

---

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

Working Draft 04, 12 FebruaryDecember 20097

## Document identifier:

sstc-saml-metadata-errata-2.0-wd-043

## Location:

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

## Editors:

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

## Contributors to the Errata:

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

## SAML V2.0 Contributors:

Conor P. Cahill, AOL  
John Hughes, Atos Origin  
Hal Lockhart, BEA Systems

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

#### 74 **Abstract:**

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

#### 86 **Status:**

87 This errata composite document is a **working draft** based on the [original](#) OASIS Standard  
88 document that had been produced by the Security Services Technical Committee and approved  
89 by the OASIS membership on 1 March 2005. While the errata corrections appearing here are  
90 non-normative, they reflect changes specified by the Approved Errata document (currently at  
91 Working Draft revision 02), which is on an OASIS standardization track. In case of any  
92 discrepancy between this document and the Approved Errata, the latter has precedence. ~~See also~~  
93 ~~the Errata Working Document (currently at revision 39), which provides background on the~~  
94 ~~changes specified here.~~

95 This document includes corrections for errata E7, E33, E34, E37, E41, ~~and E62.~~ [E66](#), [E68](#), [E69](#),  
96 [E74](#), [E76](#), and [E77](#).

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

100 For information on whether any patents have been disclosed that may be essential to  
101 implementing this specification, and any offers of patent licensing terms, please refer to the  
102 Intellectual Property Rights web page for the Security Services TC ([sstc-saml-metadata-errata-2.0-wd-043  
Copyright © OASIS Open 20097 All Rights Reserved.](http://www.oasis-</a></p></div><div data-bbox=)



---

# Table of Contents

104

105	1 Introduction.....	5
106	1.1 Notation.....	5
107	2 Metadata for SAML V2.0.....	6
108	2.1 Namespaces .....	6
109	2.2 Common Types.....	7
110	2.2.1 Simple Type entityIDType.....	7
111	2.2.2 Complex Type EndpointType.....	7
112	2.2.3 Complex Type IndexedEndpointType.....	8
113	2.2.4 Complex Type localizedNameType.....	8
114	2.2.5 Complex Type localizedURIType.....	9
115	2.3 Root Elements.....	9
116	2.3.1 Element <EntitiesDescriptor>.....	9
117	2.3.2 Element <EntityDescriptor>.....	10
118	2.3.2.1 Element <Organization>.....	12
119	2.3.2.2 Element <ContactPerson>.....	13
120	2.3.2.3 Element <AdditionalMetadataLocation>.....	14
121	2.4 Role Descriptor Elements.....	14
122	2.4.1 Element <RoleDescriptor>.....	14
123	2.4.1.1 Element <KeyDescriptor>.....	15
124	2.4.2 Complex Type SSODescriptorType.....	16
125	2.4.3 Element <IDPSSODescriptor>.....	17
126	2.4.4 Element <SPSSODescriptor>.....	18
127	2.4.4.1 Element <AttributeConsumingService>.....	19
128	2.4.4.1.1 [E34]Element <RequestedAttribute>.....	20
129	2.4.5 Element <AuthnAuthorityDescriptor>.....	20
130	2.4.6 Element <PDPDescriptor>.....	21
131	2.4.7 Element <AttributeAuthorityDescriptor>.....	22
132	2.5 Element <AffiliationDescriptor>.....	23
133	2.6 Examples.....	24
134	3 Signature Processing.....	27
135	3.1 XML Signature Profile.....	27
136	3.1.1 Signing Formats and Algorithms.....	27
137	3.1.2 References.....	27
138	3.1.3 Canonicalization Method.....	27
139	3.1.4 Transforms.....	28
140	3.1.5 KeyInfo.....	28
141	4 Metadata Publication and Resolution.....	29
142	4.1 Publication and Resolution via Well-Known Location.....	29
143	4.1.1 Publication.....	29
144	4.1.2 Resolution.....	29
145	4.2 Publishing and Resolution via DNS.....	29
146	4.2.1 Publication.....	30
147	4.2.1.1 First Well Known Rule.....	30
148	4.2.1.2 The Order Field.....	30
149	4.2.1.3 The Preference Field.....	30
150	4.2.1.4 The Flag Field.....	31
151	4.2.1.5 The Service Field.....	31

152	4.2.1.6 The Regex and Replacement Fields.....	31
153	4.2.2 NAPTR Examples.....	32
154	4.2.2.1 Entity Metadata NAPTR Examples.....	32
155	4.2.2.2 Name Identifier Examples.....	32
156	4.2.3 Resolution.....	32
157	4.2.3.1 Parsing the Unique Identifier.....	32
158	4.2.3.2 Obtaining Metadata via the DNS.....	33
159	4.2.4 Metadata Location Caching.....	33
160	4.3 Post-Processing of Metadata.....	33
161	4.3.1 Metadata Instance Caching.....	33
162	4.3.2 Handling of HTTPS Redirects.....	33
163	4.3.3 Processing of XML Signatures and General Trust Processing.....	33
164	4.3.3.1 Processing Signed DNS Zones.....	34
165	4.3.3.2 Processing Signed Documents and Fragments.....	34
166	4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL.....	34
167	5 References.....	35
168	Appendix A.Registration of MIME media type application/samlmetadata+xml.....	37
169	Appendix B. Acknowledgments.....	41
170	Appendix C. Notices.....	43

---

# 1 Introduction

171

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

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

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

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

## 1.1 Notation

184

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

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

189

190 `Example code listings appear like this.`

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

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

194

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

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

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

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

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

### 2.1 Namespaces

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

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

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

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

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

```

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

```

## 252 2.2 Common Types

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

### 255 2.2.1 Simple Type entityIDType

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

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

```

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

```

### 267 2.2.2 Complex Type EndpointType

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

#### 271 Binding [Required]

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

#### 274 Location [Required]

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

#### 277 ResponseLocation [Optional]

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

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



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

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

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

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

### 304 2.2.3 Complex Type IndexedEndpointType

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

308 **index** [Required]

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

313 **isDefault** [Optional]

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

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

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

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

### 330 2.2.4 Complex Type localizedNameType

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

```
333 <complexType name="localizedNameType">  
334   <simpleContent>  
335     <extension base="string">
```

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

## 340 2.2.5 Complex Type localizedURIType

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

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

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

## 351 2.3 Root Elements

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

### 355 2.3.1 Element `<EntitiesDescriptor>`

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

359 ID [Optional]

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

361 validUntil [Optional]

362 Optional attribute indicates the expiration time of the metadata contained in the element and any  
363 contained elements.

364 cacheDuration [Optional]

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

367 Name [Optional]

368 A string name that identifies a group of **SAML-[E77]** entities in the context of some deployment.

369 `<ds:Signature>` [Optional]

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

372 `<Extensions>` [Optional]

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

376 <EntitiesDescriptor> or <EntityDescriptor> [One or More]  
377 |         Contains the metadata for one or more [SAML\[E77\]](#) entities, or a nested group of additional  
378 |         metadata.  
  
379 |         When used as the root element of a metadata instance, this element MUST contain either a `validUntil`  
380 |         or `cacheDuration` attribute. It is RECOMMENDED that only the root element of a metadata instance  
381 |         contain either attribute.

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

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

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

### 408 | 2.3.2 Element <EntityDescriptor>

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

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

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

420 |         entityID [Required]

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

423 |         ID [Optional]

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

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

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

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

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

438 `<RoleDescriptor>`, `<IDPSSODescriptor>`, `<SPSSODescriptor>`,  
439 `<AuthnAuthorityDescriptor>`, `<AttributeAuthorityDescriptor>`, `<PDPDescriptor>` [One  
440 or More]  
441 **OR**

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

445 `<Organization>` [Optional]  
446       Optional element identifying the organization responsible for the [SAML\[E77\]](#) entity described by  
447       the element.

