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

Working Draft 054, 812 September~~February~~December
2015907

Document identifier:

sstc-saml-metadata-errata-2.0-wd-0543

Location:

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

Editors:

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

46 **SAML V2.0 Contributors:**
47 Conor P. Cahill, AOL
48 John Hughes, Atos Origin
49 Hal Lockhart, BEA Systems
50 Michael Beach, Boeing
51 Rebekah Metz, Booz Allen Hamilton
52 Rick Randall, Booz Allen Hamilton
53 Thomas Wisniewski, Entrust
54 Irving Reid, Hewlett-Packard
55 Paula Austel, IBM
56 Maryann Hondo, IBM
57 Michael McIntosh, IBM
58 Tony Nadalin, IBM
59 Nick Ragouzis, Individual
60 Scott Cantor, Internet2
61 RL 'Bob' Morgan, Internet2
62 Peter C Davis, Neustar
63 Jeff Hodges, Neustar
64 Frederick Hirsch, Nokia
65 John Kemp, Nokia
66 Paul Madsen, NTT
67 Steve Anderson, OpenNetwork
68 Prateek Mishra, Principal Identity
69 John Linn, RSA Security
70 Rob Philpott, RSA Security
71 Jahan Moreh, Sigaba
72 Anne Anderson, Sun Microsystems
73 Eve Maler, Sun Microsystems
74 Ron Monzillo, Sun Microsystems
75 Greg Whitehead, Trustgenix

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

88 **Status:**
89 This errata composite document is a **working draft** based on the [original](#) OASIS Standard
90 document that had been produced by the Security Services Technical Committee and approved
91 by the OASIS membership on 1 March 2005. While the errata corrections appearing here are
92 non-normative, they reflect changes specified by the Approved Errata document (currently at
93 Working Draft revision 02), which is on an OASIS standardization track. In case of any
94 discrepancy between this document and the Approved Errata, the latter has precedence. [See](#)
95 [also the Errata Working Document \(currently at revision 39\), which provides background on the](#)
96 [changes specified here.](#)
97 This document includes corrections for errata E7, E33, E34, E37, E41, [and E62, E66, E68, E69,](#)
98 [E74, E76, and E77, E81, E82, E83, E87, E88, E89, E91, and E94.](#)
99 Committee members should submit comments and potential errata to the [security-](mailto:security-services@lists.oasis-open.org)
100 [services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should submit them by following the instructions at
101 http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security.

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

Table of Contents

107	1 Introduction.....	6
108	1.1 Notation.....	6
109	2 Metadata for SAML V2.0.....	7
110	2.1 Namespaces	7
111	2.2 Common Types.....	8
112	2.2.1 Simple Type entityIDType.....	8
113	2.2.2 Complex Type EndpointType.....	8
114	2.2.3 Complex Type IndexedEndpointType.....	9
115	2.2.4 Complex Type localizedNameType.....	9
116	2.2.5 Complex Type localizedURIType.....	10
117	2.3 Root Elements.....	10
118	2.3.1 Element <EntitiesDescriptor>.....	10
119	2.3.2 Element <EntityDescriptor>.....	11
120	2.3.2.1 Element <Organization>.....	13
121	2.3.2.2 Element <ContactPerson>.....	14
122	2.3.2.3 Element <AdditionalMetadataLocation>.....	15
123	2.4 Role Descriptor Elements.....	15
124	2.4.1 Element <RoleDescriptor>.....	15
125	2.4.1.1 Element <KeyDescriptor>.....	17
126	2.4.2 Complex Type SSODescriptorType.....	18
127	2.4.3 Element <IDPSSODescriptor>.....	18
128	2.4.4 Element <SPSSODescriptor>.....	20
129	2.4.4.1 Element <AttributeConsumingService>.....	21
130	2.4.4.1.1 [E34]Element <RequestedAttribute>.....	21
131	2.4.5 Element <AuthnAuthorityDescriptor>.....	22
132	2.4.6 Element <PDPDescriptor>.....	23
133	2.4.7 Element <AttributeAuthorityDescriptor>.....	23
134	2.5 Element <AffiliationDescriptor>.....	24
135	2.6 Examples.....	25
136	3 Signature Processing.....	29
137	3.1 XML Signature Profile.....	29
138	3.1.1 Signing Formats and Algorithms.....	29
139	3.1.2 References.....	29
140	3.1.3 Canonicalization Method.....	30
141	3.1.4 Transforms.....	30
142	3.1.5 [E91] Object.....	30
143	3.1.6 KeyInfo.....	30
144	4 Metadata Publication and Resolution.....	31
145	4.1 Publication and Resolution via Well-Known Location.....	31
146	4.1.1 Publication.....	31
147	4.1.2 Resolution.....	31
148	4.2 Publishing and Resolution via DNS.....	31
149	4.2.1 Publication.....	32
150	4.2.1.1 First Well Known Rule.....	32
151	4.2.1.2 The Order Field.....	32
152	4.2.1.3 The Preference Field.....	32
153	4.2.1.4 The Flag Field.....	33

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

1 Introduction

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

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

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

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

1.1 Notation

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

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

```
Example code listings appear like this.
```

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

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

198

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

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

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

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

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

2.1 Namespaces

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

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

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

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

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

```

245 <annotation>
246   <documentation>
247     Document identifier: saml-schema-metadata-2.0
248     Location: http://docs.oasis-open.org/security/saml/v2.0/
249     Revision history:
250       V2.0 (March, 2005):
251         Schema for SAML metadata, first published in SAML 2.0.
252   </documentation>
253 </annotation>
254 ...
255 </schema>

```

256 2.2 Common Types

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

259 2.2.1 Simple Type entityIDType

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

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

```

266 <simpleType name="entityIDType">
267   <restriction base="anyURI">
268     <maxLength value="1024"/>
269   </restriction>
270 </simpleType>

```

271 2.2.2 Complex Type EndpointType

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

275 **Binding** [Required]

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

278 **Location** [Required]

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

281 **ResponseLocation** [Optional]

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

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

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

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

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

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

309 2.2.3 Complex Type IndexedEndpointType

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

313 **index** [Required]

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

318 **isDefault** [Optional]

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

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

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

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

335 2.2.4 Complex Type LocalizedNameType

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

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

```

339     <simpleContent>
340         <extension base="string">
341             <attribute ref="xml:lang" use="required"/>
342         </extension>
343     </simpleContent>
344 </complexType>

```

345 2.2.5 Complex Type localizedURIType

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

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

```

349 <complexType name="localizedURIType">
350     <simpleContent>
351         <extension base="anyURI">
352             <attribute ref="xml:lang" use="required"/>
353         </extension>
354     </simpleContent>
355 </complexType>

```

356 2.3 Root Elements

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

360 2.3.1 Element `<EntitiesDescriptor>`

361 The `<EntitiesDescriptor>` element contains the metadata for an optionally named group of
362 [SAML\[E77\]](#) entities. Its **EntitiesDescriptorType** complex type contains a sequence of
363 `<EntityDescriptor>` elements, `<EntitiesDescriptor>` elements, or both:

364 ID [Optional]

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

366 validUntil [Optional]

367 Optional attribute indicates the expiration time of the metadata contained in the element and any
368 contained elements.

369 cacheDuration [Optional]

370 Optional attribute indicates the maximum length of time a consumer should cache the metadata
371 contained in the element and any contained elements [\[E94\]](#) before attempting to refresh it.

372 Name [Optional]

373 A string name that identifies a group of [SAML\[E77\]](#) entities in the context of some deployment.

374 `<ds:Signature>` [Optional]

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

377 `<Extensions>` [Optional]

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

381 <EntitiesDescriptor> or <EntityDescriptor> [One or More]
382 | Contains the metadata for one or more [SAML\[E77\]](#) entities, or a nested group of additional
383 | metadata.

384 | When used as the root element of a metadata instance, this element MUST contain either a `validUntil`
385 | or `cacheDuration` attribute. It is RECOMMENDED that only the root element of a metadata instance
386 | contain either attribute.

387 | [\[E76\]](#)When not used as the root element of a metadata instance, a `validUntil` or `cacheDuration`
388 | attribute MAY be used to impose a shorter expiration or cache duration than that of the parent or root
389 | element, but never a longer one; the smaller value takes precedence.

390 | The following schema fragment defines the <EntitiesDescriptor> element and its
391 | **EntitiesDescriptorType** complex type:

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

413 | 2.3.2 Element <EntityDescriptor>

414 | The <EntityDescriptor> element specifies metadata for a single [SAML\[E77\]](#) entity. A single entity
415 | may act in many different roles in the support of multiple profiles. This specification directly supports the
416 | following concrete roles as well as the abstract <RoleDescriptor> element for extensibility (see
417 | subsequent sections for more details):

- 418 | • SSO Identity Provider
- 419 | • SSO Service Provider
- 420 | • Authentication Authority
- 421 | • Attribute Authority
- 422 | • Policy Decision Point
- 423 | • Affiliation

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

425 | entityID [Required]

426 | Specifies the unique identifier of the [SAML\[E77\]](#) entity whose metadata is described by the
427 | element's contents.

