



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

Committee Draft 03, 14 December 2004

Document identifier:

sstc-saml-profiles-2.0-cd-03

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

SAML V2.0 Contributors:

Conor P. Cahill, AOL
John Hughes, Atos Origin
Hal Lockhart, BEA Systems
Michael Beach, Boeing
Rebekah Metz, Booz Allen Hamilton
Rick Randall, Booz, Allen, Hamilton
Tim Alsop, CyberSafe Limited
Paul Madsen, Entrust
Irving Reid, Hewlett-Packard
Paula Austel, IBM
Maryann Hondo, IBM
Michael McIntosh, IBM
Tony Nadalin, IBM
Nick Ragouzis, Individual
Scott Cantor, Internet2
RL 'Bob' Morgan, Internet2
Peter C Davis, Neustar
Jeff Hodges, Neustar
Frederick Hirsch, Nokia
John Kemp, Nokia
Charles Knouse, Oblix
Steve Anderson, OpenNetwork
Prateek Mishra, Principal Identity
John Linn, RSA Security
Rob Philpott, RSA Security
Jahan Moreh, Sigaba

45 Anne Anderson, Sun Microsystems
46 Gary Ellison, Sun Microsystems
47 Eve Maler, Sun Microsystems
48 Ron Monzillo, Sun Microsystems
49 Greg Whitehead, Trustgenix

50 **Abstract:**

51 This specification defines profiles for the use of SAML assertions and request-response
52 messages in communications protocols and frameworks, as well as profiles for SAML attribute
53 value syntax and naming conventions.

54 **Status:**

55 This is a **Committee Draft** approved by the Security Services Technical Committee on 14
56 December 2004.

57 Committee members should submit comments and potential errata to the [security-](mailto:security-services@lists.oasis-open.org)
58 [services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should submit them by filling out the web form located
59 at http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security. The
60 committee will publish on its web page (<http://www.oasis-open.org/committees/security>) a catalog
61 of any changes made to this document.

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

Table of Contents

66		
67	1 Introduction.....	7
68	1.1 Profile Concepts.....	7
69	1.2 Notation.....	7
70	2 Specification of Additional Profiles.....	10
71	2.1 Guidelines for Specifying Profiles.....	10
72	2.2 Guidelines for Specifying Attribute Profiles.....	10
73	3 Confirmation Method Identifiers.....	12
74	3.1 Holder of Key.....	12
75	3.2 Sender Vouches.....	12
76	3.3 Bearer.....	13
77	4 SSO Profiles of SAML.....	14
78	4.1 Web Browser SSO Profile.....	14
79	4.1.1 Required Information.....	14
80	4.1.2 Profile Overview.....	14
81	4.1.3 Profile Description.....	16
82	4.1.3.1 HTTP Request to Service Provider.....	16
83	4.1.3.2 Service Provider Determines Identity Provider.....	16
84	4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider.....	16
85	4.1.3.4 Identity Provider Identifies Principal.....	17
86	4.1.3.5 Identity Provider Issues <Response> to Service Provider.....	17
87	4.1.3.6 Service Provider Grants or Denies Access to User Agent.....	17
88	4.1.4 Use of Authentication Request Protocol.....	18
89	4.1.4.1 <AuthnRequest> Usage.....	18
90	4.1.4.2 <Response> Usage.....	18
91	4.1.4.3 <Response> Message Processing Rules.....	19
92	4.1.4.4 Artifact-Specific <Response> Message Processing Rules.....	19
93	4.1.4.5 POST-Specific Processing Rules.....	20
94	4.1.5 Unsolicited Responses.....	20
95	4.1.6 Use of Metadata.....	20
96	4.2 Enhanced Client or Proxy (ECP) Profile.....	21
97	4.2.1 Required Information.....	21
98	4.2.2 Profile Overview.....	21
99	4.2.3 Profile Description.....	24
100	4.2.3.1 ECP issues HTTP Request to Service Provider.....	24
101	4.2.3.2 Service Provider Issues <AuthnRequest> to ECP.....	25
102	4.2.3.3 ECP Determines Identity Provider.....	25
103	4.2.3.4 ECP issues <AuthnRequest> to Identity Provider.....	25
104	4.2.3.5 Identity Provider Identifies Principal.....	25
105	4.2.3.6 Identity Provider issues <Response> to ECP, targeted at service provider.....	26
106	4.2.3.7 ECP Conveys <Response> Message to Service Provider.....	26
107	4.2.3.8 Service Provider Grants or Denies Access to Principal.....	26
108	4.2.4 ECP Profile Schema Usage.....	26
109	4.2.4.1 PAOS Request Header Block: SP to ECP.....	27

110	4.2.4.2 ECP Request Header Block: SP to ECP.....	28
111	4.2.4.3 ECP RelayState Header Block: SP to ECP.....	28
112	4.2.4.4 ECP Response Header Block: IdP to ECP.....	29
113	4.2.4.5 PAOS Response Header Block: ECP to SP.....	30
114	4.2.5 Security Considerations.....	31
115	4.3 Identity Provider Discovery Profile.....	31
116	4.3.1 Common Domain Cookie.....	31
117	4.3.2 Setting the Common Domain Cookie.....	32
118	4.3.3 Obtaining the Common Domain Cookie.....	32
119	4.4 Single Logout Profile.....	32
120	4.4.1 Required Information.....	33
121	4.4.2 Profile Overview.....	33
122	4.4.3 Profile Description.....	34
123	4.4.3.1 <LogoutRequest> Issued by Session Participant to Identity Provider.....	35
124	4.4.3.2 Identity Provider Determines Session Participants.....	35
125	4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session Participant/Authority.....	35
126	4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity Provider.....	36
127	4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant.....	36
128	4.4.4 Use of Single Logout Protocol.....	37
129	4.4.4.1 <LogoutRequest> Usage.....	37
130	4.4.4.2 <LogoutResponse> Usage.....	37
131	4.4.5 Use of Metadata.....	37
132	4.5 Name Identifier Management Profile.....	37
133	4.5.1 Required Information.....	38
134	4.5.2 Profile Overview.....	38
135	4.5.3 Profile Description.....	38
136	4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider.....	39
137	4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service Provider.....	39
138	4.5.4 Use of Name Identifier Management Protocol.....	40
139	4.5.4.1 <ManageNameIDRequest> Usage.....	40
140	4.5.4.2 <ManageNameIDResponse> Usage.....	40
141	4.5.5 Use of Metadata.....	40
142	5 Artifact Resolution Profile.....	41
143	5.1 Required Information.....	41
144	5.2 Profile Overview.....	41
145	5.3 Profile Description.....	42
146	5.3.1 <ArtifactResolve> issued by Requesting Entity.....	42
147	5.3.2 <ArtifactResponse> issued by Responding Entity.....	42
148	5.4 Use of Artifact Resolution Protocol.....	42
149	5.4.1 <ArtifactResolve> Usage.....	42
150	5.4.2 <ArtifactResponse> Usage.....	43
151	5.5 Use of Metadata.....	43
152	6 Assertion Query/Request Profile.....	44
153	6.1 Required Information.....	44
154	6.2 Profile Overview.....	44
155	6.3 Profile Description.....	45

156	6.3.1 Query/Request issued by SAML Requester.....	45
157	6.3.2 <Response> issued by SAML Authority.....	45
158	6.4 Use of Query/Request Protocol.....	45
159	6.4.1 Query/Request Usage.....	45
160	6.4.2 <Response> Usage.....	45
161	6.5 Use of Metadata.....	46
162	7 Name Identifier Mapping Profile.....	47
163	7.1 Required Information.....	47
164	7.2 Profile Overview.....	47
165	7.3 Profile Description.....	48
166	7.3.1 <NameIDMappingRequest> issued by Requesting Entity.....	48
167	7.3.2 <NameIDMappingResponse> issued by Identity Provider.....	48
168	7.4 Use of Name Identifier Mapping Protocol.....	48
169	7.4.1 <NameIDMappingRequest> Usage.....	48
170	7.4.2 <NameIDMappingResponse> Usage.....	48
171	7.4.2.1 Limiting Use of Mapped Identifier.....	49
172	7.5 Use of Metadata.....	49
173	8 SAML Attribute Profiles.....	50
174	8.1 Basic Attribute Profile.....	50
175	8.1.1 Required Information.....	50
176	8.1.2 SAML Attribute Naming.....	50
177	8.1.2.1 Attribute Name Comparison.....	50
178	8.1.3 Profile-Specific XML Attributes.....	50
179	8.1.4 SAML Attribute Values.....	50
180	8.1.5 Example.....	50
181	8.2 X.500/LDAP Attribute Profile.....	50
182	8.2.1 Required Information.....	51
183	8.2.2 SAML Attribute Naming.....	51
184	8.2.2.1 Attribute Name Comparison.....	51
185	8.2.3 Profile-Specific XML Attributes.....	51
186	8.2.4 SAML Attribute Values.....	51
187	8.2.5 Profile-Specific Schema.....	52
188	8.2.6 Example.....	53
189	8.3 UUID Attribute Profile.....	53
190	8.3.1 Required Information.....	53
191	8.3.2 UUID and GUID Background.....	53
192	8.3.3 SAML Attribute Naming.....	54
193	8.3.3.1 Attribute Name Comparison.....	54
194	8.3.4 Profile-Specific XML Attributes.....	54
195	8.3.5 SAML Attribute Values.....	54
196	8.3.6 Example.....	54
197	8.4 DCE PAC Attribute Profile.....	55
198	8.4.1 Required Information.....	55
199	8.4.2 PAC Description.....	55

200	8.4.3 SAML Attribute Naming.....	55
201	8.4.3.1 Attribute Name Comparison.....	55
202	8.4.4 Profile-Specific XML Attributes.....	55
203	8.4.5 SAML Attribute Values.....	56
204	8.4.6 Attribute Definitions.....	56
205	8.4.6.1 Realm.....	56
206	8.4.6.2 Principal.....	57
207	8.4.6.3 Primary Group.....	57
208	8.4.6.4 Groups.....	57
209	8.4.6.5 Foreign Groups.....	57
210	8.4.7 Example.....	58
211	8.5 XACML Attribute Profile.....	58
212	8.5.1 Required Information.....	59
213	8.5.2 SAML Attribute Naming.....	59
214	8.5.2.1 Attribute Name Comparison.....	59
215	8.5.3 Profile-Specific XML Attributes.....	59
216	8.5.4 SAML Attribute Values.....	59
217	8.5.5 Profile-Specific Schema.....	59
218	8.5.6 Example.....	60
219	9 References.....	61
220	Appendix A. Acknowledgments.....	64
221	Appendix B. Notices.....	66

222 1 Introduction

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

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

230 1.1 Profile Concepts

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

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

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

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

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

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

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

256 1.2 Notation

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

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

264 described in IETF RFC 2119 [RFC2119].

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

266 Example code listings appear like this.

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

268 Conventional XML namespace prefixes are used throughout this specification to stand for their respective
269 namespaces as follows, whether or not a namespace declaration is present in the example:

Prefix	XML Namespace	Comments
saml:	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace [SAMLCore]. The prefix is generally elided in mentions of SAML assertion-related elements in text.
samlp:	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace [SAMLCore]. The prefix is generally elided in mentions of XML protocol-related elements in text.
md:	urn:oasis:names:tc:SAML:2.0:metadata	This is the SAML V2.0 metadata namespace [SAMLMeta].
ecp:	urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp	This is the SAML V2.0 ECP profile namespace, specified in this document and in a schema [SAMLECP-xsd].
ds:	http://www.w3.org/2000/09/xmldsig#	This is the XML Signature namespace [XMLSig].
xenc:	http://www.w3.org/2001/04/xmlenc#	This is the XML Encryption namespace [XMLEnc].
SOAP-ENV:	http://schemas.xmlsoap.org/soap/envelope	This is the SOAP V1.1 namespace [SOAP1.1].
paos:	urn:liberty:paos:2003-08	This is the Liberty Alliance PAOS namespace.
dce:	urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE	This is the SAML V2.0 DCE PAC attribute profile namespace, specified in this document and in a schema [SAMLDCExsd].
x500:	urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500	This is the SAML V2.0 X.500/LDAP attribute profile namespace, specified in this document and in a schema [SAMLX500-xsd].
xacmlprof:	urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML	This is the SAML V2.0 XACML attribute profile namespace, specified in this document and in a schema [SAMLXAC-xsd].
xsi:	http://www.w3.org/2001/XMLSchema-instance	This namespace is defined in the W3C XML Schema specification [Schema1] for schema-related markup that appears in XML instances.