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

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

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

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

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

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

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

```

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

```

### 495 2.3.2.1 Element <Organization>

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

500 <Extensions> [Optional]

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

504 <OrganizationName> [One or More]

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

506 <OrganizationDisplayName> [One or More]

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

508 <OrganizationURL> [One or More]

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

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

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

```

515 <element name="Organization" type="md:OrganizationType"/>
516 <complexType name="OrganizationType">
517   <sequence>
518     <element ref="md:Extensions" minOccurs="0"/>
519     <element ref="md:OrganizationName" maxOccurs="unbounded"/>
520     <element ref="md:OrganizationDisplayName" maxOccurs="unbounded"/>

```

```

521     <element ref="md:OrganizationURL" maxOccurs="unbounded"/>
522   </sequence>
523   <anyAttribute namespace="##other" processContents="lax"/>
524 </complexType>
525 <element name="OrganizationName" type="md:localizedNameType"/>
526 <element name="OrganizationDisplayName" type="md:localizedNameType"/>
527 <element name="OrganizationURL" type="md:localizedURIType"/>

```

### 528 2.3.2.2 Element <ContactPerson>

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

533 **contactType** [Required]

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

536 <Extensions> [Optional]

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

540 <Company> [Optional]

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

542 <GivenName> [Optional]

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

544 <SurName> [Optional]

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

546 <EmailAddress> [Zero or More]

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

549 <TelephoneNumber> [Zero or More]

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

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

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

```

554 <element name="ContactPerson" type="md:ContactType"/>
555 <complexType name="ContactType">
556   <sequence>
557     <element ref="md:Extensions" minOccurs="0"/>
558     <element ref="md:Company" minOccurs="0"/>
559     <element ref="md:GivenName" minOccurs="0"/>
560     <element ref="md:SurName" minOccurs="0"/>
561     <element ref="md:EmailAddress" minOccurs="0" maxOccurs="unbounded"/>
562     <element ref="md:TelephoneNumber" minOccurs="0" maxOccurs="unbounded"/>
563   </sequence>
564   <attribute name="contactType" type="md:ContactTypeType" use="required"/>
565   <anyAttribute namespace="##other" processContents="lax"/>
566 </complexType>
567 <element name="Company" type="string"/>

```

```

568 <element name="GivenName" type="string"/>
569 <element name="SurName" type="string"/>
570 <element name="EmailAddress" type="anyURI"/>
571 <element name="TelephoneNumber" type="string"/>
572 <simpleType name="ContactTypeType">
573   <restriction base="string">
574     <enumeration value="technical"/>
575     <enumeration value="support"/>
576     <enumeration value="administrative"/>
577     <enumeration value="billing"/>
578     <enumeration value="other"/>
579   </restriction>
580 </simpleType>

```

### 581 2.3.2.3 Element <AdditionalMetadataLocation>

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

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

```

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

```

## 598 2.4 Role Descriptor Elements

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

### 602 2.4.1 Element <RoleDescriptor>

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

607 ID [Optional]

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

609 validUntil [Optional]

610 Optional attribute indicates the expiration time of the metadata contained in the element and any  
611 contained elements.

612 cacheDuration [Optional]

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

615 protocolSupportEnumeration [Required]

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

621 errorURL [Optional]

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

624 <ds:Signature> [Optional]

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

627 <Extensions> [Optional]

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

631 <KeyDescriptor> [Zero or More]

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

634 <Organization> [Optional]

635 Optional element specifies the organization associated with this role. Identical to the element used  
636 within the <EntityDescriptor> element.

637 <ContactPerson> [Zero or More]

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

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

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

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

```
646 <element name="RoleDescriptor" type="md:RoleDescriptorType"/>
647 <complexType name="RoleDescriptorType" abstract="true">
648   <sequence>
649     <element ref="ds:Signature" minOccurs="0"/>
650     <element ref="md:Extensions" minOccurs="0"/>
651     <element ref="md:KeyDescriptor" minOccurs="0" maxOccurs="unbounded"/>
652     <element ref="md:Organization" minOccurs="0"/>
653     <element ref="md:ContactPerson" minOccurs="0" maxOccurs="unbounded"/>
654   </sequence>
655   <attribute name="ID" type="ID" use="optional"/>
656   <attribute name="validUntil" type="dateTime" use="optional"/>
657   <attribute name="cacheDuration" type="duration" use="optional"/>
658   <attribute name="protocolSupportEnumeration" type="md:anyURIListType"
659 use="required"/>
660   <attribute name="errorURL" type="anyURI" use="optional"/>
661   <anyAttribute namespace="##other" processContents="lax"/>
662 </complexType>
663 <simpleType name="anyURIListType">
```



```
664 <list itemType="anyURI"/>
665 </simpleType>
```

### 666 2.4.1.1 Element <KeyDescriptor>

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

670 use [Optional]

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

673 <ds:KeyInfo> [Required]

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

676 <EncryptionMethod> [Zero or More]

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

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

682 [A use value of "encryption" means that the contained key information is suitable for use in wrapping](#)  
683 [encryption keys for use by the entity when acting in the enclosing role.](#)

684 [If the use attribute is omitted, then the contained key information is applicable to both of the above uses.](#)

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

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

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

```
697 <element name="KeyDescriptor" type="md:KeyDescriptorType"/>
698 <complexType name="KeyDescriptorType">
699   <sequence>
700     <element ref="ds:KeyInfo"/>
701     <element ref="md:EncryptionMethod" minOccurs="0"
702 maxOccurs="unbounded"/>
703   </sequence>
704   <attribute name="use" type="md:KeyTypes" use="optional"/>
705 </complexType>
706 <simpleType name="KeyTypes">
707   <restriction base="string">
708     <enumeration value="encryption"/>
709     <enumeration value="signing"/>
710   </restriction>
711 </simpleType>
712 <element name="EncryptionMethod" type="xenc:EncryptionMethodType"/>
```

## 713 2.4.2 Complex Type **SSODescriptorType**

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

718 <ArtifactResolutionService> [Zero or More]

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

722 <SingleLogoutService> [Zero or More]

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

725 <ManageNameIDService> [Zero or More]

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

728 <NameIDFormat> [Zero or More]

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

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

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

## 753 2.4.3 Element <IDPSSODescriptor>

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

757 WantAuthnRequestsSigned [Optional]

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

760 <SingleSignOnService> [One or More]  
761 One or more elements of type **EndpointType** that describe endpoints that support the profiles of  
762 the Authentication Request protocol defined in [SAMLProf]. All identity providers support at least  
763 one such endpoint, by definition. The `ResponseLocation` attribute MUST be omitted.

764 <NameIDMappingService> [Zero or More]  
765 Zero or more elements of type **EndpointType** that describe endpoints that support the Name  
766 Identifier Mapping profile defined in [SAMLProf]. The `ResponseLocation` attribute MUST be  
767 omitted.

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

772 <AttributeProfile> [Zero or More]  
773 Zero or more elements of type **anyURI** that enumerate the attribute profiles supported by this  
774 identity provider. See [SAMLProf] for some possible values for this element.

775 <saml:Attribute> [Zero or More]  
776 Zero or more elements that identify the SAML attributes supported by the identity provider.  
777 Specific values MAY optionally be included, indicating that only certain values permitted by the  
778 attribute's definition are supported. In this context, "support" for an attribute means that the identity  
779 provider has the capability to include it when delivering assertions during single sign-on.

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

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

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

```
792 <element name="IDPSSODescriptor" type="md:IDPSSODescriptorType"/>  
793 <complexType name="IDPSSODescriptorType">  
794   <complexContent>  
795     <extension base="md:SSODescriptorType">  
796       <sequence>  
797         <element ref="md:SingleSignOnService" maxOccurs="unbounded"/>  
798         <element ref="md:NameIDMappingService" minOccurs="0"  
799 maxOccurs="unbounded"/>  
800         <element ref="md:AssertionIDRequestService" minOccurs="0"  
801 maxOccurs="unbounded"/>  
802         <element ref="md:AttributeProfile" minOccurs="0"  
803 maxOccurs="unbounded"/>  
804         <element ref="saml:Attribute" minOccurs="0"  
805 maxOccurs="unbounded"/>  
806       </sequence>  
807       <attribute name="WantAuthnRequestsSigned" type="boolean"  
808 use="optional"/>  
809     </extension>  
810   </complexContent>  
811 </complexType>
```

```

812 <element name="SingleSignOnService" type="md:EndpointType"/>
813 <element name="NameIDMappingService" type="md:EndpointType"/>
814 <element name="AssertionIDRequestService" type="md:EndpointType"/>
815 <element name="AttributeProfile" type="anyURI"/>

```

## 816 2.4.4 Element <SPSSODescriptor>

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

820 AuthnRequestsSigned [Optional]

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

828 WantAssertionsSigned [Optional]

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

835 <AssertionConsumerService> [One or More]

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

839 <AttributeConsumingService> [Zero or More]

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

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

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

```

846 <element name="SPSSODescriptor" type="md:SPSSODescriptorType"/>
847 <complexType name="SPSSODescriptorType">
848   <complexContent>
849     <extension base="md:SSODescriptorType">
850       <sequence>
851         <element ref="md:AssertionConsumerService"
852 maxOccurs="unbounded"/>
853         <element ref="md:AttributeConsumingService" minOccurs="0"
854 maxOccurs="unbounded"/>
855       </sequence>
856       <attribute name="AuthnRequestsSigned" type="boolean"
857 use="optional"/>
858       <attribute name="WantAssertionsSigned" type="boolean"
859 use="optional"/>
860     </extension>
861   </complexContent>
862 </complexType>

```

```
863 <element name="AssertionConsumerService" type="md:IndexedEndpointType"/>
```

#### 864 2.4.4.1 Element <AttributeConsumingService>

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

868 **index** [Required]

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

871 **isDefault** [Optional]

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

874 <ServiceName> [One or More]

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

876 <ServiceDescription> [Zero or More]

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

878 <RequestedAttribute> [One or More]