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

430 validUntil [Optional]
431 Optional attribute indicates the expiration time of the metadata contained in the element and any
432 contained elements.

433 cacheDuration [Optional]
434 Optional attribute indicates the maximum length of time a consumer should cache the metadata
435 contained in the element and any contained elements [\[E94\] before attempting to refresh it](#).

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

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

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

450 <Organization> [Optional]
451 Optional element identifying the organization responsible for the [SAML\[E77\]](#) entity described by
452 the element.

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

455 <AdditionalMetadataLocation> [Zero or More]
456 Optional sequence of namespace-qualified locations where additional metadata exists for the
457 [SAML\[E77\]](#) entity. This may include metadata in alternate formats or describing adherence to
458 other non-SAML specifications.

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

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

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

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

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

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

500 2.3.2.1 Element <Organization>

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

505 <Extensions> [Optional]

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

509 <OrganizationName> [One or More]

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

511 <OrganizationDisplayName> [One or More]

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

513 <OrganizationURL> [One or More]

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

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

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

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

```

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

```

533 2.3.2.2 Element <ContactPerson>

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

538 **contactType** [Required]

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

541 <Extensions> [Optional]

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

545 <Company> [Optional]

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

547 <GivenName> [Optional]

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

549 <SurName> [Optional]

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

551 <EmailAddress> [Zero or More]

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

554 <TelephoneNumber> [Zero or More]

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

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

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

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

```

560 <element name="ContactPerson" type="md:ContactType"/>
561 <complexType name="ContactType">
562   <sequence>
563     <element ref="md:Extensions" minOccurs="0"/>
564     <element ref="md:Company" minOccurs="0"/>
565     <element ref="md:GivenName" minOccurs="0"/>

```

```

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

```

587 2.3.2.3 Element <AdditionalMetadataLocation>

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

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

```

595 <element name="AdditionalMetadataLocation"
596   type="md:AdditionalMetadataLocationType"/>
597 <complexType name="AdditionalMetadataLocationType">
598   <simpleContent>
599     <extension base="anyURI">
600       <attribute name="namespace" type="anyURI" use="required"/>
601     </extension>
602   </simpleContent>
603 </complexType>

```

604 2.4 Role Descriptor Elements

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

608 2.4.1 Element <RoleDescriptor>

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

613 ID [Optional]

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

615 `validUntil` [Optional]
 616 Optional attribute indicates the expiration time of the metadata contained in the element and any
 617 contained elements.

618 `cacheDuration` [Optional]
 619 Optional attribute indicates the maximum length of time a consumer should cache the metadata
 620 contained in the element and any contained elements [\[E94\] before attempting to refresh it.](#)

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

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

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

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

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

640 `<Organization>` [Optional]
 641 Optional element specifies the organization associated with this role. Identical to the element
 642 used within the `<EntityDescriptor>` element.

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

646 | Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.
 647 | [\[E76\]A validUntil or cacheDuration attribute MAY be used to impose a shorter expiration or cache](#)
 648 | [duration than that of the parent or root element, but never a longer one; the smaller value takes](#)
 649 | [precedence.](#)

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

```

652 <element name="RoleDescriptor" type="md:RoleDescriptorType"/>
653 <complexType name="RoleDescriptorType" abstract="true">
654   <sequence>
655     <element ref="ds:Signature" minOccurs="0"/>
656     <element ref="md:Extensions" minOccurs="0"/>
657     <element ref="md:KeyDescriptor" minOccurs="0" maxOccurs="unbounded"/>
658     <element ref="md:Organization" minOccurs="0"/>
659     <element ref="md:ContactPerson" minOccurs="0" maxOccurs="unbounded"/>
  
```



```

660     </sequence>
661     <attribute name="ID" type="ID" use="optional"/>
662     <attribute name="validUntil" type="dateTime" use="optional"/>
663     <attribute name="cacheDuration" type="duration" use="optional"/>
664     <attribute name="protocolSupportEnumeration" type="md:anyURIListType"
665 use="required"/>
666     <attribute name="errorURL" type="anyURI" use="optional"/>
667     <anyAttribute namespace="##other" processContents="lax"/>
668 </complexType>
669 <simpleType name="anyURIListType">
670   <list itemType="anyURI"/>
671 </simpleType>

```

672 2.4.1.1 Element <KeyDescriptor>

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

676 use [Optional]

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

679 <ds:KeyInfo> [Required]

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

682 <EncryptionMethod> [Zero or More]

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

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

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

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

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

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

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

```

704 <element name="KeyDescriptor" type="md:KeyDescriptorType"/>
705 <complexType name="KeyDescriptorType">
706   <sequence>
707     <element ref="ds:KeyInfo"/>

```

```

708         <element ref="md:EncryptionMethod" minOccurs="0"
709 maxOccurs="unbounded"/>
710     </sequence>
711     <attribute name="use" type="md:KeyTypes" use="optional"/>
712 </complexType>
713 <simpleType name="KeyTypes">
714     <restriction base="string">
715         <enumeration value="encryption"/>
716         <enumeration value="signing"/>
717     </restriction>
718 </simpleType>
719 <element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>

```

720 2.4.2 Complex Type SSODescriptorType

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

725 <ArtifactResolutionService> [Zero or More]

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

729 <SingleLogoutService> [Zero or More]

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

732 <ManageNameIDService> [Zero or More]

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

735 <NameIDFormat> [Zero or More]

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

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

```

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

```

760 2.4.3 Element <IDPSSODescriptor>

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

764 `WantAuthnRequestsSigned` [Optional]

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

767 <SingleSignOnService> [One or More]

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

771 <NameIDMappingService> [Zero or More]

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

775 <AssertionIDRequestService> [Zero or More]

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

779 <AttributeProfile> [Zero or More]

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

782 <saml:Attribute> [Zero or More]

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

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

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

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

```
799 <element name="IDPSSODescriptor" type="md:IDPSSODescriptorType"/>  
800 <complexType name="IDPSSODescriptorType">  
801     <complexContent>  
802         <extension base="md:SSODescriptorType">  
803             <sequence>  
804                 <element ref="md:SingleSignOnService" maxOccurs="unbounded"/>
```

```

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

```

823 2.4.4 Element <SPSSODescriptor>

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

827 AuthnRequestsSigned [Optional]

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

835 WantAssertionsSigned [Optional]

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

842 <AssertionConsumerService> [One or More]

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

846 <AttributeConsumingService> [Zero or More]

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

849 At most one <AttributeConsumingService> element can have the attribute `isDefault` set to
850 true. [E87] The default element is the first element with the `isDefault` attribute set to true. If no such
851 elements exist, the default element is the first element without the `isDefault` attribute set to false. If no
852 such elements exist, the default element is the first element in the sequence. It is permissible for none of
853 the included elements to contain an `isDefault` attribute set to true.

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

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

874 **2.4.4.1 Element <AttributeConsumingService>**

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

878 index [Required]

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

881 isDefault [Optional]

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

884 <ServiceName> [One or More]

885 One or more language-qualified names for the service [\[E88\] that are suitable for human](#)
886 [consumption](#).

887 <ServiceDescription> [Zero or More]

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

889 <RequestedAttribute> [One or More]

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

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

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

```
904 </complexType>
905 <element name="ServiceName" type="md:localizedNameType"/>
906 <element name="ServiceDescription" type="md:localizedNameType"/>
```

907 2.4.4.1.1 [E34] Element <RequestedAttribute>

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

911 **isRequired** [Optional]

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

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

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

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

```
921 <element name="RequestedAttribute" type="md:RequestedAttributeType"/>
922 <complexType name="RequestedAttributeType">
923   <complexContent>
924     <extension base="saml:AttributeType">
925       <attribute name="isRequired" type="boolean" use="optional"/>
926     </extension>
927   </complexContent>
928 </complexType>
```

929 2.4.5 Element <AuthnAuthorityDescriptor>

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

933 <AuthnQueryService> [One or More]

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

937 <AssertionIDRequestService> [Zero or More]

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

941 <NameIDFormat> [Zero or More]

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

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

```
946 <element name="AuthnAuthorityDescriptor"
947 type="md:AuthnAuthorityDescriptorType"/>
948 <complexType name="AuthnAuthorityDescriptorType">
```



```

949     <complexContent>
950       <extension base="md:RoleDescriptorType">
951         <sequence>
952           <element ref="md:AuthnQueryService" maxOccurs="unbounded"/>
953           <element ref="md:AssertionIDRequestService" minOccurs="0"
954 maxOccurs="unbounded"/>
955           <element ref="md:NameIDFormat" minOccurs="0"
956 maxOccurs="unbounded"/>
957         </sequence>
958       </extension>
959     </complexContent>
960 </complexType>
961 <element name="AuthnQueryService" type="md:EndpointType"/>

