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

Working Draft, ~~August 2006~~1312 February 2006

Document identifier:

sstc-saml-profiles-errata-2.0-wd-054

Location:

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

Editors:

John Hughes, Atos Origin
Scott Cantor, Internet2
Jeff Hodges, Neustar
Frederick Hirsch, Nokia
Prateek Mishra, Principal Identity
Rob Philpott, RSA Security
~~Jahan Moreh, Sigaba (errata document editor)~~
Eve Maler, Sun Microsystems (errata ~~composite document~~ editor)

Contributors to the Errata:

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

SAML V2.0 Contributors:

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

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

67 **Abstract:**

68 The SAML V2.0 Profiles specification defines profiles for the use of SAML assertions and request-
69 response messages in communications protocols and frameworks, as well as profiles for SAML
70 attribute value syntax and naming conventions. This document, known as an “errata composite”,
71 combines corrections to reported errata with the original specification text. By design, the
72 corrections are limited to clarifications of ambiguous or conflicting specification text. This
73 document shows deletions from the original specification as struck-through text, and additions as
74 colored blue underlined text. The “[PE~~nn~~]” designations embedded in the text refer to particular
75 errata and their dispositions.

76 **Status:**

77 ~~The SAML V2.0 Profiles specification defines profiles for the use of SAML assertions and request-~~
78 ~~response messages in communications protocols and frameworks, as well as profiles for SAML~~
79 ~~attribute value syntax and naming conventions.~~ This errata composite document is a **working**
80 **draft** based on the **original** OASIS Standard document that had been produced by the Security
81 Services Technical Committee and approved by the OASIS membership on 1 March 2005. While
82 the errata corrections appearing here are non-normative, they reflect ~~the consensus of the TC~~
83 ~~about how to interpret the specification and are likely to be incorporated into any future standards-~~
84 ~~track revision of the SAML specifications.~~ changes specified by the Approved Errata document
85 (currently at Working Draft revision 02), which is on an OASIS standardization track. In case of
86 any discrepancy between this document and the Approved Errata, the latter has precedence. See
87 also the Errata Working Document (currently at revision 39), which provides background on the
88 changes specified here.

89 This document includes ~~errata~~ corrections for errata through revision 33 of the errata document,
90 including PE12, PE14, PE17, PE18, PE20, PE22, PE26, PE27, PE32, PE35, PE38, E39, PE40,
91 PE47, PE48, PE51, E52, E53, PPE53, E54, E55, Pand E56, E58, and E63. ~~Note that PE39 has~~
92 ~~not been corrected because of a conflict between it and PE53.~~

93 Committee members should submit comments and potential errata to the [security-](mailto:security-services@lists.oasis-open.org)
94 [services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should submit them by ~~filling out the web form~~
95 ~~located following the instructions~~ at [http://www.oasis-](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security)
96 [open.org/committees/comments/form.php?wg_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security).

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

Table of Contents

102	1 Introduction.....	7
103	1.1 Profile Concepts.....	7
104	1.2 Notation.....	7
105	2 Specification of Additional Profiles.....	10
106	2.1 Guidelines for Specifying Profiles.....	10
107	2.2 Guidelines for Specifying Attribute Profiles.....	10
108	3 Confirmation Method Identifiers.....	12
109	3.1 Holder of Key.....	12
110	3.2 Sender Vouches.....	13
111	3.3 Bearer.....	13
112	4 SSO Profiles of SAML.....	14
113	4.1 Web Browser SSO Profile.....	14
114	4.1.1 Required Information.....	14
115	4.1.2 Profile Overview.....	14
116	4.1.3 Profile Description.....	16
117	4.1.3.1 HTTP Request to Service Provider.....	16
118	4.1.3.2 Service Provider Determines Identity Provider.....	16
119	4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider.....	16
120	4.1.3.4 Identity Provider Identifies Principal.....	17
121	4.1.3.5 Identity Provider Issues <Response> to Service Provider.....	17
122	4.1.3.6 Service Provider Grants or Denies Access to User Agent.....	17
123	4.1.4 Use of Authentication Request Protocol.....	18
124	4.1.4.1 <AuthnRequest> Usage.....	18
125	4.1.4.2 <Response> Usage.....	18
126	4.1.4.3 <Response> Message Processing Rules.....	20
127	4.1.4.4 Artifact-Specific <Response> Message Processing Rules.....	20
128	4.1.4.5 POST-Specific Processing Rules.....	20
129	4.1.5 Unsolicited Responses.....	20
130	4.1.6 Use of Metadata.....	21
131	4.2 Enhanced Client or Proxy (ECP) Profile.....	21
132	4.2.1 Required Information.....	22
133	4.2.2 Profile Overview.....	22
134	4.2.3 Profile Description.....	25
135	4.2.3.1 ECP issues HTTP Request to Service Provider.....	25
136	4.2.3.2 Service Provider Issues <AuthnRequest> to ECP.....	26
137	4.2.3.3 ECP Determines Identity Provider.....	26
138	4.2.3.4 ECP issues <AuthnRequest> to Identity Provider.....	26
139	4.2.3.5 Identity Provider Identifies Principal.....	26
140	4.2.3.6 Identity Provider issues <Response> to ECP, targeted at service provider.....	27
141	4.2.3.7 ECP Conveys <Response> Message to Service Provider.....	27
142	4.2.3.8 Service Provider Grants or Denies Access to Principal.....	27
143	4.2.4 ECP Profile Schema Usage.....	27
144	4.2.4.1 PAOS Request Header Block: SP to ECP.....	28
145	4.2.4.2 ECP Request Header Block: SP to ECP.....	29
146	4.2.4.3 ECP RelayState Header Block: SP to ECP.....	29

147	4.2.4.4 ECP Response Header Block: IdP to ECP.....	31
148	4.2.4.5 PAOS Response Header Block: ECP to SP.....	31
149	4.2.5 Security Considerations.....	32
150	4.2.6 [E20]Use of Metadata.....	32
151	4.3 Identity Provider Discovery Profile.....	32
152	4.3.1 [E32]Required Information.....	33
153	4.3.2 Common Domain Cookie.....	33
154	4.3.3 Setting the Common Domain Cookie.....	33
155	4.3.4 Obtaining the Common Domain Cookie.....	33
156	4.4 Single Logout Profile.....	34
157	4.4.1 Required Information.....	34
158	4.4.2 Profile Overview.....	34
159	4.4.3 Profile Description.....	36
160	4.4.3.1 <LogoutRequest> Issued by Session Participant to Identity Provider.....	36
161	4.4.3.2 Identity Provider Determines Session Participants.....	37
162	4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session Participant/Authority.....	37
163	4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity Provider.....	37
164	4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant.....	38
165	4.4.4 Use of Single Logout Protocol.....	38
166	4.4.4.1 <LogoutRequest> Usage.....	38
167	4.4.4.2 <LogoutResponse> Usage.....	38
168	4.4.5 Use of Metadata.....	39
169	4.5 Name Identifier Management Profile.....	39
170	4.5.1 Required Information.....	39
171	4.5.2 Profile Overview.....	39
172	4.5.3 Profile Description.....	40
173	4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider.....	40
174	4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service Provider.....	41
175	4.5.4 Use of Name Identifier Management Protocol.....	42
176	4.5.4.1 <ManageNameIDRequest> Usage.....	42
177	4.5.4.2 <ManageNameIDResponse> Usage.....	42
178	4.5.5 Use of Metadata.....	42
179	5 Artifact Resolution Profile.....	43
180	5.1 Required Information.....	43
181	5.2 Profile Overview.....	43
182	5.3 Profile Description.....	44
183	5.3.1 <ArtifactResolve> issued by Requesting Entity.....	44
184	5.3.2 <ArtifactResponse> issued by Responding Entity.....	44
185	5.4 Use of Artifact Resolution Protocol.....	44
186	5.4.1 <ArtifactResolve> Usage.....	44
187	5.4.2 <ArtifactResponse> Usage.....	45
188	5.5 Use of Metadata.....	45
189	6 Assertion Query/Request Profile.....	46
190	6.1 Required Information.....	46
191	6.2 Profile Overview.....	46
192	6.3 Profile Description.....	47

193	6.3.1 Query/Request issued by SAML Requester.....	47
194	6.3.2 <Response> issued by SAML Authority.....	47
195	6.4 Use of Query/Request Protocol.....	47
196	6.4.1 Query/Request Usage.....	47
197	6.4.2 <Response> Usage.....	47
198	6.5 Use of Metadata.....	48
199	7 Name Identifier Mapping Profile.....	49
200	7.1 Required Information.....	49
201	7.2 Profile Overview.....	49
202	7.3 Profile Description.....	50
203	7.3.1 <NameIDMappingRequest> issued by Requesting Entity.....	50
204	7.3.2 <NameIDMappingResponse> issued by Identity Provider.....	50
205	7.4 Use of Name Identifier Mapping Protocol.....	50
206	7.4.1 <NameIDMappingRequest> Usage.....	50
207	7.4.2 <NameIDMappingResponse> Usage.....	50
208	7.4.2.1 Limiting Use of Mapped Identifier.....	51
209	7.5 Use of Metadata.....	51
210	8 SAML Attribute Profiles.....	52
211	8.1 Basic Attribute Profile.....	52
212	8.1.1 Required Information.....	52
213	8.1.2 SAML Attribute Naming.....	52
214	8.1.2.1 Attribute Name Comparison.....	52
215	8.1.3 Profile-Specific XML Attributes.....	52
216	8.1.4 SAML Attribute Values.....	52
217	8.1.5 Example.....	52
218	8.2 X.500/LDAP Attribute Profile [E53] – Deprecated.....	52
219	8.2.1 Required Information.....	53
220	8.2.2 SAML Attribute Naming.....	53
221	8.2.2.1 Attribute Name Comparison.....	53
222	8.2.3 Profile-Specific XML Attributes.....	53
223	8.2.4 SAML Attribute Values.....	54
224	8.2.5 Profile-Specific Schema.....	55
225	8.2.6 Example.....	55
226	8.3 UUID Attribute Profile.....	55
227	8.3.1 Required Information.....	55
228	8.3.2 UUID and GUID Background.....	55
229	8.3.3 SAML Attribute Naming.....	56
230	8.3.3.1 Attribute Name Comparison.....	56
231	8.3.4 Profile-Specific XML Attributes.....	56
232	8.3.5 SAML Attribute Values.....	56
233	8.3.6 Example.....	56
234	8.4 DCE PAC Attribute Profile.....	57
235	8.4.1 Required Information.....	57
236	8.4.2 PAC Description.....	57

237	8.4.3 SAML Attribute Naming.....	57
238	8.4.3.1 Attribute Name Comparison.....	57
239	8.4.4 Profile-Specific XML Attributes.....	58
240	8.4.5 SAML Attribute Values.....	58
241	8.4.6 Attribute Definitions.....	58
242	8.4.6.1 Realm.....	59
243	8.4.6.2 Principal.....	59
244	8.4.6.3 Primary Group.....	59
245	8.4.6.4 Groups.....	59
246	8.4.6.5 Foreign Groups.....	59
247	8.4.7 Example.....	60
248	8.5 XACML Attribute Profile.....	61
249	8.5.1 Required Information.....	61
250	8.5.2 SAML Attribute Naming.....	61
251	8.5.2.1 Attribute Name Comparison.....	61
252	8.5.3 Profile-Specific XML Attributes.....	61
253	8.5.4 SAML Attribute Values.....	61
254	8.5.5 Profile-Specific Schema.....	62
255	8.5.6 Example.....	62
256	9 References.....	63
257	Appendix A. Acknowledgments.....	66
258	Appendix B. Notices.....	68

259

1 Introduction

260 This document specifies profiles that define the use of SAML assertions and request-response messages
261 in communications protocols and frameworks, as well as profiles that define SAML attribute value syntax
262 and naming conventions.

263 The SAML assertions and protocols specification [SAMLCore] defines the SAML assertions and request-
264 response protocol messages themselves, and the SAML bindings specification [SAMLBind] defines
265 bindings of SAML protocol messages to underlying communications and messaging protocols. The SAML
266 conformance document [SAMLConform] lists all of the specifications that comprise SAML V2.0.

1.1 Profile Concepts

268 One type of SAML profile outlines a set of rules describing how to embed SAML assertions into and
269 extract them from a framework or protocol. Such a profile describes how SAML assertions are embedded
270 in or combined with other objects (for example, files of various types, or protocol data units of
271 communication protocols) by an originating party, communicated from the originating party to a receiving
272 party, and subsequently processed at the destination. A particular set of rules for embedding SAML
273 assertions into and extracting them from a specific class of <FOO> objects is termed a <FOO> *profile of*
274 *SAML*.

275 For example, a SOAP profile of SAML describes how SAML assertions can be added to SOAP messages,
276 how SOAP headers are affected by SAML assertions, and how SAML-related error states should be
277 reflected in SOAP messages.

278 Another type of SAML profile defines a set of constraints on the use of a general SAML protocol or
279 assertion capability for a particular environment or context of use. Profiles of this nature may constrain
280 optionality, require the use of specific SAML functionality (for example, attributes, conditions, or bindings),
281 and in other respects define the processing rules to be followed by profile actors.

282 A particular example of the latter are those that address SAML attributes. The SAML <Attribute>
283 element provides a great deal of flexibility in attribute naming, value syntax, and including in-band
284 metadata through the use of XML attributes. Interoperability is achieved by constraining this flexibility
285 when warranted by adhering to profiles that define how to use these elements with greater specificity than
286 the generic rules defined by [SAMLCore].

287 Attribute profiles provide the definitions necessary to constrain SAML attribute expression when dealing
288 with particular types of attribute information or when interacting with external systems or other open
289 standards that require greater strictness.

290 The intent of this specification is to specify a selected set of profiles of various kinds in sufficient detail to
291 ensure that independently implemented products will interoperate.

292 For other terms and concepts that are specific to SAML, refer to the SAML glossary [SAMLGloss].

1.2 Notation

294 This specification uses schema documents conforming to W3C XML Schema [Schema1] and normative
295 text to describe the syntax and semantics of XML-encoded SAML assertions and protocol messages. In
296 cases of disagreement between the SAML profile schema documents and schema listings in this
297 specification, the schema documents take precedence. Note that in some cases the normative text of this
298 specification imposes constraints beyond those indicated by the schema documents.

299 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD
300 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as

301 described in IETF RFC 2119 [RFC2119].

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

303 Example code listings appear like this.

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

305 Conventional XML namespace prefixes are used throughout this specification to stand for their respective
306 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 [SAMLMeta].
ecp:	urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp	This is the SAML V2.0 ECP profile namespace, specified in this document and in a schema [SAMLECP-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].
SOAP-ENV:	http://schemas.xmlsoap.org/soap/envelope	This is the SOAP V1.1 namespace [SOAP1.1].
paos:	urn:liberty:paos:2003-08	This is the Liberty Alliance PAOS namespace.
dce:	urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE	This is the SAML V2.0 DCE PAC attribute profile namespace, specified in this document and in a schema [SAMLDCExsd].
x500:	urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500	This is the SAML V2.0 X.500/LDAP attribute profile namespace, specified in this document and in a schema [SAMLX500-xsd].
xacmlprof:	urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML	This is the SAML V2.0 XACML attribute profile namespace, specified in this document and in a schema [SAMLXAC-xsd].
xsi:	http://www.w3.org/2001/XMLSchema-instance	This namespace is defined in the W3C XML Schema specification [Schema1] for schema-related markup that appears in XML instances.

307 This specification uses the following typographical conventions in text: <SAMLElement>,
308 <ns:ForeignElement>, XMLAttribute, **Datatype**, OtherKeyword. In some cases, angle brackets
309 are used to indicate non-terminals, rather than XML elements; the intent will be clear from the context.

310

2 Specification of Additional Profiles

311 This specification defines a selected set of profiles, but others will possibly be developed in the future. It is
312 not possible for the OASIS Security Services Technical Committee to standardize all of these additional
313 profiles for two reasons: it has limited resources and it does not own the standardization process for all of
314 the technologies used. The following sections offer guidelines for specifying profiles.

315 The SSTC welcomes proposals for new profiles. OASIS members may wish to submit these proposals for
316 consideration by the SSTC in a future version of this specification. Other members may simply wish to
317 inform the committee of their work related to SAML. Please refer to the SSTC website [SAMLWeb] for
318 further details on how to submit such proposals to the SSTC.

2.1 Guidelines for Specifying Profiles

320 This section provides a checklist of issues that MUST be addressed by each profile.

- 321 1. Specify a URI that uniquely identifies the profile, postal or electronic contact information for the
322 author, and provide reference to previously defined profiles that the new profile updates or
323 obsoletes.
- 324 2. Describe the set of interactions between parties involved in the profile. Any restrictions on
325 applications used by each party and the protocols involved in each interaction must be explicitly
326 called out.
- 327 3. Identify the parties involved in each interaction, including how many parties are involved and
328 whether intermediaries may be involved.
- 329 4. Specify the method of authentication of parties involved in each interaction, including whether
330 authentication is required and acceptable authentication types.
- 331 5. Identify the level of support for message integrity, including the mechanisms used to ensure
332 message integrity.
- 333 6. Identify the level of support for confidentiality, including whether a third party may view the contents
334 of SAML messages and assertions, whether the profile requires confidentiality, and the
335 mechanisms recommended for achieving confidentiality.
- 336 7. Identify the error states, including the error states at each participant, especially those that receive
337 and process SAML assertions or messages.
- 338 8. Identify security considerations, including analysis of threats and description of countermeasures.
- 339 9. Identify SAML confirmation method identifiers defined and/or utilized by the profile.
- 340 10. Identify relevant SAML metadata defined and/or utilized by the profile.

2.2 Guidelines for Specifying Attribute Profiles

342 This section provides a checklist of items that MUST in particular be addressed by attribute profiles.

- 343 1. Specify a URI that uniquely identifies the profile, postal or electronic contact information for the
344 author, and provide reference to previously defined profiles that the new profile updates or
345 obsoletes.
- 346 2. Syntax and restrictions on the acceptable values of the `NameFormat` and `Name` attributes of SAML
347 `<Attribute>` elements.
- 348 3. Any additional namespace-qualified XML attributes defined by the profile that may be used in SAML
349 `<Attribute>` elements.

- 350 4. Rules for determining the equality of SAML <Attribute> elements as defined by the profile, for
351 use when processing attributes, queries, etc.
- 352 5. Syntax and restrictions on values acceptable in the SAML <AttributeValue> element, including
353 whether the `xsi:type` XML attribute can or should be used.

3 Confirmation Method Identifiers

354

355 The SAML assertion and protocol specification [SAMLCore] defines the `<SubjectConfirmation>`
356 element as a `Method` plus optional `<SubjectConfirmationData>`. The `<SubjectConfirmation>`
357 element SHOULD be used by the relying party to confirm that the request or message came from a
358 system entity that is associated with the subject of the assertion, within the context of a particular profile.

359 The `Method` attribute indicates the specific method that the relying party should use to make this
360 determination. This may or may not have any relationship to an authentication that was performed
361 previously. Unlike the authentication context, the subject confirmation method will often be accompanied
362 by additional information, such as a certificate or key, in the `<SubjectConfirmationData>` element
363 that will allow the relying party to perform the necessary verification. A common set of attributes is also
364 defined and MAY be used to constrain the conditions under which the verification can take place.

365 It is anticipated that profiles will define and use several different values for
366 [\[E56\]Confirmation<Method>](#), each corresponding to a different SAML usage scenario. The following
367 methods are defined for use by profiles defined within this specification and other profiles that find them
368 useful.

3.1 Holder of Key

369

370 **URI:** urn:oasis:names:tc:SAML:2.0:cm:holder-of-key

371 One or more `<ds:KeyInfo>` elements MUST be present within the `<SubjectConfirmationData>`
372 element. An `xsi:type` attribute MAY be present in the `<SubjectConfirmationData>` element and, if
373 present, MUST be set to **saml:KeyInfoConfirmationDataType** (the namespace prefix is arbitrary but
374 must reference the SAML assertion namespace).

375 As described in [XMLSig], each `<ds:KeyInfo>` element holds a key or information that enables an
376 application to obtain a key. The holder of [\[E47\]one or more of the specified keys](#) a specified key is
377 considered to be [\[E40\]an acceptable attesting entity for the subject of](#) the assertion by the asserting party.

378 Note that in accordance with [XMLSig], each `<ds:KeyInfo>` element MUST identify a single
379 cryptographic key. Multiple keys MAY be identified with separate `<ds:KeyInfo>` elements, such as when
380 different confirmation keys are needed for different relying parties.