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

273

2 Specification of Additional Profiles

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

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

2.1 Guidelines for Specifying Profiles

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

- 284 1. Specify a URI that uniquely identifies the profile, postal or electronic contact information for the
285 author, and provide reference to previously defined profiles that the new profile updates or
286 obsoletes.
- 287 2. Describe the set of interactions between parties involved in the profile. Any restrictions on
288 applications used by each party and the protocols involved in each interaction must be explicitly
289 called out.
- 290 3. Identify the parties involved in each interaction, including how many parties are involved and
291 whether intermediaries may be involved.
- 292 4. Specify the method of authentication of parties involved in each interaction, including whether
293 authentication is required and acceptable authentication types.
- 294 5. Identify the level of support for message integrity, including the mechanisms used to ensure
295 message integrity.
- 296 6. Identify the level of support for confidentiality, including whether a third party may view the contents
297 of SAML messages and assertions, whether the profile requires confidentiality, and the
298 mechanisms recommended for achieving confidentiality.
- 299 7. Identify the error states, including the error states at each participant, especially those that receive
300 and process SAML assertions or messages.
- 301 8. Identify security considerations, including analysis of threats and description of countermeasures.
- 302 9. Identify SAML confirmation method identifiers defined and/or utilized by the profile.
- 303 10. Identify relevant SAML metadata defined and/or utilized by the profile.

2.2 Guidelines for Specifying Attribute Profiles

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

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

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

317

3 Confirmation Method Identifiers

318 The SAML assertion and protocol specification [SAMLCore] defines the `<SubjectConfirmation>`
319 element as a `Method` plus optional `<SubjectConfirmationData>`. The `<SubjectConfirmation>`
320 element SHOULD be used by the relying party to confirm that the request or message came from a
321 system entity that corresponds to the subject of the assertion, within the context of a particular profile.

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

328 It is anticipated that profiles will define and use several different values for `<ConfirmationMethod>`,
329 each corresponding to a different SAML usage scenario. The following methods are defined for use by
330 profiles defined within this specification and other profiles that find them useful.

3.1 Holder of Key

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

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

337 As described in [XMLSig], each `<ds:KeyInfo>` element holds a key or information that enables an
338 application to obtain a key. The holder of a specified key is considered to be the subject of the assertion
339 by the asserting party.

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

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

```
345 <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">  
346   <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">  
347     <ds:KeyInfo>  
348       <ds:KeyName>By-Tor</ds:KeyName>  
349     </ds:KeyInfo>  
350     <ds:KeyInfo>  
351       <ds:KeyName>Snow Dog</ds:KeyName>  
352     </ds:KeyInfo>  
353   </SubjectConfirmationData>  
354 </SubjectConfirmation>
```

3.2 Sender Vouches

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

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

361 **3.3 Bearer**

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

363 The subject of the assertion is the bearer of the assertion, subject to optional constraints on confirmation
364 using the attributes that MAY be present in the <SubjectConfirmationData> element, as defined by
365 [SAMLCore].

366 **Example:** The bearer of the assertion can confirm itself as the subject, provided the assertion is delivered
367 in a message sent to "<https://www.serviceprovider.com/saml/consumer>" before 1:37 PM GMT on March
368 19th, 2004, in response to a request with ID "_1234567890".

```
369 <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">  
370   <SubjectConfirmationData InResponseTo="_1234567890"  
371     Recipient="https://www.serviceprovider.com/saml/consumer"  
372     NotOnOrAfter="2004-03-19T13:27:00Z"  
373   </SubjectConfirmationData>  
374 </SubjectConfirmation>
```

375 4 SSO Profiles of SAML

- 376 A set of profiles is defined to support single sign-on (SSO) of browsers and other client devices.
- 377 • A web browser-based profile of the Authentication Request protocol in [SAMLCore] is defined to
378 support web single sign-on, supporting Scenario 1-1 of the original SAML requirements document .
 - 379 • An additional web SSO profile is defined to support enhanced clients.
 - 380 • A profile of the Single Logout and Name Identifier Management protocols in [SAMLCore] is defined
381 over both front-channel (browser) and back-channel bindings.
 - 382 • An additional profile is defined for identity provider discovery using cookies.

383 4.1 Web Browser SSO Profile

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

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

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

395 4.1.1 Required Information

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

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

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

400 **Description:** Given below.

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

402 4.1.2 Profile Overview

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

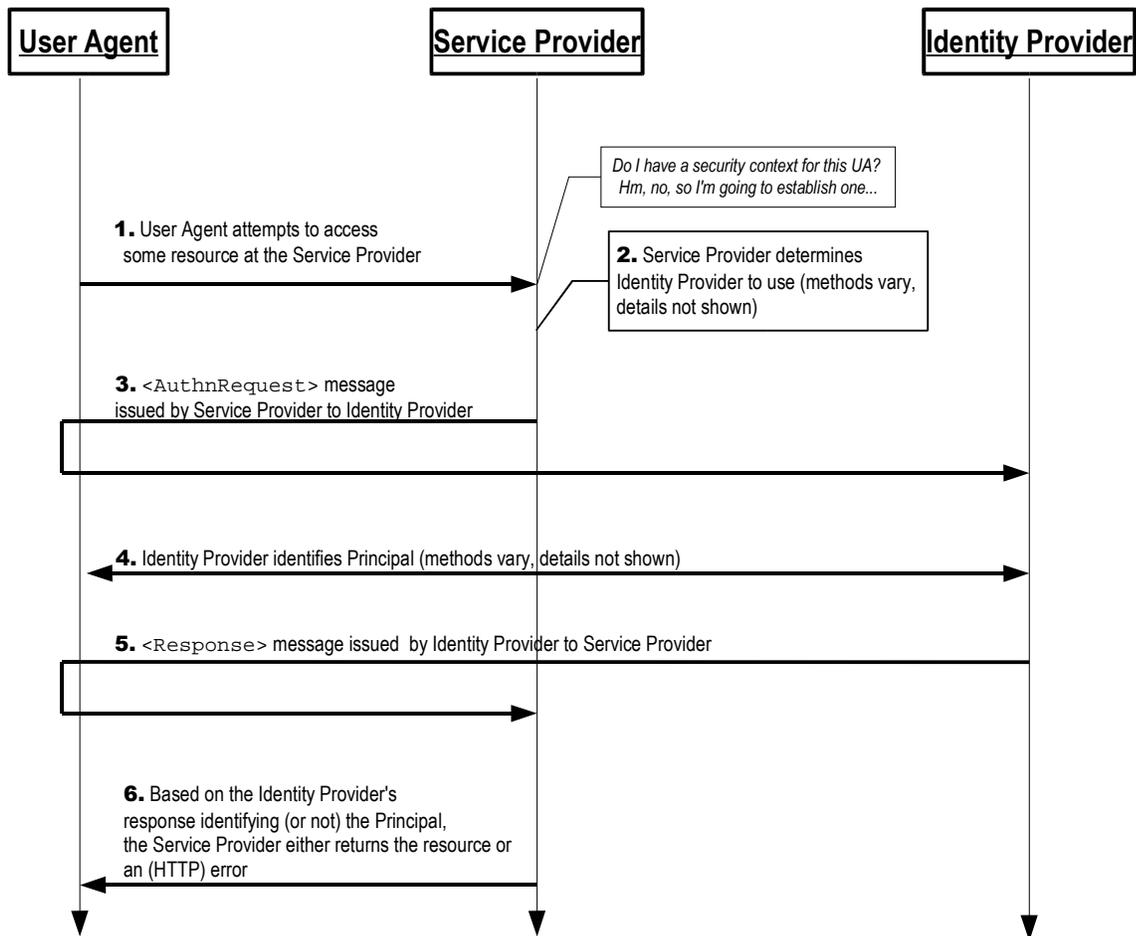


Figure 1

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

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

409 **2. Service Provider Determines Identity Provider**

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

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

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

418 **4. Identity Provider identifies Principal**

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

422 5. Identity Provider issues <Response> to Service Provider

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

428 6. Service Provider grants or denies access to Principal

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

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

434 4.1.3 Profile Description

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

437 Single Sign-On Service

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

440 Assertion Consumer Service

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

443 4.1.3.1 HTTP Request to Service Provider

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

450 4.1.3.2 Service Provider Determines Identity Provider

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

457 4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider

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

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

470 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or TLS
471 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <AuthnRequest> message MAY
472 be signed, if authentication of the request issuer is required. The HTTP Artifact binding, if used, also
473 provides for an alternate means of authenticating the request issuer when the artifact is dereferenced.

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

477 **4.1.3.4 Identity Provider Identifies Principal**

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

485 **4.1.3.5 Identity Provider Issues <Response> to Service Provider**

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

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

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

499 It is RECOMMENDED that the HTTP requests in this step be made over either SSL 3.0 ([SSL3]) or TLS
500 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <Assertion> element(s) in the
501 <Response> MUST be signed, if the HTTP POST binding is used, and MAY be signed if the HTTP-
502 Artifact binding is used.

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

505 **4.1.3.6 Service Provider Grants or Denies Access to User Agent**

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

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

511 **4.1.4 Use of Authentication Request Protocol**

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

516 **4.1.4.1 <AuthnRequest> Usage**

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

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

523 If the service provider wishes to permit the identity provider to establish a new identifier for the principal if
524 none exists, it MUST include a <NameIDPolicy> element with the AllowCreate attribute set to "true".
525 Otherwise, only a principal for whom the identity provider has previously established an identifier usable by
526 the service provider can be authenticated successfully.

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

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

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

539 **4.1.4.2 <Response> Usage**

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

- 543 • The <Issuer> element MAY be omitted, but if present it MUST contain the unique identifier of the
544 issuing identity provider; the Format attribute MUST be omitted or have a value of
545 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.
- 546 • It MUST contain at least one <Assertion>. Each assertion's <Issuer> element MUST contain the
547 unique identifier of the issuing identity provider; the Format attribute MUST be omitted or have a value
548 of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.
- 549 • The set of one or more assertions MUST contain at least one <AuthnStatement> that reflects the
550 authentication of the principal to the identity provider.

- 551 • At least one assertion containing an `<AuthnStatement>` MUST contain a `<Subject>` element with
552 at least one `<SubjectConfirmation>` element containing a `Method` of
553 `urn:oasis:names:tc:SAML:2.0:cm:bearer`. If the identity provider supports the Single Logout
554 profile, defined in Section 4.4, any such authentication statements MUST include a `SessionIndex`
555 attribute to enable per-session logout requests by the service provider.
- 556 • Any bearer `<SubjectConfirmationData>` elements MUST contain a `Recipient` attribute
557 containing the service provider's assertion consumer service URL and a `NotOnOrAfter` attribute that
558 limits the window during which the assertion can be delivered. It MAY contain an `Address` attribute
559 limiting the client address from which the assertion can be delivered. It MUST NOT contain a
560 `NotBefore` attribute. If the containing message is in response to an `<AuthnRequest>`, then the
561 `InResponseTo` attribute MUST match the request's ID.
- 562 • Other statements and confirmation methods MAY be included in the assertion(s) at the discretion of
563 the identity provider. In particular, `<AttributeStatement>` elements MAY be included. The
564 `<AuthnRequest>` MAY contain an `AttributeConsumingServiceIndex` XML attribute
565 referencing information about desired or required attributes in [SAMLMeta]. The identity provider MAY
566 ignore this, or send other attributes at its discretion.
- 567 • The assertion(s) containing a bearer subject confirmation MUST contain an
568 `<AudienceRestriction>` including the service provider's unique identifier as an `<Audience>`.
- 569 • Other conditions (and other `<Audience>` elements) MAY be included as requested by the service
570 provider or at the discretion of the identity provider. (Of course, all such conditions MUST be
571 understood by and accepted by the service provider in order for the assertion to be considered valid.)
572 The identity provider is NOT obligated to honor the requested set of `<Conditions>` in the
573 `<AuthnRequest>`, if any.

574 4.1.4.3 `<Response>` Message Processing Rules

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

- 576 • Verify any signatures present on the assertion(s) or the response
- 577 • Verify that the `Recipient` attribute in any bearer `<SubjectConfirmationData>` matches the
578 assertion consumer service URL to which the `<Response>` or artifact was delivered
- 579 • Verify that the `NotOnOrAfter` attribute in any bearer `<SubjectConfirmationData>` has not
580 passed, subject to allowable clock skew between the providers
- 581 • Verify that the `InResponseTo` attribute in the bearer `<SubjectConfirmationData>` equals the ID
582 of its original `<AuthnRequest>` message, unless the response is unsolicited (see Section 4.5), in
583 which case the attribute MUST NOT be present
- 584 • Verify that any assertions relied upon are valid in other respects

585 If any bearer `<SubjectConfirmationData>` includes an `Address` attribute, the service provider MAY
586 check the user agent's client address against it.

587 Any assertion which is not valid, or whose subject confirmation requirements cannot be met SHOULD be
588 discarded and SHOULD NOT be used to establish a security context for the principal.

589 If an `<AuthnStatement>` used to establish a security context for the principal contains a
590 `SessionNotOnOrAfter` attribute, the security context SHOULD be discarded once this time is reached,
591 unless the service provider reestablishes the principal's identity by repeating the use of this profile.

592 4.1.4.4 Artifact-Specific `<Response>` Message Processing Rules

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

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

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

599 **4.1.4.5 POST-Specific Processing Rules**

600 If the HTTP POST binding is used to deliver the <Response>, the enclosed assertion(s) MUST be
601 signed.

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

605 **4.1.5 Unsolicited Responses**

606 An identity provider MAY initiate this profile by delivering an unsolicited <Response> message to a
607 service provider.