```

962 2.4.6 Element <PDPDescriptor>

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

966 <AuthzService> [One or More]

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

970 <AssertionIDRequestService> [Zero or More]

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

974 <NameIDFormat> [Zero or More]

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

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

```

979 <element name="PDPDescriptor" type="md:PDPDescriptorType"/>
980 <complexType name="PDPDescriptorType">
981   <complexContent>
982     <extension base="md:RoleDescriptorType">
983       <sequence>
984         <element ref="md:AuthzService" maxOccurs="unbounded"/>
985         <element ref="md:AssertionIDRequestService" minOccurs="0"
986 maxOccurs="unbounded"/>
987         <element ref="md:NameIDFormat" minOccurs="0"
988 maxOccurs="unbounded"/>
989       </sequence>
990     </extension>
991   </complexContent>
992 </complexType>
993 <element name="AuthzService" type="md:EndpointType"/>

```

994 2.4.7 Element <AttributeAuthorityDescriptor>

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

- 999 <AttributeService> [One or More]
 1000 One or more elements of type **EndpointType** that describe endpoints that support the profile of
 1001 the Attribute Query protocol defined in [SAMLProf]. All attribute authorities support at least one
 1002 such endpoint, by definition.
- 1003 <AssertionIDRequestService> [Zero or More]
 1004 Zero or more elements of type **EndpointType** that describe endpoints that support the profile of
 1005 the Assertion [\[E33\]Query/Request](#) protocol defined in [SAMLProf] or the special URI binding for
 1006 assertion requests defined in [SAMLBind].
- 1007 <NameIDFormat> [Zero or More]
 1008 Zero or more elements of type **anyURI** that enumerate the name identifier formats supported by
 1009 this authority. See Section 8.3 of [SAMLCore] for some possible values for this element.
- 1010 <AttributeProfile> [Zero or More]
 1011 Zero or more elements of type **anyURI** that enumerate the attribute profiles supported by this
 1012 authority. See [SAMLProf] for some possible values for this element.
- 1013 <saml:Attribute> [Zero or More]
 1014 Zero or more elements that identify the SAML attributes supported by the authority. Specific
 1015 values MAY optionally be included, indicating that only certain values permitted by the attribute's
 1016 definition are supported.

1017 The following schema fragment defines the <AttributeAuthorityDescriptor> element and its
 1018 **AttributeAuthorityDescriptorType** complex type:

```

1019 <element name="AttributeAuthorityDescriptor"
1020 type="md:AttributeAuthorityDescriptorType"/>
1021 <complexType name="AttributeAuthorityDescriptorType">
1022   <complexContent>
1023     <extension base="md:RoleDescriptorType">
1024       <sequence>
1025         <element ref="md:AttributeService" maxOccurs="unbounded"/>
1026         <element ref="md:AssertionIDRequestService" minOccurs="0"
1027 maxOccurs="unbounded"/>
1028         <element ref="md:NameIDFormat" minOccurs="0"
1029 maxOccurs="unbounded"/>
1030         <element ref="md:AttributeProfile" minOccurs="0"
1031 maxOccurs="unbounded"/>
1032         <element ref="saml:Attribute" minOccurs="0"
1033 maxOccurs="unbounded"/>
1034       </sequence>
1035     </extension>
1036   </complexContent>
1037 </complexType>
1038 <element name="AttributeService" type="md:EndpointType"/>
  
```

1039 2.5 Element <AffiliationDescriptor>

1040 The <AffiliationDescriptor> element is an alternative to the sequence of role descriptors
 1041 described in Section 2.4 that is used when an <EntityDescriptor> describes an affiliation of
 1042 [SAML\[E77\]](#) entities (typically service providers) rather than a single entity. The
 1043 <AffiliationDescriptor> element provides a summary of the individual entities that make up the
 1044 affiliation along with general information about the affiliation itself. Its **AffiliationDescriptorType** complex
 1045 type contains the following elements and attributes:

- 1046 affiliationOwnerID [Required]
 1047 Specifies the unique identifier of the entity responsible for the affiliation. The owner is NOT

1048 presumed to be a member of the affiliation; if it is a member, its identifier MUST also appear in an
1049 <AffiliateMember> element.

1050 ID [Optional]

1051 A document-unique identifier for the element, typically used as a reference point when signing.

1052 validUntil [Optional]

1053 Optional attribute indicates the expiration time of the metadata contained in the element and any
1054 contained elements.

1055 cacheDuration [Optional]

1056 Optional attribute indicates the maximum length of time a consumer should cache the metadata
1057 contained in the element and any contained elements [\[E94\] before attempting to refresh it.](#)

1058 <ds:Signature> [Optional]

1059 An XML signature that authenticates the containing element and its contents, as described in
1060 Section 3.

1061 <Extensions> [Optional]

1062 This contains optional metadata extensions that are agreed upon between a metadata publisher
1063 and consumer. Extension elements MUST be namespace-qualified by a non-SAML-defined
1064 namespace.

1065 <AffiliateMember> [One or More]

1066 One or more elements enumerating the members of the affiliation by specifying each member's
1067 unique identifier. See also Section 8.3.6 of [SAMLCore].

1068 <KeyDescriptor> [Zero or More]

1069 Optional sequence of elements that provides information about the cryptographic keys that the
1070 affiliation uses as a whole, as distinct from keys used by individual members of the affiliation,
1071 which are published in the metadata for those entities.

1072 Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

1073 [\[E76\]A validUntil or cacheDuration attribute MAY be used to impose a shorter expiration or cache](#)
1074 [duration than that of the parent or root element, but never a longer one; the smaller value takes](#)
1075 [precedence.](#)

1076 The following schema fragment defines the <AffiliationDescriptor> element and its
1077 **AffiliationDescriptorType** complex type:

```
1078 <element name="AffiliationDescriptor" type="md:AffiliationDescriptorType"/>
1079 <complexType name="AffiliationDescriptorType">
1080   <sequence>
1081     <element ref="ds:Signature" minOccurs="0"/>
1082     <element ref="md:Extensions" minOccurs="0"/>
1083     <element ref="md:AffiliateMember" maxOccurs="unbounded"/>
1084     <element ref="md:KeyDescriptor" minOccurs="0" maxOccurs="unbounded"/>
1085   </sequence>
1086   <attribute name="affiliationOwnerID" type="md:entityIDType"
1087 use="required"/>
1088   <attribute name="validUntil" type="dateTime" use="optional"/>
1089   <attribute name="cacheDuration" type="duration" use="optional"/>
1090   <attribute name="ID" type="ID" use="optional"/>
1091   <anyAttribute namespace="##other" processContents="lax"/>
1092 </complexType>
1093 <element name="AffiliateMember" type="md:entityIDType"/>
```

2.6 Examples

1095 The following is an example of metadata for a SAML system entity acting as an identity provider and an
 1096 attribute authority. A signature is shown as a placeholder, without the actual content.
 1097

```

1098 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1099   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1100   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1101   entityID="https://IdentityProvider.com/SAML">
1102   <ds:Signature>...</ds:Signature>
1103   <IDPSSODescriptor WantAuthnRequestsSigned="true"
1104     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1105     <KeyDescriptor use="signing">
1106       <ds:KeyInfo>
1107         <ds:KeyName>IdentityProvider.com SSO Key</ds:KeyName>
1108       </ds:KeyInfo>
1109     </KeyDescriptor>
1110     <ArtifactResolutionService isDefault="true" index="0"
1111       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1112       Location="https://IdentityProvider.com/SAML/Artifact"/>
1113     <SingleLogoutService
1114       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1115       Location="https://IdentityProvider.com/SAML/SLO/SOAP"/>
1116     <SingleLogoutService
1117       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1118       Location="https://IdentityProvider.com/SAML/SLO/Browser"
1119       ResponseLocation="https://IdentityProvider.com/SAML/SLO/Response"/>
1120     <NameIDFormat>
1121       urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1122     </NameIDFormat>
1123     <NameIDFormat>
1124       urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1125     </NameIDFormat>
1126     <NameIDFormat>
1127       urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1128     </NameIDFormat>
1129     <SingleSignOnService
1130       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1131       Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1132     <SingleSignOnService
1133       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1134       Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1135     <saml:Attribute
1136       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1137       Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1138       FriendlyName="eduPersonPrincipalName">
1139     </saml:Attribute>
1140     <saml:Attribute
1141       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1142       Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1143       FriendlyName="eduPersonAffiliation">
1144       <saml:AttributeValue>member</saml:AttributeValue>
1145       <saml:AttributeValue>student</saml:AttributeValue>
1146       <saml:AttributeValue>faculty</saml:AttributeValue>
1147       <saml:AttributeValue>employee</saml:AttributeValue>
1148       <saml:AttributeValue>staff</saml:AttributeValue>
1149     </saml:Attribute>
1150   </IDPSSODescriptor>
1151   <AttributeAuthorityDescriptor
1152     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1153     <KeyDescriptor use="signing">
1154       <ds:KeyInfo>
1155         <ds:KeyName>IdentityProvider.com AA Key</ds:KeyName>
1156       </ds:KeyInfo>