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

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

```
882 <element name="AttributeConsumingService"  
883 type="md:AttributeConsumingServiceType"/>  
884 <complexType name="AttributeConsumingServiceType">  
885 <sequence>  
886 <element ref="md:ServiceName" maxOccurs="unbounded"/>  
887 <element ref="md:ServiceDescription" minOccurs="0"  
888 maxOccurs="unbounded"/>  
889 <element ref="md:RequestedAttribute" maxOccurs="unbounded"/>  
890 </sequence>  
891 <attribute name="index" type="unsignedShort" use="required"/>  
892 <attribute name="isDefault" type="boolean" use="optional"/>  
893 </complexType>  
894 <element name="ServiceName" type="md:localizedNameType"/>  
895 <element name="ServiceDescription" type="md:localizedNameType"/>
```

#### 896 2.4.4.1.1 [E34]Element <RequestedAttribute>

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

900 **isRequired** [Optional]

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

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

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

```

907 <element name="RequestedAttribute" type="md:RequestedAttributeType"/>
908 <complexType name="RequestedAttributeType">
909   <complexContent>
910     <extension base="saml:AttributeType">
911       <attribute name="isRequired" type="boolean" use="optional"/>
912     </extension>
913   </complexContent>
914 </complexType>

```

## 915 2.4.5 Element <AuthnAuthorityDescriptor>

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

919 <AuthnQueryService> [One or More]

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

923 <AssertionIDRequestService> [Zero or More]

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

927 <NameIDFormat> [Zero or More]

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

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

```

932 <element name="AuthnAuthorityDescriptor"
933   type="md:AuthnAuthorityDescriptorType"/>
934 <complexType name="AuthnAuthorityDescriptorType">
935   <complexContent>
936     <extension base="md:RoleDescriptorType">
937       <sequence>
938         <element ref="md:AuthnQueryService" maxOccurs="unbounded"/>
939         <element ref="md:AssertionIDRequestService" minOccurs="0"
940 maxOccurs="unbounded"/>
941         <element ref="md:NameIDFormat" minOccurs="0"
942 maxOccurs="unbounded"/>
943       </sequence>
944     </extension>
945   </complexContent>
946 </complexType>
947 <element name="AuthnQueryService" type="md:EndpointType"/>

```

## 948 2.4.6 Element <PDPDescriptor>

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

952 <AuthzService> [One or More]

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

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

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

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

```
965 <element name="PDPDescriptor" type="md:PDPDescriptorType"/>  
966 <complexType name="PDPDescriptorType">  
967   <complexContent>  
968     <extension base="md:RoleDescriptorType">  
969       <sequence>  
970         <element ref="md:AuthzService" maxOccurs="unbounded"/>  
971         <element ref="md:AssertionIDRequestService" minOccurs="0"  
972 maxOccurs="unbounded"/>  
973         <element ref="md:NameIDFormat" minOccurs="0"  
974 maxOccurs="unbounded"/>  
975       </sequence>  
976     </extension>  
977   </complexContent>  
978 </complexType>  
979 <element name="AuthzService" type="md:EndpointType"/>
```

## 980 **2.4.7 Element <AttributeAuthorityDescriptor>**

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

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

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

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

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

999 <saml:Attribute> [Zero or More]  
1000       Zero or more elements that identify the SAML attributes supported by the authority. Specific  
1001       values MAY optionally be included, indicating that only certain values permitted by the attribute's  
1002       definition are supported.

1003 The following schema fragment defines the <AttributeAuthorityDescriptor> element and its  
1004 **AttributeAuthorityDescriptorType** complex type:

```
1005 <element name="AttributeAuthorityDescriptor"  
1006 type="md:AttributeAuthorityDescriptorType"/>  
1007 <complexType name="AttributeAuthorityDescriptorType">  
1008 <complexContent>  
1009 <extension base="md:RoleDescriptorType">  
1010 <sequence>  
1011 <element ref="md:AttributeService" maxOccurs="unbounded"/>  
1012 <element ref="md:AssertionIDRequestService" minOccurs="0"  
1013 maxOccurs="unbounded"/>  
1014 <element ref="md:NameIDFormat" minOccurs="0"  
1015 maxOccurs="unbounded"/>  
1016 <element ref="md:AttributeProfile" minOccurs="0"  
1017 maxOccurs="unbounded"/>  
1018 <element ref="saml:Attribute" minOccurs="0"  
1019 maxOccurs="unbounded"/>  
1020 </sequence>  
1021 </extension>  
1022 </complexContent>  
1023 </complexType>  
1024 <element name="AttributeService" type="md:EndpointType"/>
```

## 1025 2.5 Element <AffiliationDescriptor>

1026 The <AffiliationDescriptor> element is an alternative to the sequence of role descriptors  
1027 described in Section 2.4 that is used when an <EntityDescriptor> describes an affiliation of  
1028 [SAML\[E77\]](#) entities (typically service providers) rather than a single entity. The  
1029 <AffiliationDescriptor> element provides a summary of the individual entities that make up the  
1030 affiliation along with general information about the affiliation itself. Its **AffiliationDescriptorType** complex  
1031 type contains the following elements and attributes:

1032 affiliationOwnerID [Required]

1033 Specifies the unique identifier of the entity responsible for the affiliation. The owner is NOT  
1034 presumed to be a member of the affiliation; if it is a member, its identifier MUST also appear in an  
1035 <AffiliateMember> element.

1036 ID [Optional]

1037 A document-unique identifier for the element, typically used as a reference point when signing.

1038 validUntil [Optional]

1039 Optional attribute indicates the expiration time of the metadata contained in the element and any  
1040 contained elements.

1041 cacheDuration [Optional]

1042 Optional attribute indicates the maximum length of time a consumer should cache the metadata  
1043 contained in the element and any contained elements.

1044 <ds:Signature> [Optional]

1045 An XML signature that authenticates the containing element and its contents, as described in  
1046 Section 3.

1047 <Extensions> [Optional]

1048 This contains optional metadata extensions that are agreed upon between a metadata publisher  
1049 and consumer. Extension elements MUST be namespace-qualified by a non-SAML-defined  
1050 namespace.



1051 <AffiliateMember> [One or More]  
1052 One or more elements enumerating the members of the affiliation by specifying each member's  
1053 unique identifier. See also Section 8.3.6 of [SAMLCore].

1054 <KeyDescriptor> [Zero or More]  
1055 Optional sequence of elements that provides information about the cryptographic keys that the  
1056 affiliation uses as a whole, as distinct from keys used by individual members of the affiliation,  
1057 which are published in the metadata for those entities.

1058 Arbitrary namespace-qualified attributes from non-SAML-defined namespaces may also be included.

1059 [E76]A validUntil or cacheDuration attribute MAY be used to impose a shorter expiration or cache  
1060 duration than that of the parent or root element, but never a longer one; the smaller value takes  
1061 precedence.

1062 The following schema fragment defines the <AffiliationDescriptor> element and its  
1063 **AffiliationDescriptorType** complex type:

```
1064 <element name="AffiliationDescriptor" type="md:AffiliationDescriptorType"/>
1065 <complexType name="AffiliationDescriptorType">
1066   <sequence>
1067     <element ref="ds:Signature" minOccurs="0"/>
1068     <element ref="md:Extensions" minOccurs="0"/>
1069     <element ref="md:AffiliateMember" maxOccurs="unbounded"/>
1070     <element ref="md:KeyDescriptor" minOccurs="0" maxOccurs="unbounded"/>
1071   </sequence>
1072   <attribute name="affiliationOwnerID" type="md:entityIDType"
1073 use="required"/>
1074   <attribute name="validUntil" type="dateTime" use="optional"/>
1075   <attribute name="cacheDuration" type="duration" use="optional"/>
1076   <attribute name="ID" type="ID" use="optional"/>
1077   <anyAttribute namespace="##other" processContents="lax"/>
1078 </complexType>
1079 <element name="AffiliateMember" type="md:entityIDType"/>
```