381 [\[E47\]If the keys contained in the <SubjectConfirmationData> element belong to an entity other than](#)
382 [the subject, then the asserting party SHOULD identify that entity to the relying party by including a SAML](#)
383 [identifier representing it in the enclosing <SubjectConfirmation> element.](#)

384 [Note that a given <SubjectConfirmation> element using the Holder of Key method SHOULD include](#)
385 [keys belonging to only a single attesting entity. If multiple attesting entities are to be permitted to use the](#)
386 [assertion, then multiple <SubjectConfirmation> elements SHOULD be included.](#)

387 **Example:** The holder of the key named "By-Tor" or the holder of the key named "Snow Dog" can confirm
388 itself as the subject.

```
389 <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
390   <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
391     <ds:KeyInfo>
392       <ds:KeyName>By-Tor</ds:KeyName>
393     </ds:KeyInfo>
394     <ds:KeyInfo>
395       <ds:KeyName>Snow Dog</ds:KeyName>
396     </ds:KeyInfo>
397   </SubjectConfirmationData>
398 </SubjectConfirmation>
```

399 3.2 Sender Vouches

400 **URI:** urn:oasis:names:tc:SAML:2.0:cm:sender-vouches

401 Indicates that no other information is available about the context of use of the assertion. The relying party
402 SHOULD utilize other means to determine if it should process the assertion further, subject to optional
403 constraints on confirmation using the attributes that MAY be present in the
404 <SubjectConfirmationData> element, as defined by [SAMLCore].

405 3.3 Bearer

406 **URI:** urn:oasis:names:tc:SAML:2.0:cm:bearer

407 The subject of the assertion is ~~[E47]the bearer of~~considered to be an acceptable attesting entity for the
408 assertion by the asserting party, subject to optional constraints on confirmation using the attributes that
409 MAY be present in the <SubjectConfirmationData> element, as defined by [SAMLCore].

410 If the intended bearer is known by the asserting party to be an entity other than the subject, then the
411 asserting party SHOULD identify that entity to the relying party by including a SAML identifier representing
412 it in the enclosing <SubjectConfirmation> element.

413 If multiple attesting entities are to be permitted to use the assertion based on bearer semantics, then
414 multiple <SubjectConfirmation> elements SHOULD be included.

415 **Example:** The bearer of the assertion can confirm itself as the subject, provided the assertion is delivered
416 in a message sent to "<https://www.serviceprovider.com/saml/consumer>" before 1:37 PM GMT on March
417 19th, 2004, in response to a request with ID "_1234567890".

```
418 <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">  
419   <SubjectConfirmationData InResponseTo="_1234567890"  
420     Recipient="https://www.serviceprovider.com/saml/consumer"  
421     NotOnOrAfter="2004-03-19T13:27:00Z"  
422   </SubjectConfirmationData>  
423 </SubjectConfirmation>
```

424 4 SSO Profiles of SAML

425 A set of profiles is defined to support single sign-on (SSO) of browsers and other client devices.

- 426 • A web browser-based profile of the Authentication Request protocol in [SAMLCore] is defined to
427 support web single sign-on, supporting Scenario 1-1 of the original SAML requirements document .
- 428 • An additional web SSO profile is defined to support enhanced clients.
- 429 • A profile of the Single Logout and Name Identifier Management protocols in [SAMLCore] is defined
430 over both front-channel (browser) and back-channel bindings.
- 431 • An additional profile is defined for identity provider discovery using cookies.

432 4.1 Web Browser SSO Profile

433 In the scenario supported by the web browser SSO profile, a web user either accesses a resource at a
434 service provider, or accesses an identity provider such that the service provider and desired resource are
435 understood or implicit. The web user authenticates (or has already authenticated) to the identity provider,
436 which then produces an authentication assertion (possibly with input from the service provider) and the
437 service provider consumes the assertion to establish a security context for the web user. During this
438 process, a name identifier might also be established between the providers for the principal, subject to the
439 parameters of the interaction and the consent of the parties.

440 To implement this scenario, a profile of the SAML Authentication Request protocol is used, in conjunction
441 with the HTTP Redirect, HTTP POST and HTTP Artifact bindings.

442 It is assumed that the user is using a standard commercial browser and can authenticate to the identity
443 provider by some means outside the scope of SAML.

444 4.1.1 Required Information

445 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:browser

446 **Contact information:** security-services-comment@lists.oasis-open.org

447 **SAML Confirmation Method Identifiers:** The SAML V2.0 "bearer" confirmation method identifier,
448 urn:oasis:names:tc:SAML:2.0:cm:bearer, is used by this profile.

449 **Description:** Given below.

450 **Updates:** SAML V1.1 browser artifact and POST profiles and bearer confirmation method.

451 4.1.2 Profile Overview

452 Figure 1 illustrates the basic template for achieving SSO. The following steps are described by the profile.
453 Within an individual step, there may be one or more actual message exchanges depending on the binding
454 used for that step and other implementation-dependent behavior.

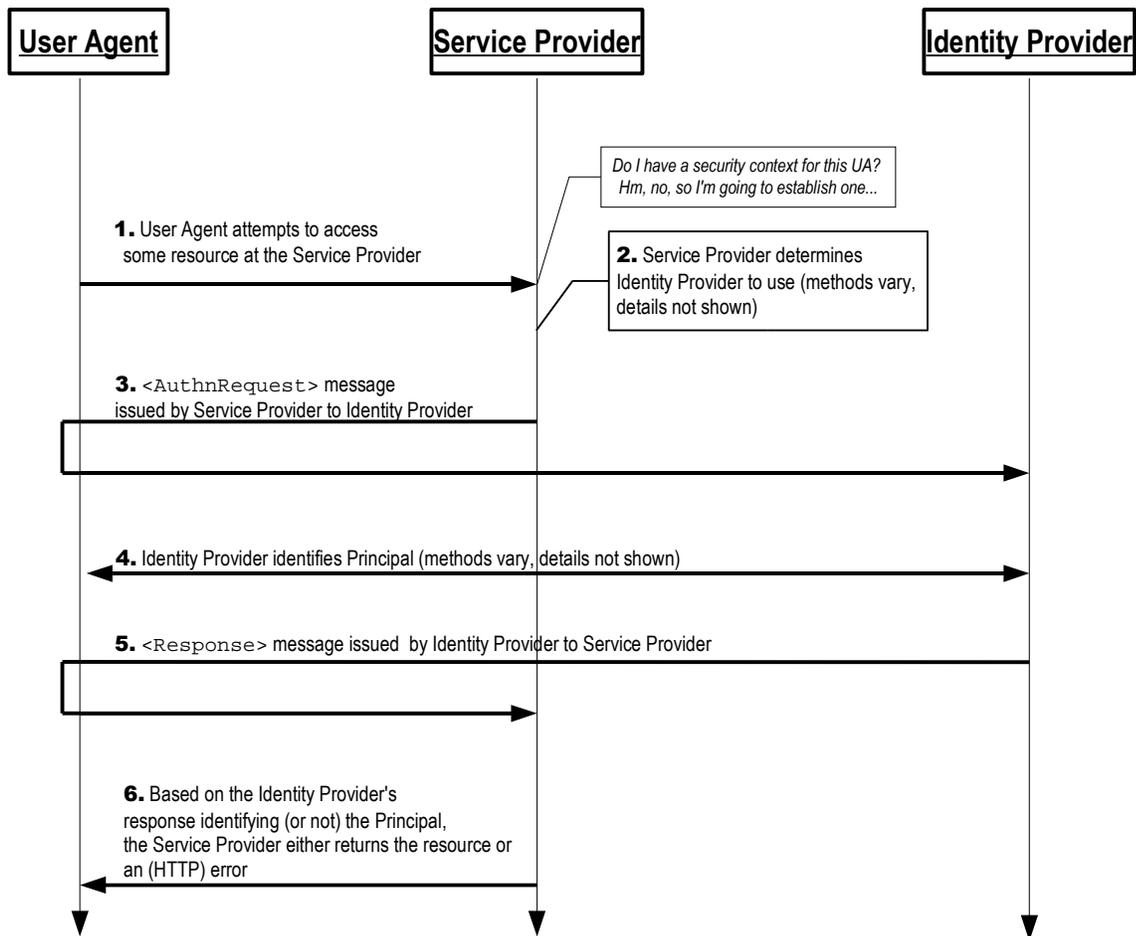


Figure 1

455 **1. HTTP Request to Service Provider**

456 In step 1, the principal, via an HTTP User Agent, makes an HTTP request for a secured resource
457 at the service provider without a security context.

458 **2. Service Provider Determines Identity Provider**

459 In step 2, the service provider obtains the location of an endpoint at an identity provider for the
460 authentication request protocol that supports its preferred binding. The means by which this is
461 accomplished is implementation-dependent. The service provider MAY use the SAML identity
462 provider discovery profile described in Section 4.3.

463 **3. <AuthnRequest> issued by Service Provider to Identity Provider**

464 In step 3, the service provider issues an <AuthnRequest> message to be delivered by the user
465 agent to the identity provider. Either the HTTP Redirect, HTTP POST, or HTTP Artifact binding
466 can be used to transfer the message to the identity provider through the user agent.

467 **4. Identity Provider identifies Principal**

468 In step 4, the principal is identified by the identity provider by some means outside the scope of
469 this profile. This may require a new act of authentication, or it may reuse an existing authenticated
470 session.

471 **5. Identity Provider issues <Response> to Service Provider**

472 In step 5, the identity provider issues a <Response> message to be delivered by the user agent
473 to the service provider. Either the HTTP POST, or HTTP Artifact binding can be used to transfer
474 the message to the service provider through the user agent. The message may indicate an error,
475 or will include (at least) an authentication assertion. The HTTP Redirect binding MUST NOT be
476 used, as the response will typically exceed the URL length permitted by most user agents.

477 **6. Service Provider grants or denies access to Principal**

478 In step 6, having received the response from the identity provider, the service provider can
479 respond to the principal's user agent with its own error, or can establish its own security context
480 for the principal and return the requested resource.

481 Note that an identity provider can initiate this profile at step 5 and issue a <Response> message to a
482 service provider without the preceding steps.

483 **4.1.3 Profile Description**

484 If the profile is initiated by the service provider, start with Section 4.1.3.1. If initiated by the identity
485 provider, start with Section 4.1.3.5. In the descriptions below, the following are referred to:

486 **Single Sign-On Service**

487 This is the authentication request protocol endpoint at the identity provider to which the
488 <AuthnRequest> message (or artifact representing it) is delivered by the user agent.

489 **Assertion Consumer Service**

490 This is the authentication request protocol endpoint at the service provider to which the
491 <Response> message (or artifact representing it) is delivered by the user agent.

492 **4.1.3.1 HTTP Request to Service Provider**

493 If the first access is to the service provider, an arbitrary request for a resource can initiate the profile.
494 There are no restrictions on the form of the request. The service provider is free to use any means it
495 wishes to associate the subsequent interactions with the original request. Each of the bindings provide a
496 RelayState mechanism that the service provider MAY use to associate the profile exchange with the
497 original request. The service provider SHOULD reveal as little of the request as possible in the RelayState
498 value unless the use of the profile does not require such privacy measures.

499 **4.1.3.2 Service Provider Determines Identity Provider**

500 This step is implementation-dependent. The service provider MAY use the SAML identity provider
501 discovery profile, described in Section 4.3. The service provider MAY also choose to redirect the user
502 agent to another service that is able to determine an appropriate identity provider. In such a case, the
503 service provider may issue an <AuthnRequest> (as in the next step) to this service to be relayed to the
504 identity provider, or it may rely on the intermediary service to issue an <AuthnRequest> message on its
505 behalf.

506 **4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider**

507 Once an identity provider is selected, the location of its single sign-on service is determined, based on the
508 SAML binding chosen by the service provider for sending the <AuthnRequest>. Metadata (as in
509 [SAMLMeta]) MAY be used for this purpose. In response to an HTTP request by the user agent, an HTTP
510 response is returned containing an <AuthnRequest> message or an artifact, depending on the SAML
511 binding used, to be delivered to the identity provider's single sign-on service.

512 The exact format of this HTTP response and the subsequent HTTP request to the single sign-on service
513 is defined by the SAML binding used. Profile-specific rules for the contents of the `<AuthnRequest>`
514 message are included in Section 4.1.4.1. If the HTTP Redirect or POST binding is used, the
515 `<AuthnRequest>` message is delivered directly to the identity provider in this step. If the HTTP Artifact
516 binding is used, the Artifact Resolution profile defined in Section 5 is used by the identity provider, which
517 makes a callback to the service provider to retrieve the `<AuthnRequest>` message, using, for example,
518 the SOAP binding.

519 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or TLS
520 1.0 [RFC2246] to maintain confidentiality and message integrity. The `<AuthnRequest>` message MAY
521 be signed, if authentication of the request issuer is required. The HTTP Artifact binding, if used, also
522 provides for an alternate means of authenticating the request issuer when the artifact is dereferenced.

523 The identity provider MUST process the `<AuthnRequest>` message as described in [SAMLCore]. This
524 may constrain the subsequent interactions with the user agent, for example if the `IsPassive` attribute is
525 included.

526 **4.1.3.4 Identity Provider Identifies Principal**

527 At any time during the previous step or subsequent to it, the identity provider MUST establish the identity
528 of the principal (unless it returns an error to the service provider). The `ForceAuthn` `<AuthnRequest>`
529 attribute, if present with a value of `true`, obligates the identity provider to freshly establish this identity,
530 rather than relying on an existing session it may have with the principal. Otherwise, and in all other
531 respects, the identity provider may use any means to authenticate the user agent, subject to any
532 requirements included in the `<AuthnRequest>` in the form of the `<RequestedAuthnContext>`
533 element.

534 **4.1.3.5 Identity Provider Issues `<Response>` to Service Provider**

535 Regardless of the success or failure of the `<AuthnRequest>`, the identity provider SHOULD produce an
536 HTTP response to the user agent containing a `<Response>` message or an artifact, depending on the
537 SAML binding used, to be delivered to the service provider's assertion consumer service.

538 The exact format of this HTTP response and the subsequent HTTP request to the assertion consumer
539 service is defined by the SAML binding used. Profile-specific rules on the contents of the `<Response>`
540 are included in Section 4.1.4.2. If the HTTP POST binding is used, the `<Response>` message is delivered
541 directly to the service provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution
542 profile defined in Section 5 is used by the service provider, which makes a callback to the identity provider
543 to retrieve the `<Response>` message, using for example the SOAP binding.

544 The location of the assertion consumer service MAY be determined using metadata (as in [SAMLMeta]).
545 The identity provider MUST have some means to establish that this location is in fact controlled by the
546 service provider. A service provider MAY indicate the SAML binding and the specific assertion consumer
547 service to use in its `<AuthnRequest>` and the identity provider MUST honor them if it can.

548 It is RECOMMENDED that the HTTP requests in this step be made over either SSL 3.0 [SSL3] or TLS 1.0
549 [RFC2246] to maintain confidentiality and message integrity. The `<Assertion>` element(s) in the
550 `<Response>` MUST be signed, if the HTTP POST binding is used, and MAY be signed if the HTTP-
551 Artifact binding is used.

552 The service provider MUST process the `<Response>` message and any enclosed `<Assertion>`
553 elements as described in [SAMLCore].

554 **4.1.3.6 Service Provider Grants or Denies Access to User Agent**

555 To complete the profile, the service provider processes the `<Response>` and `<Assertion>`(s) and
556 grants or denies access to the resource. The service provider MAY establish a security context with the

557 user agent using any session mechanism it chooses. Any subsequent use of the <Assertion>(s)
558 provided are at the discretion of the service provider and other relying parties, subject to any restrictions
559 on use contained within them.

560 4.1.4 Use of Authentication Request Protocol

561 This profile is based on the Authentication Request protocol defined in [SAMLCore]. In the nomenclature
562 of actors enumerated in Section 3.4 of that document, the service provider is the request issuer and the
563 relying party, and the principal is the presenter, requested subject, and confirming entity. There may be
564 additional relying parties or confirming entities at the discretion of the identity provider (see below).

565 4.1.4.1 <AuthnRequest> Usage

566 A service provider MAY include any message content described in [SAMLCore], Section 3.4.1. All
567 processing rules are as defined in [SAMLCore]. The <Issuer> element MUST be present and MUST
568 contain the unique identifier of the requesting service provider; the Format attribute MUST be omitted or
569 have a value of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

570 If the identity provider cannot or will not satisfy the request, it MUST respond with a <Response>
571 message containing an appropriate error status code or codes.

572 ~~[E14]If the service provider wishes to permit the identity provider to establish a new identifier for the~~
573 ~~principal if none exists, it MUST include a <NameIDPolicy> element with the AllowCreate attribute set~~
574 ~~to "true". Otherwise, only a principal for whom the identity provider has previously established an identifier~~
575 ~~usable by the service provider can be authenticated successfully. This profile does not provide any~~
576 ~~guidelines for the use of AllowCreate; see [SAMLCore] for normative rules on using AllowCreate.~~

577 Note that the service provider MAY include a <Subject> element in the request that names the actual
578 identity about which it wishes to receive an assertion. This element MUST NOT contain any
579 <SubjectConfirmation> elements. If the identity provider does not recognize the principal as that
580 identity, then it MUST respond with a <Response> message containing an error status and no assertions.

581 The <AuthnRequest> message MAY be signed (as directed by the SAML binding used). If the HTTP
582 Artifact binding is used, authentication of the parties is OPTIONAL and any mechanism permitted by the
583 binding MAY be used.

584 Note that if the <AuthnRequest> is not authenticated and/or integrity protected, the information in it
585 MUST NOT be trusted except as advisory. Whether the request is signed or not, the identity provider
586 MUST ensure that any <AssertionConsumerServiceURL> or
587 <AssertionConsumerServiceIndex> elements in the request are verified as belonging to the service
588 provider to whom the response will be sent. Failure to do so can result in a man-in-the-middle attack.

589 4.1.4.2 <Response> Usage

590 If the identity provider wishes to return an error, it MUST NOT include any assertions in the <Response>
591 message. Otherwise, if the request is successful (or if the response is not associated with a request), the
592 <Response> element MUST conform to the following:

593 • ~~[E17]The <Issuer> element MAY be omitted, but if present if the <Response> message is signed or~~
594 ~~if an enclosed assertion is encrypted, then the <Issuer> element MUST be present. Otherwise it~~
595 ~~MAY be omitted. If present~~ it MUST contain the unique identifier of the issuing identity provider; the
596 Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
597 format:entity.