```

```

1157     </KeyDescriptor>
1158     <AttributeService
1159         Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1160         Location="https://IdentityProvider.com/SAML/AA/SOAP"/>
1161     <AssertionIDRequestService
1162         Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
1163         Location="https://IdentityProvider.com/SAML/AA/URI"/>
1164     <NameIDFormat>
1165         urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1166     </NameIDFormat>
1167     <NameIDFormat>
1168         urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1169     </NameIDFormat>
1170     <NameIDFormat>
1171         urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1172     </NameIDFormat>
1173     <saml:Attribute
1174         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1175         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1176         FriendlyName="eduPersonPrincipalName">
1177     </saml:Attribute>
1178     <saml:Attribute
1179         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1180         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1181         FriendlyName="eduPersonAffiliation">
1182         <saml:AttributeValue>member</saml:AttributeValue>
1183         <saml:AttributeValue>student</saml:AttributeValue>
1184         <saml:AttributeValue>faculty</saml:AttributeValue>
1185         <saml:AttributeValue>employee</saml:AttributeValue>
1186         <saml:AttributeValue>staff</saml:AttributeValue>
1187     </saml:Attribute>
1188     </AttributeAuthorityDescriptor>
1189     <Organization>
1190         <OrganizationName xml:lang="en">Identity Providers R
1191     US</OrganizationName>
1192         <OrganizationDisplayName xml:lang="en">
1193             Identity Providers R US, a Division of Lerxst Corp.
1194         </OrganizationDisplayName>
1195         <OrganizationURL
1196     xml:lang="en">https://IdentityProvider.com</OrganizationURL>
1197         </Organization>
1198     </EntityDescriptor>
1199

```

1200 The following is an example of metadata for a SAML system entity acting as a service provider. A
1201 signature is shown as a placeholder, without the actual content. For illustrative purposes, the service is
1202 one that does not require users to uniquely identify themselves, but rather authorizes access on the basis
1203 of a role-like attribute.
1204

```

1205 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1206     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1207     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1208     entityID="https://ServiceProvider.com/SAML">
1209     <ds:Signature>...</ds:Signature>
1210     <SPSSODescriptor AuthnRequestsSigned="true"
1211         protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1212         <KeyDescriptor use="signing">
1213             <ds:KeyInfo>
1214                 <ds:KeyName>ServiceProvider.com SSO Key</ds:KeyName>
1215             </ds:KeyInfo>
1216         </KeyDescriptor>
1217         <KeyDescriptor use="encryption">
1218             <ds:KeyInfo>
1219                 <ds:KeyName>ServiceProvider.com Encrypt Key</ds:KeyName>
1220             </ds:KeyInfo>

```

```

1221         <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
1222 1_5"/>
1223     </KeyDescriptor>
1224     <SingleLogoutService
1225         Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1226         Location="https://ServiceProvider.com/SAML/SLO/SOAP"/>
1227     <SingleLogoutService
1228         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1229         Location="https://ServiceProvider.com/SAML/SLO/Browser"
1230         ResponseLocation="https://ServiceProvider.com/SAML/SLO/Response"/>
1231     <NameIDFormat>
1232         urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1233     </NameIDFormat>
1234     <AssertionConsumerService isDefault="true" index="0"
1235         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
1236         Location="https://ServiceProvider.com/SAML/SSO/Artifact"/>
1237     <AssertionConsumerService index="1"
1238         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1239         Location="https://ServiceProvider.com/SAML/SSO/POST"/>
1240     <AttributeConsumingService index="0">
1241         <ServiceName xml:lang="en">Academic Journals R US</ServiceName>
1242         <RequestedAttribute
1243             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1244             Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7"
1245             FriendlyName="eduPersonEntitlement">
1246             <saml:AttributeValue>
1247                 https://ServiceProvider.com/entitlements/123456789
1248             </saml:AttributeValue>
1249         </RequestedAttribute>
1250     </AttributeConsumingService>
1251 </SPSSODescriptor>
1252 <Organization>
1253     <OrganizationName xml:lang="en">Academic Journals R
1254 US</OrganizationName>
1255     <OrganizationDisplayName xml:lang="en">
1256 Academic Journals R US, a Division of Dirk Corp.
1257 </OrganizationDisplayName>
1258     <OrganizationURL
1259 xml:lang="en">https://ServiceProvider.com</OrganizationURL>
1260 </Organization>
1261 </EntityDescriptor>

```

1262 3 Signature Processing

1263 Various elements in a metadata instance can be digitally signed (as indicated by the element's inclusion
1264 of a `<ds:Signature>` element), with the following benefits:

- 1265 • Metadata integrity
- 1266 • Authentication of the metadata by a trusted signer

1267 A digital signature is not always required, for example if the relying party obtains the information directly
1268 from the publishing entity directly (with no intermediaries) through a secure channel, with the entity having
1269 authenticated to the relying party by some means other than a digital signature.

1270 Many different techniques are available for "direct" authentication and secure channel establishment
1271 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,
1272 the applicable security requirements depend on the communicating applications.

1273 Additionally, elements can inherit signatures on enclosing parent elements that are themselves signed.

1274 In the absence of such context, it is RECOMMENDED that at least the root element of a metadata
1275 instance be signed.

1276 3.1 XML Signature Profile

1277 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility
1278 and many choices. This section details the constraints on these facilities so that metadata processors do
1279 not have to deal with the full generality of XML Signature processing. This usage makes specific use of
1280 the `xs:ID`-typed attributes optionally present on the elements to which signatures can apply. These
1281 attributes are collectively referred to in this section as the identifier attributes.

1282 3.1.1 Signing Formats and Algorithms

1283 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and
1284 detached.

1285 SAML metadata MUST use enveloped signatures when signing the elements defined in this specification.
1286 ~~SAML processors SHOULD support the use of RSA signing and verification for public key operations in~~
1287 ~~accordance with the algorithm identified by <http://www.w3.org/2000/09/xmlsig#rsa-sha1>.^[E81] Any~~
1288 ~~algorithm defined for use with the XML Signature specification MAY be used.~~

1289 3.1.2 References

1290 Signed metadata elements MUST supply a value for the identifier attribute on the signed element. The
1291 element may or may not be the root element of the actual XML document containing the signed metadata
1292 element.

1293 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute
1294 value of the metadata element being signed. For example, if the identifier attribute value is "foo", then the
1295 URI attribute in the `<ds:Reference>` element MUST be "#foo".

1296 As a consequence, a metadata element's signature MUST apply to the content of the signed element and
1297 any child elements it contains.

1298 **3.1.3 Canonicalization Method**

1299 SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the
1300 <ds:CanonicalizationMethod> element of <ds:SignedInfo>, and as a <ds:Transform>
1301 algorithm. [\[E83\] Use of Exclusive Canonicalization facilitates the verification of signatures created over](#)
1302 [SAML messages when placed into a different XML context than present during signing.](#)

1303 [Note that use of this algorithm alone does not guarantee that a particular signed object can be moved](#)
1304 [from one context to another safely, nor is that a requirement of signed SAML objects in general, though it](#)
1305 [MAY be required by particular profiles. Use of Exclusive Canonicalization ensures that signatures created](#)
1306 [over SAML metadata embedded in an XML context can be verified independent of that context.](#)

1307 **3.1.4 Transforms**

1308 Signatures in SAML metadata SHOULD NOT contain transforms other than the enveloped signature
1309 transform (with the identifier <http://www.w3.org/2000/09/xmlsig#enveloped-signature>) or the exclusive
1310 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or
1311 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

1312 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do
1313 not, verifiers MUST ensure that no content of the signed metadata element is excluded from the
1314 signature. This can be accomplished by establishing out-of-band agreement as to what transforms are
1315 acceptable, or by applying the transforms manually to the content and reverifying the result as consisting
1316 of the same SAML metadata.

1317 **3.1.5 [E91] Object**

1318 [The <ds:Object> element is not defined for use with SAML metadata signatures, and SHOULD NOT be](#)
1319 [present. Since it can be used in service of an attacker by carrying unsigned data, verifiers SHOULD reject](#)
1320 [signatures that contain a <ds:Object> element.](#)

1321 **3.1.6 KeyInfo**

1322 XML Signature [XMLSig] defines usage of the <ds:KeyInfo> element. SAML does not require the
1323 use of <ds:KeyInfo> nor does it impose any restrictions on its use. Therefore, <ds:KeyInfo> MAY
1324 be absent.