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

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

617 **4.1.6 Use of Metadata**

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

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

625 The providers MAY document the key(s) used to sign requests, responses, and assertions with
626 <md:KeyDescriptor> elements with a use attribute of sign. When encrypting SAML elements,
627 <md:KeyDescriptor> elements with a use attribute of encrypt MAY be used to document supported
628 encryption algorithms and settings, and public keys used to receive bulk encryption keys.

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

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

638 insufficient.)

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

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

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

651 4.2 Enhanced Client or Proxy (ECP) Profile

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

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

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

664 **Note:** The means by which a p[ri]ncipal authenticates with an identity provider is outside of the
665 scope of SAML.

666 4.2.1 Required Information

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

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

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

672 **Description:** Given below.

673 **Updates:** None.

674 4.2.2 Profile Overview

675 As introduced above, the ECP profile specifies interactions between enhanced clients or proxies and
676 service providers and identity providers. It is a specific application of the SSO profile described in Section
677 4.1. If not otherwise specified by this profile, and if not specific to the use of browser-based bindings, the
678 rules specified in Section 4.1 MUST be observed.

679 An ECP is a client or proxy that satisfies the following two conditions:
680 • It has, or knows how to obtain, information about the identity provider that the principal associated with
681 the ECP wishes to use, in the context of an interaction with a service provider.

682 This allows a service provider to make an authentication request to the ECP without the need to know
683 or discover the appropriate identity provider (effectively bypassing step 2 of the SSO profile in Section
684 4.1).

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

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

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

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

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

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

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

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

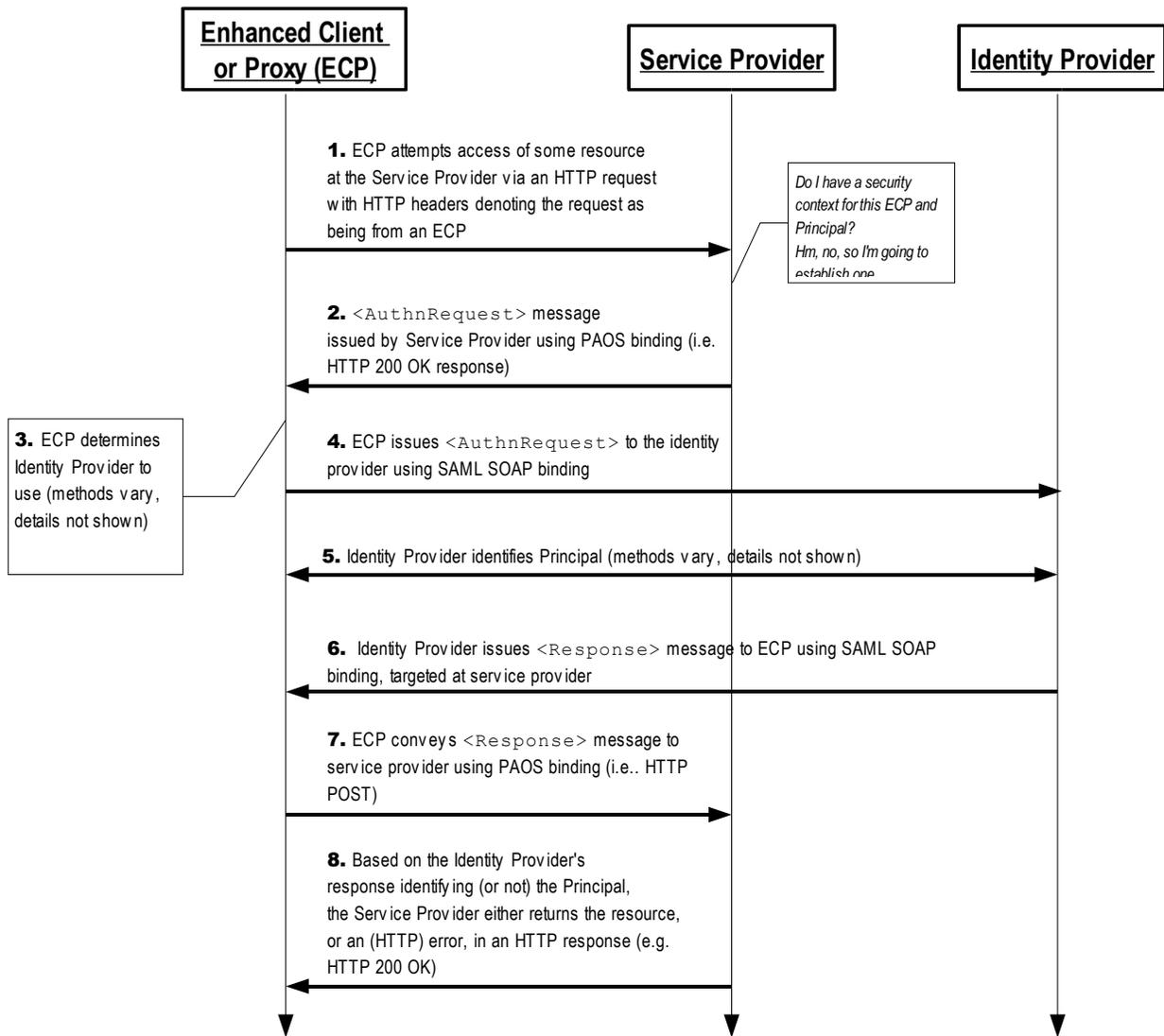


Figure 2

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

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

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

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

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

722 **3. ECP Determines Identity Provider**

723 In step 3, the ECP obtains the location of an endpoint at an identity provider for the authentication
724 request protocol that supports its preferred binding. The means by which this is accomplished is
725 implementation-dependent. The ECP MAY use the SAML identity provider discovery profile
726 described in Section 4.3.

727 **4. ECP conveys <AuthnRequest> to Identity Provider**

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

732 **5. Identity Provider identifies Principal**

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

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

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

740 **7. ECP conveys <Response> message to Service Provider**

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

743 **8. Service Provider grants or denies access to Principal**

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

747 **4.2.3 Profile Description**

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

749 **4.2.3.1 ECP issues HTTP Request to Service Provider**

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

- 752 1. The HTTP Accept Header field indicating the ability to accept the MIME type
753 "application/vnd.paos+xml"
- 754 2. The HTTP PAOS Header field specifying the PAOS version with urn:liberty:paos:2003-08 at
755 minimum.
- 756 3. Furthermore, support for this profile MUST be specified in the HTTP PAOS Header field as a service
757 value, with the value urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp. This value should
758 correspond to the service attribute in the PAOS Request SOAP header block

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

```
760 GET /index HTTP/1.1  
761 Host: identity-service.example.com  
762 Accept: text/html; application/vnd.paos+xml  
763 PAOS: ver='urn:liberty:paos:2003-08' ;  
764 'urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp'
```

765 4.2.3.2 Service Provider Issues <AuthnRequest> to ECP

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

770 The SOAP envelope MUST contain:

- 771 1. An <AuthnRequest> element in the SOAP body, intended for the ultimate SOAP recipient, the
772 identity provider.
- 773 2. A PAOS SOAP header block targeted at the ECP using the SOAP actor value of
774 `http://schemas.xmlsoap.org/soap/actor/next`. This header block provides control
775 information such as the URL to which to send the response in this solicit-response message
776 exchange pattern.
- 777 3. An ECP profile-specific Request SOAP header block targeted at the ECP using the SOAP actor
778 `http://schemas.xmlsoap.org/soap/actor/next`. The ECP Request header block defines
779 information related to the authentication request that the ECP may need to process it, such as a list
780 of identity providers acceptable to the service provider, whether the ECP may interact with the
781 principal through the client, and the service provider's human-readable name that may be displayed
782 to the principal.

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

786 4.2.3.3 ECP Determines Identity Provider

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

788 4.2.3.4 ECP issues <AuthnRequest> to Identity Provider

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

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

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

800 4.2.3.5 Identity Provider Identifies Principal

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

808 4.2.3.6 Identity Provider issues <Response> to ECP, targeted at service provider

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

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

816 4.2.3.7 ECP Conveys <Response> Message to Service Provider

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

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

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

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

834 4.2.3.8 Service Provider Grants or Denies Access to Principal

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

840 4.2.4 ECP Profile Schema Usage

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

```
843 <schema  
844   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"  
845   xmlns="http://www.w3.org/2001/XMLSchema"  
846   xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"  
847   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
848   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
849   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"  
850   elementFormDefault="unqualified"  
851   attributeFormDefault="unqualified"  
852   blockDefault="substitution"  
853   version="2.0">
```

```

854 <import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
855       schemaLocation="sstc-saml-schema-protocol-2.0.xsd"/>
856 <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
857       schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
858 <import namespace="http://schemas.xmlsoap.org/soap/envelope/"
859       schemaLocation="http://schemas.xmlsoap.org/soap/envelope/" />
860 <annotation>
861   <documentation>
862     Document identifier: sstc-saml-schema-ecp-2.0
863     Location: http://www.oasis-
864 open.org/committees/documents.php?wg_abbrev=security
865     Revision history:
866       V2.0 CD-03 (December, 2004):
867       Custom schema for ECP profile, first published in SAML 2.0.
868   </documentation>
869 </annotation>

870 <element name="Request" type="ecp:RequestType"/>
871 <complexType name="RequestType">
872   <sequence>
873     <element ref="saml:Issuer"/>
874     <element ref="samlp:IDPList" minOccurs="0"/>
875   </sequence>
876   <attribute ref="S:mustUnderstand" use="required"/>
877   <attribute ref="S:actor" use="required"/>
878   <attribute name="ProviderName" type="string" use="optional"/>
879   <attribute name="IsPassive" type="boolean" use="optional"/>
880 </complexType>

881 <element name="Response" type="ecp:ResponseType"/>
882 <complexType name="ResponseType">
883   <attribute ref="S:mustUnderstand" use="required"/>
884   <attribute ref="S:actor" use="required"/>
885   <attribute name="AssertionConsumerServiceURL" type="anyURI"
886 use="required"/>
887 </complexType>

888 <element name="RelayState" type="ecp:RelayStateType"/>
889 <complexType name="RelayStateType">
890   <simpleContent>
891     <extension base="string">
892       <attribute ref="S:mustUnderstand" use="required"/>
893       <attribute ref="S:actor" use="required"/>
894     </extension>
895   </simpleContent>
896 </complexType>
897 </schema>

```

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

901 **4.2.4.1 PAOS Request Header Block: SP to ECP**

902 The PAOS Request header block signals the use of PAOS processing and includes the following
903 attributes:

904 **responseConsumerURL [Required]**

905 Specifies where the ECP is to send an error response. Also used to verify the correctness of the
906 identity provider's response, by cross checking this location against the
907 AssertionServiceConsumerURL in the ECP response header block. This value **MUST** be the
908 same as the AssertionServiceConsumerURL (or the URL referenced in metadata) conveyed in
909 the <AuthnRequest>.

910 **service [Required]**

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

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

913 SOAP-ENV:mustUnderstand [Required]

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

916 SOAP-ENV:actor [Required]

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

918 messageID [Optional]

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

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

923 **4.2.4.2 ECP Request Header Block: SP to ECP**

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

927 SOAP-ENV:mustUnderstand [Required]

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

930 SOAP-ENV:actor [Required]

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

932 ProviderName [Optional]

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

934 IsPassive [Optional]

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

938 <saml:Issuer> [Required]

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

942 <samlp:IDPList> [Optional]

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

945 **4.2.4.3 ECP RelayState Header Block: SP to ECP**

946 The ECP RelayState SOAP header block is used to convey state information from the service provider
947 that it will need later when processing the response from the ECP. It is optional, but if used, the ECP
948 MUST include an identical header block in the response in step 5. It contains the following attributes:

949 SOAP-ENV:mustUnderstand [Required]

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

951 SOAP-ENV:actor [Required]

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

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

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

```
959 <SOAP-ENV:Envelope
960     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
961     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
962     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
963   <SOAP-ENV:Header>
964     <paos:Request xmlns:paos="urn:liberty:paos:2003-08"
965       responseConsumerURL="http://identity-service.example.com/abc"
966       messageID="6c3a4f8b9c2d" SOAP-
967 ENV:actor="http://schemas.xmlsoap.org/soap/actor/next" SOAP-
968 ENV:mustUnderstand="1"
969       service="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp">
970     </paos:Request>
971     <ecp:Request xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
972       SOAP-ENV:mustUnderstand="1" SOAP-
973 ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
974       ProviderName="Service Provider X" IsPassive="0">
975     <saml:Issuer>https://ServiceProvider.example.com</saml:Issuer>
976     <samlp:IDPList>
977       <samlp:IDPEntry ProviderID="https://IdentityProvider.example.com"
978         Name="Identity Provider X"
979         Loc="https://IdentityProvider.example.com/saml2/sso"
980       </samlp:IDPEntry>
981       <samlp:GetComplete>
982         https://ServiceProvider.example.com/idplist?id=604be136-fe91-441e-afb8
983       </samlp:GetComplete>
984     </samlp:IDPList>
985     </ecp:Request>
986     <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
987       SOAP-ENV:mustUnderstand="1" SOAP-
988 ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
989     ...
990     </ecp:RelayState>
991   </SOAP-ENV:Header>
992   <SOAP-ENV:Body>
993     <samlp:AuthnRequest> ... </samlp:AuthnRequest>
994   </SOAP-ENV:Body>
995 </SOAP-ENV:Envelope>
```

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

```
999 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
1000   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1001   <SOAP-ENV:Body>
1002     <samlp:AuthnRequest> ... </samlp:AuthnRequest>
1003   </SOAP-ENV:Body>
1004 </SOAP-ENV:Envelope>
```

1005 4.2.4.4 ECP Response Header Block: IdP to ECP

1006 The ECP response SOAP header block MUST be used on the response from the identity provider to the
1007 ECP. It contains the following attributes:

1008 SOAP-ENV:mustUnderstand [Required]
 1009 The value MUST be 1 (true). A SOAP fault MUST be generated if the ECP header block is not
 1010 understood.