598 • It MUST contain at least one <Assertion>. Each assertion's <Issuer> element MUST contain the
599 unique identifier of the ~~[E26]issuingresponding~~ identity provider; the Format attribute MUST be omitted
600 or have a value of urn:oasis:names:tc:SAML:2.0:nameid-format:entity. ~~Note that this~~

- 601 | profile assumes a single responding identity provider, and all assertions in a response MUST be issued
602 | by the same entity.
- 603 | • ~~The set of one or more assertions MUST contain at least one <AuthnStatement> that reflects the~~
604 | ~~authentication of the principal to the identity provider.~~
 - 605 | • ~~At least one assertion containing an <AuthnStatement> MUST contain a <Subject> element with~~
606 | ~~at least one <SubjectConfirmation> element containing a Method of~~
607 | ~~urn:oasis:names:tc:SAML:2.0:cm:bearer. If the identity provider supports the Single Logout~~
608 | ~~profile, defined in Section 4.4, any such authentication statements MUST include a SessionIndex~~
609 | ~~attribute to enable per-session logout requests by the service provider.~~
 - 610 | • ~~The bearer <SubjectConfirmation> element described above MUST contain a~~
611 | ~~<SubjectConfirmationData> element that contains a Recipient attribute containing the service~~
612 | ~~provider's assertion consumer service URL and a NotOnOrAfter attribute that limits the window~~
613 | ~~during which the assertion can be delivered. It MAY contain an Address attribute limiting the client~~
614 | ~~address from which the assertion can be delivered. It MUST NOT contain a NotBefore attribute. If~~
615 | ~~the containing message is in response to an <AuthnRequest>, then the InResponseTo attribute~~
616 | ~~MUST match the request's ID. If multiple assertions are included, then each assertion's <Subject>~~
617 | ~~element MUST refer to the same principal. It is allowable for the content of the <Subject> elements~~
618 | ~~to differ (e.g. using different <NameID> or alternative <SubjectConfirmation> elements).~~
 - 619 | • Any assertion issued for consumption using this profile MUST contain a <Subject> element with at
620 | least one <SubjectConfirmation> element containing a Method of
621 | urn:oasis:names:tc:SAML:2.0:cm:bearer. Such an assertion is termed a bearer assertion.
622 | Bearer assertions MAY contain additional <SubjectConfirmation> elements.
 - 623 | • Assertions without a bearer <SubjectConfirmation> MAY also be included; processing of
624 | additional assertions or <SubjectConfirmation> elements is outside the scope of this profile.
 - 625 | • At least one bearer <SubjectConfirmation> element MUST contain a
626 | <SubjectConfirmationData> element that itself MUST contain a Recipient attribute containing
627 | the service provider's assertion consumer service URL and a NotOnOrAfter attribute that limits the
628 | window during which the assertion can be [E52]confirmed by the relying party. It MAY also contain an
629 | Address attribute limiting the client address from which the assertion can be delivered. It MUST NOT
630 | contain a NotBefore attribute. If the containing message is in response to an <AuthnRequest>.
631 | then the InResponseTo attribute MUST match the request's ID.
 - 632 | • The set of one or more bearer assertions MUST contain at least one <AuthnStatement> that
633 | reflects the authentication of the principal to the identity provider. Multiple <AuthnStatement>
634 | elements MAY be included, but the semantics of multiple statements is not defined by this profile.
 - 635 | • If the identity provider supports the Single Logout profile, defined in Section 4.4, any authentication
636 | statements MUST include a SessionIndex attribute to enable per-session logout requests by the
637 | service provider.
 - 638 | • Other statements ~~and confirmation methods~~ MAY be included in the bearer assertion(s) at the
639 | discretion of the identity provider. In particular, <AttributeStatement> elements MAY be included.
640 | The <AuthnRequest> MAY contain an AttributeConsumingServiceIndex XML attribute
641 | referencing information about desired or required attributes in [SAMLMeta]. The identity provider MAY
642 | ignore this, or send other attributes at its discretion.
 - 643 | • ~~The Each bearer~~ assertion(s) ~~containing a bearer subject confirmation~~ MUST contain an
644 | <AudienceRestriction> including the service provider's unique identifier as an <Audience>.
 - 645 | • Other conditions (and other <Audience> elements) MAY be included as requested by the service
646 | provider or at the discretion of the identity provider. (Of course, all such conditions MUST be
647 | understood by and accepted by the service provider in order for the assertion to be considered valid.)
 - 648 | • ~~The identity provider is NOT obligated to honor the requested set of <Conditions> in the~~
649 | ~~<AuthnRequest>, if any.~~

650 4.1.4.3 <Response> Message Processing Rules

651 Regardless of the SAML binding used, the service provider MUST do the following:

- 652 • Verify any signatures present on the assertion(s) or the response
- 653 • Verify that the `Recipient` attribute in ~~[E26]the~~any bearer `<SubjectConfirmationData>` matches
654 the assertion consumer service URL to which the `<Response>` or artifact was delivered
- 655 • Verify that the `NotOnOrAfter` attribute in ~~the~~any bearer `<SubjectConfirmationData>` has not
656 passed, subject to allowable clock skew between the providers
- 657 • Verify that the `InResponseTo` attribute in the bearer `<SubjectConfirmationData>` equals the ID
658 of its original `<AuthnRequest>` message, unless the response is unsolicited (see Section 4.1.5), in
659 which case the attribute MUST NOT be present
- 660 • Verify that any assertions relied upon are valid in other respects. Note that while multiple bearer
661 <SubjectConfirmation> elements may be present, the successful evaluation of a single such
662 element in accordance with this profile is sufficient to confirm an assertion. However, each assertion, if
663 more than one is present, MUST be evaluated independently.
- 664 • If ~~any~~the bearer `<SubjectConfirmationData>` includes an `Address` attribute, the service provider
665 MAY check the user agent's client address against it.
- 666 • Any assertion which is not valid, or whose subject confirmation requirements cannot be met SHOULD
667 be discarded and SHOULD NOT be used to establish a security context for the principal.
- 668 • If an `<AuthnStatement>` used to establish a security context for the principal contains a
669 `SessionNotOnOrAfter` attribute, the security context SHOULD be discarded once this time is
670 reached, unless the service provider reestablishes the principal's identity by repeating the use of this
671 profile. Note that if multiple <AuthnStatement> elements are present, the SessionNotOnOrAfter
672 value closest to the present time SHOULD be honored.

673 4.1.4.4 Artifact-Specific <Response> Message Processing Rules

674 If the HTTP Artifact binding is used to deliver the `<Response>`, the dereferencing of the artifact using the
675 Artifact Resolution profile MUST be mutually authenticated, integrity protected, and confidential.

676 The identity provider MUST ensure that only the service provider to whom the `<Response>` message has
677 been issued is given the message as the result of an `<ArtifactResolve>` request.

678 Either the SAML binding used to dereference the artifact or message signatures can be used to
679 authenticate the parties and protect the messages.

680 4.1.4.5 POST-Specific Processing Rules

681 If the HTTP POST binding is used to deliver the `<Response>`, ~~[E26]the enclosed assertion(s) MUST be~~
682 ~~signed each assertion MUST be protected by a digital signature. This can be accomplished by signing~~
683 ~~each individual <Assertion> element or by signing the <Response> element.~~

684 The service provider MUST ensure that bearer assertions are not replayed, by maintaining the set of used
685 ID values for the length of time for which the assertion would be considered valid based on the
686 `NotOnOrAfter` attribute in the `<SubjectConfirmationData>`.

687 4.1.5 Unsolicited Responses

688 An identity provider MAY initiate this profile by delivering an unsolicited `<Response>` message to a
689 service provider.

690 An unsolicited <Response> MUST NOT contain an InResponseTo attribute, nor should any bearer
691 <SubjectConfirmationData> elements contain one. If metadata as specified in [SAMLMeta] is used,
692 the <Response> or artifact SHOULD be delivered to the <md:AssertionConsumerService> endpoint
693 of the service provider designated as the default.

694 Of special mention is that the identity provider MAY include a binding-specific "RelayState" parameter that
695 indicates, based on mutual agreement with the service provider, how to handle subsequent interactions
696 with the user agent. This MAY be the URL of a resource at the service provider. The service provider
697 SHOULD be prepared to handle unsolicited responses by designating a default location to send the user
698 agent subsequent to processing a response successfully.

699 **4.1.6 Use of Metadata**

700 [SAMLMeta] defines an endpoint element, <md:SingleSignOnService>, to describe supported
701 bindings and location(s) to which a service provider may send requests to an identity provider using this
702 profile.

703 The <md:IDPSSODescriptor> element's WantAuthnRequestsSigned attribute MAY be used by an
704 identity provider to document a requirement that requests be signed. The <md:SPSSODescriptor>
705 element's AuthnRequestsSigned attribute MAY be used by a service provider to document the
706 intention to sign all of its requests.

707 The providers MAY document the key(s) used to sign requests, responses, and assertions with
708 <md:KeyDescriptor> elements with a use attribute of [\[E58\]signing](#). When encrypting SAML
709 elements, <md:KeyDescriptor> elements with a use attribute of [encryption](#) MAY be used to
710 document supported encryption algorithms and settings, and public keys used to receive bulk encryption
711 keys.

712 The indexed endpoint element <md:AssertionConsumerService> is used to describe supported
713 bindings and location(s) to which an identity provider may send responses to a service provider using this
714 profile. The index attribute is used to distinguish the possible endpoints that may be specified by
715 reference in the <AuthnRequest> message. The isDefault attribute is used to specify the endpoint to
716 use if not specified in a request.

717 The <md:SPSSODescriptor> element's WantAssertionsSigned attribute MAY be used by a service
718 provider to document a requirement that assertions delivered with this profile be signed. This is in addition
719 to any requirements for signing imposed by the use of a particular binding. Note that the identity provider
720 is not obligated by this, but is being made aware of the likelihood that an unsigned assertion will be
721 insufficient.

722 If the request or response message is delivered using the HTTP Artifact binding, the artifact issuer MUST
723 provide at least one <md:ArtifactResolutionService> endpoint element in its metadata.

724 The <md:IDPSSODescriptor> MAY contain <md:NameIDFormat>, <md:AttributeProfile>, and
725 <saml:Attribute> elements to indicate the general ability to support particular name identifier formats,
726 attribute profiles, or specific attributes and values. The ability to support any such features during a given
727 authentication exchange is dependent on policy and the discretion of the identity provider.

728 The <md:SPSSODescriptor> element MAY also be used to document the service provider's need or
729 desire for SAML attributes to be delivered along with authentication information. The actual inclusion of
730 attributes is always at the discretion of the identity provider. One or more
731 <md:AttributeConsumingService> elements MAY be included in its metadata, each with an index
732 attribute to distinguish different services that MAY be specified by reference in the <AuthnRequest>
733 message. The isDefault attribute is used to specify a default set of attribute requirements.

734 **4.2 Enhanced Client or Proxy (ECP) Profile**

735 An *enhanced client or proxy* (ECP) is a system entity that knows how to contact an appropriate identity

736 provider, possibly in a context-dependent fashion, and also supports the Reverse SOAP (PAOS) binding
737 [SAMLBind].

738 An example scenario enabled by this profile is as follows: A principal, wielding an ECP, uses it to either
739 access a resource at a service provider, or access an identity provider such that the service provider and
740 desired resource are understood or implicit. The principal authenticates (or has already authenticated)
741 with the identity provider, which then produces an authentication assertion (possibly with input from the
742 service provider). The service provider then consumes the assertion and subsequently establishes a
743 security context for the principal. During this process, a name identifier might also be established between
744 the providers for the principal, subject to the parameters of the interaction and the consent of the principal.

745 This profile is based on the SAML Authentication Request protocol [SAMLCore] in conjunction with the
746 PAOS binding.

747 **Note:** The means by which a principal authenticates with an identity provider is outside of the
748 scope of SAML.

749 4.2.1 Required Information

750 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp (this is also the target namespace
751 assigned in the corresponding ECP profile schema document [SAMLECP-xsd])

752 **Contact information:** security-services-comment@lists.oasis-open.org

753 **SAML Confirmation Method Identifiers:** The SAML V2.0 "bearer" confirmation method identifier,
754 urn:oasis:names:tc:SAML:2.0:cm:bearer, is used by this profile.

755 **Description:** Given below.

756 **Updates:** None.

757 4.2.2 Profile Overview

758 As introduced above, the ECP profile specifies interactions between enhanced clients or proxies and
759 service providers and identity providers. It is a specific application of the SSO profile described in Section
760 4.1. If not otherwise specified by this profile, and if not specific to the use of browser-based bindings, the
761 rules specified in Section 4.1 MUST be observed.

762 An ECP is a client or proxy that satisfies the following two conditions:

- 763 • It has, or knows how to obtain, information about the identity provider that the principal associated with
764 the ECP wishes to use, in the context of an interaction with a service provider.

765 This allows a service provider to make an authentication request to the ECP without the need to know
766 or discover the appropriate identity provider (effectively bypassing step 2 of the SSO profile in Section
767 4.1).

- 768 • It is able to use a reverse SOAP (PAOS) binding as profiled here for an authentication request and
769 response.

770 This enables a service provider to obtain an authentication assertion via an ECP that is not otherwise
771 (i.e. outside of the context of the immediate interaction) necessarily directly addressable nor
772 continuously available. It also leverages the benefits of SOAP while using a well-defined exchange
773 pattern and profile to enable interoperability. The ECP may be viewed as a SOAP intermediary
774 between the service provider and the identity provider.

775 An *enhanced client* may be a browser or some other user agent that supports the functionality described
776 in this profile. An *enhanced proxy* is an HTTP proxy (for example a WAP gateway) that emulates an
777 enhanced client. Unless stated otherwise, all statements referring to enhanced clients are to be
778 understood as statements about both enhanced clients as well as enhanced client proxies.

779 Since the enhanced client sends and receives messages in the body of HTTP requests and responses, it
780 has no arbitrary restrictions on the size of the protocol messages.

781 This profile leverages the Reverse SOAP (PAOS) binding [SAMLBind]. Implementers of this profile MUST
782 follow the rules for HTTP indications of PAOS support specified in that binding, in addition to those
783 specified in this profile. This profile utilizes a PAOS SOAP header block conveyed between the HTTP
784 responder and the ECP but does not define PAOS itself. The SAML PAOS binding specification
785 [SAMLBind] is normative in the event of questions regarding PAOS.

786 This profile defines SOAP header blocks that accompany the SAML requests and responses. These
787 header blocks may be composed with other SOAP header blocks as necessary, for example with the
788 SOAP Message Security header block to add security features if needed, for example a digital signature
789 applied to the authentication request.

790 Two sets of request/response SOAP header blocks are used: PAOS header blocks for generic PAOS
791 information and ECP profile-specific header blocks to convey information specific to ECP profile
792 functionality.

793 Figure 2 shows the processing flow in the ECP profile.

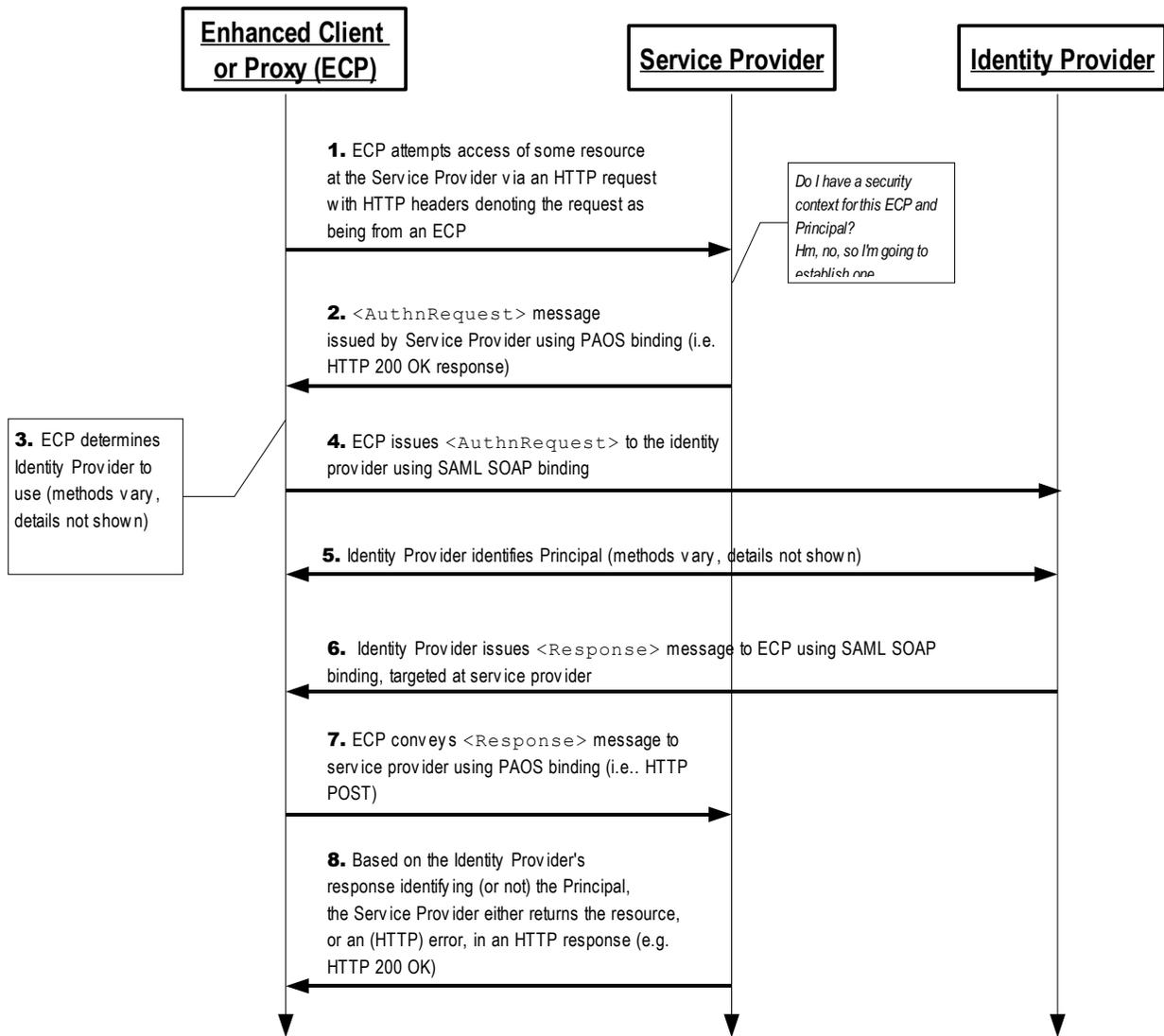


Figure 2

794 Figure 2 illustrates the basic template for SSO using an ECP. The following steps are described by the
 795 profile. Within an individual step, there may be one or more actual message exchanges depending on the
 796 binding used for that step and other implementation-dependent behavior.

797 **1. ECP issues HTTP Request to Service Provider**

798 In step 1, the Principal, via an ECP, makes an HTTP request for a secured resource at a service
 799 provider, where the service provider does not have an established security context for the ECP
 800 and Principal.

801 **2. Service Provider issues <AuthnRequest> to ECP**

802 In step 2, the service provider issues an <AuthnRequest> message to the ECP, which is to be
 803 delivered by the ECP to the appropriate identity provider. The Reverse SOAP (PAOS) binding
 804 [SAMLBind] is used here.

805 **3. ECP Determines Identity Provider**

806 In step 3, the ECP obtains the location of an endpoint at an identity provider for the authentication
807 request protocol that supports its preferred binding. The means by which this is accomplished is
808 implementation-dependent. ~~[E18] The ECP MAY use the SAML identity provider discovery profile~~
809 ~~described in Section 4.3.~~

810 4. ECP conveys <AuthnRequest> to Identity Provider

811 In step 4, the ECP conveys the <AuthnRequest> to the identity provider identified in step 3
812 using a modified form of the SAML SOAP binding [SAMLBind] with the additional allowance that
813 the identity provider may exchange arbitrary HTTP messages with the ECP before responding to
814 the SAML request.

815 5. Identity Provider identifies Principal

816 In step 5, the Principal is identified by the identity provider by some means outside the scope of
817 this profile. This may require a new act of authentication, or it may reuse an existing authenticated
818 session.

819 6. Identity Provider issues <Response> to ECP, targeted at Service Provider

820 In step 6, the identity provider issues a <Response> message, using the SAML SOAP binding, to
821 be delivered by the ECP to the service provider. The message may indicate an error, or will
822 include (at least) an authentication assertion.

823 7. ECP conveys <Response> message to Service Provider

824 In step 7, the ECP conveys the <Response> message to the service provider using the PAOS
825 binding.

826 8. Service Provider grants or denies access to Principal

827 In step 8, having received the <Response> message from the identity provider, the service
828 provider either establishes its own security context for the principal and return the requested
829 resource, or responds to the principal's ECP with an error.

830 4.2.3 Profile Description

831 The following sections provide detailed definitions of the individual steps.

832 4.2.3.1 ECP issues HTTP Request to Service Provider

833 The ECP sends an HTTP request to a service provider, specifying some resource. This HTTP request
834 MUST conform to the PAOS binding, which means it must include the following HTTP header fields:

- 835 1. The HTTP Accept Header field indicating the ability to accept the MIME type
836 "application/vnd.paos+xml"
- 837 2. The HTTP PAOS Header field specifying the PAOS version with urn:liberty:paos:2003-08 at
838 minimum.
- 839 3. Furthermore, support for this profile MUST be specified in the HTTP PAOS Header field as a service
840 value, with the value [E54] "urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp". This value
841 should correspond to the service attribute in the PAOS Request SOAP header block

842 For example, a user agent may request a page from a service provider as follows:

```
843 GET /index HTTP/1.1  
844 Host: identity-service.example.com  
845 Accept: text/html; application/vnd.paos+xml  
846 PAOS: ver="urn:liberty:paos:2003-08" ;  
847 "urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
```

848 **4.2.3.2 Service Provider Issues <AuthnRequest> to ECP**

849 When the service provider requires a security context for the principal before allowing access to the
850 specified resource, that is, before providing a service or data, it can respond to the HTTP request using
851 the PAOS binding with an <AuthnRequest> message in the HTTP response. The service provider will
852 issue an HTTP 200 OK response to the ECP containing a single SOAP envelope.