## 1080 2.6 Examples

1081 The following is an example of metadata for a SAML system entity acting as an identity provider and an  
1082 attribute authority. A signature is shown as a placeholder, without the actual content.  
1083

```
1084 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1085 xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1086 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1087 entityID="https://IdentityProvider.com/SAML">
1088   <ds:Signature>...</ds:Signature>
1089   <IDPSSODescriptor WantAuthnRequestsSigned="true"
1090 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1091     <KeyDescriptor use="signing">
1092       <ds:KeyInfo>
1093         <ds:KeyName>IdentityProvider.com SSO Key</ds:KeyName>
1094       </ds:KeyInfo>
1095     </KeyDescriptor>
1096     <ArtifactResolutionService isDefault="true" index="0"
1097       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1098       Location="https://IdentityProvider.com/SAML/Artifact"/>
1099     <SingleLogoutService
1100       Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1101       Location="https://IdentityProvider.com/SAML/SLO/SOAP"/>
1102     <SingleLogoutService
1103       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1104       Location="https://IdentityProvider.com/SAML/SLO/Browser"
1105       ResponseLocation="https://IdentityProvider.com/SAML/SLO/Response"/>
1106     <NameIDFormat>
1107       urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
```

```

1108     </NameIDFormat>
1109     <NameIDFormat>
1110         urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1111     </NameIDFormat>
1112     <NameIDFormat>
1113         urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1114     </NameIDFormat>
1115     <SingleSignOnService
1116         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1117         Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1118     <SingleSignOnService
1119         Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1120         Location="https://IdentityProvider.com/SAML/SSO/Browser"/>
1121     <saml:Attribute
1122         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1123         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1124         FriendlyName="eduPersonPrincipalName">
1125     </saml:Attribute>
1126     <saml:Attribute
1127         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1128         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1129         FriendlyName="eduPersonAffiliation">
1130         <saml:AttributeValue>member</saml:AttributeValue>
1131         <saml:AttributeValue>student</saml:AttributeValue>
1132         <saml:AttributeValue>faculty</saml:AttributeValue>
1133         <saml:AttributeValue>employee</saml:AttributeValue>
1134         <saml:AttributeValue>staff</saml:AttributeValue>
1135     </saml:Attribute>
1136 </IDPSSODescriptor>
1137 <AttributeAuthorityDescriptor
1138     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1139     <KeyDescriptor use="signing">
1140         <ds:KeyInfo>
1141             <ds:KeyName>IdentityProvider.com AA Key</ds:KeyName>
1142         </ds:KeyInfo>
1143     </KeyDescriptor>
1144     <AttributeService
1145         Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1146         Location="https://IdentityProvider.com/SAML/AA/SOAP"/>
1147     <AssertionIDRequestService
1148         Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
1149         Location="https://IdentityProvider.com/SAML/AA/URI"/>
1150     <NameIDFormat>
1151         urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
1152     </NameIDFormat>
1153     <NameIDFormat>
1154         urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
1155     </NameIDFormat>
1156     <NameIDFormat>
1157         urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1158     </NameIDFormat>
1159     <saml:Attribute
1160         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1161         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
1162         FriendlyName="eduPersonPrincipalName">
1163     </saml:Attribute>
1164     <saml:Attribute
1165         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1166         Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
1167         FriendlyName="eduPersonAffiliation">
1168         <saml:AttributeValue>member</saml:AttributeValue>
1169         <saml:AttributeValue>student</saml:AttributeValue>
1170         <saml:AttributeValue>faculty</saml:AttributeValue>
1171         <saml:AttributeValue>employee</saml:AttributeValue>
1172         <saml:AttributeValue>staff</saml:AttributeValue>
1173     </saml:Attribute>
1174 </AttributeAuthorityDescriptor>

```

```

1175     <Organization>
1176         <OrganizationName xml:lang="en">Identity Providers R
1177 US</OrganizationName>
1178         <OrganizationDisplayName xml:lang="en">
1179 Identity Providers R US, a Division of Lerxst Corp.
1180         </OrganizationDisplayName>
1181         <OrganizationURL
1182 xml:lang="en">https://IdentityProvider.com</OrganizationURL>
1183         </Organization>
1184 </EntityDescriptor>
1185

```

1186 The following is an example of metadata for a SAML system entity acting as a service provider. A  
1187 signature is shown as a placeholder, without the actual content. For illustrative purposes, the service is  
1188 one that does not require users to uniquely identify themselves, but rather authorizes access on the basis  
1189 of a role-like attribute.  
1190

```

1191 <EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
1192 xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1193 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1194 entityID="https://ServiceProvider.com/SAML">
1195 <ds:Signature>...</ds:Signature>
1196 <SPSSODescriptor AuthnRequestsSigned="true"
1197 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
1198 <KeyDescriptor use="signing">
1199 <ds:KeyInfo>
1200 <ds:KeyName>ServiceProvider.com SSO Key</ds:KeyName>
1201 </ds:KeyInfo>
1202 </KeyDescriptor>
1203 <KeyDescriptor use="encryption">
1204 <ds:KeyInfo>
1205 <ds:KeyName>ServiceProvider.com Encrypt Key</ds:KeyName>
1206 </ds:KeyInfo>
1207 <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmenc#rsa-
1208 1_5"/>
1209 </KeyDescriptor>
1210 <SingleLogoutService
1211 Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
1212 Location="https://ServiceProvider.com/SAML/SLO/SOAP"/>
1213 <SingleLogoutService
1214 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
1215 Location="https://ServiceProvider.com/SAML/SLO/Browser"
1216 ResponseLocation="https://ServiceProvider.com/SAML/SLO/Response"/>
1217 <NameIDFormat>
1218 urn:oasis:names:tc:SAML:2.0:nameid-format:transient
1219 </NameIDFormat>
1220 <AssertionConsumerService isDefault="true" index="0"
1221 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
1222 Location="https://ServiceProvider.com/SAML/SSO/Artifact"/>
1223 <AssertionConsumerService index="1"
1224 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
1225 Location="https://ServiceProvider.com/SAML/SSO/POST"/>
1226 <AttributeConsumingService index="0">
1227 <ServiceName xml:lang="en">Academic Journals R US</ServiceName>
1228 <RequestedAttribute
1229 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1230 Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7"
1231 FriendlyName="eduPersonEntitlement">
1232 <saml:AttributeValue>
1233 https://ServiceProvider.com/entitlements/123456789
1234 </saml:AttributeValue>
1235 </RequestedAttribute>
1236 </AttributeConsumingService>
1237 </SPSSODescriptor>
1238 <Organization>
1239 <OrganizationName xml:lang="en">Academic Journals R
1240 US</OrganizationName>
1241 <OrganizationDisplayName xml:lang="en">

```

```
1242         Academic Journals R US, a Division of Dirk Corp.
1243         </OrganizationDisplayName>
1244         <OrganizationURL
1245 xml:lang="en">https://ServiceProvider.com</OrganizationURL>
1246     </Organization>
1247 </EntityDescriptor>
```

---

## 1248 3 Signature Processing

1249 Various elements in a metadata instance can be digitally signed (as indicated by the element's inclusion of  
1250 a `<ds:Signature>` element), with the following benefits:

- 1251 • Metadata integrity
- 1252 • Authentication of the metadata by a trusted signer

1253 A digital signature is not always required, for example if the relying party obtains the information directly  
1254 from the publishing entity directly (with no intermediaries) through a secure channel, with the entity having  
1255 authenticated to the relying party by some means other than a digital signature.

1256 Many different techniques are available for "direct" authentication and secure channel establishment  
1257 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,  
1258 the applicable security requirements depend on the communicating applications.

1259 Additionally, elements can inherit signatures on enclosing parent elements that are themselves signed.

1260 In the absence of such context, it is RECOMMENDED that at least the root element of a metadata  
1261 instance be signed.

### 1262 3.1 XML Signature Profile

1263 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility  
1264 and many choices. This section details the constraints on these facilities so that metadata processors do  
1265 not have to deal with the full generality of XML Signature processing. This usage makes specific use of  
1266 the `xs:ID`-typed attributes optionally present on the elements to which signatures can apply. These  
1267 attributes are collectively referred to in this section as the identifier attributes.

#### 1268 3.1.1 Signing Formats and Algorithms

1269 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and  
1270 detached.

1271 SAML metadata MUST use enveloped signatures when signing the elements defined in this specification.  
1272 SAML processors SHOULD support the use of RSA signing and verification for public key operations in  
1273 accordance with the algorithm identified by <http://www.w3.org/2000/09/xmlsig#rsa-sha1>.

#### 1274 3.1.2 References

1275 Signed metadata elements MUST supply a value for the identifier attribute on the signed element. The  
1276 element may or may not be the root element of the actual XML document containing the signed metadata  
1277 element.

1278 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute  
1279 value of the metadata element being signed. For example, if the identifier attribute value is "foo", then the  
1280 URI attribute in the `<ds:Reference>` element MUST be "#foo".

1281 As a consequence, a metadata element's signature MUST apply to the content of the signed element and  
1282 any child elements it contains.

#### 1283 3.1.3 Canonicalization Method

1284 SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the

1285 <ds:CanonicalizationMethod> element of <ds:SignedInfo>, and as a <ds:Transform>  
1286 algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML metadata  
1287 embedded in an XML context can be verified independent of that context.

### 1288 **3.1.4 Transforms**

1289 Signatures in SAML metadata SHOULD NOT contain transforms other than the enveloped signature  
1290 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive  
1291 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or  
1292 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

1293 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do  
1294 not, verifiers MUST ensure that no content of the signed metadata element is excluded from the  
1295 signature. This can be accomplished by establishing out-of-band agreement as to what transforms are  
1296 acceptable, or by applying the transforms manually to the content and reverifying the result as consisting  
1297 of the same SAML metadata.

