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

Working Draft 06, 12 FebruaryDecember 20096

Document identifier:

sstc-saml-profiles-errata-2.0-wd-065

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
Eve Maler, Sun Microsystems (errata editor)

Contributors to the Errata:

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

SAML V2.0 Contributors:

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

77 **Abstract:**

78 The SAML V2.0 Profiles specification defines profiles for the use of SAML assertions and request-
79 response messages in communications protocols and frameworks, as well as profiles for SAML
80 attribute value syntax and naming conventions. This document, known as an “errata composite”,
81 combines corrections to reported errata with the original specification text. By design, the
82 corrections are limited to clarifications of ambiguous or conflicting specification text. This
83 document shows deletions from the original specification as struck-through text, and additions as
84 colored underlined text. The “[Enn]” designations embedded in the text refer to particular errata
85 and their dispositions.

86 **Status:**

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

95 This document includes corrections for errata E12, E14, E17, E18, E20, E22, E26, E27, E32, E35,
96 E38, E39, E40, E47, E48, E51, E52, E53, E54, E55, E56, E58, ~~and E63, E70, E71, and E74.~~

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

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

Table of Contents

104

105	1 Introduction.....	7
106	1.1 Profile Concepts.....	7
107	1.2 Notation.....	7
108	2 Specification of Additional Profiles.....	10
109	2.1 Guidelines for Specifying Profiles.....	10
110	2.2 Guidelines for Specifying Attribute Profiles.....	10
111	3 Confirmation Method Identifiers.....	12
112	3.1 Holder of Key.....	12
113	3.2 Sender Vouches.....	13
114	3.3 Bearer.....	13
115	4 SSO Profiles of SAML.....	14
116	4.1 Web Browser SSO Profile.....	14
117	4.1.1 Required Information.....	14
118	4.1.2 Profile Overview.....	14
119	4.1.3 Profile Description.....	16
120	4.1.3.1 HTTP Request to Service Provider.....	16
121	4.1.3.2 Service Provider Determines Identity Provider.....	16
122	4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider.....	16
123	4.1.3.4 Identity Provider Identifies Principal.....	17
124	4.1.3.5 Identity Provider Issues <Response> to Service Provider.....	17
125	4.1.3.6 Service Provider Grants or Denies Access to User Agent.....	17
126	4.1.4 Use of Authentication Request Protocol.....	18
127	4.1.4.1 <AuthnRequest> Usage.....	18
128	4.1.4.2 <Response> Usage.....	18
129	4.1.4.3 <Response> Message Processing Rules.....	20
130	4.1.4.4 Artifact-Specific <Response> Message Processing Rules.....	20
131	4.1.4.5 POST-Specific Processing Rules.....	20
132	4.1.5 Unsolicited Responses.....	20
133	4.1.6 Use of Metadata.....	21
134	4.2 Enhanced Client or Proxy (ECP) Profile.....	21
135	4.2.1 Required Information.....	22
136	4.2.2 Profile Overview.....	22
137	4.2.3 Profile Description.....	25
138	4.2.3.1 ECP issues HTTP Request to Service Provider.....	25
139	4.2.3.2 Service Provider Issues <AuthnRequest> to ECP.....	26
140	4.2.3.3 ECP Determines Identity Provider.....	26
141	4.2.3.4 ECP issues <AuthnRequest> to Identity Provider.....	26
142	4.2.3.5 Identity Provider Identifies Principal.....	26
143	4.2.3.6 Identity Provider issues <Response> to ECP, targeted at service provider.....	27
144	4.2.3.7 ECP Conveys <Response> Message to Service Provider.....	27
145	4.2.3.8 Service Provider Grants or Denies Access to Principal.....	27
146	4.2.4 ECP Profile Schema Usage.....	27
147	4.2.4.1 PAOS Request Header Block: SP to ECP.....	28
148	4.2.4.2 ECP Request Header Block: SP to ECP.....	29
149	4.2.4.3 ECP RelayState Header Block: SP to ECP.....	29

150	4.2.4.4 ECP Response Header Block: IdP to ECP.....	31
151	4.2.4.5 PAOS Response Header Block: ECP to SP.....	31
152	4.2.5 Security Considerations.....	32
153	4.2.6 [E20]Use of Metadata.....	32
154	4.3 Identity Provider Discovery Profile.....	32
155	4.3.1 [E32]Required Information.....	33
156	4.3.2 Common Domain Cookie.....	33
157	4.3.3 Setting the Common Domain Cookie.....	33
158	4.3.4 Obtaining the Common Domain Cookie.....	33
159	4.4 Single Logout Profile.....	34
160	4.4.1 Required Information.....	34
161	4.4.2 Profile Overview.....	34
162	4.4.3 Profile Description.....	36
163	4.4.3.1 <LogoutRequest> Issued by Session Participant to Identity Provider.....	36
164	4.4.3.2 Identity Provider Determines Session Participants.....	37
165	4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session Participant/Authority.....	37
166	4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity Provider.....	37
167	4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant.....	38
168	4.4.4 Use of Single Logout Protocol.....	38
169	4.4.4.1 <LogoutRequest> Usage.....	38
170	4.4.4.2 <LogoutResponse> Usage.....	38
171	4.4.5 Use of Metadata.....	39
172	4.5 Name Identifier Management Profile.....	39
173	4.5.1 Required Information.....	39
174	4.5.2 Profile Overview.....	39
175	4.5.3 Profile Description.....	40
176	4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider.....	40
177	4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service Provider.....	41
178	4.5.4 Use of Name Identifier Management Protocol.....	42
179	4.5.4.1 <ManageNameIDRequest> Usage.....	42
180	4.5.4.2 <ManageNameIDResponse> Usage.....	42
181	4.5.5 Use of Metadata.....	42
182	5 Artifact Resolution Profile.....	43
183	5.1 Required Information.....	43
184	5.2 Profile Overview.....	43
185	5.3 Profile Description.....	44
186	5.3.1 <ArtifactResolve> issued by Requesting Entity.....	44
187	5.3.2 <ArtifactResponse> issued by Responding Entity.....	44
188	5.4 Use of Artifact Resolution Protocol.....	44
189	5.4.1 <ArtifactResolve> Usage.....	44
190	5.4.2 <ArtifactResponse> Usage.....	45
191	5.5 Use of Metadata.....	45
192	6 Assertion Query/Request Profile.....	46
193	6.1 Required Information.....	46
194	6.2 Profile Overview.....	46
195	6.3 Profile Description.....	47

196	6.3.1 Query/Request issued by SAML Requester.....	47
197	6.3.2 <Response> issued by SAML Authority.....	47
198	6.4 Use of Query/Request Protocol.....	47
199	6.4.1 Query/Request Usage.....	47
200	6.4.2 <Response> Usage.....	47
201	6.5 Use of Metadata.....	48
202	7 Name Identifier Mapping Profile.....	49
203	7.1 Required Information.....	49
204	7.2 Profile Overview.....	49
205	7.3 Profile Description.....	50
206	7.3.1 <NameIDMappingRequest> issued by Requesting Entity.....	50
207	7.3.2 <NameIDMappingResponse> issued by Identity Provider.....	50
208	7.4 Use of Name Identifier Mapping Protocol.....	50
209	7.4.1 <NameIDMappingRequest> Usage.....	50
210	7.4.2 <NameIDMappingResponse> Usage.....	50
211	7.4.2.1 Limiting Use of Mapped Identifier.....	51
212	7.5 Use of Metadata.....	51
213	8 SAML Attribute Profiles.....	52
214	8.1 Basic Attribute Profile.....	52
215	8.1.1 Required Information.....	52
216	8.1.2 SAML Attribute Naming.....	52
217	8.1.2.1 Attribute Name Comparison.....	52
218	8.1.3 Profile-Specific XML Attributes.....	52
219	8.1.4 SAML Attribute Values.....	52
220	8.1.5 Example.....	52
221	8.2 X.500/LDAP Attribute Profile [E53] – Deprecated.....	52
222	8.2.1 Required Information.....	53
223	8.2.2 SAML Attribute Naming.....	53
224	8.2.2.1 Attribute Name Comparison.....	53
225	8.2.3 Profile-Specific XML Attributes.....	53
226	8.2.4 SAML Attribute Values.....	54
227	8.2.5 Profile-Specific Schema.....	55
228	8.2.6 Example.....	55
229	8.3 UUID Attribute Profile.....	55
230	8.3.1 Required Information.....	55
231	8.3.2 UUID and GUID Background.....	55
232	8.3.3 SAML Attribute Naming.....	56
233	8.3.3.1 Attribute Name Comparison.....	56
234	8.3.4 Profile-Specific XML Attributes.....	56
235	8.3.5 SAML Attribute Values.....	56
236	8.3.6 Example.....	56
237	8.4 DCE PAC Attribute Profile.....	57
238	8.4.1 Required Information.....	57
239	8.4.2 PAC Description.....	57

240	8.4.3 SAML Attribute Naming.....	57
241	8.4.3.1 Attribute Name Comparison.....	57
242	8.4.4 Profile-Specific XML Attributes.....	58
243	8.4.5 SAML Attribute Values.....	58
244	8.4.6 Attribute Definitions.....	58
245	8.4.6.1 Realm.....	59
246	8.4.6.2 Principal.....	59
247	8.4.6.3 Primary Group.....	59
248	8.4.6.4 Groups.....	59
249	8.4.6.5 Foreign Groups.....	59
250	8.4.7 Example.....	60
251	8.5 XACML Attribute Profile.....	61
252	8.5.1 Required Information.....	61
253	8.5.2 SAML Attribute Naming.....	61
254	8.5.2.1 Attribute Name Comparison.....	61
255	8.5.3 Profile-Specific XML Attributes.....	61
256	8.5.4 SAML Attribute Values.....	61
257	8.5.5 Profile-Specific Schema.....	62
258	8.5.6 Example.....	62
259	9 References.....	63
260	Appendix A. Acknowledgments.....	66
261	Appendix B. Notices.....	68

262 1 Introduction

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

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

270 1.1 Profile Concepts

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

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

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

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

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

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

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

296 1.2 Notation

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

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

304 described in IETF RFC 2119 [RFC2119].

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

306 Example code listings appear like this.

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

308 Conventional XML namespace prefixes are used throughout this specification to stand for their respective
309 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].
[E71]xs:	http://www.w3.org/2001/XMLSchema	This namespace is defined in the W3C XML Schema specification [Schema1]. In schema listings, this is the default namespace and no prefix is shown. For clarity, the prefix is generally shown in specification text when XML Schema-related constructs are

Prefix	XML Namespace	Comments
		mentioned.
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.

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

2 Specification of Additional Profiles

313

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

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

2.1 Guidelines for Specifying Profiles

322

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

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

2.2 Guidelines for Specifying Attribute Profiles

344

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

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

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

3 Confirmation Method Identifiers

357

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

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

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

3.1 Holder of Key

372

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

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

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

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

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

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

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

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

402 3.2 Sender Vouches

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

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

408 3.3 Bearer

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

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

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

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

418 **Example:** The bearer of the assertion can confirm itself as the subject, provided the assertion is delivered
419 in a message sent to "<https://www.serviceprovider.com/saml/consumer>" before 1:37 PM GMT on March
420 19th, 2004, in response to a request with ID "_1234567890".

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

427 4 SSO Profiles of SAML

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

- 429 • A web browser-based profile of the Authentication Request protocol in [SAMLCore] is defined to
430 support web single sign-on, supporting Scenario 1-1 of the original SAML requirements document
431 Error: Reference source not found.
- 432 • An additional web SSO profile is defined to support enhanced clients.
- 433 • A profile of the Single Logout and Name Identifier Management protocols in [SAMLCore] is defined
434 over both front-channel (browser) and back-channel bindings.
- 435 • An additional profile is defined for identity provider discovery using cookies.

436 4.1 Web Browser SSO Profile

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

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

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

448 4.1.1 Required Information

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

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

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

453 **Description:** Given below.

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

455 4.1.2 Profile Overview

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

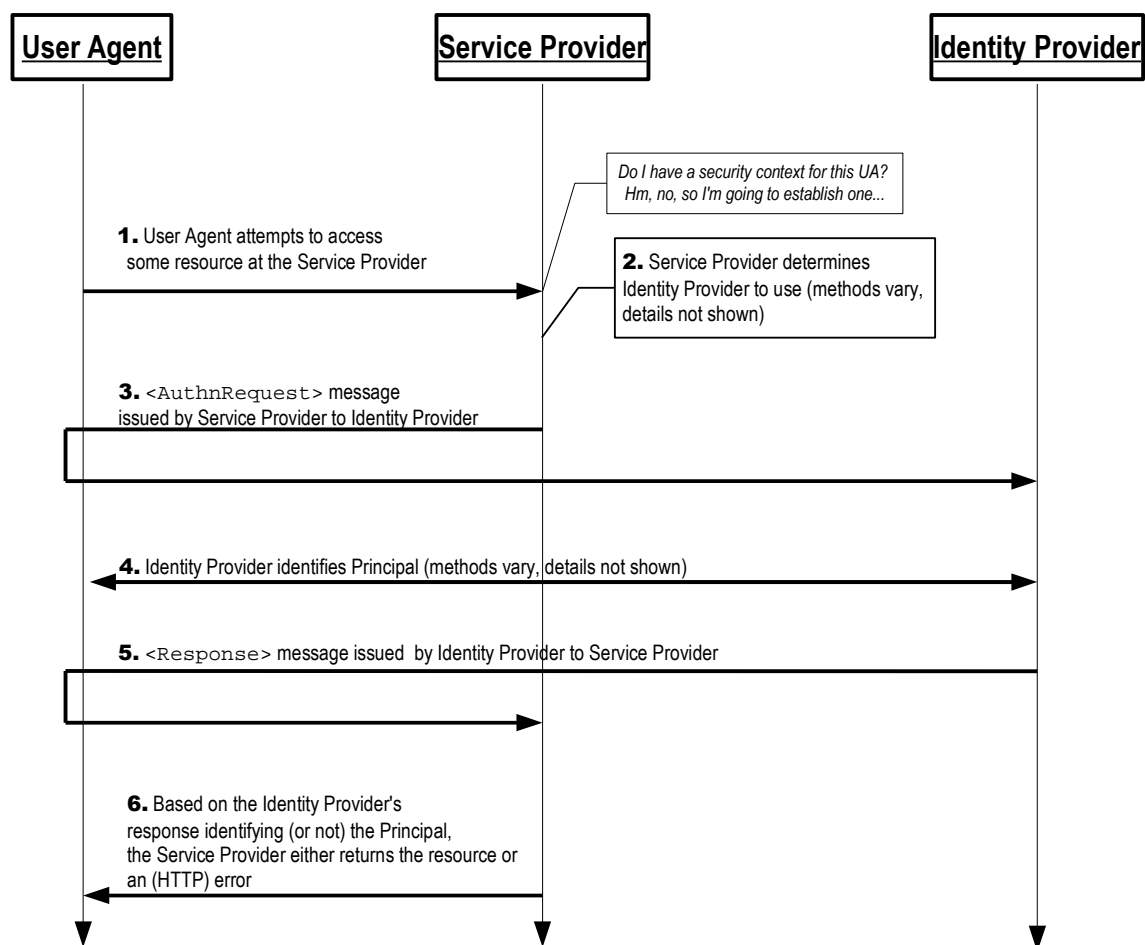


Figure 1

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

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

462 **2. Service Provider Determines Identity Provider**

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

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

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

471 **4. Identity Provider identifies Principal**

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

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

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

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

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

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

487 **4.1.3 Profile Description**

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

490 **Single Sign-On Service**

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

493 **Assertion Consumer Service**

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

496 **4.1.3.1 HTTP Request to Service Provider**

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

503 **4.1.3.2 Service Provider Determines Identity Provider**

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

510 **4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider**

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

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

523 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or TLS
524 1.0 [RFC2246] to maintain confidentiality and message integrity. The `<AuthnRequest>` message MAY
525 be signed, if authentication of the request issuer is required. The HTTP Artifact binding, if used, also
526 provides for an alternate means of authenticating the request issuer when the artifact is dereferenced.