1011 SOAP-ENV:actor [Required]
 1012 The value MUST be `http://schemas.xmlsoap.org/soap/actor/next`.

1013 AssertionConsumerServiceURL [Required]
 1014 Set by the identity provider based on the <AuthnRequest> message or the service provider's
 1015 metadata obtained by the identity provider.

1016 The ECP MUST confirm that this value corresponds to the value the ECP obtained in the
 1017 responseConsumerURL in the PAOS Request SOAP header block it received from the service
 1018 provider. Since the responseConsumerURL MAY be relative and the
 1019 AssertionConsumerServiceURL is absolute, some processing/normalization may be required.

1020 This mechanism is used for security purposes to confirm the correct response destination. If the
 1021 values do not match, then the ECP MUST generate a SOAP fault response to the service provider
 1022 and MUST NOT return the SAML response.

1023 The ECP Response SOAP header has no element content.

1024 Following is an example of an IdP-to-ECP response.

```

1025 <SOAP-ENV:Envelope
1026     xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
1027     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1028     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
1029   <SOAP-ENV:Header>
1030     <ecp:Response SOAP-ENV:mustUnderstand="1" SOAP-
1031     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
1032     AssertionConsumerServiceURL="https://ServiceProvider.example.com/ecp_assertion_
1033     consumer"/>
1034   </SOAP-ENV:Header>
1035   <SOAP-ENV:Body>
1036     <samlp:Response> ... </samlp:Response>
1037   </SOAP-ENV:Body>
1038 </SOAP-ENV:Envelope>
  
```

1039 4.2.4.5 PAOS Response Header Block: ECP to SP

1040 The PAOS Response header block includes the following attributes:

1041 SOAP-ENV:mustUnderstand [Required]
 1042 The value MUST be 1 (true). A SOAP fault MUST be generated if the PAOS header block is not
 1043 understood.

1044 SOAP-ENV:actor [Required]
 1045 The value MUST be `http://schemas.xmlsoap.org/soap/actor/next`.

1046 refToMessageID [Optional]
 1047 Allows correlation with the PAOS request. This optional attribute (and the header block as a whole)
 1048 MUST be added by the ECP if the corresponding PAOS request specified the messageID attribute.
 1049 Note that the equivalent functionality is provided in SAML using <AuthnRequest> and <Response>
 1050 correlation.

1051 The PAOS Response SOAP header has no element content.

1052 Following is an example of an ECP-to-SP response.

```

1053 <SOAP-ENV:Envelope
  
```

```

1054     xmlns:paos="urn:liberty:paos:2003-08"
1055     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1056     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
1057   <SOAP-ENV:Header>
1058     <paos:Response refToMessageID="6c3a4f8b9c2d" SOAP-
1059   ENV:actor="http://schemas.xmlsoap.org/soap/actor/next/" SOAP-
1060   ENV:mustUnderstand="1"/>
1061     <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
1062   SOAP-ENV:mustUnderstand="1" SOAP-
1063   ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
1064       ...
1065     </ecp:RelayState>
1066   </SOAP-ENV:Header>
1067   <SOAP-ENV:Body>
1068     <samlp:Response> ... </samlp:Response>
1069   </SOAP-ENV:Body>
1070 </SOAP-ENV:Envelope>

```

1071 4.2.5 Security Considerations

1072 The <AuthnRequest> message SHOULD be signed. Per the rules specified by the browser SSO profile,
 1073 the assertions enclosed in the <Response> MUST be signed. The delivery of the response in the SOAP
 1074 envelope via PAOS is essentially analogous to the use of the HTTP POST binding and security
 1075 countermeasures appropriate to that binding are used.

1076 The SOAP headers SHOULD be integrity protected, such as with SOAP Message Security or through the
 1077 use of SSL/TLS over every HTTP exchange with the client.

1078 The service provider SHOULD be authenticated to the ECP, for example with server-side TLS
 1079 authentication.

1080 The ECP SHOULD be authenticated to the identity provider, such as by maintaining an authenticated
 1081 session. Any HTTP exchanges subsequent to the delivery of the <AuthnRequest> message and before
 1082 the identity provider returns a <Response> MUST be securely associated with the original request.

1083 4.3 Identity Provider Discovery Profile

1084 This section defines a profile by which a service provider can discover which identity providers a principal
 1085 is using with the Web Browser SSO profile. In deployments having more than one identity provider,
 1086 service providers need a means to discover which identity provider(s) a principal uses. The discovery
 1087 profile relies on a cookie that is written in a domain that is common between identity providers and service
 1088 providers in a deployment. The domain that the deployment predetermines is known as the common
 1089 domain in this profile, and the cookie containing the list of identity providers is known as the common
 1090 domain cookie.

1091 Which entities host web servers in the common domain is a deployment issue and is outside the scope of
 1092 this profile.

1093 4.3.1 Common Domain Cookie

1094 The name of the cookie MUST be "_saml_idp". The format of the cookie value MUST be a set of one or
 1095 more base-64 encoded URI values separated by a single space character. Each URI is the unique
 1096 identifier of an identity provider, as defined in Section 8.3.6 of [SAMLCore]. The final set of values is then
 1097 URL encoded.

1098 The common domain cookie writing service (see below) SHOULD append the identity provider's unique
 1099 identifier to the list. If the identifier is already present in the list, it MAY remove and append it. The intent is
 1100 that the most recently established identity provider session is the last one in the list.

1101 The cookie MUST be set with a Path prefix of "/". The Domain MUST be set to ".[common-domain]" where

1102 [common-domain] is the common domain established within the deployment for use with this profile.
1103 There MUST be a leading period. The cookie MUST be marked as secure.

1104 Cookie syntax should be in accordance with IETF RFC 2965 [RFC2965] or [NSCookie]. The cookie MAY
1105 be either session-only or persistent. This choice may be made within a deployment, but should apply
1106 uniformly to all identity providers in the deployment.

1107 **4.3.2 Setting the Common Domain Cookie**

1108 After the identity provider authenticates a principal, it MAY set the common domain cookie. The means by
1109 which the identity provider sets the cookie are implementation-specific so long as the cookie is
1110 successfully set with the parameters given above. One possible implementation strategy follows and
1111 should be considered non-normative. The identity provider may:

- 1112 • Have previously established a DNS and IP alias for itself in the common domain.
- 1113 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
1114 scheme. The structure of the URL is private to the implementation and may include session
1115 information needed to identify the user agent.
- 1116 • Set the cookie on the redirected user agent using the parameters specified above.
- 1117 • Redirect the user agent back to itself, or, if appropriate, to the service provider.

1118 **4.3.3 Obtaining the Common Domain Cookie**

1119 When a service provider needs to discover which identity providers a principal uses, it invokes an
1120 exchange designed to present the common domain cookie to the service provider after it is read by an
1121 HTTP server in the common domain.

1122 If the HTTP server in the common domain is operated by the service provider or if other arrangements are
1123 in place, the service provider MAY utilize the HTTP server in the common domain to relay its
1124 <AuthnRequest> to the identity provider for an optimized single sign-on process.

1125 The specific means by which the service provider reads the cookie are implementation-specific so long as
1126 it is able to cause the user agent to present cookies that have been set with the parameters given in
1127 Section 4.3.1. One possible implementation strategy is described as follows and should be considered
1128 non-normative. Additionally, it may be sub-optimal for some applications.

- 1129 • Have previously established a DNS and IP alias for itself in the common domain.
- 1130 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
1131 scheme. The structure of the URL is private to the implementation and may include session
1132 information needed to identify the user agent.
- 1133 • Redirect the user agent back to itself, or, if appropriate, to the identity provider.

1134 **4.4 Single Logout Profile**

1135 Once a principal has authenticated to an identity provider, the authenticating entity may establish a
1136 session with the principal (typically by means of a cookie, URL re-writing, or some other implementation-
1137 specific means). The identity provider may subsequently issue assertions to service providers or other
1138 relying parties, based on this authentication event; a relying party may use this to establish *its own* session
1139 with the principal.

1140 In such a situation, the identity provider can act as a session authority and the relying parties as session
1141 participants. At some later time, the principal may wish to terminate his or her session either with an
1142 individual session participant, or with all session participants in a given session managed by the session
1143 authority. The former case is considered out of scope of this specification. The latter case, however, may
1144 be satisfied using this profile of the SAML Single Logout protocol ([SAMLCore] Section 3.7).

1145 Note that a principal (or an administrator terminating a principal's session) may choose to terminate this
1146 "global" session either by contacting the session authority, or an individual session participant. Also note
1147 that an identity provider acting as a session authority may *itself* act as a session participant in situations in
1148 which it is the relying party for another identity provider's assertions regarding that principal.

1149 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1150 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1151 front-channel binding may be required, for example, in cases in which a principal's session state exists
1152 solely in a user agent in the form of a cookie and a direct interaction between the user agent and the
1153 session participant or session authority is required.

1154 4.4.1 Required Information

1155 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:logout

1156 **Contact information:** security-services-comment@lists.oasis-open.org

1157 **Description:** Given below.

1158 **Updates:** None

1159 4.4.2 Profile Overview

1160 Figure 3 illustrates the basic template for achieving single logout:

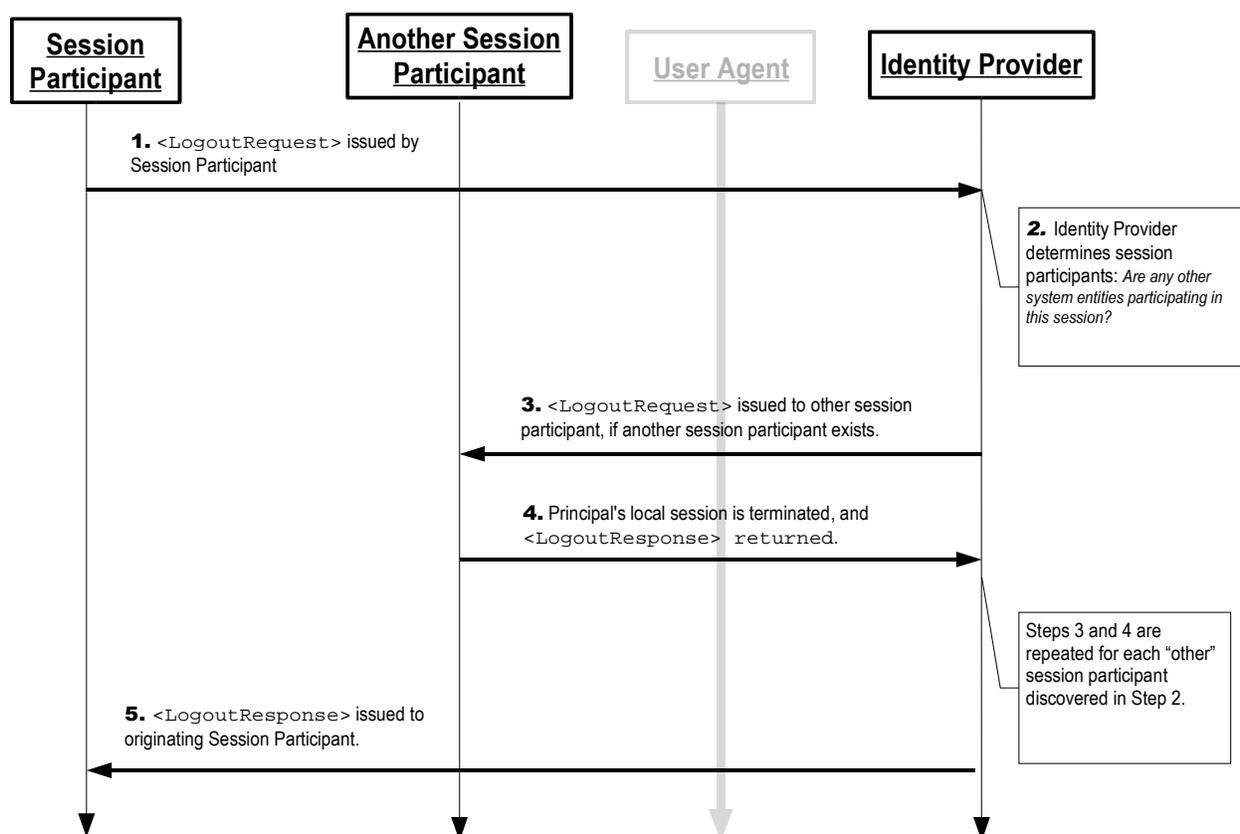


Figure 3

1161 The grayed-out user agent illustrates that the message exchange may pass through the user agent or

1162 may be a direct exchange between system entities, depending on the SAML binding used to implement
1163 the profile.