### 1298 **3.1.5 KeyInfo**

1299 XML Signature [XMLSig] defines usage of the <ds:KeyInfo> element. SAML does not require the  
1300 use of <ds:KeyInfo> nor does it impose any restrictions on its use. Therefore, <ds:KeyInfo> MAY  
1301 be absent.

---

## 1302 4 Metadata Publication and Resolution

1303 Two mechanisms are provided for an entity to publish (and for a consumer to resolve the location of)  
1304 metadata documents: via a "well-known-location" by directly dereferencing the entity's unique identifier (a  
1305 URI variously referred to as an *entityID* or *providerID*), or indirectly by publishing the location of metadata  
1306 in the DNS. Other out-of-band mechanisms are of course also permitted. A consumer that supports both  
1307 approaches defined in this document MUST attempt resolution via DNS before using the "well-known-  
1308 location" mechanism.

1309 When retrieval requires network transport of the document, the transport SHOULD be protected with  
1310 mechanisms providing server authentication and integrity protection. For example, HTTP-based resolution  
1311 SHOULD be protected with TLS/SSL [RFC2246] as amended by [RFC3546].

1312 Various mechanisms are described in this section to aid in establishing trust in the accuracy and  
1313 legitimacy of metadata, including use of XML signatures, SSL/TLS server authentication, and DNS  
1314 signatures. Regardless of the mechanism(s) used, relying parties SHOULD have some means by which to  
1315 establish trust in metadata information before relying on it.

### 1316 4.1 Publication and Resolution via Well-Known Location

1317 The following sections describe publication and resolution of metadata by means of a well-known location.

#### 1318 4.1.1 Publication

1319 Entities MAY publish their metadata documents at a well known location by placing the document at the  
1320 location denoted by its unique identifier, which MUST be in the form of a URL (rather than a URN). See  
1321 Section 8.3.6 of [SAMLCore] for more information about such identifiers. It is STRONGLY  
1322 RECOMMENDED that `https` URLs be used for this purpose. An indirection mechanism supported by the  
1323 URL scheme (such as an HTTP 1.1 302 redirect) MAY be used if the document is not placed directly at  
1324 the location. If the publishing protocol permits MIME-based identification of content types, the content type  
1325 of the metadata instance MUST be `application/samlmetadata+xml`.

1326 The XML document provided at the well-known location MUST describe the metadata only for the entity  
1327 represented by the unique identifier (that is, the root element MUST be an `<EntityDescriptor>` with  
1328 an `entityID` matching the location). If other entities need to be described, the  
1329 `<AdditionalMetadataLocation>` element MUST be used. Thus the `<EntitiesDescriptor>`  
1330 element MUST NOT be used in documents published using this mechanism, since a group of entities are  
1331 not defined by such an identifier.

#### 1332 4.1.2 Resolution

1333 If an entity's unique identifier is a URL, metadata consumers MAY attempt to resolve an entity's unique  
1334 identifier directly, in a scheme-specific manner, by dereferencing the identifier.

### 1335 4.2 Publishing and Resolution via DNS

1336 To improve the accessibility of metadata documents and provide additional indirection between an entity's  
1337 unique identifier and the location of metadata, entities MAY publish their metadata document locations in a  
1338 zone of their corresponding DNS [RFC1034]. The entity's unique identifier (a URI) is used as the input to  
1339 the process. Since URIs are flexible identifiers, location publication methods and the resolution process  
1340 are determined by the URI's scheme and fully-qualified name. URI locations for metadata are

1341 subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in [RFC2915]  
1342 and [RFC3403].

1343 It is RECOMMENDED that entities publish their resource records in signed zone files using [RFC2535]  
1344 [E66][RFC4035] such that relying parties may establish the validity of the published location and authority  
1345 of the zone, and integrity of the DNS response. If DNS zone signatures are present, relying parties MUST  
1346 properly validate the signature.

## 1347 **4.2.1 Publication**

1348 This specification makes use of the NAPTR resource record described in [RFC2915] and [RFC3403].  
1349 Familiarity with these documents is encouraged.

1350 Dynamic Delegation Discovery System (DDDS) [RFC3401] is a general purpose system for the retrieval of  
1351 information based on an application-specific input string and the application of well known rules to  
1352 transform that string until a terminal condition is reached requiring a look-up into an application-specific  
1353 defined database or resolution of a URL based on the rules defined by the application. DDDS defines a  
1354 specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS  
1355 necessary to apply DDDS rules.

1356 Entities MAY publish separate URLs when multiple metadata documents need to be distributed, or when  
1357 different metadata documents are required due to multiple trust relationships that require separate keying  
1358 material, or when service interfaces require separate metadata declarations. This may be accomplished  
1359 through the use of the optional <AdditionalMetadataLocation> element, or through the regexp  
1360 facility and multiple service definition fields in the NAPTR resource record itself.

1361 If the publishing protocol permits MIME-based identification of content types, the content type of the  
1362 metadata instance MUST be `application/samlmetadata+xml`.

1363 If the entity's unique identifier is a URN, publication of the corresponding metadata location proceeds as  
1364 specified in [RFC3404]. Otherwise, the resolution of the metadata location proceeds as specified below.

1365 The following is the application-specific profile of DDDS for SAML metadata resolution.

### 1366 **4.2.1.1 First Well Known Rule**

1367 The "first well-known-rule" for processing SAML metadata resolution is to parse the entity's unique  
1368 identifier and extract the fully-qualified domain name (subexpression 3) as described in Section 4.2.3.1.

### 1369 **4.2.1.2 The Order Field**

1370 The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY  
1371 provide multiple NAPTR resource records which MUST be processed by the resolver application in the  
1372 order indicated by this field.

### 1373 **4.2.1.3 The Preference Field**

1374 For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving  
1375 application. The resolving application MAY ignore this order, in cases where the service field value does  
1376 not meet the resolver's requirements (e.g.: the resource record returns a protocol the application does not  
1377 support).



#### 1378 4.2.1.4 The Flag Field

1379 SAML metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying  
1380 additional resource records are to be processed). The "U" flag indicates that the output of the rule is a  
1381 URI.

#### 1382 4.2.1.5 The Service Field

1383 The SAML-specific service field, as described in the following BNF, declares the modes by which instance  
1384 document(s) shall be made available:

```
1385 servicefield = 1("PID2U" / "NID2U") "+" proto [*(":" class) *(":" servicetype)]  
1386 proto = 1("https" / "uddi")  
1387 class = 1[ "entity" / "entitygroup" ]  
1388 servicetype = 1(si / "spssso" / "idpssso" / "authn" / "authnauth" / "pdp" / "attrauth" /  
1389 alphanum )  
1390 si = "si" [ ":" alphanum ] [ ":" endpoint ]  
1391 alphanum = 1*32 (ALPHA / DIGIT)
```

1392 where:

- 1393 • servicefield PID2U resolves an entity's unique identifier to metadata URL.
- 1394 • servicefield NID2U resolves a principal's <NameID> into a metadata URL.
- 1395 • proto describes the retrieval protocol (`https` or `uddi`). In the case of UDDI, the URL will be an  
1396 http(s) URL referencing a WSDL document.
- 1397 • class identifies whether the referenced metadata document describes a single entity, or multiple.  
1398 In the latter case, the referenced document MUST contain the entity defined by the original unique  
1399 identifier as a member of a group of entities within the document itself such as an  
1400 <AffiliationDescriptor> or <EntitiesDescriptor>.
- 1401 • servicetype allows an entity to publish metadata for distinct roles and services as separate  
1402 documents. Resolvers who encounter multiple servicetype declarations will dereference the  
1403 appropriate URI, depending on which service is required for an operation (e.g.: an entity operating  
1404 both as an identity provider and a service provider can publish metadata for each role at different  
1405 locations). The `authn` service type represents a <SingleSignOnService> endpoint.
- 1406 • si (with optional endpoint component) allows the publisher to either directly publish the metadata  
1407 for a service instance, or by articulating a SOAP endpoint (using `endpoint`).

1408 For example:

- 1409 • PID2U+https:entity - represents the entity's complete metadata document available via the  
1410 https protocol
- 1411 • PID2U+uddi:entity:si:foo - represents the WSDL document location that describes a service  
1412 instance "foo"
- 1413 • PID2U+https:entitygroup:idpssso - represents the metadata for a group of entities acting as  
1414 SSO identity providers, of which the original entity is a member.
- 1415 • NID2U+https:idp - represents the metadata for the SSO identity provider of a principal

#### 1416 4.2.1.6 The Regex and Replacement Fields

1417 The expected output after processing the input string through the regex MUST be a valid `https` URL or  
1418 UDDI node (WSDL document) address.