1325 4 Metadata Publication and Resolution

1326 Two mechanisms are provided for an entity to publish (and for a consumer to resolve the location of)
1327 metadata documents: via a "well-known-location" by directly dereferencing the entity's unique identifier (a
1328 URI variously referred to as an *entityID* or *providerID*), or indirectly by publishing the location of metadata
1329 in the DNS. Other out-of-band mechanisms are of course also permitted. A consumer that supports both
1330 approaches defined in this document MUST attempt resolution via DNS before using the "well-known-
1331 location" mechanism.

1332 When retrieval requires network transport of the document, the transport SHOULD be protected with
1333 mechanisms providing server authentication and integrity protection. For example, HTTP-based
1334 resolution SHOULD be protected with TLS/SSL [RFC2246] as amended by [RFC3546].

1335 Various mechanisms are described in this section to aid in establishing trust in the accuracy and
1336 legitimacy of metadata, including use of XML signatures, SSL/TLS server authentication, and DNS
1337 signatures. Regardless of the mechanism(s) used, relying parties SHOULD have some means by which
1338 to establish trust in metadata information before relying on it.

1339 4.1 Publication and Resolution via Well-Known Location

1340 The following sections describe publication and resolution of metadata by means of a well-known
1341 location.

1342 4.1.1 Publication

1343 Entities MAY publish their metadata documents at a well known location by placing the document at the
1344 location denoted by its unique identifier, which MUST be in the form of a URL (rather than a URN). See
1345 Section 8.3.6 of [SAMLCore] for more information about such identifiers. It is STRONGLY
1346 RECOMMENDED that `https` URLs be used for this purpose. An indirection mechanism supported by the
1347 URL scheme (such as an HTTP 1.1 302 redirect) MAY be used if the document is not placed directly at
1348 the location. If the publishing protocol permits MIME-based identification of content types, the content
1349 type of the metadata instance MUST be `application/samlmetadata+xml`.

1350 The XML document provided at the well-known location MUST describe the metadata only for the entity
1351 represented by the unique identifier (that is, the root element MUST be an `<EntityDescriptor>` with
1352 an `entityID` matching the location). If other entities need to be described, the
1353 `<AdditionalMetadataLocation>` element MUST be used. Thus the `<EntitiesDescriptor>`
1354 element MUST NOT be used in documents published using this mechanism, since a group of entities are
1355 not defined by such an identifier.

1356 4.1.2 Resolution

1357 If an entity's unique identifier is a URL, metadata consumers MAY attempt to resolve an entity's unique
1358 identifier directly, in a scheme-specific manner, by dereferencing the identifier.

1359 4.2 Publishing and Resolution via DNS

1360 To improve the accessibility of metadata documents and provide additional indirection between an entity's
1361 unique identifier and the location of metadata, entities MAY publish their metadata document locations in
1362 a zone of their corresponding DNS [RFC1034]. The entity's unique identifier (a URI) is used as the input
1363 to the process. Since URIs are flexible identifiers, location publication methods and the resolution process
1364 are determined by the URI's scheme and fully-qualified name. URI locations for metadata are

1365 subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in [RFC2915]
1366 and [RFC3403].

1367 It is RECOMMENDED that entities publish their resource records in signed zone files using [Error:](#)
1368 [Reference source not found\[E66\]\[RFC4035\]](#) such that relying parties may establish the validity of the
1369 published location and authority of the zone, and integrity of the DNS response. If DNS zone signatures
1370 are present, relying parties MUST properly validate the signature.

1371 **4.2.1 Publication**

1372 This specification makes use of the NAPTR resource record described in [RFC2915] and [RFC3403].
1373 Familiarity with these documents is encouraged.

1374 Dynamic Delegation Discovery System (DDDS) [RFC3401] is a general purpose system for the retrieval of
1375 information based on an application-specific input string and the application of well known rules to
1376 transform that string until a terminal condition is reached requiring a look-up into an application-specific
1377 defined database or resolution of a URL based on the rules defined by the application. DDDS defines a
1378 specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS
1379 necessary to apply DDDS rules.

1380 Entities MAY publish separate URLs when multiple metadata documents need to be distributed, or when
1381 different metadata documents are required due to multiple trust relationships that require separate keying
1382 material, or when service interfaces require separate metadata declarations. This may be accomplished
1383 through the use of the optional `<AdditionalMetadataLocation>` element, or through the regexp
1384 facility and multiple service definition fields in the NAPTR resource record itself.

1385 If the publishing protocol permits MIME-based identification of content types, the content type of the
1386 metadata instance MUST be `application/samlmetadata+xml`.

1387 If the entity's unique identifier is a URN, publication of the corresponding metadata location proceeds as
1388 specified in [RFC3404]. Otherwise, the resolution of the metadata location proceeds as specified below.

1389 The following is the application-specific profile of DDDS for SAML metadata resolution.

1390 **4.2.1.1 First Well Known Rule**

1391 The "first well-known-rule" for processing SAML metadata resolution is to parse the entity's unique
1392 identifier and extract the fully-qualified domain name (subexpression 3) as described in Section 4.2.3.1.

1393 **4.2.1.2 The Order Field**

1394 The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY
1395 provide multiple NAPTR resource records which MUST be processed by the resolver application in the
1396 order indicated by this field.

1397 **4.2.1.3 The Preference Field**

1398 For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving
1399 application. The resolving application MAY ignore this order, in cases where the service field value does
1400 not meet the resolver's requirements (e.g.: the resource record returns a protocol the application does not
1401 support).

1402 4.2.1.4 The Flag Field

1403 SAML metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying
1404 additional resource records are to be processed). The "U" flag indicates that the output of the rule is a
1405 URI.

1406 4.2.1.5 The Service Field

1407 The SAML-specific service field, as described in the following BNF, declares the modes by which instance
1408 document(s) shall be made available:

```
1409 servicefield = 1("PID2U" / "NID2U") "+" proto [*(":" class) *(":" servicetype)]  
1410 proto = 1("https" / "uddi")  
1411 class = 1[ "entity" / "entitygroup" ]  
1412 servicetype = 1(si / "spsso" / "idpsso" / "authn" / "authnauth" / "pdp" / "attrauth" /  
1413 alphanum )  
1414 si = "si" [ ":" alphanum ] [ ":" endpoint ]  
1415 alphanum = 1*32 (ALPHA / DIGIT)
```

1416 where:

- 1417 • servicefield PID2U resolves an entity's unique identifier to metadata URL.
- 1418 • servicefield NID2U resolves a principal's <NameID> into a metadata URL.
- 1419 • proto describes the retrieval protocol (`https` or `uddi`). In the case of UDDI, the URL will be an
1420 `http(s)` URL referencing a WSDL document.
- 1421 • class identifies whether the referenced metadata document describes a single entity, or multiple.
1422 In the latter case, the referenced document MUST contain the entity defined by the original unique
1423 identifier as a member of a group of entities within the document itself such as an
1424 <AffiliationDescriptor> or <EntitiesDescriptor>.
- 1425 • servicetype allows an entity to publish metadata for distinct roles and services as separate
1426 documents. Resolvers who encounter multiple servicetype declarations will dereference the
1427 appropriate URI, depending on which service is required for an operation (e.g.: an entity operating
1428 both as an identity provider and a service provider can publish metadata for each role at different
1429 locations). The `authn` service type represents a <SingleSignOnService> endpoint.
- 1430 • si (with optional endpoint component) allows the publisher to either directly publish the metadata
1431 for a service instance, or by articulating a SOAP endpoint (using `endpoint`).

1432 For example:

- 1433 • PID2U+https:entity - represents the entity's complete metadata document available via the
1434 https protocol
- 1435 • PID2U+uddi:entity:si:foo - represents the WSDL document location that describes a service
1436 instance "foo"
- 1437 • PID2U+https:entitygroup:idpsso - represents the metadata for a group of entities acting as
1438 SSO identity providers, of which the original entity is a member.
- 1439 • NID2U+https:idp - represents the metadata for the SSO identity provider of a principal

1440 4.2.1.6 The Regex and Replacement Fields

1441 The expected output after processing the input string through the regex MUST be a valid `https` URL or
1442 UDDI node (WSDL document) address.