1164 The following steps are described by the profile. Within an individual step, there may be one or more
1165 actual message exchanges depending on the binding used for that step and other implementation-
1166 dependent behavior.

1167 **1. <LogoutRequest> issued by Session Participant to Identity Provider**

1168 In step 1, the session participant initiates single logout and terminates a principal's session(s) by
1169 sending a <LogoutRequest> message to the identity provider from whom it received the
1170 corresponding authentication assertion. The request may be sent directly to the identity provider
1171 or sent indirectly through the user agent.

1172 **2. Identity Provider determines Session Participants**

1173 In step 2, the identity provider uses the contents of the <LogoutRequest> message (or if
1174 initiating logout itself, some other mechanism) to determine the session(s) being terminated. If
1175 there are no other session participants, the profile proceeds with step 5. Otherwise, steps 3 and 4
1176 are repeated for each session participant identified.

1177 **3. <LogoutRequest> issued by Identity Provider to Session Participant/Authority**

1178 In step 3, the identity provider issues a <LogoutRequest> message to a session participant or
1179 session authority related to one or more of the session(s) being terminated. The request may be
1180 sent directly to the entity or sent indirectly through the user agent (if consistent with the form of the
1181 request in step 1).

1182 **4. Session Participant/Authority issues <LogoutResponse> to Identity Provider**

1183 In step 4, a session participant or session authority terminates the principal's session(s) as
1184 directed by the request (if possible) and returns a <LogoutResponse> to the identity provider.
1185 The response may be returned directly to the identity provider or indirectly through the user agent
1186 (if consistent with the form of the request in step 3).

1187 **5. Identity Provider issues <LogoutResponse> to Session Participant**

1188 In step 5, the identity provider issues a <LogoutResponse> message to the original requesting
1189 session participant. The response may be returned directly to the session participant or indirectly
1190 through the user agent (if consistent with the form of the request in step 1).

1191 Note that an identity provider (acting as session authority) can initiate this profile at step 2 and issue a
1192 <LogoutRequest> to all session participants, also skipping step 5.

1193 **4.4.3 Profile Description**

1194 If the profile is initiated by a session participant, start with Section 4.4.3.1. If initiated by the identity
1195 provider, start with Section 4.4.3.2. In the descriptions below, the following is referred to:

1196 **Single Logout Service**

1197 This is the single logout protocol endpoint at an identity provider or session participant to which the
1198 <LogoutRequest> or <LogoutResponse> messages (or an artifact representing them) are
1199 delivered. The same or different endpoints MAY be used for requests and responses.

1200 **4.4.3.1 <LogoutRequest> Issued by Session Participant to Identity Provider**

1201 If the logout profile is initiated by a session participant, it examines the authentication assertion(s) it
1202 received pertaining to the session(s) being terminated, and collects the `SessionIndex` value(s) it
1203 received from the identity provider. If multiple identity providers are involved, then the profile MUST be
1204 repeated independently for each one.

1205 To initiate the profile, the session participant issues a `<LogoutRequest>` message to the identity
1206 provider's single logout service request endpoint containing one or more applicable `<SessionIndex>`
1207 elements. At least one element MUST be included. Metadata (as in [SAMLMeta]) MAY be used to
1208 determine the location of this endpoint and the bindings supported by the identity provider.

1209 **Synchronous Bindings (Back-Channel)**

1210 The session participant MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to
1211 send the request directly to the identity provider. The identity provider would then propagate any
1212 required logout messages to additional session participants as required using a synchronous binding.
1213 The requester MUST authenticate itself to the identity provider, either by signing the
1214 `<LogoutRequest>` or using any other binding-supported mechanism.

1215 **Asynchronous Bindings (Front-Channel)**

1216 Alternatively, the session participant MAY (if the principal's user agent is present) use an
1217 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to send the
1218 request to the identity provider through the user agent.

1219 If the HTTP Redirect or POST binding is used, then the `<LogoutRequest>` message is delivered to
1220 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile
1221 defined in Section 5 is used by the identity provider, which makes a callback to the session participant
1222 to retrieve the `<LogoutRequest>` message, using for example the SOAP binding.

1223 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1224 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The `<LogoutRequest>`
1225 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,
1226 if used, also provides for an alternate means of authenticating the request issuer when the artifact is
1227 dereferenced.

1228 Each of these bindings provide a RelayState mechanism that the session participant MAY use to
1229 associate the profile exchange with the original request. The session participant SHOULD reveal as
1230 little information as possible in the RelayState value unless the use of the profile does not require such
1231 privacy measures.

1232 Profile-specific rules for the contents of the `<LogoutRequest>` message are included in Section 4.4.4.1.

1233 **4.4.3.2 Identity Provider Determines Session Participants**

1234 If the logout profile is initiated by an identity provider, or upon receiving a valid `<LogoutRequest>`
1235 message, the identity provider processes the request as defined in [SAMLCore]. It MUST examine the
1236 identifier and `<SessionIndex>` elements and determine the set of sessions to be terminated.

1237 The identity provider then follows steps 3 and 4 for each entity participating in the session(s) being
1238 terminated, other than the original requesting session participant (if any), as described in Section 3.7.3.2
1239 of [SAMLCore].

1240 **4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session 1241 Participant/Authority**

1242 To propagate the logout, the identity provider issues its own `<LogoutRequest>` to a session authority or
1243 participant in a session being terminated. The request is sent in the same fashion as described in step 1

1244 using a SAML binding consistent with the capability of the responder and the availability of the user agent
1245 at the identity provider.

1246 Profile-specific rules for the contents of the <LogoutRequest> message are included in Section 4.4.4.1.

1247 **4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity** 1248 **Provider**

1249 The session participant/authority MUST process the <LogoutRequest> message as defined in
1250 [SAMLCore]. After processing the message or upon encountering an error, the entity MUST issue a
1251 <LogoutResponse> message containing an appropriate status code to the requesting identity provider
1252 to complete the SAML protocol exchange.

1253 **Synchronous Bindings (Back-Channel)**

1254 If the identity provider used a synchronous binding, such as the SOAP binding [SAMLBind], the
1255 response is returned directly to complete the synchronous communication. The responder MUST
1256 authenticate itself to the requesting identity provider, either by signing the <LogoutResponse> or
1257 using any other binding-supported mechanism.

1258 **Asynchronous Bindings (Front-Channel)**

1259 If the identity provider used an asynchronous binding, such as the HTTP Redirect, POST, or Artifact
1260 bindings [SAMLBind], then the <LogoutResponse> (or artifact) is returned through the user agent to
1261 the identity provider's single logout service response endpoint. Metadata (as in [SAMLMeta]) MAY be
1262 used to determine the location of this endpoint and the bindings supported by the identity provider.

1263 If the HTTP Redirect or POST binding is used, then the <LogoutResponse> message is delivered to
1264 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile
1265 defined in Section 5 is used by the identity provider, which makes a callback to the responding entity
1266 to retrieve the <LogoutResponse> message, using for example the SOAP binding.

1267 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1268 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <LogoutResponse>
1269 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,
1270 if used, also provides for an alternate means of authenticating the response issuer when the artifact is
1271 dereferenced.

1272 Profile-specific rules for the contents of the <LogoutResponse> message are included in Section
1273 4.4.4.2.

1274 **4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant**

1275 After processing the original session participant's <LogoutRequest> in step 1, or upon encountering an
1276 error, the identity provider MUST respond to the original request with a <LogoutResponse> containing
1277 an appropriate status code to complete the SAML protocol exchange.

1278 The response is sent to the original session participant in the same fashion as described in step 4, using a
1279 SAML binding consistent with the binding used in the request, the capability of the responder, and the
1280 availability of the user agent at the identity provider.

1281 Profile-specific rules for the contents of the <LogoutResponse> message are included in Section
1282 4.4.4.2.

1283 4.4.4 Use of Single Logout Protocol

1284 4.4.4.1 <LogoutRequest> Usage

1285 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1286 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1287 format:entity.

1288 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1289 the message or using a binding-specific mechanism.

1290 The principal MUST be identified in the request using an identifier that **strongly matches** the identifier in
1291 the authentication assertion the requester issued or received regarding the session being terminated, per
1292 the matching rules defined in Section 3.3.4 of [SAMLCore].

1293 If the requester is a session participant, it MUST include at least one <SessionIndex> element in the
1294 request. If the requester is a session authority (or acting on its behalf), then it MAY omit any such
1295 elements to indicate the termination of all of the principal's applicable sessions.

1296 4.4.4.2 <LogoutResponse> Usage

1297 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1298 entity; the Format attribute MUST be omitted or have a value of
1299 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1300 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1301 the message or using a binding-specific mechanism.

1302 4.4.5 Use of Metadata

1303 [SAMLMeta] defines an endpoint element, <md:SingleLogoutService>, to describe supported
1304 bindings and location(s) to which an entity may send requests and responses using this profile.

1305 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>
1306 element with a use attribute of encryption to determine an appropriate encryption algorithm and
1307 settings to use, along with a public key to use in delivering a bulk encryption key.

1308 4.5 Name Identifier Management Profile

1309 In the scenario supported by the Name Identifier Management profile, an identity provider has exchanged
1310 some form of persistent identifier for a principal with a service provider, allowing them to share a common
1311 identifier for some length of time. Subsequently, the identity provider may wish to notify the service
1312 provider of a change in the format and/or value that it will use to identify the same principal in the future.
1313 Alternatively the service provider may wish to attach its own "alias" for the principal in order to ensure that
1314 the identity provider will include it when communicating with it in the future about the principal. Finally, one
1315 of the providers may wish to inform the other that it will no longer issue or accept messages using a
1316 particular identifier. To implement these scenarios, a profile of the SAML Name Identifier Management
1317 protocol is used.

1318 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1319 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1320 front-channel binding may be required, for example, in cases in which direct interaction between the user
1321 agent and the responding provider is required in order to effect the change.

1322 **4.5.1 Required Information**

1323 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:nameid-mgmt

1324 **Contact information:** security-services-comment@lists.oasis-open.org

1325 **Description:** Given below.

1326 **Updates:** None.

1327 **4.5.2 Profile Overview**

1328 Figure 4 illustrates the basic template for the name identifier management profile.

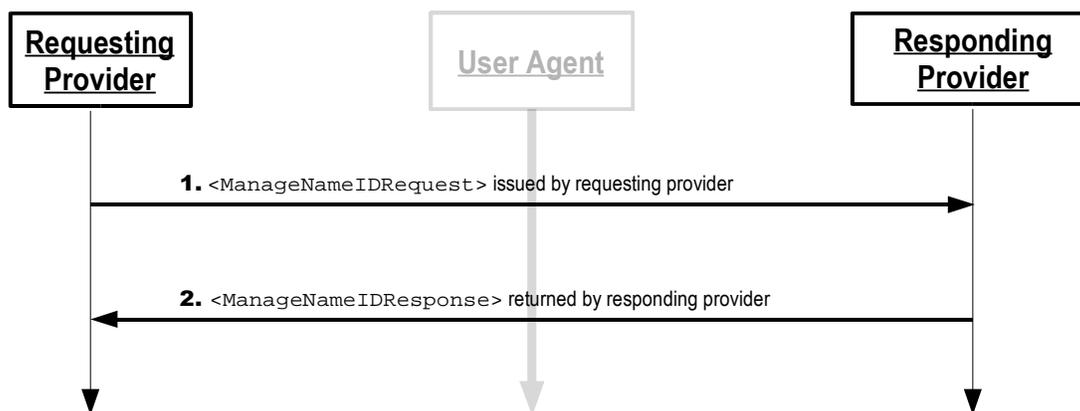


Figure 4

1329 The grayed-out user agent illustrates that the message exchange may pass through the user agent or
1330 may be a direct exchange between system entities, depending on the SAML binding used to implement
1331 the profile.

1332 The following steps are described by the profile. Within an individual step, there may be one or more
1333 actual message exchanges depending on the binding used for that step and other implementation-
1334 dependent behavior.

1335 **1. <ManageNameIDRequest> issued by Requesting Identity/Service Provider**

1336 In step 1, an identity or service provider initiates the profile by sending a
1337 <ManageNameIDRequest> message to another provider that it wishes to inform of a change.
1338 The request may be sent directly to the responding provider or sent indirectly through the user
1339 agent.

1340 **2. <ManageNameIDResponse> issued by Responding Identity/Service Provider**

1341 In step 2, the responding provider (after processing the request) issues a
1342 <ManageNameIDResponse> message to the original requesting provider. The response may be
1343 returned directly to the requesting provider or indirectly through the user agent (if consistent with
1344 the form of the request in step 1).