## 1419 4.2.2 NAPTR Examples

### 1420 4.2.2.1 Entity Metadata NAPTR Examples

1421 Entities publish metadata URLs in the following manner:

```
1422 $ORIGIN provider.biz
1423
1424 ;; order pref f service regexp or replacement
1425
1426 IN NAPTR 100 10 "U" PID2U+https:entity
1427     "!^.*$!https://host.provider.biz/some/directory/trust.xml!" ""
1428 IN NAPTR 110 10 "U" PID2U+https: entity:trust
1429     "!^.*$!https://foo.provider.biz:1443/mdtrust.xml!" ""
1430 IN NAPTR 125 10 "U" PID2U+https:"
1431 IN NAPTR 110 10 "U" PID2U+uddi:entity
1432     "!^.*$!https://this.uddi.node.provider.biz/libmd.wsdl" ""
```

### 1433 4.2.2.2 Name Identifier Examples

1434 A principal's employer `example.int` operates an identity provider which may be used by an office supply  
1435 company to authenticate authorized buyers. The supplier takes a users' email address  
1436 `buyer@example.int` as input to the resolution process, and parses the email address to extract the  
1437 FQDN (`example.int`). The employer publishes the following NAPTR record in the `example.int` DNS:

```
1438 $ORIGIN example.int
1439
1440 IN NAPTR 100 10 "U" NID2U+https:authn
1441     "!^([\^@]+)@(.*)$!https://serv.example.int:8000/cgi-bin/getmd?\1!" ""
1442 IN NAPTR 100 10 "U" NID2U+https:idp
1443     "!^([\^@]+)@(.*)$!https://auth.example.int/app/auth?\1" ""
```

## 1444 4.2.3 Resolution

1445 When resolving metadata for an entity via the DNS, the unique identifier of the entity is used as the initial  
1446 input into the resolution process, rather than as an actual location Proceed as follows:

- 1447 • If the unique identifier is a URN, proceed with the resolution steps as defined in [RFC3404].
- 1448 • Otherwise, parse the identifier to obtain the fully-qualified domain name.
- 1449 • Query the DNS for NAPTR resource records of the domain iteratively until a terminal resource  
1450 record is returned.
- 1451 • Identify which resource record to use based on the service fields, then order fields, then preference  
1452 fields of the result set.
- 1453 • Obtain the document(s) at the provided location(s) as required by the application.

### 1454 4.2.3.1 Parsing the Unique Identifier

1455 To initiate the resolution of the location of the metadata information, it will be necessary in some cases to  
1456 decompose the entity's unique identifier (expressed as a URI) into one or more atomic elements.

1457 The following regular expression should be used when initiating the decomposition process:

```
1458 ^([\^:/?#]+)?/*([\^:/?#]*@)?(((\^/??:#)*\.)*((\^/?#:\.]+)\.([\^/?#:\.]+)))
1459 (:\\d+)?([\^?#]*)(\^[^#]*)?(\#.*)?$
1460           1           2           3         4           5           6           7           8
1461           9           10          11
```

1462 Subexpression 3 MUST result in a Fully-Qualified Domain Name (FQDN), which will be the basis for  
1463 retrieving metadata locations from this zone.

#### 1464 **4.2.3.2 Obtaining Metadata via the DNS**

1465 Upon completion of the parsing of the identifier, the application then performs a DNS query for the resulting  
1466 domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses.  
1467 Applications MAY exclude from the result set any service definitions that do not concern the present  
1468 request operations.

1469 Resolving applications MUST subsequently order the result set according to the order field, and MAY  
1470 order the result set based on the preference set. Resolvers are NOT REQUIRED to follow the ordering of  
1471 the preferences field. The resulting NAPTR resource record(s) are operated on iteratively (based on the  
1472 order flag) until a terminal NAPTR resource record is reached.

1473 The result will be a well-formed, absolute URL, which is then used to retrieve the metadata document.

#### 1474 **4.2.4 Metadata Location Caching**

1475 Location caching MUST NOT exceed the TTL of the DNS zone from which the location was derived.  
1476 Resolvers MUST obtain a fresh copy of the metadata location upon reaching the expiration of the TTL of  
1477 the zone.

1478 Publishers of metadata documents should carefully consider the TTL of the zone when making changes  
1479 to metadata document locations. Should such a location change occur, a publisher MUST either keep the  
1480 document at both the old and new location until all conforming resolvers are certain to have the updated  
1481 location (e.g.: time of zone change + TTL), or provide an HTTP Redirect [RFC2616] response at the old  
1482 location specifying the new location.

### 1483 **4.3 Post-Processing of Metadata**

1484 The following sections describe the post-processing of metadata.

#### 1485 **4.3.1 Metadata Instance Caching**

1486 Document caching MUST NOT exceed the `validUntil` or `cacheDuration` attribute of the subject  
1487 element(s). If metadata elements have parent elements which contain caching policies, the parent  
1488 element takes precedence.

1489 To properly process the `cacheDuration` attribute, consumers MUST retain the date and time when the  
1490 document was retrieved.

1491 When a document or element has expired, the consumer MUST retrieve a fresh copy, which may require  
1492 a refresh of the document location(s). Consumers SHOULD process document cache processing  
1493 according to [RFC2616] Section 13, and MAY request the Last-Modified date and time from the HTTP  
1494 server. Publishers SHOULD ensure acceptable cache processing as described in [RFC2616] (Section  
1495 10.3.5 304 Not Modified).

#### 1496 **4.3.2 Handling of HTTPS Redirects**

1497 Publishers MAY issue an HTTP Redirect (301 Moved Permanently, 302 or 307 Temporary Redirect)  
1498 [RFC2616], and user agents MUST follow the specified URL in the Redirect response. Redirects  
1499 SHOULD be of the same protocol as the initial request.

#### 1500 **4.3.3 Processing of XML Signatures and General Trust Processing**

1501 Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and  
1502 for the trust ascribed to the entity described by such metadata:

- 1503 • Trust derived from the signature of the DNS zone from which the metadata location URL was

1504 resolved, ensuring accuracy of the metadata document location(s)

1505 • Trust derived from signature processing of the metadata document itself, ensuring the integrity of  
1506 the XML document

1507 • Trust derived from the SSL/TLS server authentication of the metadata location URL, ensuring the  
1508 identity of the publisher of the metadata

1509 Post-processing of the metadata document MUST include signature processing at the XML-document  
1510 level and MAY include one of the other two processes. Specifically, the relying party MAY choose to trust  
1511 any of the cited authorities in the resolution and parsing process. Publishers of metadata MUST employ a  
1512 document-integrity mechanism and MAY employ any of the other two processing profiles to establish trust  
1513 in the metadata document, governed by implementation policies.

#### 1514 **4.3.3.1 Processing Signed DNS Zones**

1515 Verification of DNS zone signature SHOULD be processed, if present, as described in [\[E66\]\[RFC2535\]](#)  
1516 [\[RFC4035\]](#).

#### 1517 **4.3.3.2 Processing Signed Documents and Fragments**

1518 Published metadata documents SHOULD be signed, as described in Section 3, either by a certificate  
1519 issued to the subject of the document, or by another trusted party. Publishers MAY consider signatures of  
1520 other parties as a means of trust conveyance.

1521 Metadata consumers MUST validate signatures, when present, on the metadata document as described  
1522 by Section 3.

#### 1523 **4.3.3.3 Processing Server Authentication during Metadata Retrieval via TLS/SSL**

1524 It is STRONGLY RECOMMENDED that publishers implement TLS/SSL URLs; therefore, consumers  
1525 SHOULD consider the trust inherited from the issuer of the TLS/SSL certificate. Publication URLs may not  
1526 always be located in the domain of the subject of the metadata document; therefore, consumers SHOULD  
1527 NOT presume certificates whose subject is the entity in question, as it may be hosted by another trusted  
1528 party.

1529 As the basis of this trust may not be available against a cached document, other mechanisms SHOULD  
1530 be used under such circumstances.