1443 4.2.2 NAPTR Examples

1444 4.2.2.1 Entity Metadata NAPTR Examples

1445 Entities publish metadata URLs in the following manner:

```
1446 $ORIGIN provider.biz
1447
1448 ;; order pref f service regexp or replacement
1449
1450 IN NAPTR 100 10 "U" PID2U+https:entity
1451     "!^.*$!https://host.provider.biz/some/directory/trust.xml!" ""
1452 IN NAPTR 110 10 "U" PID2U+https: entity:trust
1453     "!^.*$!https://foo.provider.biz:1443/mdtrust.xml!" ""
1454 IN NAPTR 125 10 "U" PID2U+https:"
1455 IN NAPTR 110 10 "U" PID2U+uddi:entity
1456     "!^.*$!https://this.uddi.node.provider.biz/libmd.wsdl" ""
```

1457 4.2.2.2 Name Identifier Examples

1458 A principal's employer `example.int` operates an identity provider which may be used by an office
1459 supply company to authenticate authorized buyers. The supplier takes a users' email address
1460 `buyer@example.int` as input to the resolution process, and parses the email address to extract the
1461 FQDN (`example.int`). The employer publishes the following NAPTR record in the `example.int` DNS:

```
1462 $ORIGIN example.int
1463
1464 IN NAPTR 100 10 "U" NID2U+https:authn
1465     "!^([\^@]+)@(.*)$!https://serv.example.int:8000/cgi-bin/getmd?\1!" ""
1466 IN NAPTR 100 10 "U" NID2U+https:idp
1467     "!^([\^@]+)@(.*)$!https://auth.example.int/app/auth?\1" ""
```

1468 4.2.3 Resolution

1469 When resolving metadata for an entity via the DNS, the unique identifier of the entity is used as the initial
1470 input into the resolution process, rather than as an actual location Proceed as follows:

- 1471 • If the unique identifier is a URN, proceed with the resolution steps as defined in [RFC3404].
- 1472 • Otherwise, parse the identifier to obtain the fully-qualified domain name.
- 1473 • Query the DNS for NAPTR resource records of the domain iteratively until a terminal resource
1474 record is returned.
- 1475 • Identify which resource record to use based on the service fields, then order fields, then preference
1476 fields of the result set.
- 1477 • Obtain the document(s) at the provided location(s) as required by the application.

1478 4.2.3.1 Parsing the Unique Identifier

1479 To initiate the resolution of the location of the metadata information, it will be necessary in some cases to
1480 decompose the entity's unique identifier (expressed as a URI) into one or more atomic elements.

1481 The following regular expression should be used when initiating the decomposition process:

```
1482 ^([\^:/?#]+)?/*([\^:/?#]*@)?(([\^/?:#]*\.)*(([\^/?#:\.]+)\.([\^/?#:\.]+)))
1483 (: \d+)?([\^?#]*)(\?[\^#]*)?(\#.*)?$
1484     1             2             34             56             7             8
1485     9             10            11
```

1486 Subexpression 3 MUST result in a Fully-Qualified Domain Name (FQDN), which will be the basis for
1487 retrieving metadata locations from this zone.

1488 **4.2.3.2 Obtaining Metadata via the DNS**

1489 Upon completion of the parsing of the identifier, the application then performs a DNS query for the
1490 resulting domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses.
1491 Applications MAY exclude from the result set any service definitions that do not concern the present
1492 request operations.

1493 Resolving applications MUST subsequently order the result set according to the order field, and MAY
1494 order the result set based on the preference set. Resolvers are NOT REQUIRED to follow the ordering of
1495 the preferences field. The resulting NAPTR resource record(s) are operated on iteratively (based on the
1496 order flag) until a terminal NAPTR resource record is reached.

1497 The result will be a well-formed, absolute URL, which is then used to retrieve the metadata document.

1498 **4.2.4 Metadata Location Caching**

1499 Location caching MUST NOT exceed the TTL of the DNS zone from which the location was derived.
1500 Resolvers MUST obtain a fresh copy of the metadata location upon reaching the expiration of the TTL of
1501 the zone.

1502 Publishers of metadata documents should carefully consider the TTL of the zone when making changes
1503 to metadata document locations. Should such a location change occur, a publisher MUST either keep the
1504 document at both the old and new location until all conforming resolvers are certain to have the updated
1505 location (e.g.: time of zone change + TTL), or provide an HTTP Redirect [RFC2616] response at the old
1506 location specifying the new location.

1507 **4.3 Post-Processing of Metadata**

1508 The following sections describe the post-processing of metadata.

1509 **4.3.1 Metadata Instance Caching**

1510 [E94] Document caching MUST be based on the duration indicated by the `cacheDuration` attribute of
1511 the subject element(s). If metadata elements have parent elements which contain caching policies, the
1512 parent element takes precedence. To properly process the `cacheDuration` attribute, consumers must
1513 retain the date and time when an instance was obtained.

1514 Note that cache expiration does not imply a lack of validity in the absence of a `validUntil` attribute or
1515 other information; failure to update a cached instance (e.g., due to network failure) need not render
1516 metadata invalid, although implementations may offer such controls to deployers. Document caching
1517 MUST NOT exceed the `validUntil` or `cacheDuration` attribute of the subject element(s). If metadata
1518 elements have parent elements which contain caching policies, the parent element takes precedence.

1520 To properly process the `cacheDuration` attribute, consumers MUST retain the date and time when the
1521 document was retrieved.

1522 When a document or element has expired, the consumer MUST retrieve a fresh copy, which may require
1523 a refresh of the document location(s). Consumers SHOULD process document cache processing
1524 according to [RFC2616] Section 13, and MAY request the Last-Modified date and time from the HTTP
1525 server. Publishers SHOULD ensure acceptable cache processing as described in [RFC2616] (Section
1526 10.3.5 304 Not Modified).

1527 **4.3.2 [E94] Metadata Instance Validity**

1528 Metadata MUST be considered invalid upon reaching the time specified in a `validUntil` attribute of the
1529 subject element(s). The effective expiration may be adjusted downward by parent element(s) with earlier
1530 expirations. Invalid metadata MUST NOT be used. This contrasts with "stale" metadata that may be

1531 [beyond its optimum cache duration but is not explicitly invalid. Such metadata remains valid and MAY be](#)
1532 [used at the discretion of the implementation.](#)

1533 **4.3.3 Handling of HTTPS Redirects**

1534 Publishers MAY issue an HTTP Redirect (301 Moved Permanently, 302 or 307 Temporary Redirect)
1535 [RFC2616], and user agents MUST follow the specified URL in the Redirect response. Redirects
1536 SHOULD be of the same protocol as the initial request.

1537 **4.3.4 Processing of XML Signatures and General Trust Processing**

1538 Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and
1539 for the trust ascribed to the entity described by such metadata:

- 1540 • Trust derived from the signature of the DNS zone from which the metadata location URL was
1541 resolved, ensuring accuracy of the metadata document location(s)
- 1542 • Trust derived from signature processing of the metadata document itself, ensuring the integrity of
1543 the XML document
- 1544 • Trust derived from the SSL/TLS server authentication of the metadata location URL, ensuring the
1545 identity of the publisher of the metadata

1546 Post-processing of the metadata document MUST include signature processing at the XML-document
1547 level and MAY include one of the other two processes. Specifically, the relying party MAY choose to trust
1548 any of the cited authorities in the resolution and parsing process. Publishers of metadata MUST employ a
1549 document-integrity mechanism and MAY employ any of the other two processing profiles to establish trust
1550 in the metadata document, governed by implementation policies.

1551 **4.3.4.1 Processing Signed DNS Zones**

1552 Verification of DNS zone signature SHOULD be processed, if present, as described in [\[E66\]Error:](#)
1553 [Reference source not found\[RFC4035\]](#).

1554 **4.3.4.2 Processing Signed Documents and Fragments**

1555 Published metadata documents SHOULD be signed, as described in Section 3, either by a certificate
1556 issued to the subject of the document, or by another trusted party. Publishers MAY consider signatures of
1557 other parties as a means of trust conveyance.

1558 Metadata consumers MUST validate signatures, when present, on the metadata document as described
1559 by Section 3.

1560 **4.3.4.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL**

1561 It is STRONGLY RECOMMENDED that publishers implement TLS/SSL URLs; therefore, consumers
1562 SHOULD consider the trust inherited from the issuer of the TLS/SSL certificate. Publication URLs may not
1563 always be located in the domain of the subject of the metadata document; therefore, consumers
1564 SHOULD NOT presume certificates whose subject is the entity in question, as it may be hosted by
1565 another trusted party.

1566 As the basis of this trust may not be available against a cached document, other mechanisms SHOULD
1567 be used under such circumstances.