1345 **4.5.3 Profile Description**

1346 In the descriptions below, the following is referred to:

1347 **Name Identifier Management Service**

1348 This is the name identifier management protocol endpoint at an identity or service provider to which
1349 the <ManageNameIDRequest> or <ManageNameIDResponse> messages (or an artifact
1350 representing them) are delivered. The same or different endpoints MAY be used for requests and
1351 responses.

1352 **4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider**

1353 To initiate the profile, the requesting provider issues a <ManageNameIDRequest> message to another
1354 provider's name identifier management service request endpoint. Metadata (as in [SAMLMeta]) MAY be
1355 used to determine the location of this endpoint and the bindings supported by the responding provider.

1356 **Synchronous Bindings (Back-Channel)**

1357 The requesting provider MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to
1358 send the request directly to the other provider. The requester MUST authenticate itself to the other
1359 provider, either by signing the <ManageNameIDRequest> or using any other binding-supported
1360 mechanism.

1361 **Asynchronous Bindings (Front-Channel)**

1362 Alternatively, the requesting provider MAY (if the principal's user agent is present) use an
1363 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to send the
1364 request to the other provider through the user agent.

1365 If the HTTP Redirect or POST binding is used, then the <ManageNameIDRequest> message is
1366 delivered to the other provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution
1367 profile defined in Section 55 is used by the other provider, which makes a callback to the requesting
1368 provider to retrieve the <ManageNameIDRequest> message, using for example the SOAP binding.

1369 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1370 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1371 <ManageNameIDRequest> message MUST be signed if the HTTP POST or Redirect binding is
1372 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1373 request issuer when the artifact is dereferenced.

1374 Each of these bindings provide a RelayState mechanism that the requesting provider MAY use to
1375 associate the profile exchange with the original request. The requesting provider SHOULD reveal as
1376 little information as possible in the RelayState value unless the use of the profile does not require such
1377 privacy measures.

1378 Profile-specific rules for the contents of the <ManageNameIDRequest> message are included in Section
1379 4.5.4.1.

1380 **4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service
1381 Provider**

1382 The recipient MUST process the <ManageNameIDRequest> message as defined in [SAMLCore]. After
1383 processing the message or upon encountering an error, the recipient MUST issue a
1384 <ManageNameIDResponse> message containing an appropriate status code to the requesting provider
1385 to complete the SAML protocol exchange.

1386 **Synchronous Bindings (Back-Channel)**

1387 If the requesting provider used a synchronous binding, such as the SOAP binding [SAMLBind], the
1388 response is returned directly to complete the synchronous communication. The responder MUST
1389 authenticate itself to the requesting provider, either by signing the <ManageNameIDResponse> or
1390 using any other binding-supported mechanism.

1391 **Asynchronous Bindings (Front-Channel)**

1392 If the requesting provider used an asynchronous binding, such as the HTTP Redirect, POST, or
1393 Artifact bindings [SAMLBind], then the <ManageNameIDResponse> (or artifact) is returned through
1394 the user agent to the requesting provider's name identifier management service response endpoint.
1395 Metadata (as in [SAMLMeta]) MAY be used to determine the location of this endpoint and the bindings
1396 supported by the requesting provider.

1397 If the HTTP Redirect or POST binding is used, then the <ManageNameIDResponse> message is
1398 delivered to the requesting provider in this step. If the HTTP Artifact binding is used, the Artifact
1399 Resolution profile defined in Section 55 is used by the requesting provider, which makes a callback to
1400 the responding provider to retrieve the <ManageNameIDResponse> message, using for example the
1401 SOAP binding.

1402 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1403 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1404 <ManageNameIDResponse> message MUST be signed if the HTTP POST or Redirect binding is
1405 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1406 response issuer when the artifact is dereferenced.

1407 Profile-specific rules for the contents of the <ManageNameIDResponse> message are included in
1408 Section 4.5.4.2.

1409 **4.5.4 Use of Name Identifier Management Protocol**

1410 **4.5.4.1 <ManageNameIDRequest> Usage**

1411 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1412 the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`
1413 `format:entity`.

1414 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1415 the message or using a binding-specific mechanism.

1416 **4.5.4.2 <ManageNameIDResponse> Usage**

1417 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1418 entity; the `Format` attribute MUST be omitted or have a value of
1419 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

1420 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1421 the message or using a binding-specific mechanism.

1422 **4.5.5 Use of Metadata**

1423 [SAMLMeta] defines an endpoint element, <md:ManageNameIDService>, to describe supported
1424 bindings and location(s) to which an entity may send requests and responses using this profile.

1425 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>
1426 element with a `use` attribute of `encryption` to determine an appropriate encryption algorithm and
1427 settings to use, along with a public key to use in delivering a bulk encryption key.

1428

5 Artifact Resolution Profile

1429 [SAMLCore] defines an Artifact Resolution protocol for dereferencing a SAML artifact into a corresponding
1430 protocol message. The HTTP Artifact binding in [SAMLBind] leverages this mechanism to pass SAML
1431 protocol messages by reference. This profile describes the use of this protocol with a synchronous
1432 binding, such as the SOAP binding defined in [SAMLBind].

5.1 Required Information

1434 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:artifact

1435 **Contact information:** security-services-comment@lists.oasis-open.org

1436 **Description:** Given below.

1437 **Updates:** None

5.2 Profile Overview

1439 The message exchange and basic processing rules that govern this profile are largely defined by Section
1440 3.5 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1441 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1442 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1443 Figure 5 illustrates the basic template for the artifact resolution profile.

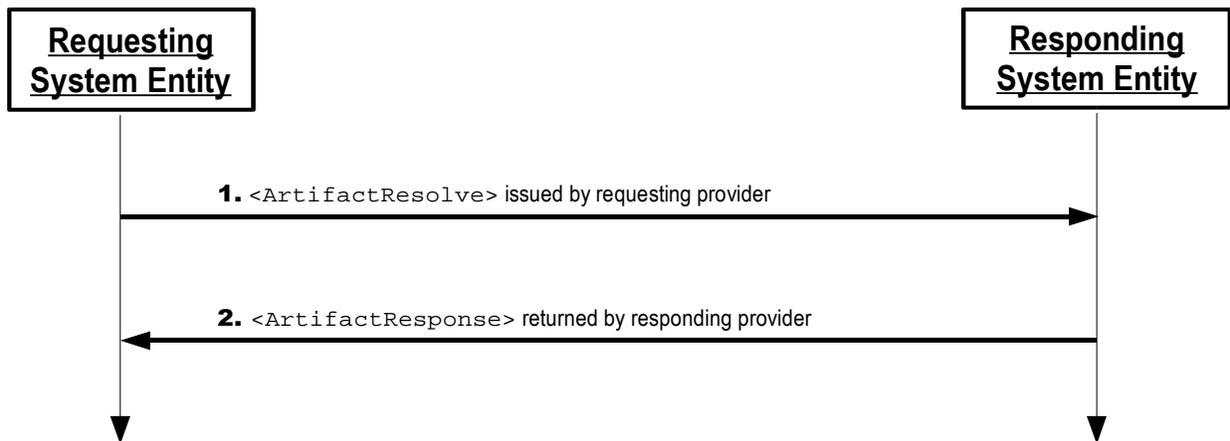


Figure 5

1444 The following steps are described by the profile.

1445 1. <ArtifactResolve> issued by Requesting Entity

1446 In step 1, a requester initiates the profile by sending an <ArtifactResolve> message to an
1447 artifact issuer.

1448 2. <ArtifactResponse> issued by Responding Entity

1449 In step 2, the responder (after processing the request) issues an <ArtifactResponse>
1450 message to the requester.

1451 5.3 Profile Description

1452 In the descriptions below, the following is referred to:

1453 Artifact Resolution Service

1454 This is the artifact resolution protocol endpoint at an artifact issuer to which <ArtifactResolve>
1455 messages are delivered.

1456 5.3.1 <ArtifactResolve> issued by Requesting Entity

1457 To initiate the profile, a requester, having received an artifact and determined the issuer using the
1458 SourceID, sends an <ArtifactResolve> message containing the artifact to an artifact issuer's artifact
1459 resolution service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of this
1460 endpoint and the bindings supported by the artifact issuer

1461 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1462 request directly to the artifact issuer. The requester SHOULD authenticate itself to the responder, either by
1463 signing the <ArtifactResolve> message or using any other binding-supported mechanism. Specific
1464 profiles that use the HTTP Artifact binding MAY impose additional requirements such that authentication is
1465 mandatory.

1466 Profile-specific rules for the contents of the <ArtifactResolve> message are included in Section 5.4.1.

1467 5.3.2 <ArtifactResponse> issued by Responding Entity

1468 The artifact issuer MUST process the <ArtifactResolve> message as defined in [SAMLCore]. After
1469 processing the message or upon encountering an error, the artifact issuer MUST return an
1470 <ArtifactResponse> message containing an appropriate status code to the requester to complete the
1471 SAML protocol exchange. If successful, the dereferenced SAML protocol message corresponding to the
1472 artifact will also be included.

1473 The responder MUST authenticate itself to the requester, either by signing the <ArtifactResponse> or
1474 using any other binding-supported mechanism.

1475 Profile-specific rules for the contents of the <ArtifactResponse> message are included in Section
1476 5.4.2.

1477 5.4 Use of Artifact Resolution Protocol

1478 5.4.1 <ArtifactResolve> Usage

1479 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1480 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1481 format:entity.

1482 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by
1483 signing the message or using a binding-specific mechanism. Specific profiles that use the HTTP Artifact
1484 binding MAY impose additional requirements such that authentication is mandatory.

1485 **5.4.2 <ArtifactResponse> Usage**

1486 The <Issuer> element MUST be present and MUST contain the unique identifier of the artifact issuer;
1487 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1488 format:entity.

1489 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1490 the message or using a binding-specific mechanism.

1491 **5.5 Use of Metadata**

1492 [SAMLMeta] defines an indexed endpoint element, <md:ArtifactResolutionService>, to describe
1493 supported bindings and location(s) to which a requester may send requests using this profile. The index
1494 attribute is used to distinguish the possible endpoints that may be specified by reference in the artifact's
1495 EndpointIndex field.

1496

6 Assertion Query/Request Profile

1497 [SAMLCore] defines a protocol for requesting existing assertions by reference or by querying on the basis
1498 of a subject and additional statement-specific criteria. This profile describes the use of this protocol with a
1499 synchronous binding, such as the SOAP binding defined in [SAMLBind].

6.1 Required Information

1501 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:query

1502 **Contact information:** security-services-comment@lists.oasis-open.org

1503 **Description:** Given below.

1504 **Updates:** None.

6.2 Profile Overview

1506 The message exchange and basic processing rules that govern this profile are largely defined by Section
1507 3.3 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1508 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1509 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1510 Figure 6 illustrates the basic template for the query/request profile.

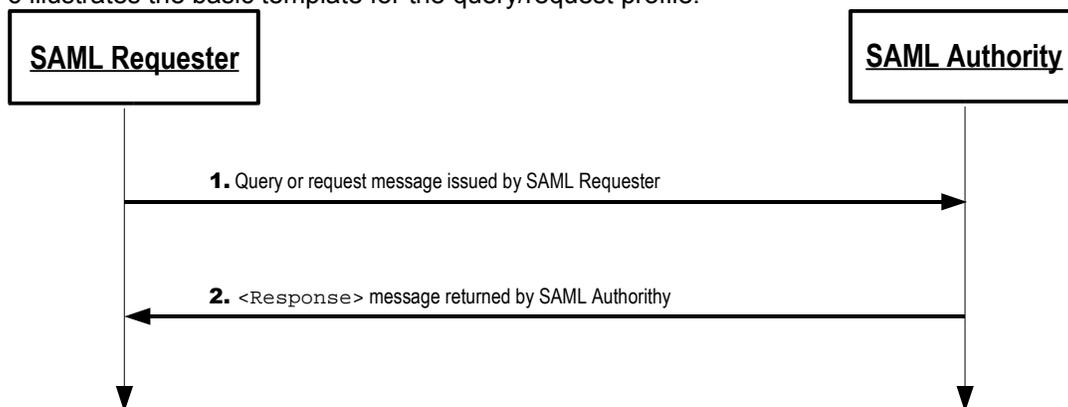


Figure 6

1511 The following steps are described by the profile.

1. Query/Request issued by SAML Requester

1513 In step 1, a SAML requester initiates the profile by sending an `<AssertionIDRequest>`,
1514 `<SubjectQuery>`, `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>`
1515 message to a SAML authority.

2. <Response> issued by SAML Authority

1517 In step 2, the responding SAML authority (after processing the query or request) issues a
1518 `<Response>` message to the SAML requester.

1519 **6.3 Profile Description**

1520 In the descriptions below, the following are referred to:

1521 **Query/Request Service**

1522 This is the query/request protocol endpoint at a SAML authority to which query or
1523 `<AssertionIDRequest>` messages are delivered.