853 The SOAP envelope MUST contain:

- 854 1. An <AuthnRequest> element in the SOAP body, intended for the ultimate SOAP recipient, the
855 identity provider.
- 856 2. A PAOS SOAP header block targeted at the ECP using the SOAP actor value of
857 `http://schemas.xmlsoap.org/soap/actor/next`. This header block provides control
858 information such as the URL to which to send the response in this solicit-response message
859 exchange pattern.
- 860 3. An ECP profile-specific Request SOAP header block targeted at the ECP using the SOAP actor
861 `http://schemas.xmlsoap.org/soap/actor/next`. The ECP Request header block defines
862 information related to the authentication request that the ECP may need to process it, such as a list
863 of identity providers acceptable to the service provider, whether the ECP may interact with the
864 principal through the client, and the service provider's human-readable name that may be displayed
865 to the principal.

866 The SOAP envelope MAY contain an ECP RelayState SOAP header block targeted at the ECP using the
867 SOAP actor value of `http://schemas.xmlsoap.org/soap/actor/next`. The header contains state information
868 to be returned by the ECP along with the SAML response.

869 **4.2.3.3 ECP Determines Identity Provider**

870 The ECP will determine which identity provider is appropriate and route the SOAP message appropriately.

871 **4.2.3.4 ECP issues <AuthnRequest> to Identity Provider**

872 The ECP MUST remove the PAOS, ECP RelayState, and ECP Request header blocks before passing the
873 <AuthnRequest> message on to the identity provider, using a modified form of the SAML SOAP binding.
874 The SAML request is submitted via SOAP in the usual fashion, but the identity provider MAY respond to
875 the ECP's HTTP request with an HTTP response containing, for example, an HTML login form or some
876 other presentation-oriented response. A sequence of HTTP exchanges MAY take place, but ultimately the
877 identity provider MUST complete the SAML SOAP exchange and return a SAML response via the SOAP
878 binding.

879 Note that the <AuthnRequest> element may itself be signed by the service provider. In this and other
880 respects, the message rules specified in the browser SSO profile in Section 4.1.4.1 MUST be followed.

881 Prior to or subsequent to this step, the identity provider MUST establish the identity of the principal by
882 some means, or it MUST return an error <Response>, as described in Section 4.2.3.6 below.

883 **4.2.3.5 Identity Provider Identifies Principal**

884 At any time during the previous step or subsequent to it, the identity provider MUST establish the identity
885 of the principal (unless it returns an error to the service provider). The `ForceAuthn` <AuthnRequest>
886 attribute, if present with a value of `true`, obligates the identity provider to freshly establish this identity,
887 rather than relying on an existing session it may have with the principal. Otherwise, and in all other
888 respects, the identity provider may use any means to authenticate the user agent, subject to any
889 requirements included in the <AuthnRequest> in the form of the <RequestedAuthnContext>
890 element.

891 **4.2.3.6 Identity Provider issues <Response> to ECP, targeted at service provider**

892 The identity provider returns a SAML <Response> message (or SOAP fault) when presented with an
893 authentication request, after having established the identity of the principal. The SAML response is
894 conveyed using the SAML SOAP binding in a SOAP message with a <Response> element in the SOAP
895 body, intended for the service provider as the ultimate SOAP receiver. The rules for the response
896 specified in the browser SSO profile in Section 4.1.4.2 MUST be followed.

897 The identity provider's response message MUST contain a profile-specific ECP Response SOAP header
898 block, and MAY contain an ECP RelayState header block, both targeted at the ECP.

899 **4.2.3.7 ECP Conveys <Response> Message to Service Provider**

900 The ECP removes the header block(s), and MAY add a PAOS Response SOAP header block and an
901 ECP RelayState header block before forwarding the SOAP response to the service provider using the
902 PAOS binding.

903 The <paos:Response> SOAP header block in the response to the service provider is generally used to
904 correlate this response to an earlier request from the service provider. In this profile, the correlation
905 refToMessageID attribute is not required since the SAML <Response> element's InResponseTo
906 attribute may be used for this purpose, but if the <paos:Request> SOAP Header block had a
907 messageID then the <paos:Response> SOAP header block MUST be used.

908 The <ecp:RelayState> header block value is typically provided by the service provider to the ECP with
909 its request, but if the identity provider is producing an unsolicited response (without having received a
910 corresponding SAML request), then it MAY include a RelayState header block that indicates, based on
911 mutual agreement with the service provider, how to handle subsequent interactions with the ECP. This
912 MAY be the URL of a resource at the service provider.

913 If the service provider included an <ecp:RelayState> SOAP header block in its request to the ECP, or
914 if the identity provider included an <ecp:RelayState> SOAP header block with its response, then the
915 ECP MUST include an identical header block with the SAML response sent to the service provider. The
916 service provider's value for this header block (if any) MUST take precedence.

917 **4.2.3.8 Service Provider Grants or Denies Access to Principal**

918 Once the service provider has received the SAML response in an HTTP request (in a SOAP envelope
919 using PAOS), it may respond with the service data in the HTTP response. In consuming the response, the
920 rules specified in the browser SSO profile in Section 4.1.4.3 and 4.1.4.5 MUST be followed. That is, the
921 same processing rules used when receiving the <Response> with the HTTP POST binding apply to the
922 use of PAOS.

923 **4.2.4 ECP Profile Schema Usage**

924 The ECP Profile XML schema [SAMLECP-xsd] defines the SOAP Request/Response header blocks used
925 by this profile. Following is a complete listing of this schema document.

```
926 <schema  
927   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"  
928   xmlns="http://www.w3.org/2001/XMLSchema"  
929   xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"  
930   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
931   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
932   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"  
933   elementFormDefault="unqualified"  
934   attributeFormDefault="unqualified"  
935   blockDefault="substitution"  
936   version="2.0">
```

```

937 <import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
938       schemaLocation="saml-schema-protocol-2.0.xsd"/>
939 <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
940       schemaLocation="saml-schema-assertion-2.0.xsd"/>
941 <import namespace="http://schemas.xmlsoap.org/soap/envelope/"
942       schemaLocation="http://schemas.xmlsoap.org/soap/envelope/" />
943 <annotation>
944   <documentation>
945     Document identifier: saml-schema-ecp-2.0
946     Location: http://docs.oasis-open.org/security/saml/v2.0/
947     Revision history:
948       V2.0 (March, 2005):
949       Custom schema for ECP profile, first published in SAML 2.0.
950   </documentation>
951 </annotation>

952 <element name="Request" type="ecp:RequestType"/>
953 <complexType name="RequestType">
954   <sequence>
955     <element ref="saml:Issuer"/>
956     <element ref="samlp:IDPList" minOccurs="0"/>
957   </sequence>
958   <attribute ref="S:mustUnderstand" use="required"/>
959   <attribute ref="S:actor" use="required"/>
960   <attribute name="ProviderName" type="string" use="optional"/>
961   <attribute name="IsPassive" type="boolean" use="optional"/>
962 </complexType>

963
964 <element name="Response" type="ecp:ResponseType"/>
965 <complexType name="ResponseType">
966   <attribute ref="S:mustUnderstand" use="required"/>
967   <attribute ref="S:actor" use="required"/>
968   <attribute name="AssertionConsumerServiceURL" type="anyURI"
969 use="required"/>
970 </complexType>

971
972 <element name="RelayState" type="ecp:RelayStateType"/>
973 <complexType name="RelayStateType">
974   <simpleContent>
975     <extension base="string">
976       <attribute ref="S:mustUnderstand" use="required"/>
977       <attribute ref="S:actor" use="required"/>
978     </extension>
979   </simpleContent>
980 </complexType>
981 </schema>

```

982 The following sections describe how these XML constructs are to be used.

983 4.2.4.1 PAOS Request Header Block: SP to ECP

984 The PAOS Request header block signals the use of PAOS processing and includes the following
985 attributes:

986 responseConsumerURL [Required]

987 Specifies where the ECP is to send an error response. Also used to verify the correctness of the
988 identity provider's response, by cross checking this location against the
989 AssertionServiceConsumerURL in the ECP response header block. This value MUST be the
990 same as the ~~[E22]AssertionServiceConsumerURL~~ AssertionConsumerServiceURL (or the
991 URL referenced in metadata) conveyed in the <AuthnRequest> ~~[E35]~~ and SHOULD NOT be a
992 relative URL.

993 service [Required]

994 Indicates that the PAOS service being used is this SAML authentication profile. The value MUST be

995 urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp.

996 SOAP-ENV:mustUnderstand [Required]

997 The value MUST be 1 (true). A SOAP fault MUST be generated if the PAOS header block is not
998 understood.

999 SOAP-ENV:actor [Required]

1000 The value MUST be `http://schemas.xmlsoap.org/soap/actor/next`.

1001 messageID [Optional]

1002 Allows optional response correlation. It MAY be used in this profile, but is NOT required, since this
1003 functionality is provided by the SAML protocol layer, via the ID attribute in the <AuthnRequest> and
1004 the InResponseTo attribute in the <Response>.

1005 The PAOS Request SOAP header block has no element content.

1006 **4.2.4.2 ECP Request Header Block: SP to ECP**

1007 The ECP Request SOAP header block is used to convey information needed by the ECP to process the
1008 authentication request. It is mandatory and its presence signals the use of this profile. It contains the
1009 following elements and attributes:

1010 SOAP-ENV:mustUnderstand [Required]

1011 The value MUST be 1 (true). A SOAP fault MUST be generated if the ECP header block is not
1012 understood.

1013 SOAP-ENV:actor [Required]

1014 The value MUST be `http://schemas.xmlsoap.org/soap/actor/next`.

1015 ProviderName [Optional]

1016 A human-readable name for the requesting service provider.

1017 IsPassive [Optional]

1018 A boolean value. If true, the identity provider and the client itself MUST NOT take control of the user
1019 interface from the request issuer and interact with the principal in a noticeable fashion. If a value is not
1020 provided, the default is true.

1021 <saml:Issuer> [Required]

1022 This element MUST contain the unique identifier of the requesting service provider; the Format
1023 attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`
1024 `format:entity`.

1025 <samlp:IDPList> [Optional]

1026 Optional list of identity providers that the service provider recognizes and from which the ECP may
1027 choose to service the request. See [SAMLCore] for details on the content of this element.

1028 **4.2.4.3 ECP RelayState Header Block: SP to ECP**

1029 The ECP RelayState SOAP header block is used to convey state information from the service provider
1030 that it will need later when processing the response from the ECP. It is optional, but if used, the ECP
1031 MUST include an identical header block in the response in step [\[E27\]57](#). It contains the following
1032 attributes:

1033 SOAP-ENV:mustUnderstand [Required]

1034 The value MUST be 1 (true). A SOAP fault MUST be generated if the header block is not understood.

1035 SOAP-ENV:actor [Required]

1036 The value MUST be `http://schemas.xmlsoap.org/soap/actor/next`.

1037 The content of the header block element is a string containing state information created by the requester.
1038 If provided, the ECP MUST include the same value in a RelayState header block when responding to the
1039 service provider in step 5. The string value MUST NOT exceed 80 bytes in length and SHOULD be
1040 integrity protected by the requester independent of any other protections that may or may not exist during
1041 message transmission.

1042 The following is an example of the SOAP authentication request from the service provider to the ECP:

```
1043 <SOAP-ENV:Envelope
1044     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1045     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1046     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
1047   <SOAP-ENV:Header>
1048     <paos:Request xmlns:paos="urn:liberty:paos:2003-08"
1049       responseConsumerURL="http://identity
1050 service.example.com/abchttps://ServiceProvider.example.com/ecp_assertion_consumer"
1051       messageID="6c3a4f8b9c2d" SOAP-
1052 ENV:actor="http://schemas.xmlsoap.org/soap/actor/next" SOAP-
1053 ENV:mustUnderstand="1"
1054       service="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp">
1055     </paos:Request>
1056     <ecp:Request xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
1057       SOAP-ENV:mustUnderstand="1" SOAP-
1058 ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
1059       ProviderName="Service Provider X" IsPassive="0">
1060     <saml:Issuer>https://ServiceProvider.example.com</saml:Issuer>
1061     <samlp:IDPList>
1062       <samlp:IDPEntry ProviderID="https://IdentityProvider.example.com"
1063         Name="Identity Provider X"
1064         Loc="https://IdentityProvider.example.com/saml2/sso"
1065       </samlp:IDPEntry>
1066       <samlp:GetComplete>
1067         https://ServiceProvider.example.com/idplist?id=604be136-fe91-441e-afb8
1068       </samlp:GetComplete>
1069     </samlp:IDPList>
1070     </ecp:Request>
1071     <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
1072       SOAP-ENV:mustUnderstand="1" SOAP-
1073 ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
1074     ...
1075     </ecp:RelayState>
1076   </SOAP-ENV:Header>
1077   <SOAP-ENV:Body>
1078     <samlp:AuthnRequest> ... </samlp:AuthnRequest>
1079   </SOAP-ENV:Body>
1080 </SOAP-ENV:Envelope>
```

1082 As noted above, the PAOS and ECP header blocks are removed from the SOAP message by the ECP
1083 before the authentication request is forwarded to the identity provider. An example authentication request
1084 from the ECP to the identity provider is as follows:

```
1085 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
1086   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1087   <SOAP-ENV:Body>
1088     <samlp:AuthnRequest> ... </samlp:AuthnRequest>
1089   </SOAP-ENV:Body>
1090 </SOAP-ENV:Envelope>
```

1091 **4.2.4.4 ECP Response Header Block: IdP to ECP**

1092 The ECP response SOAP header block **MUST** be used on the response from the identity provider to the
1093 ECP. It contains the following attributes:

1094 SOAP-ENV:mustUnderstand [Required]

1095 The value **MUST** be 1 (true). A SOAP fault **MUST** be generated if the ECP header block is not
1096 understood.

1097 SOAP-ENV:actor [Required]

1098 The value **MUST** be `http://schemas.xmlsoap.org/soap/actor/next`.

1099 AssertionConsumerServiceURL [Required]

1100 Set by the identity provider based on the <AuthnRequest> message or the service provider's
1101 metadata obtained by the identity provider.

1102 The ECP **MUST** confirm that this value corresponds to the value the ECP obtained in the
1103 responseConsumerURL in the PAOS Request SOAP header block it received from the service
1104 provider. Since the responseConsumerURL **MAY** be relative and the
1105 AssertionConsumerServiceURL is absolute, some processing/normalization may be required.

1106 This mechanism is used for security purposes to confirm the correct response destination. If the
1107 values do not match, then the ECP **MUST** generate a SOAP fault response to the service provider
1108 and **MUST NOT** return the SAML response.

1109 The ECP Response SOAP header has no element content.

1110 Following is an example of an IdP-to-ECP response.

```
1111 <SOAP-ENV:Envelope  
1112     xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"  
1113     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
1114     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">  
1115   <SOAP-ENV:Header>  
1116     <ecp:Response SOAP-ENV:mustUnderstand="1" SOAP-  
1117     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"  
1118     AssertionConsumerServiceURL="https://ServiceProvider.example.com/ecp_assertion  
1119     _consumer"/>  
1120   </SOAP-ENV:Header>  
1121   <SOAP-ENV:Body>  
1122     <samlp:Response> ... </samlp:Response>  
1123   </SOAP-ENV:Body>  
1124 </SOAP-ENV:Envelope>
```

1125 **4.2.4.5 PAOS Response Header Block: ECP to SP**

1126 The PAOS Response header block includes the following attributes:

1127 SOAP-ENV:mustUnderstand [Required]

1128 The value **MUST** be 1 (true). A SOAP fault **MUST** be generated if the PAOS header block is not
1129 understood.

1130 SOAP-ENV:actor [Required]

1131 The value **MUST** be `http://schemas.xmlsoap.org/soap/actor/next`.

1132 refToMessageID [Optional]

1133 Allows correlation with the PAOS request. This optional attribute (and the header block as a whole)
1134 **MUST** be added by the ECP if the corresponding PAOS request specified the messageID attribute.
1135 Note that the equivalent functionality is provided in SAML using <AuthnRequest> and <Response>
1136 correlation.

1137 The PAOS Response SOAP header has no element content.

1138 Following is an example of an ECP-to-SP response.

```
1139 <SOAP-ENV:Envelope
1140     xmlns:paos="urn:liberty:paos:2003-08"
1141     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1142     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
1143   <SOAP-ENV:Header>
1144     <paos:Response refToMessageID="6c3a4f8b9c2d" SOAP-
1145     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next/" SOAP-
1146     ENV:mustUnderstand="1"/>
1147     <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
1148     SOAP-ENV:mustUnderstand="1" SOAP-
1149     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
1150       ...
1151     </ecp:RelayState>
1152   </SOAP-ENV:Header>
1153   <SOAP-ENV:Body>
1154     <samlp:Response> ... </samlp:Response>
1155   </SOAP-ENV:Body>
1156 </SOAP-ENV:Envelope>
```

1157 4.2.5 Security Considerations

1158 The <AuthnRequest> message SHOULD be signed. Per the rules specified by the browser SSO profile,
1159 the assertions enclosed in the <Response> MUST be signed. The delivery of the response in the SOAP
1160 envelope via PAOS is essentially analogous to the use of the HTTP POST binding and security
1161 countermeasures appropriate to that binding are used.

1162 The SOAP headers SHOULD be integrity protected, such as with SOAP Message Security or through the
1163 use of SSL/TLS over every HTTP exchange with the client.

1164 The service provider SHOULD be authenticated to the ECP, for example with server-side TLS
1165 authentication.

1166 The ECP SHOULD be authenticated to the identity provider, such as by maintaining an authenticated
1167 session. Any HTTP exchanges subsequent to the delivery of the <AuthnRequest> message and before
1168 the identity provider returns a <Response> MUST be securely associated with the original request.

1169 4.2.6 [E20]Use of Metadata

1170 The rules specified in the browser SSO profile in Section 4.1.6 apply here as well. Specifically, the indexed
1171 endpoint element <md:AssertionConsumerService> with a binding of
1172 urn:oasis:names:tc:SAML:2.0:bindings:PAOS MAY be used to describe the supported binding
1173 and location(s) to which an identity provider may send responses to a service provider using this profile. IN
1174 addition, the endpoint <md:SingleSignOnService> with a binding of
1175 urn:oasis:names:tc:SAML:2.0:bindings:SOAP MAY be used to describe the supported binding
1176 and location(s) to which an service provider may send requests to an identity provider using this profile.

1177 4.3 Identity Provider Discovery Profile

1178 This section defines a profile by which a service provider can discover which identity providers a principal
1179 is using with the Web Browser SSO profile. In deployments having more than one identity provider,
1180 service providers need a means to discover which identity provider(s) a principal uses. The discovery
1181 profile relies on a cookie that is written in a domain that is common between identity providers and service
1182 providers in a deployment. The domain that the deployment predetermines is known as the common
1183 domain in this profile, and the cookie containing the list of identity providers is known as the common
1184 domain cookie.

1185 Which entities host web servers in the common domain is a deployment issue and is outside the scope of
1186 this profile.

1187 **4.3.1 [E32]Required Information**

1188 **Identification:** <urn:oasis:names:tc:SAML:2.0:profiles:SSO:idp-discovery>

1189 **Contact information:** security-services-comment@lists.oasis-open.org

1190 **Description:** [Given below.](#)

1191 **Updates:** [None.](#)

1192 **4.3.2 Common Domain Cookie**

1193 The name of the cookie MUST be "_saml_idp". The format of the cookie value MUST be a set of one or
1194 more base-64 encoded URI values separated by a single space character. Each URI is the unique
1195 identifier of an identity provider, as defined in Section 8.3.6 of [SAMLCore]. The final set of values is then
1196 URL encoded.

1197 The common domain cookie writing service (see below) SHOULD append the identity provider's unique
1198 identifier to the list. If the identifier is already present in the list, it MAY remove and append it. The intent is
1199 that the most recently established identity provider session is the last one in the list.

1200 The cookie MUST be set with a Path prefix of "/". The Domain MUST be set to ".[common-domain]" where
1201 [common-domain] is the common domain established within the deployment for use with this profile.
1202 There MUST be a leading period. The cookie MUST be marked as secure.

1203 Cookie syntax should be in accordance with IETF RFC 2965 [RFC2965] or [NSCookie]. The cookie MAY
1204 be either session-only or persistent. This choice may be made within a deployment, but should apply
1205 uniformly to all identity providers in the deployment. [\[E63\]Note that while a session-only cookie can be
1206 used, the intent of this profile is not to provide a means of determining whether a user actually has an
1207 active session with one or more of the identity providers stored in the cookie. The cookie merely identifies
1208 identity providers known to have been used in the past. Service providers MAY instead rely on the
1209 IsPassive attribute in their <samlp:AuthnRequest> message to probe for active sessions.](#)

1210 **4.3.3 Setting the Common Domain Cookie**

1211 After the identity provider authenticates a principal, it MAY set the common domain cookie. The means by
1212 which the identity provider sets the cookie are implementation-specific so long as the cookie is
1213 successfully set with the parameters given above. One possible implementation strategy follows and
1214 should be considered non-normative. The identity provider may:

- 1215 • Have previously established a DNS and IP alias for itself in the common domain.
- 1216 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
1217 scheme. The structure of the URL is private to the implementation and may include session
1218 information needed to identify the user agent.
- 1219 • Set the cookie on the redirected user agent using the parameters specified above.
- 1220 • Redirect the user agent back to itself, or, if appropriate, to the service provider.