---

## 5 References

1531

- 1532 [RFC1034] P. Mockapetris. *Domain Names – Concepts and Facilities*. IETF RFC 1034,  
1533 November 1987. See <http://www.ietf.org/rfc/rfc1034.txt>.
- 1534 [RFC2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*,  
1535 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1536 [RFC2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999.  
1537 See <http://www.ietf.org/rfc/rfc2246.txt>.
- 1538 [~~E66~~][RFC2535] ~~D. Eastlake. *Domain Name System Security Extensions*. IETF RFC 2535, March  
1539 1999. See <http://www.ietf.org/rfc/rfc2535.txt>.~~
- 1540 [RFC2616] R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. IETF RFC 2616, June  
1541 1999. See <http://www.ietf.org/rfc/rfc2616.txt>.
- 1542 [RFC2915] M. Mealling. *The Naming Authority Pointer (NAPTR) DNS Resource Record*.  
1543 IETF RFC 2915, September 2000. See <http://www.ietf.org/rfc/rfc2915.txt>.
- 1544 [RFC3401] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part One: The  
1545 Comprehensive DDDS*. IETF RFC 3401, October 2002. See  
1546 <http://www.ietf.org/rfc/rfc3401.txt>.
- 1547 [RFC3403] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Three: The  
1548 Domain Name System (DNS) Database*. IETF RFC 3403, October 2002. See  
1549 <http://www.ietf.org/rfc/rfc3403.txt>.
- 1550 [RFC3404] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Four: The  
1551 Uniform Resource Identifiers (URI) Resolution Application*. IETF RFC 3404,  
1552 October 2002. See <http://www.ietf.org/rfc/rfc3404.txt>.
- 1553 [RFC3546] S. Blake-Wilson et al. *Transport Layer Security (TLS) Extensions*. IETF RFC  
1554 3546, June 2003. See <http://www.ietf.org/rfc/rfc3546.txt>.
- 1555 [~~E66~~][RFC4035] ~~R. Arends et al. *Protocol Modifications for the DNS Security Extensions*. IETF  
1556 RFC 4035, March 2005. See <http://www.ietf.org/rfc/rfc4035.txt>.~~
- 1557 [SAMLBind] S. Cantor et al. *Bindings for the OASIS Security Assertion Markup Language  
1558 (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os.  
1559 See <http://www.oasis-open.org/committees/security/>.
- 1560 [SAMLConform] P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion  
1561 Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-  
1562 conformance-2.0-os. <http://www.oasis-open.org/committees/security/>.
- 1563 [SAMLCore] S. Cantor et al. *Assertions and Protocols for the OASIS Security Assertion  
1564 Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-  
1565 core-2.0-os. See <http://www.oasis-open.org/committees/security/>.
- 1566 [SAMLMeta-xsd] S. Cantor et al. *SAML metadata schema*. OASIS SSTC, March 2005. Document  
1567 ID saml-schema-metadata-2.0. See [http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>1568 open.org/committees/security/).
- 1569 [SAMLProf] S. Cantor et al. *Profiles for the OASIS Security Assertion Markup Language  
1570 (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-profiles-2.0-os. See  
1571 <http://www.oasis-open.org/committees/security/>.
- 1572 [SAMLSec] F. Hirsch et al. *Security and Privacy Considerations for the OASIS Security  
1573 Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document  
1574 ID saml-sec-consider-2.0-os. See [http://www.oasis-  
open.org/committees/security/](http://www.oasis-<br/>1575 open.org/committees/security/).
- 1576 [Schema1] H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web  
1577 Consortium Recommendation, May 2001. See <http://www.w3.org/TR/xmlschema->

1578		1/. Note that this specification normatively references [Schema2], listed below.
1579	<b>[Schema2]</b>	P. V. Biron et al. <i>XML Schema Part 2: Datatypes</i> . World Wide Web Consortium Recommendation, May 2001. See <a href="http://www.w3.org/TR/xmlschema-">http://www.w3.org/TR/xmlschema-</a>
1580		
1581	<b>[XMLEnc]</b>	D. Eastlake et al. <i>XML-Encryption Syntax and Processing</i> ,
1582		<a href="http://www.w3.org/TR/xmlenc-core/">http://www.w3.org/TR/xmlenc-core/</a> , World Wide Web Consortium.
1583	<b>[XMLSig]</b>	D. Eastlake et al. <i>XML-Signature Syntax and Processing. [E74] Second</i>
1584		<i>Edition</i> . World Wide Web Consortium, June 2008. See,
1585		<a href="http://www.w3.org/TR/xmlsig-core/">http://www.w3.org/TR/xmlsig-core/</a> , World Wide Web Consortium.

---

1586 **Appendix A.Registration of MIME media type**  
1587 **application/samlmetadata+xml**

1588 **Introduction**

1589 This document defines a MIME media type -- `application/samlmetadata+xml` -- for use  
1590 with the XML serialization of Security Assertion Markup Language metadata.  
1591

1592 SAML is a work product of the OASIS Security Services Technical Committee [SSTC]. The SAML  
1593 specifications define XML-based constructs with which one may make, and convey, security  
1594 assertions. Using SAML, one can assert that an authentication event pertaining to some subject  
1595 has occurred and convey said assertion to a relying party, for example.  
1596

1597 SAML profiles require agreements between system entities regarding identifiers, binding support,  
1598 endpoints, certificates, keys, and so forth. Such information is treated as metadata by SAML v2.0.  
1599 [SAMLv2Meta] specifies this metadata, as well as specifying metadata publication and resolution  
1600 mechanisms. If the publishing protocol permits MIME-based identification of content types, then  
1601 use of the `application/samlmetadata+xml` MIME media type is required.

1602 **MIME media type name**

1603 `application`

1604 **MIME subtype name**

1605 `samlmetadata+xml`

1606 **Required parameters**

1607 None

1608 **Optional parameters**

1609 `charset`

1610 Same as `charset` parameter of `application/xml` [RFC3023].

1611 **Encoding considerations**

1612 Same as for `application/xml` [RFC3023].

1613 **Security considerations**

1614 Per their specification, `samlmetadata+xml` typed objects do not contain executable content.  
1615 However, these objects are XML-based [XML], and thus they have all of the general security  
1616 considerations presented in Section 10 of [RFC3023].

1617 SAML metadata [SAMLv2Meta] contains information whose integrity and authenticity is important  
1618 – identity provider and service provider public keys and endpoint addresses, for example.

1619 To counter potential issues, the publisher may sign `samlmetadata+xml` typed objects. Any such  
1620 signature should be verified by the recipient of the data - both as a valid signature, and as being  
1621 the signature of the publisher.

1622 Additionally, various of the publication protocols, e.g. HTTP-over-TLS/SSL, offer means for  
1623 ensuring the authenticity of the publishing party and for protecting the metadata in transit.

1624 [SAMLv2Meta] also defines prescriptive metadata caching directives, as well as guidance on  
1625 handling HTTPS redirects, trust processing, server authentication, and related items.  
1626 For a more detailed discussion of SAML v2.0 metadata and its security considerations, please  
1627 see [SAMLv2Meta]. For a discussion of overall SAML v2.0 security considerations and specific  
1628 security-related design features, please refer to the SAML v2.0 specifications listed in the below  
1629 bibliography. The specifications containing security-specific information are explicitly listed.

## 1630 Interoperability considerations

1631 SAML v2.0 metadata explicitly supports identifying the protocols and versions supported by the  
1632 identified entities. For example, an identity provider entity can be denoted as supporting SAML  
1633 v2.0 [SAMLv2.0], SAML v1.1 [SAMLv1.1], Liberty ID-FF 1.2 [LAPFF], or even other protocols if  
1634 they are unambiguously identifiable via URI [RFC2396]. This protocol support information is  
1635 conveyed via the `protocolSupportEnumeration` attribute of metadata objects of the  
1636 `RoleDescriptorType`.

## 1637 Published specification

1638 [SAMLv2Meta] explicitly specifies use of the `application/samlmetadata+xml` MIME media  
1639 type.

## 1640 Applications which use this media type

1641 Potentially any application implementing SAML v2.0, as well as those applications implementing  
1642 specifications based on SAML, e.g. those available from the Liberty Alliance [LAP].

## 1643 Additional information

### 1644 Magic number(s)

1645 In general, the same as for `application/xml` [RFC3023]. In particular, the XML root element of  
1646 the returned object will have a namespace-qualified name with:

- 1647
- 1648 – a local name of: `EntityDescriptor`, or  
1649 `AffiliationDescriptor`, or  
1650 `EntitiesDescriptor`
  
  - 1652 – a namespace URI of: `urn:oasis:names:tc:SAML:2.0:metadata`  
1653 (the SAMLv2.0 metadata namespace)

### 1654 File extension(s)

1655 None

### 1656 Macintosh File Type Code(s)

1657 None

## 1658 Person & email address to contact for further information

1659 This registration is made on behalf of the OASIS Security Services Technical Committee (SSTC)  
1660 Please refer to the SSTC website for current information on committee chairperson(s) and their  
1661 contact addresses: <http://www.oasis-open.org/committees/security/>. Committee members should  
1662 submit comments and potential errata to the [securityservices@lists.oasis-open.org](mailto:securityservices@lists.oasis-open.org) list. Others  
1663 should submit them by filling out the web form located at [http://www.oasis-  
1664 open.org/committees/comments/form.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security).



1665  
1666 Additionally, the SAML developer community email distribution list, [saml-dev@lists.oasis-](mailto:saml-dev@lists.oasis-open.org)  
1667 [open.org](http://lists.oasis-open.org), may be employed to discuss usage of the `application/samlmetadata+xml` MIME  
1668 media type. The "saml-dev" mailing list is publicly archived here: [http://lists.oasis-](http://lists.oasis-open.org/archives/saml-dev/)  
1669 [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  
1670 subscribe, send a message with the single word "subscribe" in the message body, to: [saml-dev-](mailto:saml-dev-request@lists.oasis-open.org)  
1671 [request@lists.oasis-open.org](mailto:saml-dev-request@lists.oasis-open.org).

## 1672 Intended usage

1673 COMMON

## 1674 Author/Change controller

1675 The SAML specification sets are a work product of the OASIS Security Services Technical  
1676 Committee (SSTC). OASIS and the SSTC have change control over the SAML specification sets.