5 References

1568

- 1569 [RFC1034] P. Mockapetris. *Domain Names – Concepts and Facilities*. IETF RFC 1034,
1570 November 1987. See <http://www.ietf.org/rfc/rfc1034.txt>.
- 1571 [RFC2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*,
1572 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1573 [RFC2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999.
1574 See <http://www.ietf.org/rfc/rfc2246.txt>.
- 1575 [E66][RFC2535] D. Eastlake. *Domain Name System Security Extensions*. IETF RFC 2535, March
1576 1999. See <http://www.ietf.org/rfc/rfc2535.txt>.
- 1577 [RFC2616] R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. IETF RFC 2616, June
1578 1999. See <http://www.ietf.org/rfc/rfc2616.txt>.
- 1579 [RFC2915] M. Mealling. *The Naming Authority Pointer (NAPTR) DNS Resource Record*.
1580 IETF RFC 2915, September 2000. See <http://www.ietf.org/rfc/rfc2915.txt>.
- 1581 [RFC3401] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part One: The
1582 Comprehensive DDDS*. IETF RFC 3401, October 2002. See
1583 <http://www.ietf.org/rfc/rfc3401.txt>.
- 1584 [RFC3403] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Three: The
1585 Domain Name System (DNS) Database*. IETF RFC 3403, October 2002. See
1586 <http://www.ietf.org/rfc/rfc3403.txt>.
- 1587 [RFC3404] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Four: The
1588 Uniform Resource Identifiers (URI) Resolution Application*. IETF RFC 3404,
1589 October 2002. See <http://www.ietf.org/rfc/rfc3404.txt>.
- 1590 [RFC3546] S. Blake-Wilson et al. *Transport Layer Security (TLS) Extensions*. IETF RFC
1591 3546, June 2003. See <http://www.ietf.org/rfc/rfc3546.txt>.
- 1592 [E66][RFC4035] R. Arends et al. *Protocol Modifications for the DNS Security Extensions*. IETF
1593 RFC 4035, March 2005. See <http://www.ietf.org/rfc/rfc4035.txt>.
- 1594 [SAMLBind] S. Cantor et al. *Bindings for the OASIS Security Assertion Markup Language
1595 (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os.
1596 See <http://www.oasis-open.org/committees/security/>.
- 1597 [SAMLConform] P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion
1598 Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-
1599 conformance-2.0-os. <http://www.oasis-open.org/committees/security/>.
- 1600 [SAMLCore] S. Cantor et al. *Assertions and Protocols for the OASIS Security Assertion
1601 Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-
1602 core-2.0-os. See <http://www.oasis-open.org/committees/security/>.
- 1603 [SAMLMeta-xsd] S. Cantor et al. *SAML metadata schema*. OASIS SSTC, March 2005. Document
1604 ID saml-schema-metadata-2.0. See [http://www.oasis-
1605 open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1606 [SAMLProf] S. Cantor et al. *Profiles for the OASIS Security Assertion Markup Language
1607 (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-profiles-2.0-os. See
1608 <http://www.oasis-open.org/committees/security/>.
- 1609 [SAMLSec] F. Hirsch et al. *Security and Privacy Considerations for the OASIS Security
1610 Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005.
1611 Document ID saml-sec-consider-2.0-os. See [http://www.oasis-
open.org/committees/security/](http://www.oasis-
1612 open.org/committees/security/).
- 1613 [Schema1] H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web
1614 Consortium Recommendation, May 2001. See [http://www.w3.org/TR/xmlschema-
1/](http://www.w3.org/TR/xmlschema-
1615 1/). Note that this specification normatively references [Schema2], listed below.

1616	[Schema2]	P. V. Biron et al. <i>XML Schema Part 2: Datatypes</i> . World Wide Web Consortium Recommendation, May 2001. See http://www.w3.org/TR/xmlschema-
1617		
1618	[XMLEnc]	D. Eastlake et al. <i>XML-Encryption Syntax and Processing</i> , http://www.w3.org/TR/xmlenc-core/ , World Wide Web Consortium.
1619		
1620	[XMLSig]	D. Eastlake et al. <i>XML-Signature Syntax and Processing. [E74] Second Edition</i> . World Wide Web Consortium, June 2008. See,
1621		http://www.w3.org/TR/xmldsig-core/ , World Wide Web Consortium.
1622		

1623 **Appendix A.Registration of MIME media type**
1624 **application/samlmetadata+xml**

1625 **Introduction**

1626 This document defines a MIME media type -- `application/samlmetadata+xml` -- for use
1627 with the XML serialization of Security Assertion Markup Language metadata.
1628

1629 SAML is a work product of the OASIS Security Services Technical Committee [SSTC]. The SAML
1630 specifications define XML-based constructs with which one may make, and convey, security
1631 assertions. Using SAML, one can assert that an authentication event pertaining to some subject
1632 has occurred and convey said assertion to a relying party, for example.
1633

1634 SAML profiles require agreements between system entities regarding identifiers, binding support,
1635 endpoints, certificates, keys, and so forth. Such information is treated as metadata by SAML v2.0.
1636 [SAMLv2Meta] specifies this metadata, as well as specifying metadata publication and resolution
1637 mechanisms. If the publishing protocol permits MIME-based identification of content types, then
1638 use of the `application/samlmetadata+xml` MIME media type is required.

1639 **MIME media type name**

1640 `application`

1641 **MIME subtype name**

1642 `samlmetadata+xml`

1643 **Required parameters**

1644 None

1645 **Optional parameters**

1646 `charset`

1647 Same as `charset` parameter of `application/xml` [RFC3023].

1648 **Encoding considerations**

1649 Same as for `application/xml` [RFC3023].

1650 **Security considerations**

1651 Per their specification, `samlmetadata+xml` typed objects do not contain executable content.
1652 However, these objects are XML-based [XML], and thus they have all of the general security
1653 considerations presented in Section 10 of [RFC3023].

1654 SAML metadata [SAMLv2Meta] contains information whose integrity and authenticity is important
1655 – identity provider and service provider public keys and endpoint addresses, for example.

1656 To counter potential issues, the publisher may sign `samlmetadata+xml` typed objects. Any such
1657 signature should be verified by the recipient of the data - both as a valid signature, and as being
1658 the signature of the publisher.

1659 Additionally, various of the publication protocols, e.g. HTTP-over-TLS/SSL, offer means for
1660 ensuring the authenticity of the publishing party and for protecting the metadata in transit.

1661 [SAMLv2Meta] also defines prescriptive metadata caching directives, as well as guidance on
1662 handling HTTPS redirects, trust processing, server authentication, and related items.
1663 For a more detailed discussion of SAML v2.0 metadata and its security considerations, please
1664 see [SAMLv2Meta]. For a discussion of overall SAML v2.0 security considerations and specific
1665 security-related design features, please refer to the SAML v2.0 specifications listed in the below
1666 bibliography. The specifications containing security-specific information are explicitly listed.

1667 **Interoperability considerations**

1668 SAML v2.0 metadata explicitly supports identifying the protocols and versions supported by the
1669 identified entities. For example, an identity provider entity can be denoted as supporting SAML
1670 v2.0 [SAMLv2.0], SAML v1.1 [SAMLv1.1], Liberty ID-FF 1.2 [LAPFF], or even other protocols if
1671 they are unambiguously identifiable via URI [RFC2396]. This protocol support information is
1672 conveyed via the `protocolSupportEnumeration` attribute of metadata objects of the
1673 **RoleDescriptorType**.

1674 **Published specification**

1675 [SAMLv2Meta] explicitly specifies use of the `application/samlmetadata+xml` MIME media
1676 type.

1677 **Applications which use this media type**

1678 Potentially any application implementing SAML v2.0, as well as those applications implementing
1679 specifications based on SAML, e.g. those available from the Liberty Alliance [LAP].

1680 **Additional information**

1681 **Magic number(s)**

1682 In general, the same as for `application/xml` [RFC3023]. In particular, the XML root element of
1683 the returned object will have a namespace-qualified name with:
1684

- 1685 – a local name of: `EntityDescriptor`, or
1686 `AffiliationDescriptor`, or
1687 `EntitiesDescriptor`
- 1689 – a namespace URI of: `urn:oasis:names:tc:SAML:2.0:metadata`
1690 (the SAMLv2.0 metadata namespace)

1691 **File extension(s)**

1692 None

1693 **Macintosh File Type Code(s)**

1694 None

1695 **Person & email address to contact for further information**

1696 This registration is made on behalf of the OASIS Security Services Technical Committee (SSTC)
1697 Please refer to the SSTC website for current information on committee chairperson(s) and their
1698 contact addresses: <http://www.oasis-open.org/committees/security/>. Committee members should
1699 submit comments and potential errata to the securityservices@lists.oasis-open.org list. Others
1700 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)
1701 [open.org/committees/comments/form.php?wg_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security).

1702
1703 Additionally, the SAML developer community email distribution list, [saml-dev@lists.oasis-](mailto:saml-dev@lists.oasis-open.org)
1704 [open.org](http://lists.oasis-open.org), may be employed to discuss usage of the `application/samlmetadata+xml` MIME
1705 media type. The "saml-dev" mailing list is publicly archived here: [http://lists.oasis-](http://lists.oasis-open.org/archives/saml-dev/)
1706 [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
1707 subscribe, send a message with the single word "subscribe" in the message body, to: [saml-dev-](mailto:saml-dev-request@lists.oasis-open.org)
1708 [request@lists.oasis-open.org](mailto:saml-dev-request@lists.oasis-open.org).