1221 **4.3.4 Obtaining the Common Domain Cookie**

1222 When a service provider needs to discover which identity providers a principal uses, it invokes an
1223 exchange designed to present the common domain cookie to the service provider after it is read by an
1224 HTTP server in the common domain.

1225 If the HTTP server in the common domain is operated by the service provider or if other arrangements are
1226 in place, the service provider MAY utilize the HTTP server in the common domain to relay its
1227 <AuthnRequest> to the identity provider for an optimized single sign-on process.

1228 The specific means by which the service provider reads the cookie are implementation-specific so long as
1229 it is able to cause the user agent to present cookies that have been set with the parameters given in
1230 Section 4.3.2. One possible implementation strategy is described as follows and should be considered
1231 non-normative. Additionally, it may be sub-optimal for some applications.

- 1232 • Have previously established a DNS and IP alias for itself in the common domain.
- 1233 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
1234 scheme. The structure of the URL is private to the implementation and may include session
1235 information needed to identify the user agent.
- 1236 • Redirect the user agent back to itself, or, if appropriate, to the identity provider.

1237 4.4 Single Logout Profile

1238 Once a principal has authenticated to an identity provider, the authenticating entity may establish a
1239 session with the principal (typically by means of a cookie, URL re-writing, or some other implementation-
1240 specific means). The identity provider may subsequently issue assertions to service providers or other
1241 relying parties, based on this authentication event; a relying party may use this to establish *its own* session
1242 with the principal.

1243 In such a situation, the identity provider can act as a session authority and the relying parties as session
1244 participants. At some later time, the principal may wish to terminate his or her session either with an
1245 individual session participant, or with all session participants in a given session managed by the session
1246 authority. The former case is considered out of scope of this specification. The latter case, however, may
1247 be satisfied using this profile of the SAML Single Logout protocol ([SAMLCore] Section 3.7).

1248 Note that a principal (or an administrator terminating a principal's session) may choose to terminate this
1249 "global" session either by contacting the session authority, or an individual session participant. Also note
1250 that an identity provider acting as a session authority may *itself* act as a session participant in situations in
1251 which it is the relying party for another identity provider's assertions regarding that principal.

1252 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1253 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1254 front-channel binding may be required, for example, in cases in which a principal's session state exists
1255 solely in a user agent in the form of a cookie and a direct interaction between the user agent and the
1256 session participant or session authority is required. As will be discussed below, session participants
1257 should if possible use a "front-channel" binding when initiating this profile to maximize the likelihood that
1258 the session authority can propagate the logout successfully to all participants.

1259 4.4.1 Required Information

1260 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:logout

1261 **Contact information:** security-services-comment@lists.oasis-open.org

1262 **Description:** Given below.

1263 **Updates:** None

1264 4.4.2 Profile Overview

1265 Figure 3 illustrates the basic template for achieving single logout:

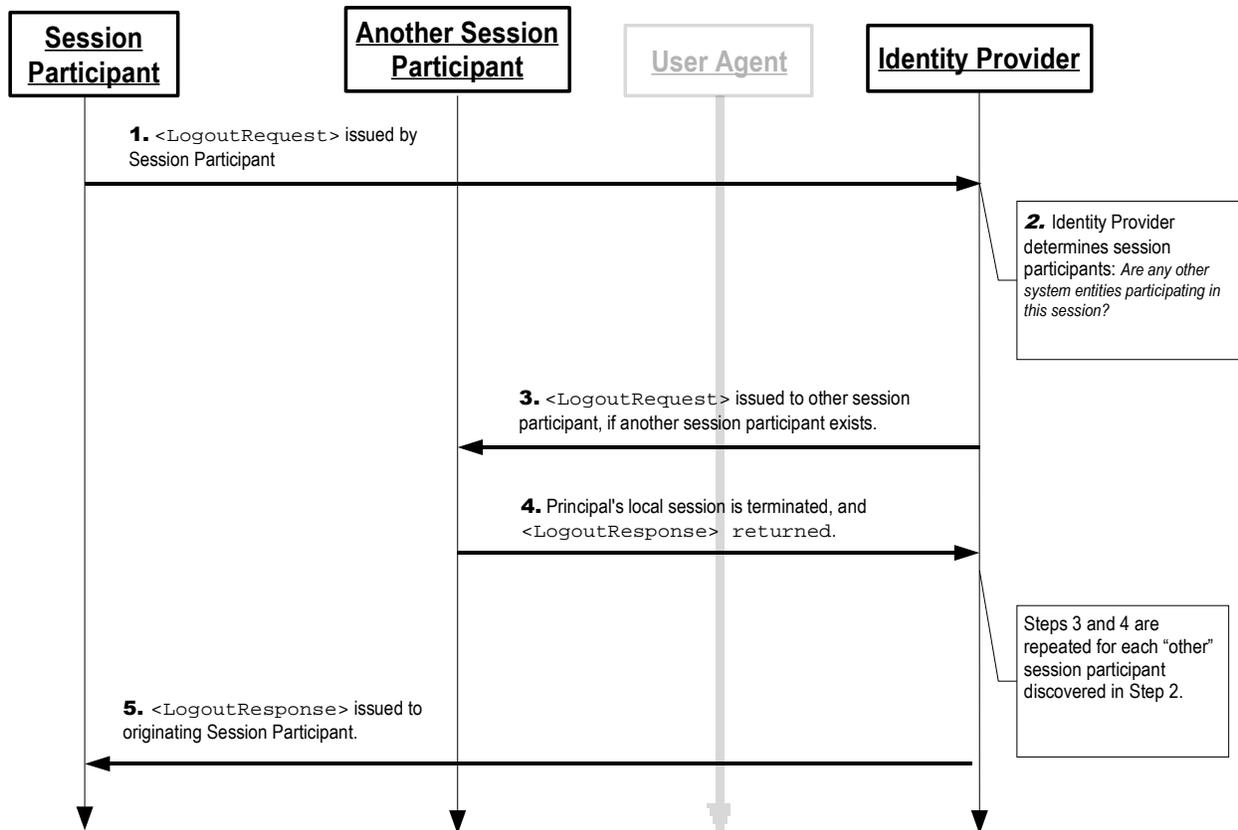


Figure 3

1266 The grayed-out user agent illustrates that the message exchange may pass through the user agent or
 1267 may be a direct exchange between system entities, depending on the SAML binding used to implement
 1268 the profile.

1269 The following steps are described by the profile. Within an individual step, there may be one or more
 1270 actual message exchanges depending on the binding used for that step and other implementation-
 1271 dependent behavior.

1272 **1. <LogoutRequest> issued by Session Participant to Identity Provider**

1273 In step 1, the session participant initiates single logout and terminates a principal's session(s) by
 1274 sending a <LogoutRequest> message to the identity provider from whom it received the
 1275 corresponding authentication assertion. The request may be sent directly to the identity provider
 1276 or sent indirectly through the user agent.

1277 **2. Identity Provider determines Session Participants**

1278 In step 2, the identity provider uses the contents of the <LogoutRequest> message (or if
 1279 initiating logout itself, some other mechanism) to determine the session(s) being terminated. If
 1280 there are no other session participants, the profile proceeds with step 5. Otherwise, steps 3 and 4
 1281 are repeated for each session participant identified.

1282 **3. <LogoutRequest> issued by Identity Provider to Session Participant/Authority**

1283 In step 3, the identity provider issues a <LogoutRequest> message to a session participant or
 1284 session authority related to one or more of the session(s) being terminated. The request may be
 1285 sent directly to the entity or sent indirectly through the user agent (if consistent with the form of the
 1286 request in step 1).

1287 4. Session Participant/Authority issues <LogoutResponse> to Identity Provider

1288 In step 4, a session participant or session authority terminates the principal's session(s) as
1289 directed by the request (if possible) and returns a <LogoutResponse> to the identity provider.
1290 The response may be returned directly to the identity provider or indirectly through the user agent
1291 (if consistent with the form of the request in step 3).

1292 5. Identity Provider issues <LogoutResponse> to Session Participant

1293 In step 5, the identity provider issues a <LogoutResponse> message to the original requesting
1294 session participant. The response may be returned directly to the session participant or indirectly
1295 through the user agent (if consistent with the form of the request in step 1).

1296 Note that an identity provider (acting as session authority) can initiate this profile at step 2 and issue a
1297 <LogoutRequest> to all session participants, also skipping step 5.

1298 4.4.3 Profile Description

1299 If the profile is initiated by a session participant, start with Section 4.4.3.1. If initiated by the identity
1300 provider, start with Section 4.4.3.2. In the descriptions below, the following is referred to:

1301 Single Logout Service

1302 This is the single logout protocol endpoint at an identity provider or session participant to which the
1303 <LogoutRequest> or <LogoutResponse> messages (or an artifact representing them) are
1304 delivered. The same or different endpoints MAY be used for requests and responses.

1305 4.4.3.1 <LogoutRequest> Issued by Session Participant to Identity Provider

1306 If the logout profile is initiated by a session participant, it examines the authentication assertion(s) it
1307 received pertaining to the session(s) being terminated, and collects the `SessionIndex` value(s) it
1308 received from the identity provider. If multiple identity providers are involved, then the profile MUST be
1309 repeated independently for each one.

1310 To initiate the profile, the session participant issues a <LogoutRequest> message to the identity
1311 provider's single logout service request endpoint containing one or more applicable <SessionIndex>
1312 elements. At least one element MUST be included. Metadata (as in [SAMLMeta]) MAY be used to
1313 determine the location of this endpoint and the bindings supported by the identity provider.

1314 Asynchronous Bindings (Front-Channel)

1315 The session participant SHOULD (if the principal's user agent is present) use an asynchronous
1316 binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind], to send the request to
1317 the identity provider through the user agent. The identity provider SHOULD then propagate any
1318 required logout messages to additional session participants as required using either a synchronous or
1319 asynchronous binding. The use of an asynchronous binding for the original request is preferred
1320 because it gives the identity provider the best chance of successfully propagating the logout to the
1321 other session participants during step 3.

1322 If the HTTP Redirect or POST binding is used, then the <LogoutRequest> message is delivered to
1323 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile
1324 defined in Section 5 is used by the identity provider, which makes a callback to the session participant
1325 to retrieve the <LogoutRequest> message, using for example the SOAP binding.

1326 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or
1327 TLS 1.0 [RFC2246] to maintain confidentiality and message integrity. The <LogoutRequest>
1328 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,
1329 if used, also provides for an alternate means of authenticating the request issuer when the artifact is
1330 dereferenced.

1331 Each of these bindings provide a RelayState mechanism that the session participant MAY use to
1332 associate the profile exchange with the original request. The session participant SHOULD reveal as
1333 little information as possible in the RelayState value unless the use of the profile does not require such
1334 privacy measures.

1335 **Synchronous Bindings (Back-Channel)**

1336 Alternatively, the session participant MAY use a synchronous binding, such as the SOAP binding
1337 [SAMLBind], to send the request directly to the identity provider. The identity provider SHOULD then
1338 propagate any required logout messages to additional session participants as required using a
1339 synchronous binding. The requester MUST authenticate itself to the identity provider, either by signing
1340 the <LogoutRequest> or using any other binding-supported mechanism.

1341 Profile-specific rules for the contents of the <LogoutRequest> message are included in Section 4.4.4.1.

1342 **4.4.3.2 Identity Provider Determines Session Participants**

1343 If the logout profile is initiated by an identity provider, or upon receiving a valid <LogoutRequest>
1344 message, the identity provider processes the request as defined in [SAMLCore]. It MUST examine the
1345 identifier and <SessionIndex> elements and determine the set of sessions to be terminated.

1346 The identity provider then follows steps 3 and 4 for each entity participating in the session(s) being
1347 terminated, other than the original requesting session participant (if any), as described in Section 3.7.3.2
1348 of [SAMLCore].

1349 **4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session 1350 Participant/Authority**

1351 To propagate the logout, the identity provider issues its own <LogoutRequest> to a session authority or
1352 participant in a session being terminated. The request is sent using a SAML binding consistent with the
1353 capability of the responder and the availability of the user agent at the identity provider.

1354 In general, the binding with which the original request was received in step 1 does not dictate the binding
1355 that may be used in this step except that as noted in step 1, using a synchronous binding that bypasses
1356 the user agent constrains the identity provider to use a similar binding to propagate additional requests.

1357 Profile-specific rules for the contents of the <LogoutRequest> message are included in Section 4.4.4.1.

1358 **4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity 1359 Provider**

1360 The session participant/authority MUST process the <LogoutRequest> message as defined in
1361 [SAMLCore]. After processing the message or upon encountering an error, the entity MUST issue a
1362 <LogoutResponse> message containing an appropriate status code to the requesting identity provider
1363 to complete the SAML protocol exchange.

1364 **Synchronous Bindings (Back-Channel)**

1365 If the identity provider used a synchronous binding, such as the SOAP binding [SAMLBind], the
1366 response is returned directly to complete the synchronous communication. The responder MUST
1367 authenticate itself to the requesting identity provider, either by signing the <LogoutResponse> or
1368 using any other binding-supported mechanism.

1369 **Asynchronous Bindings (Front-Channel)**

1370 If the identity provider used an asynchronous binding, such as the HTTP Redirect, POST, or Artifact
1371 bindings [SAMLBind], then the <LogoutResponse> (or artifact) is returned through the user agent to
1372 the identity provider's single logout service response endpoint. Metadata (as in [SAMLMeta]) MAY be
1373 used to determine the location of this endpoint and the bindings supported by the identity provider.

1374 Any asynchronous binding supported by both entities MAY be used.

1375 If the HTTP Redirect or POST binding is used, then the `<LogoutResponse>` message is delivered to
1376 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile
1377 defined in Section 5 is used by the identity provider, which makes a callback to the responding entity
1378 to retrieve the `<LogoutResponse>` message, using for example the SOAP binding.

1379 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or
1380 TLS 1.0 [RFC2246] to maintain confidentiality and message integrity. The `<LogoutResponse>`
1381 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,
1382 if used, also provides for an alternate means of authenticating the response issuer when the artifact is
1383 dereferenced.

1384 Profile-specific rules for the contents of the `<LogoutResponse>` message are included in Section
1385 4.4.4.2.

1386 **4.4.3.5 Identity Provider Issues `<LogoutResponse>` to Session Participant**

1387 After processing the original session participant's `<LogoutRequest>` as described in the previous steps
1388 the identity provider MUST respond to the original request with a `<LogoutResponse>` containing an
1389 appropriate status code to complete the SAML protocol exchange.

1390 The response is sent to the original session participant, using a SAML binding consistent with the binding
1391 used in the original request, the capability of the responder, and the availability of the user agent at the
1392 identity provider. Assuming an asynchronous binding was used in step 1, then any binding supported by
1393 both entities MAY be used.

1394 Profile-specific rules for the contents of the `<LogoutResponse>` message are included in Section
1395 4.4.4.2.

1396 **4.4.4 Use of Single Logout Protocol**

1397 **4.4.4.1 `<LogoutRequest>` Usage**

1398 The `<Issuer>` element MUST be present and MUST contain the unique identifier of the requesting entity;
1399 the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-
1400 format:entity`.

1401 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1402 the message or using a binding-specific mechanism.

1403 The principal MUST be identified in the request using an identifier that **strongly matches** the identifier in
1404 the authentication assertion the requester issued or received regarding the session being terminated, per
1405 the matching rules defined in Section 3.3.4 of [SAMLCore].

1406 If the requester is a session participant, it MUST include at least one `<SessionIndex>` element in the
1407 request. [\[E38\]\(Note that the session participant always receives a `SessionIndex` attribute in the
1408 `<saml:AuthnStatement>` elements that it receives to initiate the session, per Section 4.1.4.2 of the
1409 \[Web Browser SSO Profile\]\(#\).\)](#) If the requester is a session authority (or acting on its behalf), then it MAY
1410 omit any such elements to indicate the termination of all of the principal's applicable sessions.

1411 **4.4.4.2 `<LogoutResponse>` Usage**

1412 The `<Issuer>` element MUST be present and MUST contain the unique identifier of the responding
1413 entity; the `Format` attribute MUST be omitted or have a value of
1414 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

1415 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1416 the message or using a binding-specific mechanism.

1417 4.4.5 Use of Metadata

1418 [SAMLMeta] defines an endpoint element, <md:SingleLogoutService>, to describe supported
1419 bindings and location(s) to which an entity may send requests and responses using this profile.

1420 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>
1421 element with a use attribute of encryption to determine an appropriate encryption algorithm and
1422 settings to use, along with a public key to use in delivering a bulk encryption key.

1423 4.5 Name Identifier Management Profile

1424 In the scenario supported by the Name Identifier Management profile, an identity provider has exchanged
1425 some form of ~~[E55]persistent~~long-term identifier ~~(including but not limited to identifiers with a Format of~~
1426 ~~urn:oasis:names:tc:SAML:2.0:nameid-format:persistent)~~ for a principal with a service
1427 provider, allowing them to share a common identifier for some length of time. Subsequently, the identity
1428 provider may wish to notify the service provider of a change in the ~~[E12]format and/or~~value that it will use
1429 to identify the same principal in the future. Alternatively the service provider may wish to attach its own
1430 "alias" for the principal in order to ensure that the identity provider will include it when communicating with
1431 it in the future ~~about the principal~~[E55]using that identifier. Finally, one of the providers may wish to inform
1432 the other that it will no longer issue or accept messages using a particular identifier. To implement these
1433 scenarios, a profile of the SAML Name Identifier Management protocol is used.

1434 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1435 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1436 front-channel binding may be required, for example, in cases in which direct interaction between the user
1437 agent and the responding provider is required in order to effect the change.

1438 4.5.1 Required Information

1439 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:nameid-mgmt

1440 **Contact information:** security-services-comment@lists.oasis-open.org

1441 **Description:** Given below.

1442 **Updates:** None.

1443 4.5.2 Profile Overview

1444 Figure 4 illustrates the basic template for the name identifier management profile.

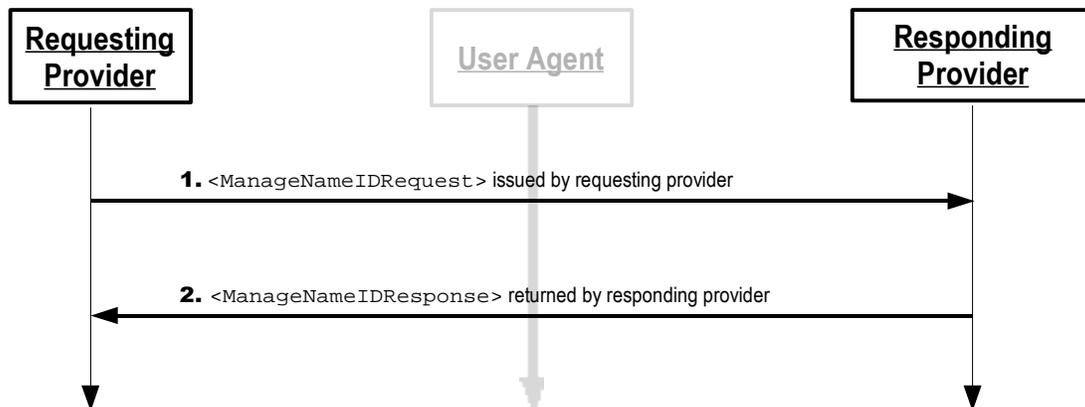


Figure 4

1445 The grayed-out user agent illustrates that the message exchange may pass through the user agent or
 1446 may be a direct exchange between system entities, depending on the SAML binding used to implement
 1447 the profile.

1448 The following steps are described by the profile. Within an individual step, there may be one or more
 1449 actual message exchanges depending on the binding used for that step and other implementation-
 1450 dependent behavior.

1451 **1. <ManageNameIDRequest> issued by Requesting Identity/Service Provider**

1452 In step 1, an identity or service provider initiates the profile by sending a
 1453 <ManageNameIDRequest> message to another provider that it wishes to inform of a change.
 1454 The request may be sent directly to the responding provider or sent indirectly through the user
 1455 agent.