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

530 **4.1.3.4 Identity Provider Identifies Principal**

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

538 **4.1.3.5 Identity Provider Issues `<Response>` to Service Provider**

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

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

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

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

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

558 **4.1.3.6 Service Provider Grants or Denies Access to User Agent**

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

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

564 4.1.4 Use of Authentication Request Protocol

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

569 4.1.4.1 <AuthnRequest> Usage

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

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

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

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

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

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

593 4.1.4.2 <Response> Usage

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

- 597 • ~~[E17]The <Issuer> element MAY be omitted, but if present If the <Response> message is signed or~~
598 ~~if an enclosed assertion is encrypted, then the <Issuer> element MUST be present. Otherwise it~~
599 ~~MAY be omitted. If present~~ it MUST contain the unique identifier of the issuing identity provider; the
600 Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
601 format:entity.
- 602 • It MUST contain at least one <Assertion>. Each assertion's <Issuer> element MUST contain the
603 unique identifier of the [\[E26\]issuingresponding](#) identity provider; the Format attribute MUST be omitted
604 or have a value of urn:oasis:names:tc:SAML:2.0:nameid-format:entity. [Note that this](#)

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

654 4.1.4.3 <Response> Message Processing Rules

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

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

677 4.1.4.4 Artifact-Specific <Response> Message Processing Rules

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

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

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

684 4.1.4.5 POST-Specific Processing Rules

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

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

691 4.1.5 Unsolicited Responses

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

694 An unsolicited `<Response>` MUST NOT contain an `InResponseTo` attribute, nor should any bearer

695 <SubjectConfirmationData> elements contain one. If metadata as specified in [SAMLMeta] is used,
696 the <Response> or artifact SHOULD be delivered to the <md:AssertionConsumerService> endpoint
697 of the service provider designated as the default.

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

703 **4.1.6 Use of Metadata**

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

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

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

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

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

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

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

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

738 4.2 Enhanced Client or Proxy (ECP) Profile

739 An *enhanced client or proxy* (ECP) is a system entity that knows how to contact an appropriate identity
740 provider, possibly in a context-dependent fashion, and also supports the Reverse SOAP (PAOS) binding
741 [SAMLBind].

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

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

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

753 4.2.1 Required Information

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

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

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

759 **Description:** Given below.

760 **Updates:** None.

761 4.2.2 Profile Overview

762 As introduced above, the ECP profile specifies interactions between enhanced clients or proxies and
763 service providers and identity providers. It is a specific application of the SSO profile described in Section
764 4.1. If not otherwise specified by this profile, and if not specific to the use of browser-based bindings, the
765 rules specified in Section 4.1 MUST be observed.

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

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

769 This allows a service provider to make an authentication request to the ECP without the need to know
770 or discover the appropriate identity provider (effectively bypassing step 2 of the SSO profile in Section
771 4.1).

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

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

779 An *enhanced client* may be a browser or some other user agent that supports the functionality described

780 in this profile. An *enhanced proxy* is an HTTP proxy (for example a WAP gateway) that emulates an
781 enhanced client. Unless stated otherwise, all statements referring to enhanced clients are to be
782 understood as statements about both enhanced clients as well as enhanced client proxies.

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

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

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

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

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

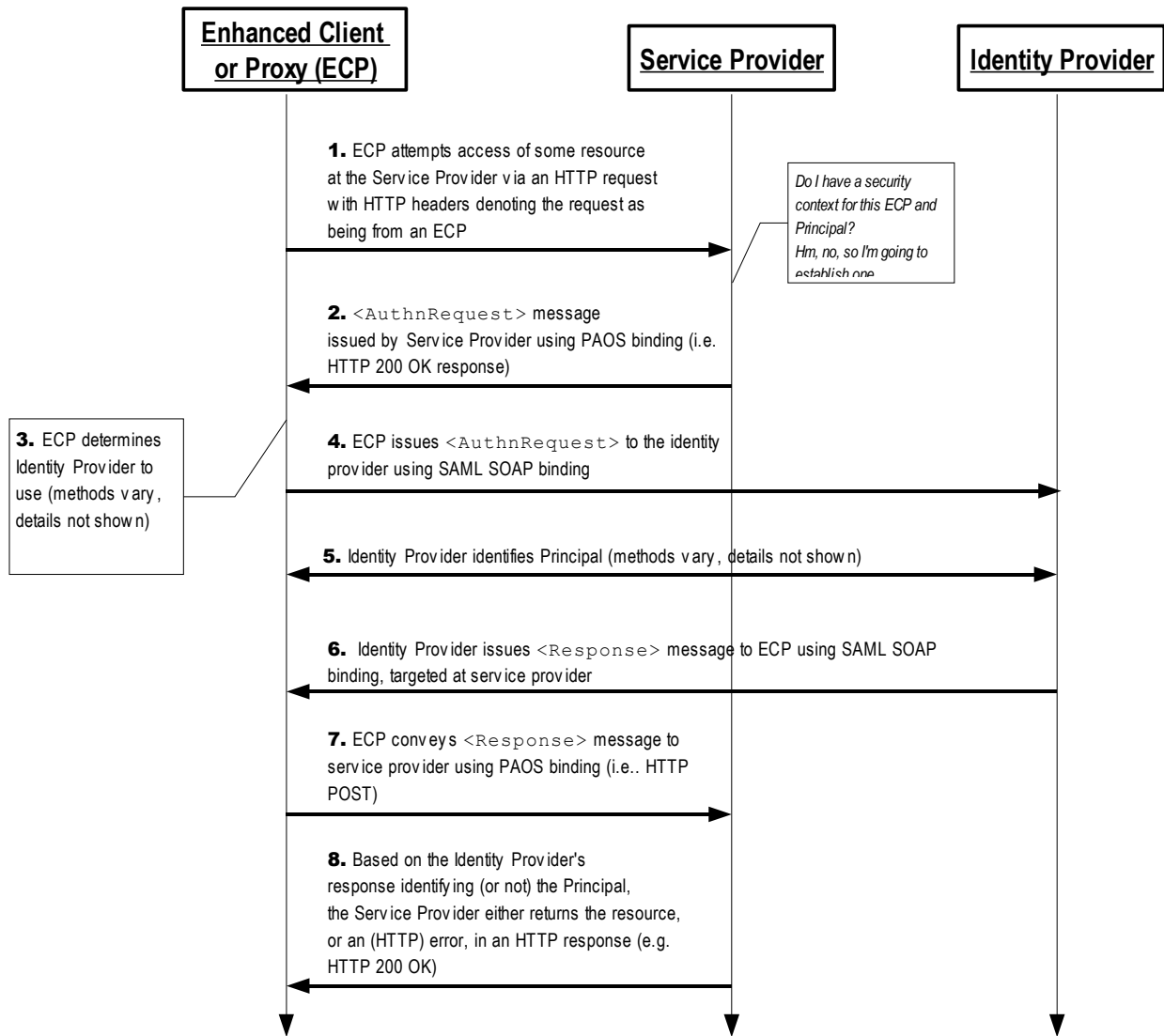


Figure 2

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

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

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

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

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

809 3. ECP Determines Identity Provider

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

814 4. ECP conveys <AuthnRequest> to Identity Provider

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

819 5. Identity Provider identifies Principal

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

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

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

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

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

830 8. Service Provider grants or denies access to Principal

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

834 4.2.3 Profile Description

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

836 4.2.3.1 ECP issues HTTP Request to Service Provider

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

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

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

```
847 GET /index HTTP/1.1  
848 Host: identity-service.example.com  
849 Accept: text/html; application/vnd.paos+xml
```

850 | PAOS: ver="urn:liberty:paos:2003-08" ;
851 | urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"

852 **4.2.3.2 Service Provider Issues <AuthnRequest> to ECP**

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

857 The SOAP envelope MUST contain:

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

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

873 **4.2.3.3 ECP Determines Identity Provider**

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

875 **4.2.3.4 ECP issues <AuthnRequest> to Identity Provider**

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

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

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

887 **4.2.3.5 Identity Provider Identifies Principal**

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

893 requirements included in the <AuthnRequest> in the form of the <RequestedAuthnContext>
894 element.

895 **4.2.3.6 Identity Provider issues <Response> to ECP, targeted at service provider**

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

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

903 **4.2.3.7 ECP Conveys <Response> Message to Service Provider**

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

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

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

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

921 **4.2.3.8 Service Provider Grants or Denies Access to Principal**

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

927 **4.2.4 ECP Profile Schema Usage**

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

```
930 <schema  
931   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"  
932   xmlns="http://www.w3.org/2001/XMLSchema"  
933   xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"  
934   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
935   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
936   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
```

```

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

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

967     <element name="Response" type="ecp:ResponseType"/>
968     <complexType name="ResponseType">
969       <attribute ref="S:mustUnderstand" use="required"/>
970       <attribute ref="S:actor" use="required"/>
971       <attribute name="AssertionConsumerServiceURL" type="anyURI"
972 use="required"/>
973     </complexType>

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

```

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

987 **4.2.4.1 PAOS Request Header Block: SP to ECP**

988 The PAOS Request header block signals the use of PAOS processing and includes the following
989 attributes:

990 responseConsumerURL [Required]

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

996 | [relative URL](#).

997 service [Required]

998 Indicates that the PAOS service being used is this SAML authentication profile. The value MUST be
999 urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp.

1000 SOAP-ENV:mustUnderstand [Required]

1001 The value MUST be 1 (true). A SOAP fault MUST be generated if the PAOS header block is not
1002 understood.

1003 SOAP-ENV:actor [Required]

1004 The value MUST be <http://schemas.xmlsoap.org/soap/actor/next>.

1005 messageID [Optional]

1006 Allows optional response correlation. It MAY be used in this profile, but is NOT required, since this
1007 functionality is provided by the SAML protocol layer, via the ID attribute in the <AuthnRequest> and
1008 the InResponseTo attribute in the <Response>.

1009 The PAOS Request SOAP header block has no element content.

1010 **4.2.4.2 ECP Request Header Block: SP to ECP**

1011 The ECP Request SOAP header block is used to convey information needed by the ECP to process the
1012 authentication request. It is mandatory and its presence signals the use of this profile. It contains the
1013 following elements and attributes:

1014 SOAP-ENV:mustUnderstand [Required]

1015 The value MUST be 1 (true). A SOAP fault MUST be generated if the ECP header block is not
1016 understood.

1017 SOAP-ENV:actor [Required]

1018 The value MUST be <http://schemas.xmlsoap.org/soap/actor/next>.

1019 ProviderName [Optional]

1020 A human-readable name for the requesting service provider.

1021 IsPassive [Optional]

1022 A boolean value. If true, the identity provider and the client itself MUST NOT take control of the user
1023 interface from the request issuer and interact with the principal in a noticeable fashion. If a value is not
1024 provided, the default is true.

1025 <saml:Issuer> [Required]

1026 This element MUST contain the unique identifier of the requesting service provider; the Format
1027 attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1028 format:entity.

1029 <samlp:IDPList> [Optional]

1030 Optional list of identity providers that the service provider recognizes and from which the ECP may
1031 choose to service the request. See [SAMLCore] for details on the content of this element.

1032 **4.2.4.3 ECP RelayState Header Block: SP to ECP**

1033 The ECP RelayState SOAP header block is used to convey state information from the service provider
1034 that it will need later when processing the response from the ECP. It is optional, but if used, the ECP
1035 MUST include an identical header block in the response in step [\[E27\]57](#). It contains the following

1036 attributes:

1037 SOAP-ENV:mustUnderstand [Required]

1038 The value MUST be 1 (true). A SOAP fault MUST be generated if the header block is not understood.

1039 SOAP-ENV:actor [Required]

1040 The value MUST be `http://schemas.xmlsoap.org/soap/actor/next`.

1041 The content of the header block element is a string containing state information created by the requester.
1042 If provided, the ECP MUST include the same value in a RelayState header block when responding to the
1043 service provider in step 5. The string value MUST NOT exceed 80 bytes in length and SHOULD be
1044 integrity protected by the requester independent of any other protections that may or may not exist during
1045 message transmission.