1524 **6.3.1 Query/Request issued by SAML Requester**

1525 To initiate the profile, a SAML requester issues an `<AssertionIDRequest>`, `<SubjectQuery>`,
1526 `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>` message to a SAML authority's
1527 query/request service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of
1528 this endpoint and the bindings supported by the SAML authority.

1529 The SAML requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send
1530 the request directly to the identity provider. The requester SHOULD authenticate itself to the SAML
1531 authority either by signing the message or using any other binding-supported mechanism.

1532 Profile-specific rules for the contents of the various messages are included in Section 6.4.1.

1533 **6.3.2 `<Response>` issued by SAML Authority**

1534 The SAML authority MUST process the query or request message as defined in [SAMLCore]. After
1535 processing the message or upon encountering an error, the SAML authority MUST return a `<Response>`
1536 message containing an appropriate status code to the SAML requester to complete the SAML protocol
1537 exchange. If the request is successful in locating one or more matching assertions, they will also be
1538 included in the response.

1539 The responder SHOULD authenticate itself to the requester, either by signing the `<Response>` or using
1540 any other binding-supported mechanism.

1541 Profile-specific rules for the contents of the `<Response>` message are included in Section 6.4.2.

1542 **6.4 Use of Query/Request Protocol**

1543 **6.4.1 Query/Request Usage**

1544 The `<Issuer>` element MUST be present.

1545 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by
1546 signing the message or using a binding-specific mechanism.

1547 **6.4.2 `<Response>` Usage**

1548 The `<Issuer>` element MUST be present and MUST contain the unique identifier of the responding
1549 SAML authority; the `Format` attribute MUST be omitted or have a value of
1550 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`. Note that this need not necessarily
1551 match the `<Issuer>` element in the returned assertion(s).

1552 The responder SHOULD authenticate itself to the requester and ensure message integrity, either by
1553 signing the message or using a binding-specific mechanism.

1554 **6.5 Use of Metadata**

1555 [SAMLMeta] defines several endpoint elements, `<md:AssertionIDRequestService>`,
1556 `<md:AuthnQueryService>`, `<md:AttributeService>`, and `<md:AuthzService>`, to describe
1557 supported bindings and location(s) to which a requester may send requests or queries using this profile.

1558 The SAML authority, if encrypting the resulting assertions or assertion contents for a particular entity, can
1559 use that entity's `<md:KeyDescriptor>` element with a `use` attribute of `encryption` to determine an
1560 appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk
1561 encryption key.

1562 The various role descriptors MAY contain `<md:NameIDFormat>`, `<md:AttributeProfile>`, and
1563 `<saml:Attribute>` elements (as applicable) to indicate the general ability to support particular name
1564 identifier formats, attribute profiles, or specific attributes and values. The ability to support any such
1565 features during a given request is dependent on policy and the discretion of the authority.

1566

7 Name Identifier Mapping Profile

1567 [SAMLCore] defines a Name Identifier Mapping protocol for mapping a principal's name identifier into a
1568 different name identifier for the same principal. This profile describes the use of this protocol with a
1569 synchronous binding, such as the SOAP binding defined in [SAMLBind], and additional guidelines for
1570 protecting the privacy of the principal with encryption and limiting the use of the mapped identifier.

7.1 Required Information

1572 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:nameidmapping

1573 **Contact information:** security-services-comment@lists.oasis-open.org

1574 **Description:** Given below.

1575 **Updates:** None.

7.2 Profile Overview

1577 The message exchange and basic processing rules that govern this profile are largely defined by Section
1578 3.8 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1579 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1580 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1581 Figure 7 illustrates the basic template for the name identifier mapping profile.



Figure 7

1582 The following steps are described by the profile.

1583 1. <NameIDMappingRequest> issued by Requesting Entity

1584 In step 1, a requester initiates the profile by sending a <NameIDMappingRequest> message to
1585 an identity provider.

1586 2. <NameIDMappingResponse> issued by Identity Provider

1587 In step 2, the responding identity provider (after processing the request) issues a
1588 <NameIDMappingResponse> message to the requester.

1589 **7.3 Profile Description**

1590 In the descriptions below, the following is referred to:

1591 **Name Identifier Mapping Service**

1592 This is the name identifier mapping protocol endpoint at an identity provider to which
1593 <NameIDMappingRequest> messages are delivered.

1594 **7.3.1 <NameIDMappingRequest> issued by Requesting Entity**

1595 To initiate the profile, a requester issues a <NameIDMappingRequest> message to an identity provider's
1596 name identifier mapping service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the
1597 location of this endpoint and the bindings supported by the identity provider.

1598 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1599 request directly to the identity provider. The requester MUST authenticate itself to the identity provider,
1600 either by signing the <NameIDMappingRequest> or using any other binding-supported mechanism.

1601 Profile-specific rules for the contents of the <NameIDMappingRequest> message are included in
1602 Section 7.4.1.

1603 **7.3.2 <NameIDMappingResponse> issued by Identity Provider**

1604 The identity provider MUST process the <ManageNameIDRequest> message as defined in [SAMLCore].
1605 After processing the message or upon encountering an error, the identity provider MUST return a
1606 <NameIDMappingResponse> message containing an appropriate status code to the requester to
1607 complete the SAML protocol exchange.

1608 The responder MUST authenticate itself to the requester, either by signing the
1609 <NameIDMappingResponse> or using any other binding-supported mechanism.

1610 Profile-specific rules for the contents of the <NameIDMappingResponse> message are included in
1611 Section 7.4.2.

1612 **7.4 Use of Name Identifier Mapping Protocol**

1613 **7.4.1 <NameIDMappingRequest> Usage**

1614 The <Issuer> element MUST be present.

1615 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1616 the message or using a binding-specific mechanism.

1617 **7.4.2 <NameIDMappingResponse> Usage**

1618 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1619 identity provider; the Format attribute MUST be omitted or have a value of
1620 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1621 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1622 the message or using a binding-specific mechanism.

1623 Section 2.2.3 of [SAMLCore] defines the use of encryption to apply confidentiality to a name identifier. In
1624 most cases, the identity provider SHOULD encrypt the mapped name identifier it returns to the requester
1625 to protect the privacy of the principal. The requester can extract the <EncryptedID> element and place it
1626 in subsequent protocol messages or assertions.

1627 **7.4.2.1 Limiting Use of Mapped Identifier**

1628 Additional limits on the use of the resulting identifier MAY be applied by the identity provider by returning
1629 the mapped name identifier in the form of an <Assertion> containing the identifier in its <Subject> but
1630 without any statements. The assertion is then encrypted and the result used as the <EncryptedData>
1631 element in the <EncryptedID> returned to the requester. The assertion MAY include a <Conditions>
1632 element to limit use, as defined by [SAMLCore], such as time-based constraints or use by specific relying
1633 parties, and MUST be signed for integrity protection.

1634 **7.5 Use of Metadata**

1635 [SAMLMeta] defines an endpoint element, <md:NameIDMappingService>, to describe supported
1636 bindings and location(s) to which a requester may send requests using this profile.

1637 The identity provider, if encrypting the resulting identifier for a particular entity, can use that entity's
1638 <md:KeyDescriptor> element with a use attribute of encryption to determine an appropriate
1639 encryption algorithm and settings to use, along with a public key to use in delivering a bulk encryption key.

1640 8 SAML Attribute Profiles

1641 8.1 Basic Attribute Profile

1642 The Basic attribute profile specifies simplified, but non-unique, naming of SAML attributes together with
1643 attribute values based on the built-in XML Schema data types, eliminating the need for extension schemas
1644 to validate syntax.

1645 8.1.1 Required Information

1646 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:basic

1647 **Contact information:** security-services-comment@lists.oasis-open.org

1648 **Description:** Given below.

1649 **Updates:** None.

1650 8.1.2 SAML Attribute Naming

1651 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
1652 `urn:oasis:names:tc:SAML:2.0:attrname-format:basic`.

1653 The `Name` XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

1654 8.1.2.1 Attribute Name Comparison

1655 Two `<Attribute>` elements refer to the same SAML attribute if and only if the values of their `Name` XML
1656 attributes are equal in the sense of Section 3.3.6 of [Schema2].

1657 8.1.3 Profile-Specific XML Attributes

1658 No additional XML attributes are defined for use with the `<Attribute>` element.

1659 8.1.4 SAML Attribute Values

1660 The schema type of the contents of the `<AttributeValue>` element MUST be drawn from one of the
1661 types defined in Section 3.3 of [Schema2]. The `xsi:type` attribute MUST be present and be given the
1662 appropriate value.

1663 8.1.5 Example

```
1664     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"  
1665         Name="FirstName">  
1666         <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>  
1667     </saml:Attribute>
```

1668 8.2 X.500/LDAP Attribute Profile

1669 Directories based on the ITU-T X.500 specifications [X.500] and the related IETF Lightweight Directory
1670 Access Protocol specifications [LDAP] are widely deployed. Directory schema is used to model
1671 information to be stored in these directories. In particular, in X.500, attribute type definitions are used to
1672 specify the syntax and other features of attributes, the basic information storage unit in a directory (this

1673 document refers to these as “directory attributes”). Directory attribute types are defined in schema in the
1674 X.500 and LDAP specifications themselves, schema in other public documents (such as the
1675 Internet2/Educause EduPerson schema [eduPerson], or the inetOrgperson schema [RFC2798]), and
1676 schema defined for private purposes. In any of these cases, it is useful for deployers to take advantage of
1677 these directory attribute types in the context of SAML attribute statements, without having to manually
1678 create SAML-specific attribute definitions for them, and to do this in an interoperable fashion.
1679 The X.500/LDAP attribute profile defines a common convention for the naming and representation of such
1680 attributes when expressed as SAML attributes.

1681 8.2.1 Required Information

1682 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500 (this is also the target namespace
1683 assigned in the corresponding X.500/LDAP profile schema document [SAMLX500-xsd])

1684 **Contact information:** security-services-comment@lists.oasis-open.org

1685 **Description:** Given below.

1686 **Updates:** None.

1687 8.2.2 SAML Attribute Naming

1688 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
1689 urn:oasis:names:tc:SAML:2.0:attrname-format:uri.

1690 To construct attribute names, the URN `oid` namespace described in IETF RFC 3061 [RFC3061] is used.
1691 In this approach the `Name` XML attribute is based on the OBJECT IDENTIFIER assigned to the directory
1692 attribute type.

1693 Example:

```
1694 urn:oid:2.5.4.3
```

1695 Since X.500 procedures require that every attribute type be identified with a unique OBJECT IDENTIFIER,
1696 this naming scheme ensures that the derived SAML attribute names are unambiguous.

1697 For purposes of human readability, there may also be a requirement for some applications to carry an
1698 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
1699 [SAMLCore]) MAY be used for this purpose. If the definition of the directory attribute type includes one or
1700 more descriptors (short names) for the attribute type, the `FriendlyName` value, if present, SHOULD be
1701 one of the defined descriptors.

1702 8.2.2.1 Attribute Name Comparison

1703 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
1704 values are equal in the sense of [RFC3061]. The `FriendlyName` attribute plays no role in the
1705 comparison.

1706 8.2.3 Profile-Specific XML Attributes

1707 No additional XML attributes are defined for use with the `<Attribute>` element.

1708 8.2.4 SAML Attribute Values

1709 Directory attribute type definitions for use in native X.500 directories specify the syntax of the attribute
1710 using ASN.1 [ASN.1]. For use in LDAP, directory attribute definitions additionally include an LDAP syntax
1711 which specifies how attribute or assertion values conforming to the syntax are to be represented when
1712 transferred in the LDAP protocol (known as an LDAP-specific encoding). The LDAP-specific encoding

1713 commonly produces Unicode characters in UTF-8 form. This SAML attribute profile specifies the form of
1714 SAML attribute values only for those directory attributes which have LDAP syntaxes. Future extensions to
1715 this profile may define attribute value formats for directory attributes whose syntaxes specify other
1716 encodings.

1717 To represent the encoding rules in use for a particular attribute value, the <AttributeValue> element
1718 MUST contain an XML attribute named `Encoding` defined in the XML namespace
1719 `urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500`.

1720 For any directory attribute with a syntax whose LDAP-specific encoding exclusively produces UTF-8
1721 character strings as values, the SAML attribute value is encoded as simply the UTF-8 string itself, as the
1722 content of the <AttributeValue> element, with no additional whitespace. In such cases, the
1723 `xsi:type` XML attribute MUST be set to **xs:string**. The profile-specific `Encoding` XML attribute is
1724 provided, with a value of `LDAP`.