1456 **2. <ManageNameIDResponse> issued by Responding Identity/Service Provider**

1457 In step 2, the responding provider (after processing the request) issues a
 1458 <ManageNameIDResponse> message to the original requesting provider. The response may be
 1459 returned directly to the requesting provider or indirectly through the user agent (if consistent with
 1460 the form of the request in step 1).

1461 **4.5.3 Profile Description**

1462 In the descriptions below, the following is referred to:

1463 **Name Identifier Management Service**

1464 This is the name identifier management protocol endpoint at an identity or service provider to which
 1465 the <ManageNameIDRequest> or <ManageNameIDResponse> messages (or an artifact
 1466 representing them) are delivered. The same or different endpoints MAY be used for requests and
 1467 responses.

1468 **4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider**

1469 To initiate the profile, the requesting provider issues a <ManageNameIDRequest> message to another
 1470 provider's name identifier management service request endpoint. Metadata (as in [SAMLMeta]) MAY be
 1471 used to determine the location of this endpoint and the bindings supported by the responding provider.

1472 **Synchronous Bindings (Back-Channel)**

1473 The requesting provider MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to

1474 send the request directly to the other provider. The requester MUST authenticate itself to the other
1475 provider, either by signing the <ManageNameIDRequest> or using any other binding-supported
1476 mechanism.

1477 **Asynchronous Bindings (Front-Channel)**

1478 Alternatively, the requesting provider MAY (if the principal's user agent is present) use an
1479 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to send the
1480 request to the other provider through the user agent.

1481 If the HTTP Redirect or POST binding is used, then the <ManageNameIDRequest> message is
1482 delivered to the other provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution
1483 profile defined in Section 5 is used by the other provider, which makes a callback to the requesting
1484 provider to retrieve the <ManageNameIDRequest> message, using for example the SOAP binding.

1485 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or
1486 TLS 1.0 [RFC2246] to maintain confidentiality and message integrity. The
1487 <ManageNameIDRequest> message MUST be signed if the HTTP POST or Redirect binding is
1488 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1489 request issuer when the artifact is dereferenced.

1490 Each of these bindings provide a RelayState mechanism that the requesting provider MAY use to
1491 associate the profile exchange with the original request. The requesting provider SHOULD reveal as
1492 little information as possible in the RelayState value unless the use of the profile does not require such
1493 privacy measures.

1494 Profile-specific rules for the contents of the <ManageNameIDRequest> message are included in Section
1495 4.5.4.1.

1496 **4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service 1497 Provider**

1498 The recipient MUST process the <ManageNameIDRequest> message as defined in [SAMLCore]. After
1499 processing the message or upon encountering an error, the recipient MUST issue a
1500 <ManageNameIDResponse> message containing an appropriate status code to the requesting provider
1501 to complete the SAML protocol exchange.

1502 **Synchronous Bindings (Back-Channel)**

1503 If the requesting provider used a synchronous binding, such as the SOAP binding [SAMLBind], the
1504 response is returned directly to complete the synchronous communication. The responder MUST
1505 authenticate itself to the requesting provider, either by signing the <ManageNameIDResponse> or
1506 using any other binding-supported mechanism.

1507 **Asynchronous Bindings (Front-Channel)**

1508 If the requesting provider used an asynchronous binding, such as the HTTP Redirect, POST, or
1509 Artifact bindings [SAMLBind], then the <ManageNameIDResponse> (or artifact) is returned through
1510 the user agent to the requesting provider's name identifier management service response endpoint.
1511 Metadata (as in [SAMLMeta]) MAY be used to determine the location of this endpoint and the bindings
1512 supported by the requesting provider. Any binding supported by both entities MAY be used.

1513 If the HTTP Redirect or POST binding is used, then the <ManageNameIDResponse> message is
1514 delivered to the requesting provider in this step. If the HTTP Artifact binding is used, the Artifact
1515 Resolution profile defined in Section 5 is used by the requesting provider, which makes a callback to
1516 the responding provider to retrieve the <ManageNameIDResponse> message, using for example the
1517 SOAP binding.

1518 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or
1519 TLS 1.0 [RFC2246] to maintain confidentiality and message integrity. The
1520 <ManageNameIDResponse> message MUST be signed if the HTTP POST or Redirect binding is

1521 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1522 response issuer when the artifact is dereferenced.

1523 Profile-specific rules for the contents of the <ManageNameIDResponse> message are included in
1524 Section 4.5.4.2.

1525 **4.5.4 Use of Name Identifier Management Protocol**

1526 **4.5.4.1 <ManageNameIDRequest> Usage**

1527 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1528 the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`
1529 `format:entity`.

1530 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1531 the message or using a binding-specific mechanism.

1532 **4.5.4.2 <ManageNameIDResponse> Usage**

1533 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1534 entity; the `Format` attribute MUST be omitted or have a value of
1535 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

1536 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1537 the message or using a binding-specific mechanism.

1538 **4.5.5 Use of Metadata**

1539 [SAMLMeta] defines an endpoint element, <md:ManageNameIDService>, to describe supported
1540 bindings and location(s) to which an entity may send requests and responses using this profile.

1541 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>
1542 element with a `use` attribute of `encryption` to determine an appropriate encryption algorithm and
1543 settings to use, along with a public key to use in delivering a bulk encryption key.

1544 5 Artifact Resolution Profile

1545 [SAMLCore] defines an Artifact Resolution protocol for dereferencing a SAML artifact into a corresponding
1546 protocol message. The HTTP Artifact binding in [SAMLBind] leverages this mechanism to pass SAML
1547 protocol messages by reference. This profile describes the use of this protocol with a synchronous
1548 binding, such as the SOAP binding defined in [SAMLBind].

1549 5.1 Required Information

1550 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:artifact

1551 **Contact information:** security-services-comment@lists.oasis-open.org

1552 **Description:** Given below.

1553 **Updates:** None

1554 5.2 Profile Overview

1555 The message exchange and basic processing rules that govern this profile are largely defined by Section
1556 3.5 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1557 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1558 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1559 Figure 5 illustrates the basic template for the artifact resolution profile.

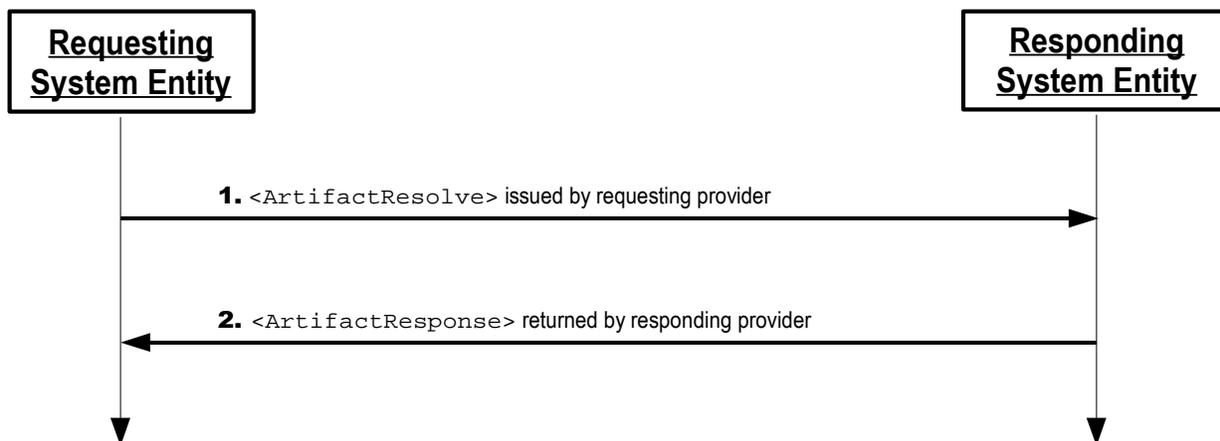


Figure 5

1560 The following steps are described by the profile.

1561 1. <ArtifactResolve> issued by Requesting Entity

1562 In step 1, a requester initiates the profile by sending an <ArtifactResolve> message to an
1563 artifact issuer.

1564 2. <ArtifactResponse> issued by Responding Entity

1565 In step 2, the responder (after processing the request) issues an <ArtifactResponse>
1566 message to the requester.

1567 5.3 Profile Description

1568 In the descriptions below, the following is referred to:

1569 Artifact Resolution Service

1570 This is the artifact resolution protocol endpoint at an artifact issuer to which <ArtifactResolve>
1571 messages are delivered.

1572 5.3.1 <ArtifactResolve> issued by Requesting Entity

1573 To initiate the profile, a requester, having received an artifact and determined the issuer using the
1574 SourceID, sends an <ArtifactResolve> message containing the artifact to an artifact issuer's artifact
1575 resolution service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of this
1576 endpoint and the bindings supported by the artifact issuer.

1577 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1578 request directly to the artifact issuer. The requester SHOULD authenticate itself to the responder, either by
1579 signing the <ArtifactResolve> message or using any other binding-supported mechanism. Specific
1580 profiles that use the HTTP Artifact binding MAY impose additional requirements such that authentication is
1581 mandatory.

1582 Profile-specific rules for the contents of the <ArtifactResolve> message are included in Section 5.4.1.

1583 5.3.2 <ArtifactResponse> issued by Responding Entity

1584 The artifact issuer MUST process the <ArtifactResolve> message as defined in [SAMLCore]. After
1585 processing the message or upon encountering an error, the artifact issuer MUST return an
1586 <ArtifactResponse> message containing an appropriate status code to the requester to complete the
1587 SAML protocol exchange. If successful, the dereferenced SAML protocol message corresponding to the
1588 artifact will also be included.

1589 The responder MUST authenticate itself to the requester, either by signing the <ArtifactResponse> or
1590 using any other binding-supported mechanism.

1591 Profile-specific rules for the contents of the <ArtifactResponse> message are included in Section
1592 5.4.2.

1593 5.4 Use of Artifact Resolution Protocol

1594 5.4.1 <ArtifactResolve> Usage

1595 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1596 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1597 format:entity.

1598 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by
1599 signing the message or using a binding-specific mechanism. Specific profiles that use the HTTP Artifact
1600 binding MAY impose additional requirements such that authentication is mandatory.

1601 **5.4.2 <ArtifactResponse> Usage**

1602 The <Issuer> element MUST be present and MUST contain the unique identifier of the artifact issuer;
1603 the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`
1604 `format:entity`.

1605 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1606 the message or using a binding-specific mechanism.

1607 **5.5 Use of Metadata**

1608 [SAMLMeta] defines an indexed endpoint element, <md:ArtifactResolutionService>, to describe
1609 supported bindings and location(s) to which a requester may send requests using this profile. The `index`
1610 attribute is used to distinguish the possible endpoints that may be specified by reference in the artifact's
1611 `EndpointIndex` field.

1612 6 Assertion Query/Request Profile

1613 [SAMLCore] defines a protocol for requesting existing assertions by reference or by querying on the basis
1614 of a subject and additional statement-specific criteria. This profile describes the use of this protocol with a
1615 synchronous binding, such as the SOAP binding defined in [SAMLBind].

1616 6.1 Required Information

1617 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:query

1618 **Contact information:** security-services-comment@lists.oasis-open.org

1619 **Description:** Given below.

1620 **Updates:** None.

1621 6.2 Profile Overview

1622 The message exchange and basic processing rules that govern this profile are largely defined by Section
1623 3.3 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1624 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1625 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1626 Figure 6 illustrates the basic template for the query/request profile.

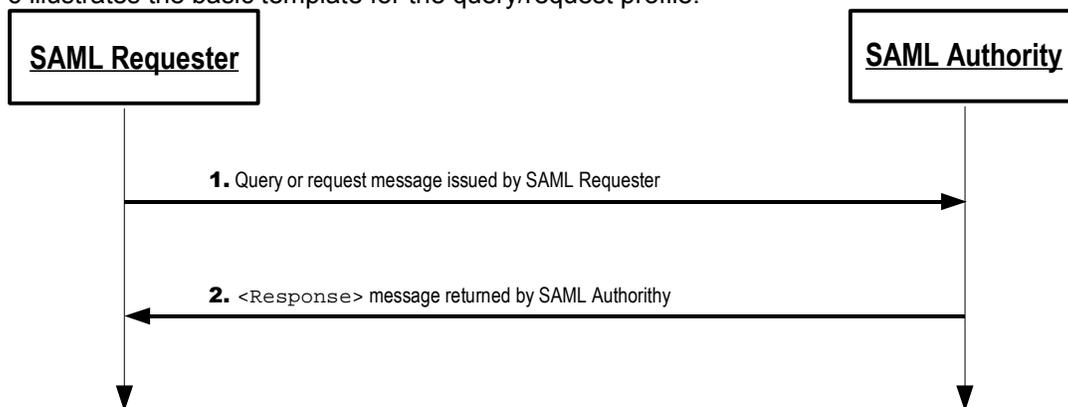


Figure 6

1627 The following steps are described by the profile.

1628 1. Query/Request issued by SAML Requester

1629 In step 1, a SAML requester initiates the profile by sending an `<AssertionIDRequest>`,
1630 `<SubjectQuery>`, `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>`
1631 message to a SAML authority.

1632 2. <Response> issued by SAML Authority

1633 In step 2, the responding SAML authority (after processing the query or request) issues a
1634 `<Response>` message to the SAML requester.

1635 **6.3 Profile Description**

1636 In the descriptions below, the following are referred to:

1637 **Query/Request Service**

1638 This is the query/request protocol endpoint at a SAML authority to which query or
1639 `<AssertionIDRequest>` messages are delivered.

1640 **6.3.1 Query/Request issued by SAML Requester**

1641 To initiate the profile, a SAML requester issues an `<AssertionIDRequest>`, `<SubjectQuery>`,
1642 `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>` message to a SAML authority's
1643 query/request service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of
1644 this endpoint and the bindings supported by the SAML authority.

1645 The SAML requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send
1646 the request directly to the identity provider. The requester SHOULD authenticate itself to the SAML
1647 authority either by signing the message or using any other binding-supported mechanism.

1648 Profile-specific rules for the contents of the various messages are included in Section 6.4.1.

1649 **6.3.2 `<Response>` issued by SAML Authority**

1650 The SAML authority MUST process the query or request message as defined in [SAMLCore]. After
1651 processing the message or upon encountering an error, the SAML authority MUST return a `<Response>`
1652 message containing an appropriate status code to the SAML requester to complete the SAML protocol
1653 exchange. If the request is successful in locating one or more matching assertions, they will also be
1654 included in the response.

1655 The responder SHOULD authenticate itself to the requester, either by signing the `<Response>` or using
1656 any other binding-supported mechanism.

1657 Profile-specific rules for the contents of the `<Response>` message are included in Section 6.4.2.

1658 **6.4 Use of Query/Request Protocol**

1659 **6.4.1 Query/Request Usage**

1660 The `<Issuer>` element MUST be present.

1661 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by
1662 signing the message or using a binding-specific mechanism.

1663 **6.4.2 `<Response>` Usage**

1664 The `<Issuer>` element MUST be present and MUST contain the unique identifier of the responding
1665 SAML authority; the `Format` attribute MUST be omitted or have a value of
1666 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`. Note that this need not necessarily
1667 match the `<Issuer>` element in the returned assertion(s).

1668 The responder SHOULD authenticate itself to the requester and ensure message integrity, either by
1669 signing the message or using a binding-specific mechanism.

1670 **6.5 Use of Metadata**

1671 [SAMLMeta] defines several endpoint elements, `<md:AssertionIDRequestService>`,
1672 `<md:AuthnQueryService>`, `<md:AttributeService>`, and `<md:AuthzService>`, to describe
1673 supported bindings and location(s) to which a requester may send requests or queries using this profile.

1674 The SAML authority, if encrypting the resulting assertions or assertion contents for a particular entity, can
1675 use that entity's `<md:KeyDescriptor>` element with a `use` attribute of `encryption` to determine an
1676 appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk
1677 encryption key.

1678 The various role descriptors MAY contain `<md:NameIDFormat>`, `<md:AttributeProfile>`, and
1679 `<saml:Attribute>` elements (as applicable) to indicate the general ability to support particular name
1680 identifier formats, attribute profiles, or specific attributes and values. The ability to support any such
1681 features during a given request is dependent on policy and the discretion of the authority.

1682

7 Name Identifier Mapping Profile

1683 [SAMLCore] defines a Name Identifier Mapping protocol for mapping a principal's name identifier into a
1684 different name identifier for the same principal. This profile describes the use of this protocol with a
1685 synchronous binding, such as the SOAP binding defined in [SAMLBind], and additional guidelines for
1686 protecting the privacy of the principal with encryption and limiting the use of the mapped identifier.

7.1 Required Information

1688 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:nameidmapping

1689 **Contact information:** security-services-comment@lists.oasis-open.org

1690 **Description:** Given below.

1691 **Updates:** None.

7.2 Profile Overview

1693 The message exchange and basic processing rules that govern this profile are largely defined by Section
1694 3.8 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1695 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1696 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1697 Figure 7 illustrates the basic template for the name identifier mapping profile.

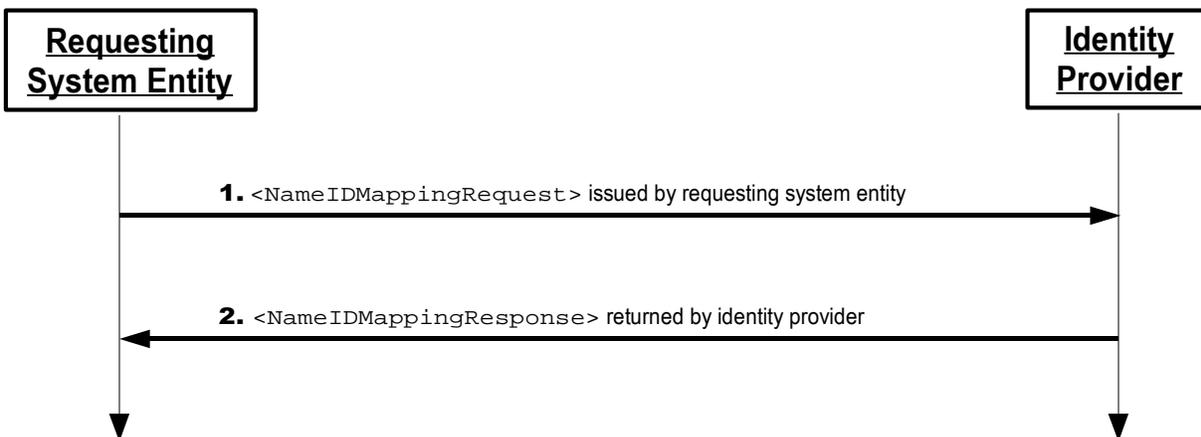


Figure 7

1698 The following steps are described by the profile.

1699 1. <NameIDMappingRequest> issued by Requesting Entity

1700 In step 1, a requester initiates the profile by sending a <NameIDMappingRequest> message to
1701 an identity provider.

1702 2. <NameIDMappingResponse> issued by Identity Provider

1703 In step 2, the responding identity provider (after processing the request) issues a
1704 <NameIDMappingResponse> message to the requester.

1705 **7.3 Profile Description**

1706 In the descriptions below, the following is referred to:

1707 **Name Identifier Mapping Service**

1708 This is the name identifier mapping protocol endpoint at an identity provider to which
1709 <NameIDMappingRequest> messages are delivered.

1710 **7.3.1 <NameIDMappingRequest> issued by Requesting Entity**

1711 To initiate the profile, a requester issues a <NameIDMappingRequest> message to an identity provider's
1712 name identifier mapping service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the
1713 location of this endpoint and the bindings supported by the identity provider.

1714 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1715 request directly to the identity provider. The requester MUST authenticate itself to the identity provider,
1716 either by signing the <NameIDMappingRequest> or using any other binding-supported mechanism.

1717 Profile-specific rules for the contents of the <NameIDMappingRequest> message are included in
1718 Section 7.4.1.

1719 **7.3.2 <NameIDMappingResponse> issued by Identity Provider**

1720 The identity provider MUST process the <ManageNameIDRequest> message as defined in [SAMLCore].
1721 After processing the message or upon encountering an error, the identity provider MUST return a
1722 <NameIDMappingResponse> message containing an appropriate status code to the requester to
1723 complete the SAML protocol exchange.

1724 The responder MUST authenticate itself to the requester, either by signing the
1725 <NameIDMappingResponse> or using any other binding-supported mechanism.