1046 The following is an example of the SOAP authentication request from the service provider to the ECP:

```
1047 <SOAP-ENV:Envelope
1048     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1049     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1050     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
1051   <SOAP-ENV:Header>
1052     <paos:Request xmlns:paos="urn:liberty:paos:2003-08"
1053       responseConsumerURL="\[E35\]http://identity-
1054       service.example.com/abchttps://ServiceProvider.example.com/ecp\_assertion\_consumer"
1055     messageID="6c3a4f8b9c2d" SOAP-
1056     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next" SOAP-
1057     ENV:mustUnderstand="1"
1058     service="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp">
1059     </paos:Request>
1060     <ecp:Request xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
1061       SOAP-ENV:mustUnderstand="1" SOAP-
1062       ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
1063       ProviderName="Service Provider X" IsPassive="0">
1064       <saml:Issuer>https://ServiceProvider.example.com</saml:Issuer>
1065       <samlp:IDPList>
1066         <samlp:IDPEntry ProviderID="https://IdentityProvider.example.com"
1067           Name="Identity Provider X"
1068           Loc="https://IdentityProvider.example.com/saml2/sso"
1069         </samlp:IDPEntry>
1070         <samlp:GetComplete>
1071           https://ServiceProvider.example.com/idplist?id=604bel36-fe91-441e-afb8
1072         </samlp:GetComplete>
1073       </samlp:IDPList>
1074     </ecp:Request>
1075     <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
1076       SOAP-ENV:mustUnderstand="1" SOAP-
1077       ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
1078       ...
1079     </ecp:RelayState>
1080   </SOAP-ENV:Header>
1081   <SOAP-ENV:Body>
1082     <samlp:AuthnRequest> ... </samlp:AuthnRequest>
1083   </SOAP-ENV:Body>
1084 </SOAP-ENV:Envelope>
```

1086 As noted above, the PAOS and ECP header blocks are removed from the SOAP message by the ECP
1087 before the authentication request is forwarded to the identity provider. An example authentication request
1088 from the ECP to the identity provider is as follows:

```
1089 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
1090   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1091   <SOAP-ENV:Body>
1092     <samlp:AuthnRequest> ... </samlp:AuthnRequest>
1093   </SOAP-ENV:Body>
```

1094 </SOAP-ENV:Envelope>

1095 **4.2.4.4 ECP Response Header Block: IdP to ECP**

1096 The ECP response SOAP header block **MUST** be used on the response from the identity provider to the
1097 ECP. It contains the following attributes:

1098 SOAP-ENV:mustUnderstand [Required]

1099 The value **MUST** be 1 (true). A SOAP fault **MUST** be generated if the ECP header block is not
1100 understood.

1101 SOAP-ENV:actor [Required]

1102 The value **MUST** be `http://schemas.xmlsoap.org/soap/actor/next`.

1103 AssertionConsumerServiceURL [Required]

1104 Set by the identity provider based on the <AuthnRequest> message or the service provider's
1105 metadata obtained by the identity provider.

1106 The ECP **MUST** confirm that this value corresponds to the value the ECP obtained in the
1107 responseConsumerURL in the PAOS Request SOAP header block it received from the service
1108 provider. Since the responseConsumerURL **MAY** be relative and the
1109 AssertionConsumerServiceURL is absolute, some processing/normalization may be required.

1110 This mechanism is used for security purposes to confirm the correct response destination. If the
1111 values do not match, then the ECP **MUST** generate a SOAP fault response to the service provider
1112 and **MUST NOT** return the SAML response.

1113 The ECP Response SOAP header has no element content.

1114 Following is an example of an IdP-to-ECP response.

```
1115 <SOAP-ENV:Envelope  
1116     xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"  
1117     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
1118     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">  
1119   <SOAP-ENV:Header>  
1120     <ecp:Response SOAP-ENV:mustUnderstand="1" SOAP-  
1121     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"  
1122     AssertionConsumerServiceURL="https://ServiceProvider.example.com/ecp_assertion  
1123     _consumer"/>  
1124   </SOAP-ENV:Header>  
1125   <SOAP-ENV:Body>  
1126     <samlp:Response> ... </samlp:Response>  
1127   </SOAP-ENV:Body>  
1128 </SOAP-ENV:Envelope>
```

1129 **4.2.4.5 PAOS Response Header Block: ECP to SP**

1130 The PAOS Response header block includes the following attributes:

1131 SOAP-ENV:mustUnderstand [Required]

1132 The value **MUST** be 1 (true). A SOAP fault **MUST** be generated if the PAOS header block is not
1133 understood.

1134 SOAP-ENV:actor [Required]

1135 The value **MUST** be `http://schemas.xmlsoap.org/soap/actor/next`.

1136 refToMessageID [Optional]

1137 Allows correlation with the PAOS request. This optional attribute (and the header block as a whole)
1138 **MUST** be added by the ECP if the corresponding PAOS request specified the messageID attribute.

1139 Note that the equivalent functionality is provided in SAML using <AuthnRequest> and <Response>
1140 correlation.

1141 The PAOS Response SOAP header has no element content.

1142 Following is an example of an ECP-to-SP response.

```
1143 <SOAP-ENV:Envelope  
1144     xmlns:paos="urn:liberty:paos:2003-08"  
1145     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
1146     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">  
1147   <SOAP-ENV:Header>  
1148     <paos:Response refToMessageID="6c3a4f8b9c2d" SOAP-  
1149     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next/" SOAP-  
1150     ENV:mustUnderstand="1"/>  
1151     <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"  
1152     SOAP-ENV:mustUnderstand="1" SOAP-  
1153     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">  
1154       ...  
1155     </ecp:RelayState>  
1156   </SOAP-ENV:Header>  
1157   <SOAP-ENV:Body>  
1158     <samlp:Response> ... </samlp:Response>  
1159   </SOAP-ENV:Body>  
1160 </SOAP-ENV:Envelope>
```

1161 4.2.5 Security Considerations

1162 The <AuthnRequest> message SHOULD be signed. Per the rules specified by the browser SSO profile,
1163 the assertions enclosed in the <Response> MUST be signed. The delivery of the response in the SOAP
1164 envelope via PAOS is essentially analogous to the use of the HTTP POST binding and security
1165 countermeasures appropriate to that binding are used.

1166 The SOAP headers SHOULD be integrity protected, such as with SOAP Message Security or through the
1167 use of SSL/TLS over every HTTP exchange with the client.

1168 The service provider SHOULD be authenticated to the ECP, for example with server-side TLS
1169 authentication.

1170 The ECP SHOULD be authenticated to the identity provider, such as by maintaining an authenticated
1171 session. Any HTTP exchanges subsequent to the delivery of the <AuthnRequest> message and before
1172 the identity provider returns a <Response> MUST be securely associated with the original request.

1173 4.2.6 [E20]Use of Metadata

1174 [The rules specified in the browser SSO profile in Section 4.1.6 apply here as well. Specifically, the indexed](#)
1175 [endpoint element <md:AssertionConsumerService> with a binding of](#)
1176 [urn:oasis:names:tc:SAML:2.0:bindings:PAOS MAY be used to describe the supported binding](#)
1177 [and location\(s\) to which an identity provider may send responses to a service provider using this profile. IN](#)
1178 [addition, the endpoint <md:SingleSignOnService> with a binding of](#)
1179 [urn:oasis:names:tc:SAML:2.0:bindings:SOAP MAY be used to describe the supported binding](#)
1180 [and location\(s\) to which an service provider may send requests to an identity provider using this profile.](#)

1181 4.3 Identity Provider Discovery Profile

1182 This section defines a profile by which a service provider can discover which identity providers a principal
1183 is using with the Web Browser SSO profile. In deployments having more than one identity provider,
1184 service providers need a means to discover which identity provider(s) a principal uses. The discovery
1185 profile relies on a cookie that is written in a domain that is common between identity providers and service
1186 providers in a deployment. The domain that the deployment predetermines is known as the common

1187 domain in this profile, and the cookie containing the list of identity providers is known as the common
1188 domain cookie.

1189 Which entities host web servers in the common domain is a deployment issue and is outside the scope of
1190 this profile.

1191 **4.3.1 [E32]Required Information**

1192 **Identification:** [urn:oasis:names:tc:SAML:2.0:profiles:SSO:idp-discovery](#)

1193 **Contact information:** security-services-comment@lists.oasis-open.org

1194 **Description:** [Given below.](#)

1195 **Updates:** [None.](#)

1196 **4.3.2 Common Domain Cookie**

1197 The name of the cookie MUST be "_saml_idp". The format of the cookie value MUST be a set of one or
1198 more base-64 encoded URI values separated by a single space character. Each URI is the unique
1199 identifier of an identity provider, as defined in Section 8.3.6 of [SAMLCore]. The final set of values is then
1200 URL encoded.

1201 The common domain cookie writing service (see below) SHOULD append the identity provider's unique
1202 identifier to the list. If the identifier is already present in the list, it MAY remove and append it. The intent is
1203 that the most recently established identity provider session is the last one in the list.

1204 The cookie MUST be set with a Path prefix of "/". The Domain MUST be set to "[common-domain]" where
1205 [common-domain] is the common domain established within the deployment for use with this profile.
1206 There MUST be a leading period. The cookie MUST be marked as secure.

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

1214 **4.3.3 Setting the Common Domain Cookie**

1215 After the identity provider authenticates a principal, it MAY set the common domain cookie. The means by
1216 which the identity provider sets the cookie are implementation-specific so long as the cookie is
1217 successfully set with the parameters given above. One possible implementation strategy follows and
1218 should be considered non-normative. The identity provider may:

- 1219 • Have previously established a DNS and IP alias for itself in the common domain.
- 1220 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
1221 scheme. The structure of the URL is private to the implementation and may include session
1222 information needed to identify the user agent.
- 1223 • Set the cookie on the redirected user agent using the parameters specified above.
- 1224 • Redirect the user agent back to itself, or, if appropriate, to the service provider.

1225 **4.3.4 Obtaining the Common Domain Cookie**

1226 When a service provider needs to discover which identity providers a principal uses, it invokes an

1227 exchange designed to present the common domain cookie to the service provider after it is read by an
1228 HTTP server in the common domain.

1229 If the HTTP server in the common domain is operated by the service provider or if other arrangements are
1230 in place, the service provider MAY utilize the HTTP server in the common domain to relay its
1231 <AuthnRequest> to the identity provider for an optimized single sign-on process.

1232 The specific means by which the service provider reads the cookie are implementation-specific so long as
1233 it is able to cause the user agent to present cookies that have been set with the parameters given in
1234 Section 4.3.2. One possible implementation strategy is described as follows and should be considered
1235 non-normative. Additionally, it may be sub-optimal for some applications.

- 1236 • Have previously established a DNS and IP alias for itself in the common domain.
- 1237 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
1238 scheme. The structure of the URL is private to the implementation and may include session
1239 information needed to identify the user agent.
- 1240 • Redirect the user agent back to itself, or, if appropriate, to the identity provider.

1241 4.4 Single Logout Profile

1242 Once a principal has authenticated to an identity provider, the authenticating entity may establish a
1243 session with the principal (typically by means of a cookie, URL re-writing, or some other implementation-
1244 specific means). The identity provider may subsequently issue assertions to service providers or other
1245 relying parties, based on this authentication event; a relying party may use this to establish *its own* session
1246 with the principal.

1247 In such a situation, the identity provider can act as a session authority and the relying parties as session
1248 participants. At some later time, the principal may wish to terminate his or her session either with an
1249 individual session participant, or with all session participants in a given session managed by the session
1250 authority. The former case is considered out of scope of this specification. The latter case, however, may
1251 be satisfied using this profile of the SAML Single Logout protocol ([SAMLCore] Section 3.7).

1252 Note that a principal (or an administrator terminating a principal's session) may choose to terminate this
1253 "global" session either by contacting the session authority, or an individual session participant. Also note
1254 that an identity provider acting as a session authority may *itself* act as a session participant in situations in
1255 which it is the relying party for another identity provider's assertions regarding that principal.

1256 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1257 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1258 front-channel binding may be required, for example, in cases in which a principal's session state exists
1259 solely in a user agent in the form of a cookie and a direct interaction between the user agent and the
1260 session participant or session authority is required. As will be discussed below, session participants
1261 should if possible use a "front-channel" binding when initiating this profile to maximize the likelihood that
1262 the session authority can propagate the logout successfully to all participants.

1263 4.4.1 Required Information

1264 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:logout

1265 **Contact information:** security-services-comment@lists.oasis-open.org

1266 **Description:** Given below.

1267 **Updates:** None

1268 4.4.2 Profile Overview

1269 Figure 3 illustrates the basic template for achieving single logout:

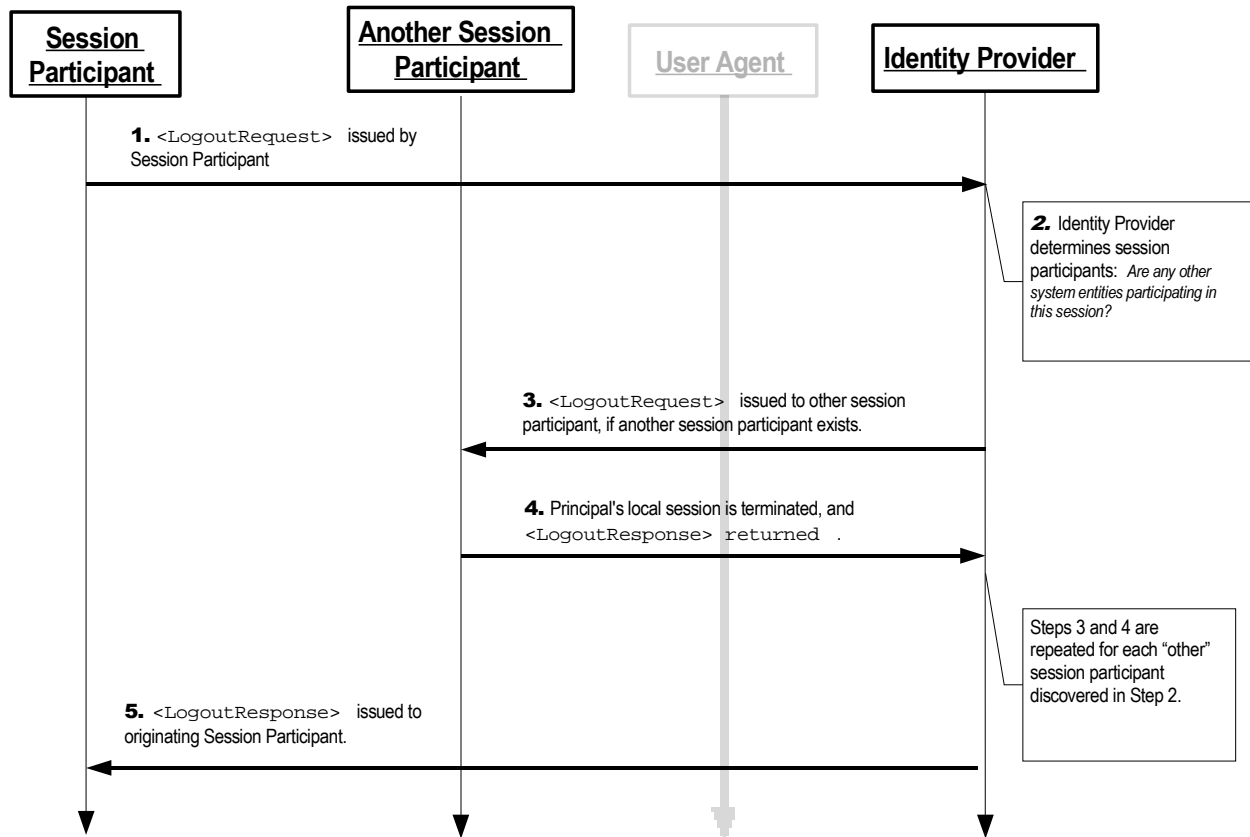


Figure 3

1270 The grayed-out user agent illustrates that the message exchange may pass through the user agent or
 1271 may be a direct exchange between system entities, depending on the SAML binding used to implement
 1272 the profile.