## 1677 Bibliography

- 1678 [LAP] “*Liberty Alliance Project*”. See <http://www.projectliberty.org/>  
1679 [LAPFF] “Liberty Alliance Project: Federation Framework”. See  
1680 <http://www.projectliberty.org/resources/specifications.php#box1>  
1681 [OASIS] “*Organization for the Advancement of Structured Information Systems*”.  
1682 See <http://www.oasis-open.org/>  
1683 [RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifiers*  
1684 *(URI): Generic Syntax*. IETF RFC 2396, August 1998. Available at  
1685 <http://www.ietf.org/rfc/rfc2396.txt>  
1686 [RFC3023] M. Murata, S. St.Laurent, D. Kohn, “*XML Media Types*”, IETF Request for  
1687 Comments 3023, January 2001. Available as [http://www.rfc-](http://www.rfc-editor.org/rfc/rfc3023.txt)  
1688 [editor.org/rfc/rfc3023.txt](http://www.rfc-editor.org/rfc/rfc3023.txt)  
1689 [SAMLv1.1] OASIS Security Services Technical Committee, “*Security Assertion*  
1690 *Markup Language (SAML) Version 1.1 Specification Set*”. OASIS  
1691 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)  
1692 [open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip](http://www.oasis-open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip)  
1693 [SAMLv2.0] OASIS Security Services Technical Committee, “*Security Assertion*  
1694 *Markup Language (SAML) Version 2.0 Specification Set*”. OASIS  
1695 Standard, 15-Mar-2005. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip)  
1696 [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)  
1697 [SAMLv2Bind] S. Cantor et al., “*Bindings for the OASIS Security Assertion Markup*  
1698 *Language (SAML) V2.0*”. OASIS, March 2005. Document ID `saml-`  
1699 `bindings-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf)  
1700 [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)  
1701 [SAMLv2Core] S. Cantor et al., “*Assertions and Protocols for the OASIS Security*  
1702 *Assertion Markup Language (SAML) V2.0*”. OASIS, March 2005.  
1703 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)  
1704 [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)  
1705 [SAMLv2Meta] S. Cantor et al., “*Metadata for the OASIS Security Assertion Markup*  
1706 *Language (SAML) V2.0*”. OASIS SSTC, August 2004. Document ID `saml-`  
1707 `metadata-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf)  
1708 [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)  
1709 [SAMLv2Prof] S. Cantor et al., “*Profiles for the OASIS Security Assertion Markup*  
1710 *Language (SAML) V2.0*”. OASIS, March 2005. Document ID `saml-`  
1711 `profiles-2.0-os`. Available at: [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf)  
1712 [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)

1713	[SAMLv2Sec]	F. Hirsch et al., "Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0". OASIS, March 2005. Document ID saml-sec-consider-2.0-os. Available at: <a href="http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf">http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf</a>
1714		
1715		
1716		
1717	[SSTC]	"OASIS Security Services Technical Committee". See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a>
1718		
1719	[XML]	Bray, T., Paoli, J., Sperberg-McQueen, C.M. and E. Maler, François Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium Recommendation REC-xml, Feb 2004, Available as <a href="http://www.w3.org/TR/REC-xml/">http://www.w3.org/TR/REC-xml/</a>
1720		
1721		
1722		
1723		

---

## 1724 Appendix B. Acknowledgments

1725 The editors would like to acknowledge the contributions of the OASIS Security Services Technical  
1726 Committee, whose voting members at the time of publication were:

- 1727 • Conor Cahill, AOL
- 1728 • John Hughes, Atos Origin
- 1729 • Hal Lockhart, BEA Systems
- 1730 • Mike Beach, Boeing
- 1731 • Rebekah Metz, Booz Allen Hamilton
- 1732 • Rick Randall, Booz Allen Hamilton
- 1733 • Ronald Jacobson, Computer Associates
- 1734 • Gavenraj Sodhi, Computer Associates
- 1735 • Thomas Wisniewski, Entrust
- 1736 • Carolina Canales-Valenzuela, Ericsson
- 1737 • Dana Kaufman, Forum Systems
- 1738 • Irving Reid, Hewlett-Packard
- 1739 • Guy Denton, IBM
- 1740 • Heather Hinton, IBM
- 1741 • Maryann Hondo, IBM
- 1742 • Michael McIntosh, IBM
- 1743 • Anthony Nadalin, IBM
- 1744 • Nick Ragouzis, Individual
- 1745 • Scott Cantor, Internet2
- 1746 • Bob Morgan, Internet2
- 1747 • Peter Davis, Neustar
- 1748 • Jeff Hodges, Neustar
- 1749 • Frederick Hirsch, Nokia
- 1750 • Senthil Sengodan, Nokia
- 1751 • Abbie Barbir, Nortel Networks
- 1752 • Scott Kiestler, Novell
- 1753 • Cameron Morris, Novell
- 1754 • Paul Madsen, NTT
- 1755 • Steve Anderson, OpenNetwork
- 1756 • Ari Kermaier, Oracle
- 1757 • Vamsi Motukuru, Oracle
- 1758 • Darren Platt, Ping Identity
- 1759 • Prateek Mishra, Principal Identity
- 1760 • Jim Lien, RSA Security
- 1761 • John Linn, RSA Security
- 1762 • Rob Philpott, RSA Security
- 1763 • Dipak Chopra, SAP
- 1764 • Jahan Moreh, Sigaba
- 1765 • Bhavna Bhatnagar, Sun Microsystems
- 1766 • Eve Maler, Sun Microsystems

- 1767 • Ronald Monzillo, Sun Microsystems
- 1768 • Emily Xu, Sun Microsystems
- 1769 • Greg Whitehead, Trustgenix
- 1770

1771 The editors also would like to acknowledge the following former SSTC members for their contributions to  
1772 this or previous versions of the OASIS Security Assertions Markup Language Standard:

- 1773 • Stephen Farrell, Baltimore Technologies
- 1774 • David Orchard, BEA Systems
- 1775 • Krishna Sankar, Cisco Systems
- 1776 • Zahid Ahmed, CommerceOne
- 1777 • Tim Alsop, CyberSafe Limited
- 1778 • Carlisle Adams, Entrust
- 1779 • Tim Moses, Entrust
- 1780 • Nigel Edwards, Hewlett-Packard
- 1781 • Joe Pato, Hewlett-Packard
- 1782 • Bob Blakley, IBM
- 1783 • Marlena Erdos, IBM
- 1784 • Marc Chanliau, Netegrity
- 1785 • Chris McLaren, Netegrity
- 1786 • Lynne Rosenthal, NIST
- 1787 • Mark Skall, NIST
- 1788 • Charles Knouse, Oblix
- 1789 • Simon Godik, Overxeer
- 1790 • Charles Norwood, SAIC
- 1791 • Evan Prodromou, Securant
- 1792 • Robert Griffin, RSA Security (former editor)
- 1793 • Sai Allarvarpu, Sun Microsystems
- 1794 • Gary Ellison, Sun Microsystems
- 1795 • Chris Ferris, Sun Microsystems
- 1796 • Mike Myers, Traceroute Security
- 1797 • Phillip Hallam-Baker, VeriSign (former editor)
- 1798 • James Vanderbeek, Vodafone
- 1799 • Mark O'Neill, Vordel
- 1800 • Tony Palmer, Vordel

1801  
1802 Finally, the editors wish to acknowledge the following people for their contributions of material used as  
1803 input to the OASIS Security Assertions Markup Language specifications:

- 1804 • Thomas Gross, IBM
- 1805 • Birgit Pfitzmann, IBM

1806 [The editors also would like to gratefully acknowledge Jahan Moreh of Sigaba, who during his tenure on](#)  
1807 [the SSTC was the primary editor of the errata working document and who made major substantive](#)  
1808 [contributions to all of the errata materials.](#)

---

## 1809 Appendix C. Notices

1810 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
1811 might be claimed to pertain to the implementation or use of the technology described in this document or  
1812 the extent to which any license under such rights might or might not be available; neither does it represent  
1813 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to  
1814 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made  
1815 available for publication and any assurances of licenses to be made available, or the result of an attempt  
1816 made to obtain a general license or permission for the use of such proprietary rights by implementors or  
1817 users of this specification, can be obtained from the OASIS Executive Director.

1818 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or  
1819 other proprietary rights which may cover technology that may be required to implement this specification.  
1820 Please address the information to the OASIS Executive Director.

1821 **Copyright © OASIS Open 2005. All Rights Reserved.**

1822 This document and translations of it may be copied and furnished to others, and derivative works that  
1823 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and  
1824 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and  
1825 this paragraph are included on all such copies and derivative works. However, this document itself may  
1826 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as  
1827 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights  
1828 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it  
1829 into languages other than English.

1830 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
1831 or assigns.

1832 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
1833 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
1834 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR  
1835 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.