1726 Profile-specific rules for the contents of the <NameIDMappingResponse> message are included in
1727 Section 7.4.2.

1728 **7.4 Use of Name Identifier Mapping Protocol**

1729 **7.4.1 <NameIDMappingRequest> Usage**

1730 The <Issuer> element MUST be present.

1731 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1732 the message or using a binding-specific mechanism.

1733 **7.4.2 <NameIDMappingResponse> Usage**

1734 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1735 identity provider; the Format attribute MUST be omitted or have a value of
1736 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1737 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1738 the message or using a binding-specific mechanism.

1739 Section 2.2.3 of [SAMLCore] defines the use of encryption to apply confidentiality to a name identifier. In
1740 most cases, the identity provider SHOULD encrypt the mapped name identifier it returns to the requester
1741 to protect the privacy of the principal. The requester can extract the <EncryptedID> element and place it
1742 in subsequent protocol messages or assertions.

1743 **7.4.2.1 Limiting Use of Mapped Identifier**

1744 Additional limits on the use of the resulting identifier MAY be applied by the identity provider by returning
1745 the mapped name identifier in the form of an <Assertion> containing the identifier in its <Subject> but
1746 without any statements. The assertion is then encrypted and the result used as the <EncryptedData>
1747 element in the <EncryptedID> returned to the requester. The assertion MAY include a <Conditions>
1748 element to limit use, as defined by [SAMLCore], such as time-based constraints or use by specific relying
1749 parties, and MUST be signed for integrity protection.

1750 **7.5 Use of Metadata**

1751 [SAMLMeta] defines an endpoint element, <md:NameIDMappingService>, to describe supported
1752 bindings and location(s) to which a requester may send requests using this profile.

1753 The identity provider, if encrypting the resulting identifier for a particular entity, can use that entity's
1754 <md:KeyDescriptor> element with a use attribute of encryption to determine an appropriate
1755 encryption algorithm and settings to use, along with a public key to use in delivering a bulk encryption key.

1756 8 SAML Attribute Profiles

1757 8.1 Basic Attribute Profile

1758 The Basic attribute profile specifies simplified, but non-unique, naming of SAML attributes together with
1759 attribute values based on the built-in XML Schema data types, eliminating the need for extension schemas
1760 to validate syntax.

1761 8.1.1 Required Information

1762 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:basic

1763 **Contact information:** security-services-comment@lists.oasis-open.org

1764 **Description:** Given below.

1765 **Updates:** None.

1766 8.1.2 SAML Attribute Naming

1767 The NameFormat XML attribute in <Attribute> elements MUST be
1768 urn:oasis:names:tc:SAML:2.0:attrname-format:basic.

1769 The Name XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

1770 8.1.2.1 Attribute Name Comparison

1771 Two <Attribute> elements refer to the same SAML attribute if and only if the values of their Name XML
1772 attributes are equal in the sense of Section 3.3.6 of [Schema2].

1773 8.1.3 Profile-Specific XML Attributes

1774 No additional XML attributes are defined for use with the <Attribute> element.

1775 8.1.4 SAML Attribute Values

1776 The schema type of the contents of the <AttributeValue> element MUST be drawn from one of the
1777 types defined in Section 3[E51]-3 of [Schema2]. The xsi:type attribute MUST be present and be given
1778 the appropriate value.

1779 8.1.5 Example

```
1780 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"  
1781     Name="FirstName">  
1782     <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>  
1783 </saml:Attribute>
```

1784 8.2 X.500/LDAP Attribute Profile [E53] – Deprecated

1785 **[E53]Note:** [This attribute profile is deprecated because of a flaw that makes it schema-](#)
1786 [invalid. The SSTC has replaced it with a separately published SAML V2.0 X.500/LDAP](#)
1787 [Attribute Profile specification that removes this flaw.](#)

1788 Directories based on the ITU-T X.500 specifications [X.500] and the related IETF Lightweight Directory
1789 Access Protocol specifications [LDAP] are widely deployed. Directory schema is used to model
1790 information to be stored in these directories. In particular, in X.500, attribute type definitions are used to
1791 specify the syntax and other features of attributes, the basic information storage unit in a directory (this
1792 document refers to these as “directory attributes”). Directory attribute types are defined in schema in the
1793 X.500 and LDAP specifications themselves, schema in other public documents (such as the
1794 Internet2/Educause EduPerson schema [eduPerson], or the inetOrgperson schema [RFC2798]), and
1795 schema defined for private purposes. In any of these cases, it is useful for deployers to take advantage of
1796 these directory attribute types in the context of SAML attribute statements, without having to manually
1797 create SAML-specific attribute definitions for them, and to do this in an interoperable fashion.
1798 The X.500/LDAP attribute profile defines a common convention for the naming and representation of such
1799 attributes when expressed as SAML attributes.

1800 8.2.1 Required Information

1801 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500 (this is also the target namespace
1802 assigned in the corresponding X.500/LDAP profile schema document [SAMLX500-xsd])

1803 **Contact information:** security-services-comment@lists.oasis-open.org

1804 **Description:** Given below.

1805 **Updates:** None.

1806 8.2.2 SAML Attribute Naming

1807 The `NameFormat` XML attribute in `<Attribute>` elements **MUST** be
1808 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1809 To construct attribute names, the URN `oid` namespace described in IETF RFC 3061 [RFC3061] is used.
1810 In this approach the `Name` XML attribute is based on the OBJECT IDENTIFIER assigned to the directory
1811 attribute type.

1812 **Example:**

1813 `urn:oid:2.5.4.3`

1814 Since X.500 procedures require that every attribute type be identified with a unique OBJECT IDENTIFIER,
1815 this naming scheme ensures that the derived SAML attribute names are unambiguous.

1816 For purposes of human readability, there may also be a requirement for some applications to carry an
1817 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
1818 [SAMLCore]) **MAY** be used for this purpose. If the definition of the directory attribute type includes one or
1819 more descriptors (short names) for the attribute type, the `FriendlyName` value, if present, **SHOULD** be
1820 one of the defined descriptors.

1821 8.2.2.1 Attribute Name Comparison

1822 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
1823 values are equal in the sense of [RFC3061]. The `FriendlyName` attribute plays no role in the
1824 comparison.

1825 8.2.3 Profile-Specific XML Attributes

1826 No additional XML attributes are defined for use with the `<Attribute>` element.

1827 8.2.4 SAML Attribute Values

1828 Directory attribute type definitions for use in native X.500 directories specify the syntax of the attribute
1829 using ASN.1 [ASN.1]. For use in LDAP, directory attribute definitions additionally include an LDAP syntax
1830 which specifies how attribute or assertion values conforming to the syntax are to be represented when
1831 transferred in the LDAP protocol (known as an LDAP-specific encoding). The LDAP-specific encoding
1832 commonly produces Unicode characters in UTF-8 form. This SAML attribute profile specifies the form of
1833 SAML attribute values only for those directory attributes which have LDAP syntaxes. Future extensions to
1834 this profile may define attribute value formats for directory attributes whose syntaxes specify other
1835 encodings.

1836 To represent the encoding rules in use for a particular attribute value, the <AttributeValue> element
1837 MUST contain an XML attribute named `Encoding` defined in the XML namespace
1838 `urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500`. (See [E53] for an issue with this
1839 [attribute](#).)

1840 For any directory attribute with a syntax whose LDAP-specific encoding exclusively produces UTF-8
1841 character strings as values, the SAML attribute value is encoded as simply the UTF-8 string itself, as the
1842 content of the <AttributeValue> element, with no additional whitespace. In such cases, the
1843 `xsi:type` XML attribute MUST be set to `xs:string`. The profile-specific `Encoding` XML attribute is
1844 provided, with a value of `LDAP`.

1845 A list of some LDAP attribute syntaxes to which this applies is:

1846	Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3
1847	Bit String	1.3.6.1.4.1.1466.115.121.1.6
1848	Boolean	1.3.6.1.4.1.1466.115.121.1.7
1849	Country String	1.3.6.1.4.1.1466.115.121.1.11
1850	DN	1.3.6.1.4.1.1466.115.121.1.12
1851	Directory String	1.3.6.1.4.1.1466.115.121.1.15
1852	Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22
1853	Generalized Time	1.3.6.1.4.1.1466.115.121.1.24
1854	IA5 String	1.3.6.1.4.1.1466.115.121.1.26
1855	INTEGER	1.3.6.1.4.1.1466.115.121.1.27
1856	LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54
1857	Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30
1858	Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31
1859	Name And Optional UID	1.3.6.1.4.1.1466.115.121.1.34
1860	Name Form Description	1.3.6.1.4.1.1466.115.121.1.35
1861	Numeric String	1.3.6.1.4.1.1466.115.121.1.36
1862	Object Class Description	1.3.6.1.4.1.1466.115.121.1.37
1863	Octet String	1.3.6.1.4.1.1466.115.121.1.40
1864	OID	1.3.6.1.4.1.1466.115.121.1.38
1865	Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39
1866	Postal Address	1.3.6.1.4.1.1466.115.121.1.41
1867	Presentation Address	1.3.6.1.4.1.1466.115.121.1.43
1868	Printable String	1.3.6.1.4.1.1466.115.121.1.44
1869	Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58
1870	Telephone Number	1.3.6.1.4.1.1466.115.121.1.50
1871	UTC Time	1.3.6.1.4.1.1466.115.121.1.53

1872 For all other LDAP syntaxes, the attribute value is encoded, as the content of the <AttributeValue>
1873 element, by base64-encoding [RFC2045] the [\[E48\]encompassingcontents of the](#) ASN.1 OCTET STRING-
1874 encoded LDAP attribute value ([not including the ASN.1 OCTET STRING wrapper](#)). The `xsi:type` XML
1875 attribute MUST be set to `xs:base64Binary`. The profile-specific `Encoding` XML attribute is provided,
1876 with a value of `LDAP`.

1877 When comparing SAML attribute values for equality, the matching rules specified for the corresponding
1878 directory attribute type MUST be observed (case sensitivity, for example).

1879 8.2.5 Profile-Specific Schema

1880 The following schema listing shows how the profile-specific Encoding XML attribute is defined
1881 [SAMLX500-xsd]:

```
1882 <schema  
1883   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"  
1884   xmlns="http://www.w3.org/2001/XMLSchema"  
1885   elementFormDefault="unqualified"  
1886   attributeFormDefault="unqualified"  
1887   blockDefault="substitution"  
1888   version="2.0">  
1889   <annotation>  
1890     <documentation>  
1891       Document identifier: saml-schema-x500-2.0  
1892       Location: http://docs.oasis-open.org/security/saml/v2.0/  
1893       Revision history:  
1894         V2.0 (March, 2005):  
1895         Custom schema for X.500 attribute profile, first published in  
1896 SAML 2.0.  
1897     </documentation>  
1898   </annotation>  
1899   <attribute name="Encoding" type="string"/>  
1900 </schema>
```

1901 8.2.6 Example

1902 The following is an example of a mapping of the "givenName" directory attribute, representing the SAML
1903 assertion subject's first name. It's OBJECT IDENTIFIER is 2.5.4.42 and its LDAP syntax is Directory
1904 String.

```
1905 <saml:Attribute  
1906   xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"  
1907   NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1908   Name="urn:oid:2.5.4.42" FriendlyName="givenName">  
1909   <saml:AttributeValue xsi:type="xs:string"  
1910     x500:Encoding="LDAP">Steven</saml:AttributeValue>  
1911 </saml:Attribute>
```

1912 8.3 UUID Attribute Profile

1913 The UUID attribute profile standardizes the expression of UUID values as SAML attribute names and
1914 values. It is applicable when the attribute's source system is one that identifies an attribute or its value with
1915 a UUID.

1916 8.3.1 Required Information

1917 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:UUID

1918 **Contact information:** security-services-comment@lists.oasis-open.org

1919 **Description:** Given below.

1920 **Updates:** None.

1921 8.3.2 UUID and GUID Background

1922 UUIDs (Universally Unique Identifiers), also known as GUIDs (Globally Unique Identifiers), are used to
1923 define objects and subjects such that they are guaranteed uniqueness across space and time. UUIDs

1924 were originally used in the Network Computing System (NCS), and then used in the Open Software
1925 Foundation's (OSF) Distributed Computing Environment (DCE). Recently GUIDs have been used in
1926 Microsoft's COM and Active Directory/Windows 2000/2003 platform.

1927 A UUID is a 128 bit number, generated such that it should never be duplicated within the domain of
1928 interest. UUIDs are used to represent a wide range of objects including, but not limited to, subjects/users,
1929 groups of users and node names. A UUID, represented as a hexadecimal string, is as follows:

1930 `f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1931 In DCE and Microsoft Windows, the UUID is usually presented to the administrator in the form of a
1932 "friendly name". For instance the above UUID could represent the user john.doe@example.com.

1933 **8.3.3 SAML Attribute Naming**

1934 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
1935 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1936 If the underlying representation of the attribute's name is a UUID, then the URN `uuid` namespace
1937 described in [Mealling] is used. In this approach the `Name` XML attribute is based on the URN form of the
1938 underlying UUID that identifies the attribute.

1939 Example:

1940 `urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1941 If the underlying representation of the attribute's name is not a UUID, then any form of URI MAY be used
1942 in the `Name` XML attribute.

1943 For purposes of human readability, there may also be a requirement for some applications to carry an
1944 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in
1945 [SAMLCore]) MAY be used for this purpose.

1946 **8.3.3.1 Attribute Name Comparison**

1947 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
1948 values are equal in the sense of [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-05.txt]. The
1949 `FriendlyName` attribute plays no role in the comparison.

1950 **8.3.4 Profile-Specific XML Attributes**

1951 No additional XML attributes are defined for use with the `<Attribute>` element.

1952 **8.3.5 SAML Attribute Values**

1953 In cases in which the attribute's value is also a UUID, the same URN syntax described above MUST be
1954 used to express the value within the `<AttributeValue>` element. The `xsi:type` XML attribute MUST
1955 be set to `xs:anyURI`.

1956 If the attribute's value is not a UUID, then there are no restrictions on the use of the `<AttributeValue>`
1957 element.

1958 **8.3.6 Example**

1959 The following is an example of a DCE Extended Registry Attribute, the "pre_auth_req" setting, which has a
1960 well-known UUID of 6c9d0ec8-dd2d-11cc-abdd-080009353559 and is integer-valued.

```
1961 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1962         Name="urn:uuid:6c9d0ec8-dd2d-11cc-abdd-080009353559"
1963         FriendlyName="pre_auth_req">
1964     <saml:AttributeValue xsi:type="xs:integer">1</saml:AttributeValue>
1965 </saml:Attribute>
```

1966 8.4 DCE PAC Attribute Profile

1967 The DCE PAC attribute profile defines the expression of DCE PAC information as SAML attribute names
1968 and values. It is used to standardize a mapping between the primary information that makes up a DCE
1969 principal's identity and a set of SAML attributes. This profile builds on the UUID attribute profile defined in
1970 Section 8.3.

1971 8.4.1 Required Information

1972 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE (this is also the target namespace
1973 assigned in the corresponding DCE PAC attribute profile schema document [SAMLDCExsd])

1974 **Contact information:** security-services-comment@lists.oasis-open.org

1975 **Description:** Given below.

1976 **Updates:** None.

1977 8.4.2 PAC Description

1978 A DCE PAC is an extensible structure that can carry arbitrary DCE registry attributes, but a core set of
1979 information is common across principals and makes up the bulk of a DCE identity:

- 1980 • The principal's DCE "realm" or "cell"
- 1981 • The principal's unique identifier
- 1982 • The principal's primary DCE local group membership
- 1983 • The principal's set of DCE local group memberships (multi-valued)
- 1984 • The principal's set of DCE foreign group memberships (multi-valued)

1985 The primary value(s) of each of these attributes is a UUID.

1986 8.4.3 SAML Attribute Naming

1987 This profile defines a mapping of specific DCE information into SAML attributes, and thus defines actual
1988 specific attribute names, rather than a naming convention.

1989 For all attributes defined by this profile, the `NameFormat` XML attribute in `<Attribute>` elements MUST
1990 have the value `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1991 For purposes of human readability, there may also be a requirement for some applications to carry an
1992 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in
1993 [SAMLCORE]) MAY be used for this purpose.

1994 See Section 8.4.6 for the specific attribute names defined by this profile.

1995 8.4.3.1 Attribute Name Comparison

1996 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
1997 values are equal in the sense of [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-05.txt]. The

1998 FriendlyName attribute plays no role in the comparison.

1999 8.4.4 Profile-Specific XML Attributes

2000 No additional XML attributes are defined for use with the <Attribute> element.

2001 8.4.5 SAML Attribute Values

2002 The primary value(s) of each of the attributes defined by this profile is a UUID. The URN syntax described
2003 in Section 8.3.5 of the UUID profile is used to represent such values.

2004 However, additional information associated with the UUID value is permitted by this profile, consisting of a
2005 friendly, human-readable string, and an additional UUID representing a DCE cell or realm. The additional
2006 information is carried in the <AttributeValue> element in FriendlyName and Realm XML attributes
2007 defined in the XML namespace urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE. Note
2008 that this is not the same as the FriendlyName XML attribute defined in [SAMLCore], although it has the
2009 same basic purpose.

2010 The following schema listing shows how the profile-specific XML attributes and complex type used in an
2011 xsi:type specification are defined [SAML DCE-xsd]:

```
2012 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
2013   xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
2014   xmlns="http://www.w3.org/2001/XMLSchema"  
2015   elementFormDefault="unqualified"  
2016   attributeFormDefault="unqualified"  
2017   blockDefault="substitution"  
2018   version="2.0">  
2019   <annotation>  
2020     <documentation>  
2021       Document identifier: saml-schema-dce-2.0  
2022       Location: http://docs.oasis-open.org/security/saml/v2.0/  
2023       Revision history:  
2024       V2.0 (March, 2005):  
2025         Custom schema for DCE attribute profile, first published in  
2026 SAML 2.0.  
2027     </documentation>  
2028   </annotation>  
2029   <complexType name="DCEValueType">  
2030     <simpleContent>  
2031       <extension base="anyURI">  
2032         <attribute ref="dce:Realm" use="optional"/>  
2033         <attribute ref="dce:FriendlyName" use="optional"/>  
2034       </extension>  
2035     </simpleContent>  
2036   </complexType>  
2037   <attribute name="Realm" type="anyURI"/>  
2038   <attribute name="FriendlyName" type="string"/>  
2039 </schema>
```

2040 8.4.6 Attribute Definitions

2041 The following are the set of SAML attributes defined by this profile. In each case, an xsi:type XML
2042 attribute MAY be included in the <AttributeValue> element, but MUST have the value
2043 **dce:DCEValueType**, where the dce prefix is arbitrary and MUST be bound to the XML namespace
2044 urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE.

2045 Note that such use of xsi:type will require validating attribute consumers to include the extension
2046 schema defined by this profile.

2047 **8.4.6.1 Realm**

2048 This single-valued attribute represents the SAML assertion subject's DCE realm or cell.

2049 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm

2050 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion
2051 subject's DCE realm/cell, with an optional profile-specific `FriendlyName` XML attribute containing the
2052 realm's string name.

2053 **8.4.6.2 Principal**

2054 This single-valued attribute represents the SAML assertion subject's DCE principal identity.

2055 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal

2056 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion
2057 subject's DCE principal identity, with an optional profile-specific `FriendlyName` XML attribute containing
2058 the principal's string name.

2059 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
2060 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
2061 8.4.6.1).

2062 **8.4.6.3 Primary Group**

2063 This single-valued attribute represents the SAML assertion subject's primary DCE group membership.

2064 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group

2065 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion
2066 subject's primary DCE group, with an optional profile-specific `FriendlyName` XML attribute containing
2067 the group's string name.

2068 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
2069 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
2070 8.4.6.1).

2071 **8.4.6.4 Groups**

2072 This multi-valued attribute represents the SAML assertion subject's DCE local group memberships.

2073 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups

2074 Each `<AttributeValue>` element contains a UUID in URN form identifying a DCE group membership
2075 of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute containing
2076 the group's string name.

2077 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
2078 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
2079 8.4.6.1).

2080 **8.4.6.5 Foreign Groups**

2081 This multi-valued attribute represents the SAML assertion subject's DCE foreign group memberships.

2082 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-groups

2083 Each <AttributeValue> element contains a UUID in URN form identifying a DCE foreign group
2084 membership of the SAML assertion subject, with an optional profile-specific FriendlyName XML attribute
2085 containing the group's string name.

2086 The profile-specific Realm XML attribute MUST be included and MUST contain a UUID in URN form
2087 identifying the DCE realm/cell of the foreign group.