1273 The following steps are described by the profile. Within an individual step, there may be one or more
 1274 actual message exchanges depending on the binding used for that step and other implementation-
 1275 dependent behavior.

1276 **1. <LogoutRequest> issued by Session Participant to Identity Provider**

1277 In step 1, the session participant initiates single logout and terminates a principal's session(s) by
 1278 sending a <LogoutRequest> message to the identity provider from whom it received the
 1279 corresponding authentication assertion. The request may be sent directly to the identity provider
 1280 or sent indirectly through the user agent.

1281 **2. Identity Provider determines Session Participants**

1282 In step 2, the identity provider uses the contents of the <LogoutRequest> message (or if
 1283 initiating logout itself, some other mechanism) to determine the session(s) being terminated. If
 1284 there are no other session participants, the profile proceeds with step 5. Otherwise, steps 3 and 4
 1285 are repeated for each session participant identified.

1286 **3. <LogoutRequest> issued by Identity Provider to Session Participant/Authority**

1287 In step 3, the identity provider issues a <LogoutRequest> message to a session participant or
 1288 session authority related to one or more of the session(s) being terminated. The request may be

1289 sent directly to the entity or sent indirectly through the user agent (if consistent with the form of the
1290 request in step 1).

1291 **4. Session Participant/Authority issues <LogoutResponse> to Identity Provider**

1292 In step 4, a session participant or session authority terminates the principal's session(s) as
1293 directed by the request (if possible) and returns a <LogoutResponse> to the identity provider.
1294 The response may be returned directly to the identity provider or indirectly through the user agent
1295 (if consistent with the form of the request in step 3).

1296 **5. Identity Provider issues <LogoutResponse> to Session Participant**

1297 In step 5, the identity provider issues a <LogoutResponse> message to the original requesting
1298 session participant. The response may be returned directly to the session participant or indirectly
1299 through the user agent (if consistent with the form of the request in step 1).

1300 Note that an identity provider (acting as session authority) can initiate this profile at step 2 and issue a
1301 <LogoutRequest> to all session participants, also skipping step 5.

1302 **4.4.3 Profile Description**

1303 If the profile is initiated by a session participant, start with Section 4.4.3.1. If initiated by the identity
1304 provider, start with Section 4.4.3.2. In the descriptions below, the following is referred to:

1305 **Single Logout Service**

1306 This is the single logout protocol endpoint at an identity provider or session participant to which the
1307 <LogoutRequest> or <LogoutResponse> messages (or an artifact representing them) are
1308 delivered. The same or different endpoints MAY be used for requests and responses.

1309 **4.4.3.1 <LogoutRequest> Issued by Session Participant to Identity Provider**

1310 If the logout profile is initiated by a session participant, it examines the authentication assertion(s) it
1311 received pertaining to the session(s) being terminated, and collects the `SessionIndex` value(s) it
1312 received from the identity provider. If multiple identity providers are involved, then the profile MUST be
1313 repeated independently for each one.

1314 To initiate the profile, the session participant issues a <LogoutRequest> message to the identity
1315 provider's single logout service request endpoint containing one or more applicable <SessionIndex>
1316 elements. At least one element MUST be included. Metadata (as in [SAMLMeta]) MAY be used to
1317 determine the location of this endpoint and the bindings supported by the identity provider.

1318 **Asynchronous Bindings (Front-Channel)**

1319 The session participant SHOULD (if the principal's user agent is present) use an asynchronous
1320 binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind], to send the request to
1321 the identity provider through the user agent. The identity provider SHOULD then propagate any
1322 required logout messages to additional session participants as required using either a synchronous or
1323 asynchronous binding. The use of an asynchronous binding for the original request is preferred
1324 because it gives the identity provider the best chance of successfully propagating the logout to the
1325 other session participants during step 3.

1326 If the HTTP Redirect or POST binding is used, then the <LogoutRequest> message is delivered to
1327 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile
1328 defined in Section 5 is used by the identity provider, which makes a callback to the session participant
1329 to retrieve the <LogoutRequest> message, using for example the SOAP binding.

1330 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or
1331 TLS 1.0 [RFC2246] to maintain confidentiality and message integrity. The <LogoutRequest>
1332 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,

1333 if used, also provides for an alternate means of authenticating the request issuer when the artifact is
1334 dereferenced.

1335 Each of these bindings provide a RelayState mechanism that the session participant MAY use to
1336 associate the profile exchange with the original request. The session participant SHOULD reveal as
1337 little information as possible in the RelayState value unless the use of the profile does not require such
1338 privacy measures.

1339 **Synchronous Bindings (Back-Channel)**

1340 Alternatively, the session participant MAY use a synchronous binding, such as the SOAP binding
1341 [SAMLBind], to send the request directly to the identity provider. The identity provider SHOULD then
1342 propagate any required logout messages to additional session participants as required using a
1343 synchronous binding. The requester MUST authenticate itself to the identity provider, either by signing
1344 the <LogoutRequest> or using any other binding-supported mechanism.

1345 Profile-specific rules for the contents of the <LogoutRequest> message are included in Section 4.4.4.1.

1346 **4.4.3.2 Identity Provider Determines Session Participants**

1347 If the logout profile is initiated by an identity provider, or upon receiving a valid <LogoutRequest>
1348 message, the identity provider processes the request as defined in [SAMLCore]. It MUST examine the
1349 identifier and <SessionIndex> elements and determine the set of sessions to be terminated.

1350 The identity provider then follows steps 3 and 4 for each entity participating in the session(s) being
1351 terminated, other than the original requesting session participant (if any), as described in Section 3.7.3.2
1352 of [SAMLCore].

1353 **4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session 1354 Participant/Authority**

1355 To propagate the logout, the identity provider issues its own <LogoutRequest> to a session authority or
1356 participant in a session being terminated. The request is sent using a SAML binding consistent with the
1357 capability of the responder and the availability of the user agent at the identity provider.

1358 In general, the binding with which the original request was received in step 1 does not dictate the binding
1359 that may be used in this step except that as noted in step 1, using a synchronous binding that bypasses
1360 the user agent constrains the identity provider to use a similar binding to propagate additional requests.

1361 Profile-specific rules for the contents of the <LogoutRequest> message are included in Section 4.4.4.1.

1362 **4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity 1363 Provider**

1364 The session participant/authority MUST process the <LogoutRequest> message as defined in
1365 [SAMLCore]. After processing the message or upon encountering an error, the entity MUST issue a
1366 <LogoutResponse> message containing an appropriate status code to the requesting identity provider
1367 to complete the SAML protocol exchange.

1368 **Synchronous Bindings (Back-Channel)**

1369 If the identity provider used a synchronous binding, such as the SOAP binding [SAMLBind], the
1370 response is returned directly to complete the synchronous communication. The responder MUST
1371 authenticate itself to the requesting identity provider, either by signing the <LogoutResponse> or
1372 using any other binding-supported mechanism.

1373 **Asynchronous Bindings (Front-Channel)**

1374 If the identity provider used an asynchronous binding, such as the HTTP Redirect, POST, or Artifact

1375 bindings [SAMLBind], then the <LogoutResponse> (or artifact) is returned through the user agent to
1376 the identity provider's single logout service response endpoint. Metadata (as in [SAMLMeta]) MAY be
1377 used to determine the location of this endpoint and the bindings supported by the identity provider.
1378 Any asynchronous binding supported by both entities MAY be used.

1379 If the HTTP Redirect or POST binding is used, then the <LogoutResponse> message is delivered to
1380 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile
1381 defined in Section 5 is used by the identity provider, which makes a callback to the responding entity
1382 to retrieve the <LogoutResponse> message, using for example the SOAP binding.

1383 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or
1384 TLS 1.0 [RFC2246] to maintain confidentiality and message integrity. The <LogoutResponse>
1385 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,
1386 if used, also provides for an alternate means of authenticating the response issuer when the artifact is
1387 dereferenced.

1388 Profile-specific rules for the contents of the <LogoutResponse> message are included in Section
1389 4.4.4.2.

1390 **4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant**

1391 After processing the original session participant's <LogoutRequest> as described in the previous steps
1392 the identity provider MUST respond to the original request with a <LogoutResponse> containing an
1393 appropriate status code to complete the SAML protocol exchange.

1394 The response is sent to the original session participant, using a SAML binding consistent with the binding
1395 used in the original request, the capability of the responder, and the availability of the user agent at the
1396 identity provider. Assuming an asynchronous binding was used in step 1, then any binding supported by
1397 both entities MAY be used.

1398 Profile-specific rules for the contents of the <LogoutResponse> message are included in Section
1399 4.4.4.2.

1400 **4.4.4 Use of Single Logout Protocol**

1401 **4.4.4.1 <LogoutRequest> Usage**

1402 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1403 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1404 format:entity.

1405 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1406 the message or using a binding-specific mechanism.

1407 The principal MUST be identified in the request using an identifier that **strongly matches** the identifier in
1408 the authentication assertion the requester issued or received regarding the session being terminated, per
1409 the matching rules defined in Section 3.3.4 of [SAMLCore].

1410 If the requester is a session participant, it MUST include at least one <SessionIndex> element in the
1411 request. [\[E38\]\(Note that the session participant always receives a SessionIndex attribute in the](#)
1412 [<saml:AuthnStatement> elements that it receives to initiate the session, per Section 4.1.4.2 of the](#)
1413 [Web Browser SSO Profile.\)](#) If the requester is a session authority (or acting on its behalf), then it MAY
1414 omit any such elements to indicate the termination of all of the principal's applicable sessions.

1415 **4.4.4.2 <LogoutResponse> Usage**

1416 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding

1417 entity; the `Format` attribute MUST be omitted or have a value of
1418 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

1419 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1420 the message or using a binding-specific mechanism.

1421 4.4.5 Use of Metadata

1422 [SAMLMeta] defines an endpoint element, `<md:SingleLogoutService>`, to describe supported
1423 bindings and location(s) to which an entity may send requests and responses using this profile.

1424 A requester, if encrypting the principal's identifier, can use the responder's `<md:KeyDescriptor>`
1425 element with a `use` attribute of `encryption` to determine an appropriate encryption algorithm and
1426 settings to use, along with a public key to use in delivering a bulk encryption key.

1427 4.5 Name Identifier Management Profile

1428 In the scenario supported by the Name Identifier Management profile, an identity provider has exchanged
1429 some form of ~~[E55]persistentlong-term~~ identifier ~~(including but not limited to identifiers with a Format of~~
1430 ~~`urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`)~~ for a principal with a service
1431 provider, allowing them to share a common identifier for some length of time. Subsequently, the identity
1432 provider may wish to notify the service provider of a change in the ~~[E12]format and/or~~ value that it will use
1433 to identify the same principal in the future. Alternatively the service provider may wish to attach its own
1434 "alias" for the principal in order to ensure that the identity provider will include it when communicating with
1435 it in the future ~~about the principal[E55]using that identifier~~. Finally, one of the providers may wish to inform
1436 the other that it will no longer issue or accept messages using a particular identifier. To implement these
1437 scenarios, a profile of the SAML Name Identifier Management protocol is used.

1438 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1439 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1440 front-channel binding may be required, for example, in cases in which direct interaction between the user
1441 agent and the responding provider is required in order to effect the change.

1442 4.5.1 Required Information

1443 **Identification:** `urn:oasis:names:tc:SAML:2.0:profiles:SSO:nameid-mgmt`

1444 **Contact information:** security-services-comment@lists.oasis-open.org

1445 **Description:** Given below.

1446 **Updates:** None.

1447 4.5.2 Profile Overview

1448 Figure 4 illustrates the basic template for the name identifier management profile.

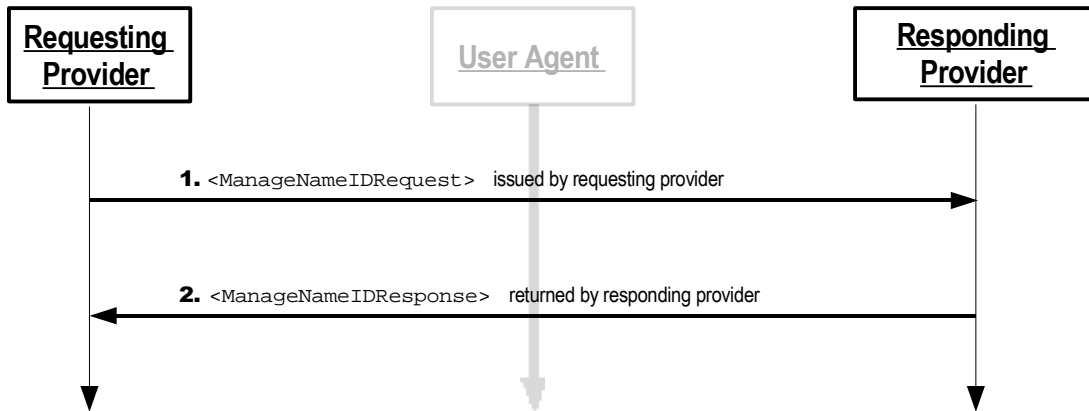


Figure 4

1449 The grayed-out user agent illustrates that the message exchange may pass through the user agent or
 1450 may be a direct exchange between system entities, depending on the SAML binding used to implement
 1451 the profile.

1452 The following steps are described by the profile. Within an individual step, there may be one or more
 1453 actual message exchanges depending on the binding used for that step and other implementation-
 1454 dependent behavior.

1455 **1. <ManageNameIDRequest> issued by Requesting Identity/Service Provider**

1456 In step 1, an identity or service provider initiates the profile by sending a
 1457 <ManageNameIDRequest> message to another provider that it wishes to inform of a change.
 1458 The request may be sent directly to the responding provider or sent indirectly through the user
 1459 agent.

1460 **2. <ManageNameIDResponse> issued by Responding Identity/Service Provider**

1461 In step 2, the responding provider (after processing the request) issues a
 1462 <ManageNameIDResponse> message to the original requesting provider. The response may be
 1463 returned directly to the requesting provider or indirectly through the user agent (if consistent with
 1464 the form of the request in step 1).

1465 **4.5.3 Profile Description**

1466 In the descriptions below, the following is referred to:

1467 **Name Identifier Management Service**

1468 This is the name identifier management protocol endpoint at an identity or service provider to which
 1469 the <ManageNameIDRequest> or <ManageNameIDResponse> messages (or an artifact
 1470 representing them) are delivered. The same or different endpoints MAY be used for requests and
 1471 responses.

