage

1695 COMMON

## 1696 Author/Change controller

1697 The SAML specification sets are a work product of the OASIS Security Services Technical  
1698 Committee (SSTC). OASIS and the SSTC have change control over the SAML specification sets.

## 1699 Bibliography

- 1700 [LAP] “*Liberty Alliance Project*”. See <http://www.projectliberty.org/>
- 1701 [LAPFF] “Liberty Alliance Project: Federation Framework”. See  
1702 <http://www.projectliberty.org/resources/specifications.php#box1>
- 1703 [OASIS] “*Organization for the Advancement of Structured Information Systems*”.  
1704 See <http://www.oasis-open.org/>
- 1705 [RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifiers*  
1706 *(URI): Generic Syntax*. IETF RFC 2396, August 1998. Available at  
1707 <http://www.ietf.org/rfc/rfc2396.txt>
- 1708 [RFC3023] M. Murata, S. St.Laurent, D. Kohn, “*XML Media Types*”, IETF Request  
1709 for Comments 3023, January 2001. Available as [http://www.rfc-](http://www.rfc-editor.org/rfc/rfc3023.txt)  
1710 [editor.org/rfc/rfc3023.txt](http://www.rfc-editor.org/rfc/rfc3023.txt)
- 1711 [SAMLv1.1] OASIS Security Services Technical Committee, “*Security Assertion*  
1712 *Markup Language (SAML) Version 1.1 Specification Set*”. OASIS  
1713 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)  
1714 [open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-](http://www.oasis-open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip)  
1715 [xsd.zip](http://www.oasis-open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip)
- 1716 [SAMLv2.0] OASIS Security Services Technical Committee, “*Security Assertion*  
1717 *Markup Language (SAML) Version 2.0 Specification Set*”. OASIS  
1718 Standard, 15-Mar-2005. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip)  
1719 [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)
- 1720 [SAMLv2Bind] S. Cantor et al., “*Bindings for the OASIS Security Assertion Markup*  
1721 *Language (SAML) V2.0*”. OASIS, March 2005. Document ID `saml-`  
1722 `bindings-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf)  
1723 [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)
- 1724 [SAMLv2Core] S. Cantor et al., “*Assertions and Protocols for the OASIS Security*  
1725 *Assertion Markup Language (SAML) V2.0*”. OASIS, March 2005.  
1726 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)  
1727 [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)
- 1728 [SAMLv2Meta] S. Cantor et al., “*Metadata for the OASIS Security Assertion Markup*  
1729 *Language (SAML) V2.0*”. OASIS SSTC, August 2004. Document ID `saml-`  
1730 `metadata-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf)  
1731 [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)
- 1732 [SAMLv2Prof] S. Cantor et al., “*Profiles for the OASIS Security Assertion Markup*  
1733 *Language (SAML) V2.0*”. OASIS, March 2005. Document ID `saml-`  
1734 `profiles-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf)  
1735 [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)

1736	[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:
1737		
1738		
1739		<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>
1740	[SSTC]	"OASIS Security Services Technical Committee". See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a>
1741		
1742	[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
1743		as <a href="http://www.w3.org/TR/REC-xml/">http://www.w3.org/TR/REC-xml/</a>
1744		
1745		
1746		

---

## 1747 Appendix B. Acknowledgments

1748 The editors would like to acknowledge the contributions of the OASIS Security Services Technical  
1749 Committee, whose voting members at the time of publication were:

- 1750 • Conor Cahill, AOL
- 1751 • John Hughes, Atos Origin
- 1752 • Hal Lockhart, BEA Systems
- 1753 • Mike Beach, Boeing
- 1754 • Rebekah Metz, Booz Allen Hamilton
- 1755 • Rick Randall, Booz Allen Hamilton
- 1756 • Ronald Jacobson, Computer Associates
- 1757 • Gavenraj Sodhi, Computer Associates
- 1758 • Thomas Wisniewski, Entrust
- 1759 • Carolina Canales-Valenzuela, Ericsson
- 1760 • Dana Kaufman, Forum Systems
- 1761 • Irving Reid, Hewlett-Packard
- 1762 • Guy Denton, IBM
- 1763 • Heather Hinton, IBM
- 1764 • Maryann Hondo, IBM
- 1765 • Michael McIntosh, IBM
- 1766 • Anthony Nadalin, IBM
- 1767 • Nick Ragouzis, Individual
- 1768 • Scott Cantor, Internet2
- 1769 • Bob Morgan, Internet2
- 1770 • Peter Davis, Neustar
- 1771 • Jeff Hodges, Neustar
- 1772 • Frederick Hirsch, Nokia
- 1773 • Senthil Sengodan, Nokia
- 1774 • Abbie Barbir, Nortel Networks
- 1775 • Scott Kiestler, Novell
- 1776 • Cameron Morris, Novell
- 1777 • Paul Madsen, NTT
- 1778 • Steve Anderson, OpenNetwork
- 1779 • Ari Kermaier, Oracle
- 1780 • Vamsi Motukuru, Oracle
- 1781 • Darren Platt, Ping Identity
- 1782 • Prateek Mishra, Principal Identity
- 1783 • Jim Lien, RSA Security
- 1784 • John Linn, RSA Security
- 1785 • Rob Philpott, RSA Security
- 1786 • Dipak Chopra, SAP
- 1787 • Jahan Moreh, Sigaba
- 1788 • Bhavna Bhatnagar, Sun Microsystems
- 1789 • Eve Maler, Sun Microsystems