2088 8.4.7 Example

2089 The following is an example of the transformation of PAC data into SAML attributes belonging to a DCE
2090 principal named "jdoe" in realm "example.com", a member of the "cubicle-dwellers" and "underpaid" local
2091 groups and an "engineers" foreign group.

```
2092 <saml:Assertion xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
2093 ...>  
2094   <saml:Issuer>...</saml:Issuer>  
2095   <saml:Subject>...</saml:Subject>  
2096   <saml:AttributeStatement>  
2097     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
2098       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm">  
2099       <saml:AttributeValue xsi:type="dce:DCEValueType"  
2100 dce:FriendlyName="example.com">  
2101         urn:uuid:003c6cc1-9ff8-10f9-990f-004005b13a2b  
2102       </saml:AttributeValue>  
2103     </saml:Attribute>  
2104     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
2105       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal">  
2106       <saml:AttributeValue xsi:type="dce:DCEValueType" dce:FriendlyName="jdoe">  
2107         urn:uuid:00305ed1-a1bd-10f9-a2d0-004005b13a2b  
2108       </saml:AttributeValue>  
2109     </saml:Attribute>  
2110     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
2111       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group">  
2112       <saml:AttributeValue xsi:type="dce:DCEValueType"  
2113         dce:FriendlyName="cubicle-dwellers">  
2114         urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b  
2115       </saml:AttributeValue>  
2116     </saml:Attribute>  
2117     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
2118       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups">  
2119       <saml:AttributeValue xsi:type="dce:DCEValueType"  
2120         dce:FriendlyName="cubicle-dwellers">  
2121         urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b  
2122       </saml:AttributeValue>  
2123       <saml:AttributeValue xsi:type="dce:DCEValueType"  
2124 dce:FriendlyName="underpaid">  
2125         urn:uuid:006a5a91-a2b7-10f9-824d-004005b13a2b  
2126       </saml:AttributeValue>  
2127     </saml:Attribute>  
2128     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
2129       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-  
2130 groups">  
2131       <saml:AttributeValue xsi:type="dce:DCEValueType"  
2132 dce:FriendlyName="engineers"  
2133         dce:Realm="urn:uuid:00583221-a35f-10f9-8b6e-004005b13a2b">  
2134         urn:uuid:00099cf1-a355-10f9-9e95-004005b13a2b  
2135       </saml:AttributeValue>  
2136     </saml:Attribute>  
2137   </saml:AttributeStatement>  
2138 </saml:Assertion>
```

2139 **8.5 XACML Attribute Profile**

2140 SAML attribute assertions may be used as input to authorization decisions made according to the OASIS
2141 eXtensible Access Control Markup Language [XACML] standard specification. Since the SAML attribute
2142 format differs from the XACML attribute format, there is a mapping that must be performed. The XACML
2143 attribute profile facilitates this mapping by standardizing naming, value syntax, and additional attribute
2144 metadata. SAML attributes generated in conformance with this profile can be mapped automatically into
2145 XACML attributes and used as input to XACML authorization decisions.

2146 **8.5.1 Required Information**

2147 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML (this is also the target namespace
2148 assigned in the corresponding XACML profile schema document [SAMLXAC-xsd])

2149 **Contact information:** security-services-comment@lists.oasis-open.org

2150 **Description:** Given below.

2151 **Updates:** None.

2152 **8.5.2 SAML Attribute Naming**

2153 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
2154 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

2155 The `Name` XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

2156 For purposes of human readability, there may also be a requirement for some applications to carry an
2157 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
2158 [SAMLCore]) MAY be used for this purpose, but is not translatable into an XACML attribute equivalent.

2159 **8.5.2.1 Attribute Name Comparison**

2160 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
2161 values are equal in a binary comparison. The `FriendlyName` attribute plays no role in the comparison.

2162 **8.5.3 Profile-Specific XML Attributes**

2163 XACML requires each attribute to carry an explicit data type. To supply this data type value, a new URI-
2164 valued XML attribute called `DataType` is defined in the XML namespace
2165 `urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML`.

2166 SAML `<Attribute>` elements conforming to this profile MUST include the namespace-qualified
2167 `DataType` attribute, or the value is presumed to be <http://www.w3.org/2001/XMLSchema#string>.

2168 While in principle any URI reference can be used as a data type, the standard values to be used are
2169 specified in Appendix A of the XACML 2.0 Specification [XACML]. If non-standard values are used, then
2170 each XACML PDP that will be consuming mapped SAML attributes with non-standard `DataType` values
2171 must be extended to support the new data types.

2172 **8.5.4 SAML Attribute Values**

2173 The syntax of the `<AttributeValue>` element's content MUST correspond to the data type expressed
2174 in the profile-specific `DataType` XML attribute appearing in the parent `<Attribute>` element. For data
2175 types corresponding to the types defined in Section 3.3 of [Schema2], the `xsi:type` XML attribute
2176 SHOULD also be used on the `<AttributeValue>` element(s).

2177 8.5.5 Profile-Specific Schema

2178 The following schema listing shows how the profile-specific `DataType` XML attribute is defined
2179 [SAMLXAC-xsd]:

```
2180 <schema
2181   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
2182   xmlns="http://www.w3.org/2001/XMLSchema"
2183   elementFormDefault="unqualified"
2184   attributeFormDefault="unqualified"
2185   blockDefault="substitution"
2186   version="2.0">
2187   <annotation>
2188     <documentation>
2189       Document identifier: saml-schema-xacml-2.0
2190       Location: http://docs.oasis-open.org/security/saml/v2.0/
2191       Revision history:
2192       V2.0 (March, 2005):
2193         Custom schema for XACML attribute profile, first published in
2194 SAML 2.0.
2195     </documentation>
2196   </annotation>
2197   <attribute name="DataType" type="anyURI"/>
2198 </schema>
```

2199 8.5.6 Example

2200 The following is an example of a mapping of the "givenName" LDAP/X.500 attribute, representing the
2201 SAML assertion subject's first name. It also illustrates that a single SAML attribute can conform to multiple
2202 attribute profiles when they are compatible with each other.

```
2203 <saml:Attribute
2204   xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
2205   xmlns:ldapprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP"
2206   xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string"
2207   [E39] ldapprof:Encoding="LDAP"
2208   NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
2209   Name="urn:oid:2.5.4.42" FriendlyName="givenName">
2210   <saml:AttributeValue xsi:type="xs:string"
2211     ldapprof:Encoding="LDAP">By-Tor</saml:AttributeValue>
2212 </saml:Attribute>
```

9 References

2213

- 2214 **[AES]** FIPS-197, Advanced Encryption Standard (AES). See <http://www.nist.gov/>.
- 2215 **[Anders]** A suggestion on how to implement SAML browser bindings without using “Artifacts”.
2216 See <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- 2217 **[ASN.1]** Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic
2218 notation, ITU-T Recommendation X.680, July 2002. See
2219 [http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-](http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.680)
2220 [X.680](http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.680).
- 2221 **[eduPerson]** eduPerson.Idif. See <http://www.educause.edu/eduperson>.
- 2222 **[LDAP]** J. Hodges et al. *Lightweight Directory Access Protocol (v3): Technical Specification*.
2223 IETF RFC 3377, September 2002. See <http://www.ietf.org/rfc/rfc3377.txt>.
- 2224 **[Mealling]** P Leach et al. *A UUID URN Namespace*. IETF Internet-Draft, December 2004. See
2225 <http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-05.txt>.
- 2226 **[MSURL]** Microsoft technical support article. See
2227 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- 2228 **[NSCookie]** Persistent Client State HTTP Cookies, Netscape documentation. See
2229 http://wp.netscape.com/newsref/std/cookie_spec.html.
- 2230 **[PAOS]** R. Aarts. *Liberty Reverse HTTP Binding for SOAP Specification* Version 1.0. Liberty
2231 Alliance Project, 2003. See <https://www.projectliberty.org/specs/liberty-paos-v1.0.pdf>.
- 2232 **[Rescorla-Sec]** E. Rescorla et al. *Guidelines for Writing RFC Text on Security Considerations*. IETF
2233 RFC 3552, July 2003. See <http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt>.
- 2234 **[RFC1738]** T. Berners-Lee et al. *Uniform Resource Locators (URL)*. IETF RFC 1738, December
2235 1994. See <http://www.ietf.org/rfc/rfc1738.txt>.
- 2236 **[RFC1750]** D. Eastlake et al. *Randomness Recommendations for Security*. IETF RFC 1750,
2237 December 1994. See <http://www.ietf.org/rfc/rfc1750.txt>.
- 2238 **[RFC1945]** T. Berners-Lee et al. *Hypertext Transfer Protocol – HTTP/1.0*. IETF RFC 1945, May
2239 1996. See <http://www.ietf.org/rfc/rfc1945.txt>.
- 2240 **[RFC2045]** N. Freed et al. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of*
2241 *Internet Message Bodies*. IETF RFC 2045, November 1996. See
2242 <http://www.ietf.org/rfc/rfc2045.txt>.
- 2243 **[RFC2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC
2244 2119, March 1997. See <http://www.ietf.org/rfc/rfc2119.txt>.
- 2245 **[RFC2246]** T. Dierks. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. See
2246 <http://www.ietf.org/rfc/rfc2246.txt>.
- 2247 **[RFC2256]** M. Wahl. *A Summary of the X.500(96) User Schema for use with LDAPv3*. IETF RFC
2248 2256, December 1997. See <http://www.ietf.org/rfc/rfc2256.txt>.
- 2249 **[RFC2279]** F. Yergeau. *UTF-8, a transformation format of ISO 10646*. IETF RFC 2279, January
2250 1998. See <http://www.ietf.org/rfc/rfc2279.txt>.
- 2251 **[RFC2616]** R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. IETF RFC 2616, June 1999.
2252 See <http://www.ietf.org/rfc/rfc2616.txt>.
- 2253 **[RFC2617]** J. Franks et al. *HTTP Authentication: Basic and Digest Access Authentication*. IETF
2254 RFC 2617, June 1999. See <http://www.ietf.org/rfc/rfc2617.txt>.
- 2255 **[RFC2798]** M. Smith. *Definition of the inetOrgPerson LDAP Object Class*. IETF RFC 2798, April
2256 2000. See <http://www.ietf.org/rfc/rfc2798.txt>.
- 2257 **[RFC2965]** D. Cristol et al. *HTTP State Management Mechanism*. IETF RFC 2965, October 2000.
2258 See <http://www.ietf.org/rfc/rfc2965.txt>.

2259	[RFC3061]	M. Mealling. <i>A URN Namespace of Object Identifiers</i> . IETF RFC 3061, February 2001. See http://www.ietf.org/rfc/rfc3061.txt .
2260		
2261	[SAMLBind]	S. Cantor et al. <i>Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os. See http://www.oasis-open.org/committees/security/ .
2262		
2263		
2264	[SAMLConform]	P. Mishra et al. <i>Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-conformance-2.0-os. See http://www.oasis-open.org/committees/security/ .
2265		
2266		
2267	[SAMLCore]	S. Cantor et al. <i>Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-core-2.0-os. See http://www.oasis-open.org/committees/security/ .
2268		
2269		
2270	[SAML DCE-xsd]	S. Cantor et al. SAML DCE PAC attribute profile schema. OASIS SSTC, March 2005. Document ID saml-schema-dce-2.0. See http://www.oasis-open.org/committees/security/ .
2271		
2272		
2273	[SAML ECP-xsd]	S. Cantor et al. SAML ECP profile schema. OASIS SSTC, March 2005. Document ID saml-schema-ecp-2.0. See http://www.oasis-open.org/committees/security/ .
2274		
2275	[SAML Gloss]	J. Hodges et al. <i>Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-glossary-2.0-os. See http://www.oasis-open.org/committees/security/ .
2276		
2277		
2278	[SAML X500-xsd]	S. Cantor et al. SAML X.500/LDAP attribute profile schema. OASIS SSTC, March 2005. Document ID saml-schema-x500-2.0. See http://www.oasis-open.org/committees/security/ .
2279		
2280		
2281	[SAML Meta]	S. Cantor et al. <i>Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-metadata-2.0-os. See http://www.oasis-open.org/committees/security/ .
2282		
2283		
2284	[SAML Reqs]	Darren Platt et al. <i>OASIS Security Services Use Cases and Requirements</i> . OASIS SSTC, May 2001. Document ID draft-sstc-saml-reqs-01. See http://www.oasis-open.org/committees/security/ .
2285		
2286		
2287	[SAML Sec]	F. Hirsch et al. <i>Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-sec-consider-2.0-os. See http://www.oasis-open.org/committees/security/ .
2288		
2289		
2290	[SAML Web]	OASIS Security Services Technical Committee website, http://www.oasis-open.org/committees/security/ .
2291		
2292	[SAML XAC-xsd]	S. Cantor et al. SAML XACML attribute profile schema. OASIS SSTC, March 2005. Document ID saml-schema-xacml-2.0. See http://www.oasis-open.org/committees/security/ .
2293		
2294		
2295	[Schema1]	H. S. Thompson et al. <i>XML Schema Part 1: Structures</i> . World Wide Web Consortium Recommendation, May 2001. http://www.w3.org/TR/xmlschema-1/ . Note that this specification normatively references [Schema2], listed below.
2296		
2297		
2298	[Schema2]	Paul V. Biron, Ashok Malhotra. <i>XML Schema Part 2: Datatypes</i> . World Wide Web Consortium Recommendation, May 2001. See http://www.w3.org/TR/xmlschema-2/ .
2299		
2300	[SESSION]	RL 'Bob' Morgan. <i>Support of target web server sessions in Shibboleth</i> . Shibboleth, May 2001. See http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt .
2301		
2302		
2303	[ShibMarlena]	Marlena Erdos et al. <i>Shibboleth Architecture DRAFT v05</i> . Shibboleth, May 2002. See http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html .
2304		
2305	[SOAP1.1]	D. Box et al. <i>Simple Object Access Protocol (SOAP) 1.1</i> . World Wide Web Consortium Note, May 2000. See http://www.w3.org/TR/SOAP .
2306		
2307	[SSL3]	A. Frier et al. <i>The SSL 3.0 Protocol</i> . Netscape Communications Corp, November 1996.
2308	[WEBSSO]	RL 'Bob' Morgan. <i>Interactions between Shibboleth and local-site web sign-on services</i> . Shibboleth, April 2001. See http://middleware.internet2.edu/shibboleth/docs/draft-
2309		

2310		morgan-shibboleth-websso-00.txt .
2311	[X.500]	Information technology - Open Systems Interconnection - The Directory: Overview of
2312		concepts, models and services. ITU-T Recommendation X.500, February 2001. See
2313		http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-
2314		X.500 .
2315	[XMLEnc]	D. Eastlake et al. <i>XML Encryption Syntax and Processing</i> . World Wide Web
2316		Consortium Recommendation, December 2002. See
2317		http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/ .
2318	[XMLSig]	D. Eastlake et al. <i>XML-Signature Syntax and Processing</i> . World Wide Web
2319		Consortium Recommendation, February 2002. See http://www.w3.org/TR/xmlsig-
2320		core/ .
2321	[XACML]	T. Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Versions
2322		1.0, 1.1, and 2.0. Available on the OASIS XACML TC web page at http://www.oasis-
2323		open.org/committees/tc_home.php?wg_abbrev=xacml .

Appendix A. Acknowledgments

2325 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
2326 Committee, whose voting members at the time of publication were:

- 2327 • Conor Cahill, AOL
- 2328 • John Hughes, Atos Origin
- 2329 • Hal Lockhart, BEA Systems
- 2330 • Mike Beach, Boeing
- 2331 • Rebekah Metz, Booz Allen Hamilton
- 2332 • Rick Randall, Booz Allen Hamilton
- 2333 • Ronald Jacobson, Computer Associates
- 2334 • Gavenraj Sodhi, Computer Associates
- 2335 • Thomas Wisniewski, Entrust
- 2336 • Carolina Canales-Valenzuela, Ericsson
- 2337 • Dana Kaufman, Forum Systems
- 2338 • Irving Reid, Hewlett-Packard
- 2339 • Guy Denton, IBM
- 2340 • Heather Hinton, IBM
- 2341 • Maryann Hondo, IBM
- 2342 • Michael McIntosh, IBM
- 2343 • Anthony Nadalin, IBM
- 2344 • Nick Ragouzis, Individual
- 2345 • Scott Cantor, Internet2
- 2346 • Bob Morgan, Internet2
- 2347 • Peter Davis, Neustar
- 2348 • Jeff Hodges, Neustar
- 2349 • Frederick Hirsch, Nokia
- 2350 • Senthil Sengodan, Nokia
- 2351 • Abbie Barbir, Nortel Networks
- 2352 • Scott Kiestler, Novell
- 2353 • Cameron Morris, Novell
- 2354 • Paul Madsen, NTT
- 2355 • Steve Anderson, OpenNetwork
- 2356 • Ari Kermaier, Oracle
- 2357 • Vamsi Motukuru, Oracle
- 2358 • Darren Platt, Ping Identity
- 2359 • Prateek Mishra, Principal Identity
- 2360 • Jim Lien, RSA Security
- 2361 • John Linn, RSA Security
- 2362 • Rob Philpott, RSA Security
- 2363 • Dipak Chopra, SAP
- 2364 • Jahan Moreh, Sigaba
- 2365 • Bhavna Bhatnagar, Sun Microsystems
- 2366 • Eve Maler, Sun Microsystems

- 2367 • Ronald Monzillo, Sun Microsystems
- 2368 • Emily Xu, Sun Microsystems
- 2369 • Greg Whitehead, Trustgenix

2370 The editors also would like to acknowledge the following former SSTC members for their contributions to
2371 this or previous versions of the OASIS Security Assertions Markup Language Standard:

- 2372 • Stephen Farrell, Baltimore Technologies
- 2373 • David Orchard, BEA Systems
- 2374 • Krishna Sankar, Cisco Systems
- 2375 • Zahid Ahmed, CommerceOne
- 2376 • Tim Alsop, CyberSafe Limited
- 2377 • Carlisle Adams, Entrust
- 2378 • Tim Moses, Entrust
- 2379 • Nigel Edwards, Hewlett-Packard
- 2380 • Joe Pato, Hewlett-Packard
- 2381 • Bob Blakley, IBM
- 2382 • Marlena Erdos, IBM
- 2383 • Marc Chanliau, Netegrity
- 2384 • Chris McLaren, Netegrity
- 2385 • Lynne Rosenthal, NIST
- 2386 • Mark Skall, NIST
- 2387 • Charles Knouse, Oblix
- 2388 • Simon Godik, Overxeer
- 2389 • Charles Norwood, SAIC
- 2390 • Evan Prodromou, Securant
- 2391 • Robert Griffin, RSA Security (former editor)
- 2392 • Sai Allarvarpu, Sun Microsystems
- 2393 • Gary Ellison, Sun Microsystems
- 2394 • Chris Ferris, Sun Microsystems
- 2395 • Mike Myers, Traceroute Security
- 2396 • Phillip Hallam-Baker, VeriSign (former editor)
- 2397 • James Vanderbeek, Vodafone
- 2398 • Mark O'Neill, Vordel
- 2399 • Tony Palmer, Vordel

2400 Finally, the editors wish to acknowledge the following people for their contributions of material used as
2401 input to the OASIS Security Assertions Markup Language specifications:

- 2402 • Thomas Gross, IBM
- 2403 • Birgit Pfitzmann, IBM

2404 The editors also would like to gratefully acknowledge Jahan Moreh of Sigaba, who during his tenure on
2405 the SSTC was the primary editor of the errata working document and who made major substantive
2406 contributions to all of the errata materials.

2407 **Appendix B. Notices**

2408 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
2409 might be claimed to pertain to the implementation or use of the technology described in this document or
2410 the extent to which any license under such rights might or might not be available; neither does it represent
2411 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
2412 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
2413 available for publication and any assurances of licenses to be made available, or the result of an attempt
2414 made to obtain a general license or permission for the use of such proprietary rights by implementors or
2415 users of this specification, can be obtained from the OASIS Executive Director.

2416 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
2417 other proprietary rights which may cover technology that may be required to implement this specification.
2418 Please address the information to the OASIS Executive Director.

2419 **Copyright © OASIS Open 2005. All Rights Reserved.**

2420 This document and translations of it may be copied and furnished to others, and derivative works that
2421 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
2422 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
2423 this paragraph are included on all such copies and derivative works. However, this document itself may
2424 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
2425 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
2426 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
2427 into languages other than English.

2428 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
2429 or assigns.

2430 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2431 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
2432 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
2433 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.