1472 **4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider**

1473 To initiate the profile, the requesting provider issues a <ManageNameIDRequest> message to another
 1474 provider's name identifier management service request endpoint. Metadata (as in [SAMLMeta]) MAY be
 1475 used to determine the location of this endpoint and the bindings supported by the responding provider.

1476 **Synchronous Bindings (Back-Channel)**

1477 The requesting provider MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to
1478 send the request directly to the other provider. The requester MUST authenticate itself to the other
1479 provider, either by signing the <ManageNameIDRequest> or using any other binding-supported
1480 mechanism.

1481 **Asynchronous Bindings (Front-Channel)**

1482 Alternatively, the requesting provider MAY (if the principal's user agent is present) use an
1483 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to send the
1484 request to the other provider through the user agent.

1485 If the HTTP Redirect or POST binding is used, then the <ManageNameIDRequest> message is
1486 delivered to the other provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution
1487 profile defined in Section 5 is used by the other provider, which makes a callback to the requesting
1488 provider to retrieve the <ManageNameIDRequest> message, using for example the SOAP binding.

1489 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or
1490 TLS 1.0 [RFC2246] to maintain confidentiality and message integrity. The
1491 <ManageNameIDRequest> message MUST be signed if the HTTP POST or Redirect binding is
1492 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1493 request issuer when the artifact is dereferenced.

1494 Each of these bindings provide a RelayState mechanism that the requesting provider MAY use to
1495 associate the profile exchange with the original request. The requesting provider SHOULD reveal as
1496 little information as possible in the RelayState value unless the use of the profile does not require such
1497 privacy measures.

1498 Profile-specific rules for the contents of the <ManageNameIDRequest> message are included in Section
1499 4.5.4.1.

1500 **4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service 1501 Provider**

1502 The recipient MUST process the <ManageNameIDRequest> message as defined in [SAMLCore]. After
1503 processing the message or upon encountering an error, the recipient MUST issue a
1504 <ManageNameIDResponse> message containing an appropriate status code to the requesting provider
1505 to complete the SAML protocol exchange.

1506 **Synchronous Bindings (Back-Channel)**

1507 If the requesting provider used a synchronous binding, such as the SOAP binding [SAMLBind], the
1508 response is returned directly to complete the synchronous communication. The responder MUST
1509 authenticate itself to the requesting provider, either by signing the <ManageNameIDResponse> or
1510 using any other binding-supported mechanism.

1511 **Asynchronous Bindings (Front-Channel)**

1512 If the requesting provider used an asynchronous binding, such as the HTTP Redirect, POST, or
1513 Artifact bindings [SAMLBind], then the <ManageNameIDResponse> (or artifact) is returned through
1514 the user agent to the requesting provider's name identifier management service response endpoint.
1515 Metadata (as in [SAMLMeta]) MAY be used to determine the location of this endpoint and the bindings
1516 supported by the requesting provider. Any binding supported by both entities MAY be used.

1517 If the HTTP Redirect or POST binding is used, then the <ManageNameIDResponse> message is
1518 delivered to the requesting provider in this step. If the HTTP Artifact binding is used, the Artifact
1519 Resolution profile defined in Section 5 is used by the requesting provider, which makes a callback to
1520 the responding provider to retrieve the <ManageNameIDResponse> message, using for example the
1521 SOAP binding.

1522 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 [SSL3] or
1523 TLS 1.0 [RFC2246] to maintain confidentiality and message integrity. The

1524 <ManageNameIDResponse> message MUST be signed if the HTTP POST or Redirect binding is
1525 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1526 response issuer when the artifact is dereferenced.

1527 Profile-specific rules for the contents of the <ManageNameIDResponse> message are included in
1528 Section 4.5.4.2.

1529 **4.5.4 Use of Name Identifier Management Protocol**

1530 **4.5.4.1 <ManageNameIDRequest> Usage**

1531 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1532 the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`
1533 `format:entity`.

1534 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1535 the message or using a binding-specific mechanism.

1536 **4.5.4.2 <ManageNameIDResponse> Usage**

1537 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1538 entity; the `Format` attribute MUST be omitted or have a value of
1539 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

1540 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1541 the message or using a binding-specific mechanism.

1542 **4.5.5 Use of Metadata**

1543 [SAMLMeta] defines an endpoint element, <md:ManageNameIDService>, to describe supported
1544 bindings and location(s) to which an entity may send requests and responses using this profile.

1545 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>
1546 element with a `use` attribute of `encryption` to determine an appropriate encryption algorithm and
1547 settings to use, along with a public key to use in delivering a bulk encryption key.

1548

5 Artifact Resolution Profile

1549 [SAMLCore] defines an Artifact Resolution protocol for dereferencing a SAML artifact into a corresponding
1550 protocol message. The HTTP Artifact binding in [SAMLBind] leverages this mechanism to pass SAML
1551 protocol messages by reference. This profile describes the use of this protocol with a synchronous
1552 binding, such as the SOAP binding defined in [SAMLBind].

5.1 Required Information

1554 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:artifact

1555 **Contact information:** security-services-comment@lists.oasis-open.org

1556 **Description:** Given below.

1557 **Updates:** None

5.2 Profile Overview

1559 The message exchange and basic processing rules that govern this profile are largely defined by Section
1560 3.5 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1561 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1562 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1563 Figure 5 illustrates the basic template for the artifact resolution profile.

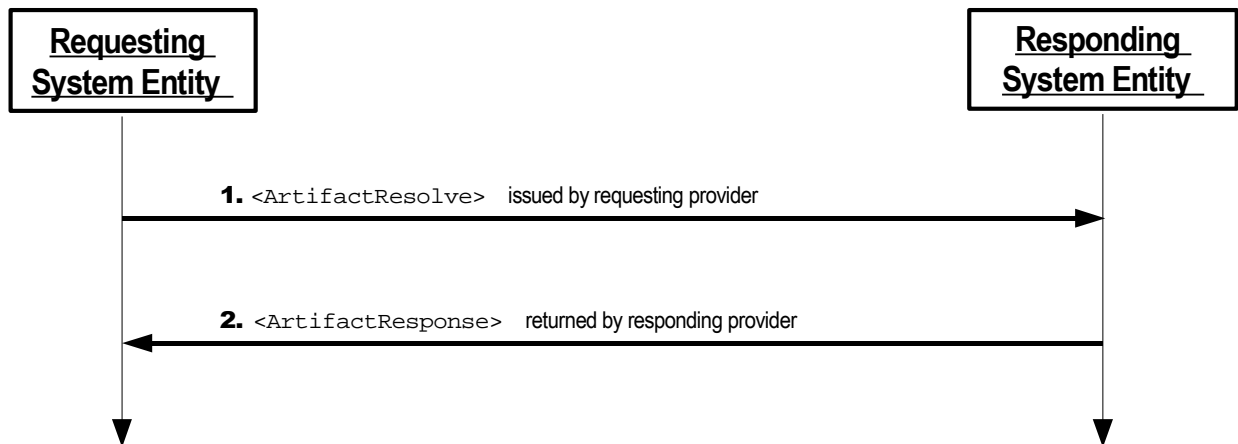


Figure 5

1564 The following steps are described by the profile.

1565 1. <ArtifactResolve> issued by Requesting Entity

1566 In step 1, a requester initiates the profile by sending an <ArtifactResolve> message to an
1567 artifact issuer.

1568 2. <ArtifactResponse> issued by Responding Entity

1569 In step 2, the responder (after processing the request) issues an <ArtifactResponse>
1570 message to the requester.

1571 5.3 Profile Description

1572 In the descriptions below, the following is referred to:

1573 Artifact Resolution Service

1574 This is the artifact resolution protocol endpoint at an artifact issuer to which <ArtifactResolve>
1575 messages are delivered.

1576 5.3.1 <ArtifactResolve> issued by Requesting Entity

1577 To initiate the profile, a requester, having received an artifact and determined the issuer using the
1578 SourceID, sends an <ArtifactResolve> message containing the artifact to an artifact issuer's artifact
1579 resolution service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of this
1580 endpoint and the bindings supported by the artifact issuer.

1581 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1582 request directly to the artifact issuer. The requester SHOULD authenticate itself to the responder, either by
1583 signing the <ArtifactResolve> message or using any other binding-supported mechanism. Specific
1584 profiles that use the HTTP Artifact binding MAY impose additional requirements such that authentication is
1585 mandatory.

1586 Profile-specific rules for the contents of the <ArtifactResolve> message are included in Section 5.4.1.

1587 5.3.2 <ArtifactResponse> issued by Responding Entity

1588 The artifact issuer MUST process the <ArtifactResolve> message as defined in [SAMLCore]. After
1589 processing the message or upon encountering an error, the artifact issuer MUST return an
1590 <ArtifactResponse> message containing an appropriate status code to the requester to complete the
1591 SAML protocol exchange. If successful, the dereferenced SAML protocol message corresponding to the
1592 artifact will also be included.

1593 The responder MUST authenticate itself to the requester, either by signing the <ArtifactResponse> or
1594 using any other binding-supported mechanism.

1595 Profile-specific rules for the contents of the <ArtifactResponse> message are included in Section
1596 5.4.2.

1597 5.4 Use of Artifact Resolution Protocol

1598 5.4.1 <ArtifactResolve> Usage

1599 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1600 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1601 format:entity.

1602 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by
1603 signing the message or using a binding-specific mechanism. Specific profiles that use the HTTP Artifact
1604 binding MAY impose additional requirements such that authentication is mandatory.

1605 **5.4.2 <ArtifactResponse> Usage**

1606 The <Issuer> element MUST be present and MUST contain the unique identifier of the artifact issuer;
1607 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1608 format:entity.

1609 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1610 the message or using a binding-specific mechanism.

1611 **5.5 Use of Metadata**

1612 [SAMLMeta] defines an indexed endpoint element, <md:ArtifactResolutionService>, to describe
1613 supported bindings and location(s) to which a requester may send requests using this profile. The index
1614 attribute is used to distinguish the possible endpoints that may be specified by reference in the artifact's
1615 EndpointIndex field.

1616

6 Assertion Query/Request Profile

1617 [SAMLCore] defines a protocol for requesting existing assertions by reference or by querying on the basis
1618 of a subject and additional statement-specific criteria. This profile describes the use of this protocol with a
1619 synchronous binding, such as the SOAP binding defined in [SAMLBind].

6.1 Required Information

1621 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:query

1622 **Contact information:** security-services-comment@lists.oasis-open.org

1623 **Description:** Given below.

1624 **Updates:** None.

6.2 Profile Overview

1626 The message exchange and basic processing rules that govern this profile are largely defined by Section
1627 3.3 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1628 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1629 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1630 Figure 6 illustrates the basic template for the query/request profile.

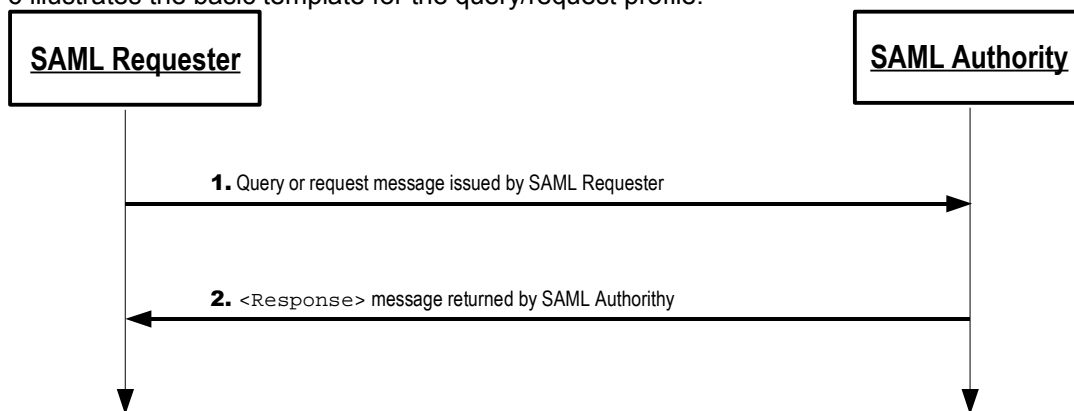


Figure 6

1631 The following steps are described by the profile.

1. Query/Request issued by SAML Requester

1633 In step 1, a SAML requester initiates the profile by sending an `<AssertionIDRequest>`,
1634 `<SubjectQuery>`, `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>`
1635 message to a SAML authority.

2. <Response> issued by SAML Authority

1637 In step 2, the responding SAML authority (after processing the query or request) issues a
1638 `<Response>` message to the SAML requester.

1639 **6.3 Profile Description**

1640 In the descriptions below, the following are referred to:

1641 **Query/Request Service**

1642 This is the query/request protocol endpoint at a SAML authority to which query or
1643 `<AssertionIDRequest>` messages are delivered.

1644 **6.3.1 Query/Request issued by SAML Requester**

1645 To initiate the profile, a SAML requester issues an `<AssertionIDRequest>`, `<SubjectQuery>`,
1646 `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>` message to a SAML authority's
1647 query/request service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of
1648 this endpoint and the bindings supported by the SAML authority.

1649 The SAML requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send
1650 the request directly to the identity provider. The requester SHOULD authenticate itself to the SAML
1651 authority either by signing the message or using any other binding-supported mechanism.

1652 Profile-specific rules for the contents of the various messages are included in Section 6.4.1.

1653 **6.3.2 `<Response>` issued by SAML Authority**

1654 The SAML authority MUST process the query or request message as defined in [SAMLCore]. After
1655 processing the message or upon encountering an error, the SAML authority MUST return a `<Response>`
1656 message containing an appropriate status code to the SAML requester to complete the SAML protocol
1657 exchange. If the request is successful in locating one or more matching assertions, they will also be
1658 included in the response.

1659 The responder SHOULD authenticate itself to the requester, either by signing the `<Response>` or using
1660 any other binding-supported mechanism.

1661 Profile-specific rules for the contents of the `<Response>` message are included in Section 6.4.2.

1662 **6.4 Use of Query/Request Protocol**

1663 **6.4.1 Query/Request Usage**

1664 The `<Issuer>` element MUST be present.

1665 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by
1666 signing the message or using a binding-specific mechanism.

1667 **6.4.2 `<Response>` Usage**

1668 The `<Issuer>` element MUST be present and MUST contain the unique identifier of the responding
1669 SAML authority; the `Format` attribute MUST be omitted or have a value of
1670 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`. Note that this need not necessarily
1671 match the `<Issuer>` element in the returned assertion(s).

1672 The responder SHOULD authenticate itself to the requester and ensure message integrity, either by
1673 signing the message or using a binding-specific mechanism.