- 1790 • Ronald Monzillo, Sun Microsystems
- 1791 • Emily Xu, Sun Microsystems
- 1792 • Greg Whitehead, Trustgenix
- 1793

1794 The editors also would like to acknowledge the following former SSTC members for their contributions to  
1795 this or previous versions of the OASIS Security Assertions Markup Language Standard:

- 1796 • Stephen Farrell, Baltimore Technologies
- 1797 • David Orchard, BEA Systems
- 1798 • Krishna Sankar, Cisco Systems
- 1799 • Zahid Ahmed, CommerceOne
- 1800 • Tim Alsop, CyberSafe Limited
- 1801 • Carlisle Adams, Entrust
- 1802 • Tim Moses, Entrust
- 1803 • Nigel Edwards, Hewlett-Packard
- 1804 • Joe Pato, Hewlett-Packard
- 1805 • Bob Blakley, IBM
- 1806 • Marlena Erdos, IBM
- 1807 • Marc Chanliau, Netegrity
- 1808 • Chris McLaren, Netegrity
- 1809 • Lynne Rosenthal, NIST
- 1810 • Mark Skall, NIST
- 1811 • Charles Knouse, Oblix
- 1812 • Simon Godik, Overxeer
- 1813 • Charles Norwood, SAIC
- 1814 • Evan Prodromou, Securant
- 1815 • Robert Griffin, RSA Security (former editor)
- 1816 • Sai Allarvarpu, Sun Microsystems
- 1817 • Gary Ellison, Sun Microsystems
- 1818 • Chris Ferris, Sun Microsystems
- 1819 • Mike Myers, Traceroute Security
- 1820 • Phillip Hallam-Baker, VeriSign (former editor)
- 1821 • James Vanderbeek, Vodafone
- 1822 • Mark O'Neill, Vordel
- 1823 • Tony Palmer, Vordel

1824  
1825 Finally, the editors wish to acknowledge the following people for their contributions of material used as  
1826 input to the OASIS Security Assertions Markup Language specifications:

- 1827 • Thomas Gross, IBM
- 1828 • Birgit Pfitzmann, IBM

1829 The editors also would like to gratefully acknowledge Jahan Moreh of Sigaba, who during his tenure on  
1830 the SSTC was the primary editor of the errata working document and who made major substantive  
1831 contributions to all of the errata materials.

---

1832 **Appendix C. Notices**

1833 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
1834 might be claimed to pertain to the implementation or use of the technology described in this document or  
1835 the extent to which any license under such rights might or might not be available; neither does it  
1836 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with  
1837 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights  
1838 made available for publication and any assurances of licenses to be made available, or the result of an  
1839 attempt made to obtain a general license or permission for the use of such proprietary rights by  
1840 implementors or users of this specification, can be obtained from the OASIS Executive Director.

1841 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,  
1842 or other proprietary rights which may cover technology that may be required to implement this  
1843 specification. Please address the information to the OASIS Executive Director.

1844 **Copyright © OASIS Open 2005. All Rights Reserved.**

1845 This document and translations of it may be copied and furnished to others, and derivative works that  
1846 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published  
1847 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice  
1848 and this paragraph are included on all such copies and derivative works. However, this document itself  
1849 may not be modified in any way, such as by removing the copyright notice or references to OASIS, except  
1850 as needed for the purpose of developing OASIS specifications, in which case the procedures for  
1851 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to  
1852 translate it into languages other than English.

1853 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
1854 or assigns.

1855 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
1856 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
1857 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR  
1858 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.