1709 Intended usage

1710 COMMON

1711 Author/Change controller

1712 The SAML specification sets are a work product of the OASIS Security Services Technical
1713 Committee (SSTC). OASIS and the SSTC have change control over the SAML specification sets.

1714 Bibliography

- 1715 [LAP] “*Liberty Alliance Project*”. See <http://www.projectliberty.org/>
- 1716 [LAPFF] “*Liberty Alliance Project: Federation Framework*”. See
1717 <http://www.projectliberty.org/resources/specifications.php#box1>
- 1718 [OASIS] “*Organization for the Advancement of Structured Information Systems*”.
1719 See <http://www.oasis-open.org/>
- 1720 [RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifiers*
1721 *(URI): Generic Syntax*. IETF RFC 2396, August 1998. Available at
1722 <http://www.ietf.org/rfc/rfc2396.txt>
- 1723 [RFC3023] M. Murata, S. St.Laurent, D. Kohn, “*XML Media Types*”, IETF Request
1724 for Comments 3023, January 2001. Available as [http://www.rfc-](http://www.rfc-editor.org/rfc/rfc3023.txt)
1725 [editor.org/rfc/rfc3023.txt](http://www.rfc-editor.org/rfc/rfc3023.txt)
- 1726 [SAMLv1.1] OASIS Security Services Technical Committee, “*Security Assertion*
1727 *Markup Language (SAML) Version 1.1 Specification Set*”. OASIS
1728 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)
1729 [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)
1730 [xsd.zip](http://www.oasis-open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip)
- 1731 [SAMLv2.0] OASIS Security Services Technical Committee, “*Security Assertion*
1732 *Markup Language (SAML) Version 2.0 Specification Set*”. OASIS
1733 Standard, 15-Mar-2005. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip)
1734 [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)
- 1735 [SAMLv2Bind] S. Cantor et al., “*Bindings for the OASIS Security Assertion Markup*
1736 *Language (SAML) V2.0*”. OASIS, March 2005. Document ID `saml-`
1737 `bindings-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf)
1738 [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)
- 1739 [SAMLv2Core] S. Cantor et al., “*Assertions and Protocols for the OASIS Security*
1740 *Assertion Markup Language (SAML) V2.0*”. OASIS, March 2005.
1741 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)
1742 [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)
- 1743 [SAMLv2Meta] S. Cantor et al., “*Metadata for the OASIS Security Assertion Markup*
1744 *Language (SAML) V2.0*”. OASIS SSTC, August 2004. Document ID `saml-`
1745 `metadata-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf)
1746 [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)
- 1747 [SAMLv2Prof] S. Cantor et al., “*Profiles for the OASIS Security Assertion Markup*
1748 *Language (SAML) V2.0*”. OASIS, March 2005. Document ID `saml-`
1749 `profiles-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf)
1750 [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)

1751	[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: http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf
1752		
1753		
1754		
1755	[SSTC]	"OASIS Security Services Technical Committee". See http://www.oasis-open.org/committees/security/
1756		
1757	[XML]	Bray, T., Paoli, J., Sperberg-McQueen, C.M. and E. Maler, François Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium Recommendation REC-xml, Feb 2004, Available as http://www.w3.org/TR/REC-xml/
1758		
1759		
1760		
1761		

1762 Appendix B. Acknowledgments

1763 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
1764 Committee, whose voting members at the time of publication were:

- 1765 • Conor Cahill, AOL
- 1766 • John Hughes, Atos Origin
- 1767 • Hal Lockhart, BEA Systems
- 1768 • Mike Beach, Boeing
- 1769 • Rebekah Metz, Booz Allen Hamilton
- 1770 • Rick Randall, Booz Allen Hamilton
- 1771 • Ronald Jacobson, Computer Associates
- 1772 • Gavenraj Sodhi, Computer Associates
- 1773 • Thomas Wisniewski, Entrust
- 1774 • Carolina Canales-Valenzuela, Ericsson
- 1775 • Dana Kaufman, Forum Systems
- 1776 • Irving Reid, Hewlett-Packard
- 1777 • Guy Denton, IBM
- 1778 • Heather Hinton, IBM
- 1779 • Maryann Hondo, IBM
- 1780 • Michael McIntosh, IBM
- 1781 • Anthony Nadalin, IBM
- 1782 • Nick Ragouzis, Individual
- 1783 • Scott Cantor, Internet2
- 1784 • Bob Morgan, Internet2
- 1785 • Peter Davis, Neustar
- 1786 • Jeff Hodges, Neustar
- 1787 • Frederick Hirsch, Nokia
- 1788 • Senthil Sengodan, Nokia
- 1789 • Abbie Barbir, Nortel Networks
- 1790 • Scott Kiestler, Novell
- 1791 • Cameron Morris, Novell
- 1792 • Paul Madsen, NTT
- 1793 • Steve Anderson, OpenNetwork
- 1794 • Ari Kermaier, Oracle
- 1795 • Vamsi Motukuru, Oracle
- 1796 • Darren Platt, Ping Identity
- 1797 • Prateek Mishra, Principal Identity
- 1798 • Jim Lien, RSA Security
- 1799 • John Linn, RSA Security
- 1800 • Rob Philpott, RSA Security
- 1801 • Dipak Chopra, SAP
- 1802 • Jahan Moreh, Sigaba
- 1803 • Bhavna Bhatnagar, Sun Microsystems
- 1804 • Eve Maler, Sun Microsystems

- 1805 • Ronald Monzillo, Sun Microsystems
- 1806 • Emily Xu, Sun Microsystems
- 1807 • Greg Whitehead, Trustgenix
- 1808

1809 The editors also would like to acknowledge the following former SSTC members for their contributions to
1810 this or previous versions of the OASIS Security Assertions Markup Language Standard:

- 1811 • Stephen Farrell, Baltimore Technologies
- 1812 • David Orchard, BEA Systems
- 1813 • Krishna Sankar, Cisco Systems
- 1814 • Zahid Ahmed, CommerceOne
- 1815 • Tim Alsop, CyberSafe Limited
- 1816 • Carlisle Adams, Entrust
- 1817 • Tim Moses, Entrust
- 1818 • Nigel Edwards, Hewlett-Packard
- 1819 • Joe Pato, Hewlett-Packard
- 1820 • Bob Blakley, IBM
- 1821 • Marlena Erdos, IBM
- 1822 • Marc Chanliau, Netegrity
- 1823 • Chris McLaren, Netegrity
- 1824 • Lynne Rosenthal, NIST
- 1825 • Mark Skall, NIST
- 1826 • Charles Knouse, Oblix
- 1827 • Simon Godik, Overxeer
- 1828 • Charles Norwood, SAIC
- 1829 • Evan Prodromou, Securant
- 1830 • Robert Griffin, RSA Security (former editor)
- 1831 • Sai Allarvarpu, Sun Microsystems
- 1832 • Gary Ellison, Sun Microsystems
- 1833 • Chris Ferris, Sun Microsystems
- 1834 • Mike Myers, Traceroute Security
- 1835 • Phillip Hallam-Baker, VeriSign (former editor)
- 1836 • James Vanderbeek, Vodafone
- 1837 • Mark O'Neill, Vordel
- 1838 • Tony Palmer, Vordel

1839
1840 Finally, the editors wish to acknowledge the following people for their contributions of material used as
1841 input to the OASIS Security Assertions Markup Language specifications:

- 1842 • Thomas Gross, IBM
- 1843 • Birgit Pfitzmann, IBM

1844 The editors also would like to gratefully acknowledge Jahan Moreh of Sigaba, who during his tenure on
1845 the SSTC was the primary editor of the errata working document and who made major substantive
1846 contributions to all of the errata materials.

1847 **Appendix C. Notices**

1848 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
1849 might be claimed to pertain to the implementation or use of the technology described in this document or
1850 the extent to which any license under such rights might or might not be available; neither does it
1851 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with
1852 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights
1853 made available for publication and any assurances of licenses to be made available, or the result of an
1854 attempt made to obtain a general license or permission for the use of such proprietary rights by
1855 implementors or users of this specification, can be obtained from the OASIS Executive Director.

1856 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,
1857 or other proprietary rights which may cover technology that may be required to implement this
1858 specification. Please address the information to the OASIS Executive Director.

1859 **Copyright © OASIS Open 2005. All Rights Reserved.**

1860 This document and translations of it may be copied and furnished to others, and derivative works that
1861 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published
1862 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice
1863 and this paragraph are included on all such copies and derivative works. However, this document itself
1864 may not be modified in any way, such as by removing the copyright notice or references to OASIS, except
1865 as needed for the purpose of developing OASIS specifications, in which case the procedures for
1866 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to
1867 translate it into languages other than English.

1868 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
1869 or assigns.

1870 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1871 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1872 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1873 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.