1674 **6.5 Use of Metadata**

1675 [SAMLMeta] defines several endpoint elements, `<md:AssertionIDRequestService>`,
1676 `<md:AuthnQueryService>`, `<md:AttributeService>`, and `<md:AuthzService>`, to describe
1677 supported bindings and location(s) to which a requester may send requests or queries using this profile.

1678 The SAML authority, if encrypting the resulting assertions or assertion contents for a particular entity, can
1679 use that entity's `<md:KeyDescriptor>` element with a `use` attribute of encryption to determine an
1680 appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk
1681 encryption key.

1682 The various role descriptors MAY contain `<md:NameIDFormat>`, `<md:AttributeProfile>`, and
1683 `<saml:Attribute>` elements (as applicable) to indicate the general ability to support particular name
1684 identifier formats, attribute profiles, or specific attributes and values. The ability to support any such
1685 features during a given request is dependent on policy and the discretion of the authority.

1686

7 Name Identifier Mapping Profile

1687 [SAMLCore] defines a Name Identifier Mapping protocol for mapping a principal's name identifier into a
1688 different name identifier for the same principal. This profile describes the use of this protocol with a
1689 synchronous binding, such as the SOAP binding defined in [SAMLBind], and additional guidelines for
1690 protecting the privacy of the principal with encryption and limiting the use of the mapped identifier.

7.1 Required Information

1692 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:nameidmapping

1693 **Contact information:** security-services-comment@lists.oasis-open.org

1694 **Description:** Given below.

1695 **Updates:** None.

7.2 Profile Overview

1697 The message exchange and basic processing rules that govern this profile are largely defined by Section
1698 3.8 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1699 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1700 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1701 Figure 7 illustrates the basic template for the name identifier mapping profile.

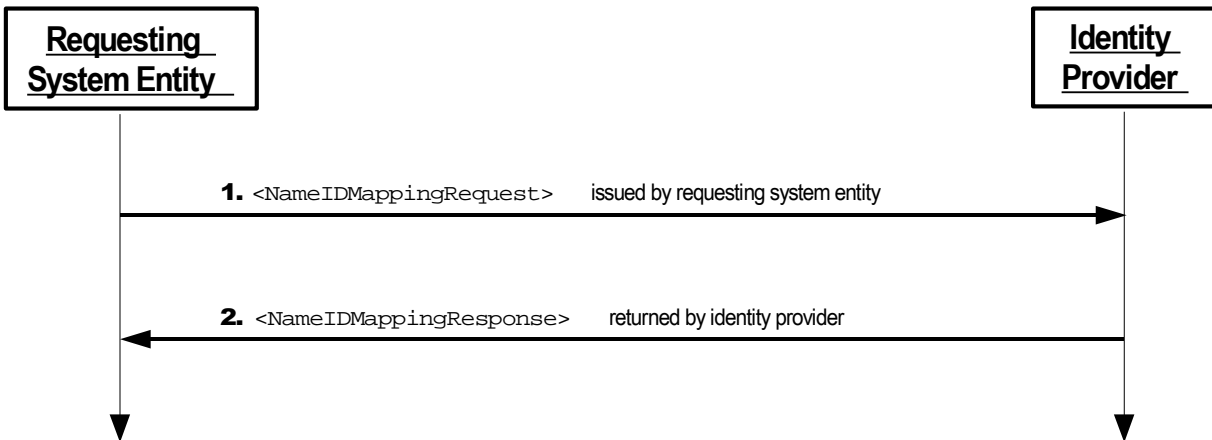


Figure 7

1702 The following steps are described by the profile.

1703 1. <NameIDMappingRequest> issued by Requesting Entity

1704 In step 1, a requester initiates the profile by sending a <NameIDMappingRequest> message to
1705 an identity provider.

1706 2. <NameIDMappingResponse> issued by Identity Provider

1707 In step 2, the responding identity provider (after processing the request) issues a
1708 <NameIDMappingResponse> message to the requester.

1709 **7.3 Profile Description**

1710 In the descriptions below, the following is referred to:

1711 **Name Identifier Mapping Service**

1712 This is the name identifier mapping protocol endpoint at an identity provider to which
1713 <NameIDMappingRequest> messages are delivered.

1714 **7.3.1 <NameIDMappingRequest> issued by Requesting Entity**

1715 To initiate the profile, a requester issues a <NameIDMappingRequest> message to an identity provider's
1716 name identifier mapping service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the
1717 location of this endpoint and the bindings supported by the identity provider.

1718 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1719 request directly to the identity provider. The requester MUST authenticate itself to the identity provider,
1720 either by signing the <NameIDMappingRequest> or using any other binding-supported mechanism.

1721 Profile-specific rules for the contents of the <NameIDMappingRequest> message are included in
1722 Section 7.4.1.

1723 **7.3.2 <NameIDMappingResponse> issued by Identity Provider**

1724 The identity provider MUST process the <ManageNameIDRequest> message as defined in [SAMLCore].
1725 After processing the message or upon encountering an error, the identity provider MUST return a
1726 <NameIDMappingResponse> message containing an appropriate status code to the requester to
1727 complete the SAML protocol exchange.

1728 The responder MUST authenticate itself to the requester, either by signing the
1729 <NameIDMappingResponse> or using any other binding-supported mechanism.

1730 Profile-specific rules for the contents of the <NameIDMappingResponse> message are included in
1731 Section 7.4.2.

1732 **7.4 Use of Name Identifier Mapping Protocol**

1733 **7.4.1 <NameIDMappingRequest> Usage**

1734 The <Issuer> element MUST be present.

1735 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1736 the message or using a binding-specific mechanism.

1737 **7.4.2 <NameIDMappingResponse> Usage**

1738 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1739 identity provider; the Format attribute MUST be omitted or have a value of
1740 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1741 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1742 the message or using a binding-specific mechanism.

1743 Section 2.2.3 of [SAMLCore] defines the use of encryption to apply confidentiality to a name identifier. In
1744 most cases, the identity provider SHOULD encrypt the mapped name identifier it returns to the requester
1745 to protect the privacy of the principal. The requester can extract the <EncryptedID> element and place it
1746 in subsequent protocol messages or assertions.

1747 **7.4.2.1 Limiting Use of Mapped Identifier**

1748 Additional limits on the use of the resulting identifier MAY be applied by the identity provider by returning
1749 the mapped name identifier in the form of an <Assertion> containing the identifier in its <Subject> but
1750 without any statements. The assertion is then encrypted and the result used as the <EncryptedData>
1751 element in the <EncryptedID> returned to the requester. The assertion MAY include a <Conditions>
1752 element to limit use, as defined by [SAMLCore], such as time-based constraints or use by specific relying
1753 parties, and MUST be signed for integrity protection.

1754 **7.5 Use of Metadata**

1755 [SAMLMeta] defines an endpoint element, <md:NameIDMappingService>, to describe supported
1756 bindings and location(s) to which a requester may send requests using this profile.

1757 The identity provider, if encrypting the resulting identifier for a particular entity, can use that entity's
1758 <md:KeyDescriptor> element with a use attribute of encryption to determine an appropriate
1759 encryption algorithm and settings to use, along with a public key to use in delivering a bulk encryption key.

1760 8 SAML Attribute Profiles

1761 8.1 Basic Attribute Profile

1762 The Basic attribute profile specifies simplified, but non-unique, naming of SAML attributes together with
1763 attribute values based on the built-in XML Schema data types, eliminating the need for extension schemas
1764 to validate syntax.

1765 8.1.1 Required Information

1766 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:basic

1767 **Contact information:** security-services-comment@lists.oasis-open.org

1768 **Description:** Given below.

1769 **Updates:** None.

1770 8.1.2 SAML Attribute Naming

1771 The NameFormat XML attribute in <Attribute> elements MUST be
1772 urn:oasis:names:tc:SAML:2.0:attrname-format:basic.

1773 The Name XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

1774 8.1.2.1 Attribute Name Comparison

1775 Two <Attribute> elements refer to the same SAML attribute if and only if the values of their Name XML
1776 attributes are equal in the sense of Section 3.3.6 of [Schema2].

1777 8.1.3 Profile-Specific XML Attributes

1778 No additional XML attributes are defined for use with the <Attribute> element.

1779 8.1.4 SAML Attribute Values

1780 The schema type of the contents of the <AttributeValue> element MUST be drawn from one of the
1781 types defined in Section 3[E51]-3 of [Schema2]. The xsi:type attribute MUST be present and be given
1782 the appropriate value.

1783 8.1.5 Example

```
1784 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"  
1785     Name="FirstName">  
1786     <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>  
1787 </saml:Attribute>
```

1788 8.2 X.500/LDAP Attribute Profile [E53] – Deprecated

1789 [\[E53\]Note: This attribute profile is deprecated because of a flaw that makes it schema-](#)
1790 [invalid. The SSTC has replaced it with a separately published SAML V2.0 X.500/LDAP](#)
1791 [Attribute Profile specification that removes this flaw.](#)

1792 | Directories based on the ITU-T X.500 specifications [X.500] and the related IETF Lightweight Directory
1793 Access Protocol specifications [LDAP] are widely deployed. Directory schema is used to model
1794 information to be stored in these directories. In particular, in X.500, attribute type definitions are used to
1795 specify the syntax and other features of attributes, the basic information storage unit in a directory (this
1796 document refers to these as “directory attributes”). Directory attribute types are defined in schema in the
1797 X.500 and LDAP specifications themselves, schema in other public documents (such as the
1798 Internet2/Educause EduPerson schema [eduPerson], or the inetOrgperson schema [RFC2798]), and
1799 schema defined for private purposes. In any of these cases, it is useful for deployers to take advantage of
1800 these directory attribute types in the context of SAML attribute statements, without having to manually
1801 create SAML-specific attribute definitions for them, and to do this in an interoperable fashion.
1802 The X.500/LDAP attribute profile defines a common convention for the naming and representation of such
1803 attributes when expressed as SAML attributes.

1804 8.2.1 Required Information

1805 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500 (this is also the target namespace
1806 assigned in the corresponding X.500/LDAP profile schema document [SAMLX500-xsd])

1807 **Contact information:** security-services-comment@lists.oasis-open.org

1808 **Description:** Given below.

1809 **Updates:** None.

1810 8.2.2 SAML Attribute Naming

1811 The `NameFormat` XML attribute in `<Attribute>` elements **MUST** be
1812 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1813 To construct attribute names, the URN `oid` namespace described in IETF RFC 3061 [RFC3061] is used.
1814 In this approach the `Name` XML attribute is based on the OBJECT IDENTIFIER assigned to the directory
1815 attribute type.

1816 **Example:**

1817 `urn:oid:2.5.4.3`

1818 Since X.500 procedures require that every attribute type be identified with a unique OBJECT IDENTIFIER,
1819 this naming scheme ensures that the derived SAML attribute names are unambiguous.

1820 For purposes of human readability, there may also be a requirement for some applications to carry an
1821 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
1822 [SAMLCore]) **MAY** be used for this purpose. If the definition of the directory attribute type includes one or
1823 more descriptors (short names) for the attribute type, the `FriendlyName` value, if present, **SHOULD** be
1824 one of the defined descriptors.

1825 8.2.2.1 Attribute Name Comparison

1826 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
1827 values are equal in the sense of [RFC3061]. The `FriendlyName` attribute plays no role in the
1828 comparison.

1829 8.2.3 Profile-Specific XML Attributes

1830 No additional XML attributes are defined for use with the `<Attribute>` element.

1831 8.2.4 SAML Attribute Values

1832 Directory attribute type definitions for use in native X.500 directories specify the syntax of the attribute
1833 using ASN.1 [ASN.1]. For use in LDAP, directory attribute definitions additionally include an LDAP syntax
1834 which specifies how attribute or assertion values conforming to the syntax are to be represented when
1835 transferred in the LDAP protocol (known as an LDAP-specific encoding). The LDAP-specific encoding
1836 commonly produces Unicode characters in UTF-8 form. This SAML attribute profile specifies the form of
1837 SAML attribute values only for those directory attributes which have LDAP syntaxes. Future extensions to
1838 this profile may define attribute value formats for directory attributes whose syntaxes specify other
1839 encodings.

1840 To represent the encoding rules in use for a particular attribute value, the <AttributeValue> element
1841 MUST contain an XML attribute named `Encoding` defined in the XML namespace
1842 `urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500`. (See [E53] for an issue with this
1843 [attribute](#).)

1844 For any directory attribute with a syntax whose LDAP-specific encoding exclusively produces UTF-8
1845 character strings as values, the SAML attribute value is encoded as simply the UTF-8 string itself, as the
1846 content of the <AttributeValue> element, with no additional whitespace. In such cases, the
1847 `xsi:type` XML attribute MUST be set to `xs:string`. The profile-specific `Encoding` XML attribute is
1848 provided, with a value of `LDAP`.

1849 A list of some LDAP attribute syntaxes to which this applies is:

1850	Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3
1851	Bit String	1.3.6.1.4.1.1466.115.121.1.6
1852	Boolean	1.3.6.1.4.1.1466.115.121.1.7
1853	Country String	1.3.6.1.4.1.1466.115.121.1.11
1854	DN	1.3.6.1.4.1.1466.115.121.1.12
1855	Directory String	1.3.6.1.4.1.1466.115.121.1.15
1856	Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22
1857	Generalized Time	1.3.6.1.4.1.1466.115.121.1.24
1858	IA5 String	1.3.6.1.4.1.1466.115.121.1.26
1859	INTEGER	1.3.6.1.4.1.1466.115.121.1.27
1860	LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54
1861	Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30
1862	Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31
1863	Name And Optional UID	1.3.6.1.4.1.1466.115.121.1.34
1864	Name Form Description	1.3.6.1.4.1.1466.115.121.1.35
1865	Numeric String	1.3.6.1.4.1.1466.115.121.1.36
1866	Object Class Description	1.3.6.1.4.1.1466.115.121.1.37
1867	Octet String	1.3.6.1.4.1.1466.115.121.1.40
1868	OID	1.3.6.1.4.1.1466.115.121.1.38
1869	Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39
1870	Postal Address	1.3.6.1.4.1.1466.115.121.1.41
1871	Presentation Address	1.3.6.1.4.1.1466.115.121.1.43
1872	Printable String	1.3.6.1.4.1.1466.115.121.1.44
1873	Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58
1874	Telephone Number	1.3.6.1.4.1.1466.115.121.1.50
1875	UTC Time	1.3.6.1.4.1.1466.115.121.1.53

1876 For all other LDAP syntaxes, the attribute value is encoded, as the content of the <AttributeValue>
1877 element, by base64-encoding [RFC2045] the [\[E48\]encompassingcontents of the ASN.1 OCTET STRING-](#)
1878 [encoded LDAP attribute value \(not including the ASN.1 OCTET STRING wrapper\)](#). The `xsi:type` XML
1879 attribute MUST be set to `xs:base64Binary`. The profile-specific `Encoding` XML attribute is provided,
1880 with a value of `LDAP`.

1881 When comparing SAML attribute values for equality, the matching rules specified for the corresponding
1882 directory attribute type MUST be observed (case sensitivity, for example).

1883 8.2.5 Profile-Specific Schema

1884 The following schema listing shows how the profile-specific Encoding XML attribute is defined
1885 [SAMLX500-xsd]:

```
1886 <schema
1887   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
1888   xmlns="http://www.w3.org/2001/XMLSchema"
1889   elementFormDefault="unqualified"
1890   attributeFormDefault="unqualified"
1891   blockDefault="substitution"
1892   version="2.0">
1893   <annotation>
1894     <documentation>
1895       Document identifier: saml-schema-x500-2.0
1896       Location: http://docs.oasis-open.org/security/saml/v2.0/
1897       Revision history:
1898         V2.0 (March, 2005):
1899         Custom schema for X.500 attribute profile, first published in
1900 SAML 2.0.
1901     </documentation>
1902   </annotation>
1903   <attribute name="Encoding" type="string"/>
1904 </schema>
```

1905 8.2.6 Example

1906 The following is an example of a mapping of the "givenName" directory attribute, representing the SAML
1907 assertion subject's first name. It's OBJECT IDENTIFIER is 2.5.4.42 and its LDAP syntax is Directory
1908 String.

```
1909 <saml:Attribute
1910   xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
1911     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1912     Name="urn:oid:2.5.4.42" FriendlyName="givenName">
1913   <saml:AttributeValue xsi:type="xs:string"
1914     x500:Encoding="LDAP">Steven</saml:AttributeValue>
1915 </saml:Attribute>
```

1916 8.3 UUID Attribute Profile

1917 The UUID attribute profile standardizes the expression of UUID values as SAML attribute names and
1918 values. It is applicable when the attribute's source system is one that identifies an attribute or its value with
1919 a UUID.

1920 8.3.1 Required Information

1921 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:UUID

1922 **Contact information:** security-services-comment@lists.oasis-open.org

1923 **Description:** Given below.

1924 **Updates:** None.

1925 8.3.2 UUID and GUID Background

1926 UUIDs (Universally Unique Identifiers), also known as GUIDs (Globally Unique Identifiers), are used to
1927 define objects and subjects such that they are guaranteed uniqueness across space and time. UUIDs

1928 were originally used in the Network Computing System (NCS), and then used in the Open Software
1929 Foundation's (OSF) Distributed Computing Environment (DCE). Recently GUIDs have been used in
1930 Microsoft's COM and Active Directory/Windows 2000/2003 platform.

1931 A UUID is a 128 bit number, generated such that it should never be duplicated within the domain of
1932 interest. UUIDs are used to represent a wide range of objects including, but not limited to, subjects/users,
1933 groups of users and node names. A UUID, represented as a hexadecimal string, is as follows:

1934 `f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1935 In DCE and Microsoft Windows, the UUID is usually presented to the administrator in the form of a
1936 "friendly name". For instance the above UUID could represent the user john.doe@example.com.

1937 **8.3.3 SAML Attribute Naming**

1938 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
1939 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1940 If the underlying representation of the attribute's name is a UUID, then the URN `uuid` namespace
1941 described in [E70][RFC4122][Mealling] is used. In this approach the `Name` XML attribute is based on the
1942 URN form of the underlying UUID that identifies the attribute.

1943 Example:

1944 `urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1945 If the underlying representation of the attribute's name is not a UUID, then any form of URI MAY be used
1946 in the `Name` XML attribute.

1947 For purposes of human readability, there may also be a requirement for some applications to carry an
1948 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in
1949 [SAMLCore]) MAY be used for this purpose.

1950 **8.3.3.1 Attribute Name Comparison**

1951 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
1952 values are equal in the sense of [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-05.txt][E70]
1953 [RFC4122]. The `FriendlyName` attribute plays no role in the comparison.

1954 **8.3.4 Profile-Specific XML Attributes**

1955 No additional XML attributes are defined for use with the `<Attribute>` element.

1956 **8.3.5 SAML Attribute Values**

1957 In cases in which the attribute's value is also a UUID, the same URN syntax described above MUST be
1958 used to express the value within the `<AttributeValue>` element. The `xsi:type` XML attribute MUST
1959 be set to `xs:anyURI`.

1960 If the attribute's value is not a UUID, then there are no restrictions on the use of the `<AttributeValue>`
1961 element.

1962 **8.3.6 Example**

1963 The following is an example of a DCE Extended Registry Attribute, the "pre_auth_req" setting, which has a
1964 well-known UUID of 6c9d0ec8-dd2d-11cc-abdd-080009353559 and is integer-valued.

```
1965 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1966           Name="urn:uuid:6c9d0ec8-dd2d-11cc-abdd-080009353559"
1967           FriendlyName="pre_auth_req">
1968     <saml:AttributeValue xsi:type="xs:integer">1</saml:AttributeValue>
1969 </saml:Attribute>
```

1970 8.4 DCE PAC Attribute Profile

1971 The DCE PAC attribute profile defines the expression of DCE PAC information as SAML attribute names
1972 and values. It is used to standardize a mapping between the primary information that makes up a DCE
1973 principal's identity and a set of SAML attributes. This profile builds on the UUID attribute profile defined in
1974 Section 8.3.

1975 8.4.1 Required Information

1976 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE (this is also the target namespace
1977 assigned in the corresponding DCE PAC attribute profile schema document [SAMLDCExsd])

1978 **Contact information:** security-services-comment@lists.oasis-open.org

1979 **Description:** Given below.

1980 **Updates:** None.

1981 8.4.2 PAC Description

1982 A DCE PAC is an extensible structure that can carry arbitrary DCE registry attributes, but a core set of
1983 information is common across principals and makes up the bulk of a DCE identity:

- 1984 • The principal's DCE "realm" or "cell"
- 1985 • The principal's unique identifier
- 1986 • The principal's primary DCE local group membership
- 1987 • The principal's set of DCE local group memberships (multi-valued)
- 1988 • The principal's set of DCE foreign group memberships (multi-valued)

1989 The primary value(s) of each of these attributes is a UUID.

1990 8.4.3 SAML Attribute Naming

1991 This profile defines a mapping of specific DCE information into SAML attributes, and thus defines actual
1992 specific attribute names, rather than a naming convention.

1993 For all attributes defined by this profile, the `NameFormat` XML attribute in `<Attribute>` elements MUST
1994 have the value `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1995 For purposes of human readability, there may also be a requirement for some applications to carry an
1996 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in
1997 [SAMLCore]) MAY be used for this purpose.

1998 See Section 8.4.6 for the specific attribute names defined by this profile.

1999 8.4.3.1 Attribute Name Comparison

2000 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
2001 values are equal in the sense of [<http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-05.txt>][E70]

2002 | [\[RFC4122\]](#). The `FriendlyName` attribute plays no role in the comparison.

2003 8.4.4 Profile-Specific XML Attributes

2004 No additional XML attributes are defined for use with the `<Attribute>` element.

2005 8.4.5 SAML Attribute Values

2006 The primary value(s) of each of the attributes defined by this profile is a UUID. The URN syntax described
2007 in Section 8.3.5 of the UUID profile is used to represent such values.

2008 However, additional information associated with the UUID value is permitted by this profile, consisting of a
2009 friendly, human-readable string, and an additional UUID representing a DCE cell or realm. The additional
2010 information is carried in the `<AttributeValue>` element in `FriendlyName` and `Realm` XML attributes
2011 defined in the XML namespace `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE`. Note
2012 that this is not the same as the `FriendlyName` XML attribute defined in [\[SAMLCore\]](#), although it has the
2013 same basic purpose.

2014 The following schema listing shows how the profile-specific XML attributes and complex type used in an
2015 `xsi:type` specification are defined [\[SAMLDCE-xsd\]](#):

```
2016 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"
2017   xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"
2018   xmlns="http://www.w3.org/2001/XMLSchema"
2019   elementFormDefault="unqualified"
2020   attributeFormDefault="unqualified"
2021   blockDefault="substitution"
2022   version="2.0">
2023   <annotation>
2024     <documentation>
2025       Document identifier: saml-schema-dce-2.0
2026       Location: http://docs.oasis-open.org/security/saml/v2.0/
2027       Revision history:
2028       V2.0 (March, 2005):
2029         Custom schema for DCE attribute profile, first published in
2030 SAML 2.0.
2031     </documentation>
2032   </annotation>
2033   <complexType name="DCEValueType">
2034     <simpleContent>
2035       <extension base="anyURI">
2036         <attribute ref="dce:Realm" use="optional"/>
2037         <attribute ref="dce:FriendlyName" use="optional"/>
2038       </extension>
2039     </simpleContent>
2040   </complexType>
2041   <attribute name="Realm" type="anyURI"/>
2042   <attribute name="FriendlyName" type="string"/>
2043 </schema>
```

2044 8.4.6 Attribute Definitions

2045 The following are the set of SAML attributes defined by this profile. In each case, an `xsi:type` XML
2046 attribute MAY be included in the `<AttributeValue>` element, but MUST have the value
2047 **dce:DCEValueType**, where the `dce` prefix is arbitrary and MUST be bound to the XML namespace
2048 `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE`.

2049 Note that such use of `xsi:type` will require validating attribute consumers to include the extension
2050 schema defined by this profile.

2051 **8.4.6.1 Realm**

2052 This single-valued attribute represents the SAML assertion subject's DCE realm or cell.

2053 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm

2054 The single <AttributeValue> element contains a UUID in URN form identifying the SAML assertion
2055 subject's DCE realm/cell, with an optional profile-specific `FriendlyName` XML attribute containing the
2056 realm's string name.

2057 **8.4.6.2 Principal**

2058 This single-valued attribute represents the SAML assertion subject's DCE principal identity.

2059 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal

2060 The single <AttributeValue> element contains a UUID in URN form identifying the SAML assertion
2061 subject's DCE principal identity, with an optional profile-specific `FriendlyName` XML attribute containing
2062 the principal's string name.

2063 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
2064 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
2065 8.4.6.1).

2066 **8.4.6.3 Primary Group**

2067 This single-valued attribute represents the SAML assertion subject's primary DCE group membership.

2068 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group

2069 The single <AttributeValue> element contains a UUID in URN form identifying the SAML assertion
2070 subject's primary DCE group, with an optional profile-specific `FriendlyName` XML attribute containing
2071 the group's string name.

2072 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
2073 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
2074 8.4.6.1).

2075 **8.4.6.4 Groups**

2076 This multi-valued attribute represents the SAML assertion subject's DCE local group memberships.

2077 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups

2078 Each <AttributeValue> element contains a UUID in URN form identifying a DCE group membership
2079 of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute containing
2080 the group's string name.

2081 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
2082 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
2083 8.4.6.1).

2084 **8.4.6.5 Foreign Groups**

2085 This multi-valued attribute represents the SAML assertion subject's DCE foreign group memberships.

2086 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-groups

2087 Each <AttributeValue> element contains a UUID in URN form identifying a DCE foreign group
2088 membership of the SAML assertion subject, with an optional profile-specific FriendlyName XML attribute
2089 containing the group's string name.

2090 The profile-specific Realm XML attribute MUST be included and MUST contain a UUID in URN form
2091 identifying the DCE realm/cell of the foreign group.

2092 8.4.7 Example

2093 The following is an example of the transformation of PAC data into SAML attributes belonging to a DCE
2094 principal named "jdoe" in realm "example.com", a member of the "cubicle-dwellers" and "underpaid" local
2095 groups and an "engineers" foreign group.

```
2096 <saml:Assertion xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
2097 ...>  
2098   <saml:Issuer>...</saml:Issuer>  
2099   <saml:Subject>...</saml:Subject>  
2100   <saml:AttributeStatement>  
2101     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
2102       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm">  
2103       <saml:AttributeValue xsi:type="dce:DCEValueType"  
2104 dce:FriendlyName="example.com">  
2105         urn:uuid:003c6cc1-9ff8-10f9-990f-004005b13a2b  
2106       </saml:AttributeValue>  
2107     </saml:Attribute>  
2108     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
2109       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal">  
2110       <saml:AttributeValue xsi:type="dce:DCEValueType" dce:FriendlyName="jdoe">  
2111         urn:uuid:00305ed1-a1bd-10f9-a2d0-004005b13a2b  
2112       </saml:AttributeValue>  
2113     </saml:Attribute>  
2114     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
2115       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group">  
2116       <saml:AttributeValue xsi:type="dce:DCEValueType"  
2117         dce:FriendlyName="cubicle-dwellers">  
2118         urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b  
2119       </saml:AttributeValue>  
2120     </saml:Attribute>  
2121     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
2122       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups">  
2123       <saml:AttributeValue xsi:type="dce:DCEValueType"  
2124         dce:FriendlyName="cubicle-dwellers">  
2125         urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b  
2126       </saml:AttributeValue>  
2127       <saml:AttributeValue xsi:type="dce:DCEValueType"  
2128 dce:FriendlyName="underpaid">  
2129         urn:uuid:006a5a91-a2b7-10f9-824d-004005b13a2b  
2130       </saml:AttributeValue>  
2131     </saml:Attribute>  
2132     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
2133       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-  
2134 groups">  
2135       <saml:AttributeValue xsi:type="dce:DCEValueType"  
2136 dce:FriendlyName="engineers"  
2137         dce:Realm="urn:uuid:00583221-a35f-10f9-8b6e-004005b13a2b">  
2138         urn:uuid:00099cf1-a355-10f9-9e95-004005b13a2b  
2139       </saml:AttributeValue>  
2140     </saml:Attribute>  
2141   </saml:AttributeStatement>  
2142 </saml:Assertion>
```

2143 8.5 XACML Attribute Profile

2144 SAML attribute assertions may be used as input to authorization decisions made according to the OASIS
2145 eXtensible Access Control Markup Language [XACML] standard specification. Since the SAML attribute
2146 format differs from the XACML attribute format, there is a mapping that must be performed. The XACML
2147 attribute profile facilitates this mapping by standardizing naming, value syntax, and additional attribute
2148 metadata. SAML attributes generated in conformance with this profile can be mapped automatically into
2149 XACML attributes and used as input to XACML authorization decisions.

2150 8.5.1 Required Information

2151 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML (this is also the target namespace
2152 assigned in the corresponding XACML profile schema document [SAMLXAC-xsd])

2153 **Contact information:** security-services-comment@lists.oasis-open.org

2154 **Description:** Given below.

2155 **Updates:** None.

2156 8.5.2 SAML Attribute Naming

2157 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
2158 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

2159 The `Name` XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

2160 For purposes of human readability, there may also be a requirement for some applications to carry an
2161 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
2162 [SAMLCore]) MAY be used for this purpose, but is not translatable into an XACML attribute equivalent.

2163 8.5.2.1 Attribute Name Comparison

2164 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
2165 values are equal in a binary comparison. The `FriendlyName` attribute plays no role in the comparison.

2166 8.5.3 Profile-Specific XML Attributes

2167 XACML requires each attribute to carry an explicit data type. To supply this data type value, a new URI-
2168 valued XML attribute called `DataType` is defined in the XML namespace
2169 `urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML`.

2170 SAML `<Attribute>` elements conforming to this profile MUST include the namespace-qualified
2171 `DataType` attribute, or the value is presumed to be <http://www.w3.org/2001/XMLSchema#string>.

2172 While in principle any URI reference can be used as a data type, the standard values to be used are
2173 specified in Appendix A of the XACML 2.0 Specification [XACML]. If non-standard values are used, then
2174 each XACML PDP that will be consuming mapped SAML attributes with non-standard `DataType` values
2175 must be extended to support the new data types.

2176 8.5.4 SAML Attribute Values

2177 The syntax of the `<AttributeValue>` element's content MUST correspond to the data type expressed
2178 in the profile-specific `DataType` XML attribute appearing in the parent `<Attribute>` element. For data
2179 types corresponding to the types defined in Section 3.3 of [Schema2], the `xsi:type` XML attribute
2180 SHOULD also be used on the `<AttributeValue>` element(s).

2181 8.5.5 Profile-Specific Schema

2182 The following schema listing shows how the profile-specific `DataType` XML attribute is defined
2183 [SAMLXAC-xsd]:

```
2184 <schema
2185   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
2186   xmlns="http://www.w3.org/2001/XMLSchema"
2187   elementFormDefault="unqualified"
2188   attributeFormDefault="unqualified"
2189   blockDefault="substitution"
2190   version="2.0">
2191   <annotation>
2192     <documentation>
2193       Document identifier: saml-schema-xacml-2.0
2194       Location: http://docs.oasis-open.org/security/saml/v2.0/
2195       Revision history:
2196       V2.0 (March, 2005):
2197         Custom schema for XACML attribute profile, first published in
2198 SAML 2.0.
2199     </documentation>
2200   </annotation>
2201   <attribute name="DataType" type="anyURI"/>
2202 </schema>
```

2203 8.5.6 Example

2204 The following is an example of a mapping of the "givenName" LDAP/X.500 attribute, representing the
2205 SAML assertion subject's first name. It also illustrates that a single SAML attribute can conform to multiple
2206 attribute profiles when they are compatible with each other.

```
2207 <saml:Attribute
2208   xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
2209   xmlns:ldapprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP"
2210   xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string"
2211   ldapprof:Encoding="LDAP"
2212   NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
2213   Name="urn:oid:2.5.4.42" FriendlyName="givenName">
2214   <saml:AttributeValue xsi:type="xs:string"
2215     ldapprof:Encoding="LDAP">By-Tor</saml:AttributeValue>
2216 </saml:Attribute>
```

2217

9 References

- 2218 **[AES]** FIPS-197, Advanced Encryption Standard (AES). See <http://www.nist.gov/>.
- 2219 **[Anders]** A suggestion on how to implement SAML browser bindings without using “Artifacts”.
2220 See <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- 2221 **[ASN.1]** Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic
2222 notation, ITU-T Recommendation X.680, July 2002. See
2223 [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)
2224 [X.680](http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.680).
- 2225 **[eduPerson]** eduPerson.Idif. See <http://www.educause.edu/eduperson>.
- 2226 **[LDAP]** J. Hodges et al. *Lightweight Directory Access Protocol (v3): Technical Specification*.
2227 IETF RFC 3377, September 2002. See <http://www.ietf.org/rfc/rfc3377.txt>.
- 2228 ~~**[E70][Mealling]** P. Leach et al. *A UUID-URN Namespace*. IETF Internet Draft, December 2004. See~~
2229 ~~<http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-05.txt>.~~
- 2230 **[MSURL]** Microsoft technical support article. See
2231 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- 2232 **[NSCookie]** Persistent Client State HTTP Cookies, Netscape documentation. See
2233 http://wp.netscape.com/newsref/std/cookie_spec.html.
- 2234 **[PAOS]** R. Aarts. *Liberty Reverse HTTP Binding for SOAP Specification Version 1.0*. Liberty
2235 Alliance Project, 2003. See <https://www.projectliberty.org/specs/liberty-paos-v1.0.pdf>.
- 2236 **[Rescorla-Sec]** E. Rescorla et al. *Guidelines for Writing RFC Text on Security Considerations*. IETF
2237 RFC 3552, July 2003. See <http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt>.
- 2238 **[RFC1738]** T. Berners-Lee et al. *Uniform Resource Locators (URL)*. IETF RFC 1738, December
2239 1994. See <http://www.ietf.org/rfc/rfc1738.txt>.
- 2240 **[RFC1750]** D. Eastlake et al. *Randomness Recommendations for Security*. IETF RFC 1750,
2241 December 1994. See <http://www.ietf.org/rfc/rfc1750.txt>.
- 2242 **[RFC1945]** T. Berners-Lee et al. *Hypertext Transfer Protocol – HTTP/1.0*. IETF RFC 1945, May
2243 1996. See <http://www.ietf.org/rfc/rfc1945.txt>.
- 2244 **[RFC2045]** N. Freed et al. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of*
2245 *Internet Message Bodies*. IETF RFC 2045, November 1996. See
2246 <http://www.ietf.org/rfc/rfc2045.txt>.
- 2247 **[RFC2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC
2248 2119, March 1997. See <http://www.ietf.org/rfc/rfc2119.txt>.
- 2249 **[RFC2246]** T. Dierks. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. See
2250 <http://www.ietf.org/rfc/rfc2246.txt>.
- 2251 **[RFC2256]** M. Wahl. *A Summary of the X.500(96) User Schema for use with LDAPv3*. IETF RFC
2252 2256, December 1997. See <http://www.ietf.org/rfc/rfc2256.txt>.
- 2253 **[RFC2279]** F. Yergeau. *UTF-8, a transformation format of ISO 10646*. IETF RFC 2279, January
2254 1998. See <http://www.ietf.org/rfc/rfc2279.txt>.
- 2255 **[RFC2616]** R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. IETF RFC 2616, June 1999.
2256 See <http://www.ietf.org/rfc/rfc2616.txt>.
- 2257 **[RFC2617]** J. Franks et al. *HTTP Authentication: Basic and Digest Access Authentication*. IETF
2258 RFC 2617, June 1999. See <http://www.ietf.org/rfc/rfc2617.txt>.
- 2259 **[RFC2798]** M. Smith. *Definition of the inetOrgPerson LDAP Object Class*. IETF RFC 2798, April
2260 2000. See <http://www.ietf.org/rfc/rfc2798.txt>.
- 2261 **[RFC2965]** D. Cristol et al. *HTTP State Management Mechanism*. IETF RFC 2965, October 2000.
2262 See <http://www.ietf.org/rfc/rfc2965.txt>.

2263	[RFC3061]	M. Mealling. <i>A URN Namespace of Object Identifiers</i> . IETF RFC 3061, February 2001. See http://www.ietf.org/rfc/rfc3061.txt .
2264		
2265	[E70][RFC4122]	P. Leach et al. <i>A Universally Unique Identifier (UUID) URN Namespace</i> . IETF RFC 4122, July 2005. See http://www.ietf.org/rfc/rfc4122.txt .
2266		
2267	[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/ .
2268		
2269		
2270	[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/ .
2271		
2272		
2273	[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/ .
2274		
2275		
2276	[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/ .
2277		
2278		
2279	[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/ .
2280		
2281	[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/ .
2282		
2283		
2284	[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/ .
2285		
2286		
2287	[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/ .
2288		
2289		
2290	[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/ .
2291		
2292		
2293	[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/ .
2294		
2295		
2296	[SAML Web]	OASIS Security Services Technical Committee website, http://www.oasis-open.org/committees/security .
2297		
2298	[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/ .
2299		
2300		
2301	[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.
2302		
2303		
2304	[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/ .
2305		
2306	[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 .
2307		
2308		
2309	[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 .
2310		
2311	[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 .
2312		
2313	[SSL3]	A. Frier et al. <i>The SSL 3.0 Protocol</i> . Netscape Communications Corp, November 1996.

2314	[WEBSO]	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-morgan-shibboleth-websso-00.txt .
2315		
2316		
2317	[X.500]	Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services. ITU-T Recommendation X.500, February 2001. See http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.500 .
2318		
2319		
2320		
2321	[XMLEnc]	D. Eastlake et al. <i>XML Encryption Syntax and Processing</i> . World Wide Web Consortium Recommendation, December 2002. See http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/ .
2322		
2323		
2324	[XMLSig]	D. Eastlake et al. <i>XML-Signature Syntax and Processing</i> . [E74] <i>Second Edition</i> . World Wide Web Consortium Recommendation, June 2008 <i>February 2002</i> . See http://www.w3.org/TR/xmlsig-core/ .
2325		
2326		
2327	[XACML]	T. Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Versions 1.0, 1.1, and 2.0. Available on the OASIS XACML TC web page at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml .
2328		
2329		

Appendix A. Acknowledgments

2331 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
2332 Committee, whose voting members at the time of publication were:

- 2333 • Conor Cahill, AOL
- 2334 • John Hughes, Atos Origin
- 2335 • Hal Lockhart, BEA Systems
- 2336 • Mike Beach, Boeing
- 2337 • Rebekah Metz, Booz Allen Hamilton
- 2338 • Rick Randall, Booz Allen Hamilton
- 2339 • Ronald Jacobson, Computer Associates
- 2340 • Gavenraj Sodhi, Computer Associates
- 2341 • Thomas Wisniewski, Entrust
- 2342 • Carolina Canales-Valenzuela, Ericsson
- 2343 • Dana Kaufman, Forum Systems
- 2344 • Irving Reid, Hewlett-Packard
- 2345 • Guy Denton, IBM
- 2346 • Heather Hinton, IBM
- 2347 • Maryann Hondo, IBM
- 2348 • Michael McIntosh, IBM
- 2349 • Anthony Nadalin, IBM
- 2350 • Nick Ragouzis, Individual
- 2351 • Scott Cantor, Internet2
- 2352 • Bob Morgan, Internet2
- 2353 • Peter Davis, Neustar
- 2354 • Jeff Hodges, Neustar
- 2355 • Frederick Hirsch, Nokia
- 2356 • Senthil Sengodan, Nokia
- 2357 • Abbie Barbir, Nortel Networks
- 2358 • Scott Kiestler, Novell
- 2359 • Cameron Morris, Novell
- 2360 • Paul Madsen, NTT
- 2361 • Steve Anderson, OpenNetwork
- 2362 • Ari Kermaier, Oracle
- 2363 • Vamsi Motukuru, Oracle
- 2364 • Darren Platt, Ping Identity
- 2365 • Prateek Mishra, Principal Identity
- 2366 • Jim Lien, RSA Security
- 2367 • John Linn, RSA Security
- 2368 • Rob Philpott, RSA Security
- 2369 • Dipak Chopra, SAP
- 2370 • Jahan Moreh, Sigaba
- 2371 • Bhavna Bhatnagar, Sun Microsystems
- 2372 • Eve Maler, Sun Microsystems

- 2373 • Ronald Monzillo, Sun Microsystems
- 2374 • Emily Xu, Sun Microsystems
- 2375 • Greg Whitehead, Trustgenix

2376 The editors also would like to acknowledge the following former SSTC members for their contributions to
2377 this or previous versions of the OASIS Security Assertions Markup Language Standard:

- 2378 • Stephen Farrell, Baltimore Technologies
- 2379 • David Orchard, BEA Systems
- 2380 • Krishna Sankar, Cisco Systems
- 2381 • Zahid Ahmed, CommerceOne
- 2382 • Tim Alsop, CyberSafe Limited
- 2383 • Carlisle Adams, Entrust
- 2384 • Tim Moses, Entrust
- 2385 • Nigel Edwards, Hewlett-Packard
- 2386 • Joe Pato, Hewlett-Packard
- 2387 • Bob Blakley, IBM
- 2388 • Marlena Erdos, IBM
- 2389 • Marc Chanliau, Netegrity
- 2390 • Chris McLaren, Netegrity
- 2391 • Lynne Rosenthal, NIST
- 2392 • Mark Skall, NIST
- 2393 • Charles Knouse, Oblix
- 2394 • Simon Godik, Overxeer
- 2395 • Charles Norwood, SAIC
- 2396 • Evan Prodromou, Securant
- 2397 • Robert Griffin, RSA Security (former editor)
- 2398 • Sai Allarvarpu, Sun Microsystems
- 2399 • Gary Ellison, Sun Microsystems
- 2400 • Chris Ferris, Sun Microsystems
- 2401 • Mike Myers, Traceroute Security
- 2402 • Phillip Hallam-Baker, VeriSign (former editor)
- 2403 • James Vanderbeek, Vodafone
- 2404 • Mark O'Neill, Vordel
- 2405 • Tony Palmer, Vordel

2406 Finally, the editors wish to acknowledge the following people for their contributions of material used as
2407 input to the OASIS Security Assertions Markup Language specifications:

- 2408 • Thomas Gross, IBM
- 2409 • Birgit Pfitzmann, IBM

2410 [The editors also would like to gratefully acknowledge Jahan Moreh of Sigaba, who during his tenure on](#)
2411 [the SSTC was the primary editor of the errata working document and who made major substantive](#)
2412 [contributions to all of the errata materials.](#)

2413 **Appendix B. Notices**

2414 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
2415 might be claimed to pertain to the implementation or use of the technology described in this document or
2416 the extent to which any license under such rights might or might not be available; neither does it represent
2417 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
2418 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
2419 available for publication and any assurances of licenses to be made available, or the result of an attempt
2420 made to obtain a general license or permission for the use of such proprietary rights by implementors or
2421 users of this specification, can be obtained from the OASIS Executive Director.

2422 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
2423 other proprietary rights which may cover technology that may be required to implement this specification.
2424 Please address the information to the OASIS Executive Director.

2425 **Copyright © OASIS Open 2005. All Rights Reserved.**

2426 This document and translations of it may be copied and furnished to others, and derivative works that
2427 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
2428 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
2429 this paragraph are included on all such copies and derivative works. However, this document itself may
2430 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
2431 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
2432 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
2433 into languages other than English.

2434 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
2435 or assigns.

2436 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2437 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
2438 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
2439 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.