1725 A list of some LDAP attribute syntaxes to which this applies is:

1726	Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3
1727	Bit String	1.3.6.1.4.1.1466.115.121.1.6
1728	Boolean	1.3.6.1.4.1.1466.115.121.1.7
1729	Country String	1.3.6.1.4.1.1466.115.121.1.11
1730	DN	1.3.6.1.4.1.1466.115.121.1.12
1731	Directory String	1.3.6.1.4.1.1466.115.121.1.15
1732	Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22
1733	Generalized Time	1.3.6.1.4.1.1466.115.121.1.24
1734	IA5 String	1.3.6.1.4.1.1466.115.121.1.26
1735	INTEGER	1.3.6.1.4.1.1466.115.121.1.27
1736	LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54
1737	Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30
1738	Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31
1739	Name And Optional UID	1.3.6.1.4.1.1466.115.121.1.34
1740	Name Form Description	1.3.6.1.4.1.1466.115.121.1.35
1741	Numeric String	1.3.6.1.4.1.1466.115.121.1.36
1742	Object Class Description	1.3.6.1.4.1.1466.115.121.1.37
1743	Octet String	1.3.6.1.4.1.1466.115.121.1.40
1744	OID	1.3.6.1.4.1.1466.115.121.1.38
1745	Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39
1746	Postal Address	1.3.6.1.4.1.1466.115.121.1.41
1747	Presentation Address	1.3.6.1.4.1.1466.115.121.1.43
1748	Printable String	1.3.6.1.4.1.1466.115.121.1.44
1749	Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58
1750	Telephone Number	1.3.6.1.4.1.1466.115.121.1.50
1751	UTC Time	1.3.6.1.4.1.1466.115.121.1.53

1752 For all other LDAP syntaxes, the attribute value is encoded, as the content of the <AttributeValue>
1753 element, by base64-encoding [RFC2045] the encompassing ASN.1 OCTET STRING-encoded LDAP
1754 attribute value. The `xsi:type` XML attribute MUST be set to **xs:base64Binary**. The profile-specific
1755 `Encoding` XML attribute is provided, with a value of `"LDAP"`.

1756 When comparing SAML attribute values for equality, the matching rules specified for the corresponding
1757 directory attribute type MUST be observed (case sensitivity, for example).

1758 **8.2.5 Profile-Specific Schema**

1759 The following schema listing shows how the profile-specific `Encoding` XML attribute is defined
1760 [SAMLX500-xsd]:

```

1761 <schema
1762   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
1763   xmlns="http://www.w3.org/2001/XMLSchema"
1764   elementFormDefault="unqualified"
1765   attributeFormDefault="unqualified"
1766   blockDefault="substitution"
1767   version="2.0">
1768   <annotation>
1769     <documentation>
1770       Document identifier: sstc-saml-schema-x500-2.0
1771       Location: http://www.oasis-
1772 open.org/committees/documents.php?wg_abbrev=security
1773       Revision history:
1774         V2.0 CD-03 (December, 2004):
1775         Custom schema for X.500 attribute profile, first published in
1776 SAML 2.0.
1777     </documentation>
1778   </annotation>
1779   <attribute name="Encoding" type="string"/>
1780 </schema>

```

1781 8.2.6 Example

1782 The following is an example of a mapping of the "givenName" directory attribute, representing the SAML
1783 assertion subject's first name. It's OBJECT IDENTIFIER is 2.5.4.42 and its LDAP syntax is Directory
1784 String.

```

1785 <saml:Attribute xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
1786   NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1787   Name="urn:oid:2.5.4.42" FriendlyName="givenName">
1788   <saml:AttributeValue xsi:type="xs:string"
1789     x500:Encoding="LDAP">Steven</saml:AttributeValue>
1790 </saml:Attribute>

```

1791 8.3 UUID Attribute Profile

1792 The UUID attribute profile standardizes the expression of UUID values as SAML attribute names and
1793 values. It is applicable when the attribute's source system is one that identifies an attribute or its value with
1794 a UUID.

1795 8.3.1 Required Information

1796 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:UUID

1797 **Contact information:** security-services-comment@lists.oasis-open.org

1798 **Description:** Given below.

1799 **Updates:** None.

1800 8.3.2 UUID and GUID Background

1801 UUIDs (Universally Unique Identifiers), also known as GUIDs (Globally Unique Identifiers), are used to
1802 define objects and subjects such that they are guaranteed uniqueness across space and time. UUIDs
1803 were originally used in the Network Computing System (NCS), and then used in the Open Software
1804 Foundation's (OSF) Distributed Computing Environment (DCE). Recently GUIDs have been used in
1805 Microsoft's COM and Active Directory/Windows 2000/2003 platform.

1806 A UUID is a 128 bit number, generated such that it should never be duplicated within the domain of
1807 interest. UUIDs are used to represent a wide range of objects including, but not limited to, subjects/users,
1808 groups of users and node names. A UUID, represented as a hexadecimal string, is as follows:

1809 `f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1810 In DCE and Microsoft Windows, the UUID is usually presented to the administrator in the form of a
1811 “friendly name”. For instance the above UUID could represent the user john.doe@example.com.

1812 **8.3.3 SAML Attribute Naming**

1813 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
1814 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1815 If the underlying representation of the attribute's name is a UUID, then the URN `uuid` namespace
1816 described in [<http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt>] is used. In this approach the
1817 `Name` XML attribute is based on the URN form of the underlying UUID that identifies the attribute.

1818 Example:

1819 `urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1820 If the underlying representation of the attribute's name is not a UUID, then any form of URI MAY be used
1821 in the `Name` XML attribute.

1822 For purposes of human readability, there may also be a requirement for some applications to carry an
1823 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in
1824 [SAMLCore]) MAY be used for this purpose.

1825 **8.3.3.1 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 [<http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt>]. The
1828 `FriendlyName` attribute plays no role in the comparison.

1829 **8.3.4 Profile-Specific XML Attributes**

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

1831 **8.3.5 SAML Attribute Values**

1832 In cases in which the attribute's value is also a UUID, the same URN syntax described above MUST be
1833 used to express the value within the `<AttributeValue>` element. The `xsi:type` XML attribute MUST
1834 be set to `xs:anyURI`.

1835 If the attribute's value is not a UUID, then there are no restrictions on the use of the `<AttributeValue>`
1836 element.

1837 **8.3.6 Example**

1838 The following is an example of a DCE Extended Registry Attribute, the "pre_auth_req" setting, which has a
1839 well-known UUID of 6c9d0ec8-dd2d-11cc-abdd-080009353559 and is integer-valued.

```
1840 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1841           Name="urn:uuid:6c9d0ec8-dd2d-11cc-abdd-080009353559"  
1842           FriendlyName="pre_auth_req">  
1843     <saml:AttributeValue xsi:type="xs:integer">1</saml:AttributeValue>  
1844 </saml:Attribute>
```

1845 8.4 DCE PAC Attribute Profile

1846 The DCE PAC attribute profile defines the expression of DCE PAC information as SAML attribute names
1847 and values. It is used to standardize a mapping between the primary information that makes up a DCE
1848 principal's identity and a set of SAML attributes. This profile builds on the UUID attribute profile defined in
1849 Section 8.3.

1850 8.4.1 Required Information

1851 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE (this is also the target namespace
1852 assigned in the corresponding DCE PAC attribute profile schema document [SAMLDCExsd])

1853 **Contact information:** security-services-comment@lists.oasis-open.org

1854 **Description:** Given below.

1855 **Updates:** None.

1856 8.4.2 PAC Description

1857 A DCE PAC is an extensible structure that can carry arbitrary DCE registry attributes, but a core set of
1858 information is common across principals and makes up the bulk of a DCE identity:

- 1859 • The principal's DCE "realm" or "cell"
- 1860 • The principal's unique identifier
- 1861 • The principal's primary DCE local group membership
- 1862 • The principal's set of DCE local group memberships (multi-valued)
- 1863 • The principal's set of DCE foreign group memberships (multi-valued)

1864 The primary value(s) of each of these attributes is a UUID.

1865 8.4.3 SAML Attribute Naming

1866 This profile defines a mapping of specific DCE information into SAML attributes, and thus defines actual
1867 specific attribute names, rather than a naming convention.

1868 For all attributes defined by this profile, the `NameFormat` XML attribute in `<Attribute>` elements MUST
1869 have the value `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1870 For purposes of human readability, there may also be a requirement for some applications to carry an
1871 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in
1872 [SAMLCore]) MAY be used for this purpose.

1873 See Section 8.4.6 for the specific attribute names defined by this profile.

1874 8.4.3.1 Attribute Name Comparison

1875 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
1876 values are equal in the sense of [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt]. The
1877 `FriendlyName` attribute plays no role in the comparison.

1878 8.4.4 Profile-Specific XML Attributes

1879 No additional XML attributes are defined for use with the `<Attribute>` element.

1880 8.4.5 SAML Attribute Values

1881 The primary value(s) of each of the attributes defined by this profile is a UUID. The URN syntax described
1882 in Section 8.3.5 of the UUID profile is used to represent such values.

1883 However, additional information associated with the UUID value is permitted by this profile, consisting of a
1884 friendly, human-readable string, and an additional UUID representing a DCE cell or realm. The additional
1885 information is carried in the <AttributeValue> element in FriendlyName and Realm XML attributes
1886 defined in the XML namespace urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE. Note
1887 that this is not the same as the FriendlyName XML attribute defined in [SAMLCore], although it has the
1888 same basic purpose.

1889 The following schema listing shows how the profile-specific XML attributes and complex type used in an
1890 xsi:type specification are defined [SAML DCE-xsd]:

```
1891 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
1892   xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
1893   xmlns="http://www.w3.org/2001/XMLSchema"  
1894   elementFormDefault="unqualified"  
1895   attributeFormDefault="unqualified"  
1896   blockDefault="substitution"  
1897   version="2.0">  
1898   <annotation>  
1899     <documentation>  
1900       Document identifier: sstc-saml-schema-dce-2.0  
1901       Location: http://www.oasis-  
1902 open.org/committees/documents.php?wg_abbrev=security  
1903       Revision history:  
1904         V2.0 CD-03 (December, 2004):  
1905         Custom schema for DCE attribute profile, first published in  
1906 SAML 2.0.  
1907     </documentation>  
1908   </annotation>  
1909   <complexType name="DCEValueType">  
1910     <simpleContent>  
1911       <extension base="anyURI">  
1912         <attribute ref="dce:Realm" use="optional"/>  
1913         <attribute ref="dce:FriendlyName" use="optional"/>  
1914       </extension>  
1915     </simpleContent>  
1916   </complexType>  
1917   <attribute name="Realm" type="anyURI"/>  
1918   <attribute name="FriendlyName" type="string"/>  
1919 </schema>
```

1920 8.4.6 Attribute Definitions

1921 The following are the set of SAML attributes defined by this profile. In each case, an xsi:type XML
1922 attribute MAY be included in the <AttributeValue> element, but MUST have the value
1923 **dce:DCEValueType**, where the dce prefix is arbitrary and MUST be bound to the XML namespace
1924 urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE.

1925 Note that such use of xsi:type will require validating attribute consumers to include the extension
1926 schema defined by this profile.

1927 8.4.6.1 Realm

1928 This single-valued attribute represents the SAML assertion subject's DCE realm or cell.

1929 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm

1930 The single <AttributeValue> element contains a UUID in URN form identifying the SAML assertion
1931 subject's DCE realm/cell, with an optional profile-specific `FriendlyName` XML attribute containing the
1932 realm's string name.

1933 **8.4.6.2 Principal**

1934 This single-valued attribute represents the SAML assertion subject's DCE principal identity.

1935 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal

1936 The single <AttributeValue> element contains a UUID in URN form identifying the SAML assertion
1937 subject's DCE principal identity, with an optional profile-specific `FriendlyName` XML attribute containing
1938 the principal's string name.

1939 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
1940 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
1941 8.4.6.1).

1942 **8.4.6.3 Primary Group**

1943 This single-valued attribute represents the SAML assertion subject's primary DCE group membership.

1944 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group

1945 The single <AttributeValue> element contains a UUID in URN form identifying the SAML assertion
1946 subject's primary DCE group, with an optional profile-specific `FriendlyName` XML attribute containing
1947 the group's string name.

1948 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
1949 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
1950 8.4.6.1).

1951 **8.4.6.4 Groups**

1952 This multi-valued attribute represents the SAML assertion subject's DCE local group memberships.

1953 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups

1954 Each <AttributeValue> element contains a UUID in URN form identifying a DCE group membership
1955 of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute containing
1956 the group's string name.

1957 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
1958 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
1959 8.4.6.1).

1960 **8.4.6.5 Foreign Groups**

1961 This multi-valued attribute represents the SAML assertion subject's DCE foreign group memberships.

1962 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-groups

1963 Each <AttributeValue> element contains a UUID in URN form identifying a DCE foreign group
1964 membership of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute
1965 containing the group's string name.

1966 The profile-specific `Realm` XML attribute MUST be included and MUST contain a UUID in URN form
1967 identifying the DCE realm/cell of the foreign group.

1968 8.4.7 Example

1969 The following is an example of the transformation of PAC data into SAML attributes belonging to a DCE
1970 principal named "jdoe" in realm "example.com", a member of the "cubicle-dwellers" and "underpaid" local
1971 groups and an "engineers" foreign group.

```
1972 <saml:Assertion  
1973   xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE" ...>  
1974   <saml:Issuer>...</saml:Issuer>  
1975   <saml:Subject>...</saml:Subject>  
1976   <saml:AttributeStatement>  
1977     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1978       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm">  
1979       <saml:AttributeValue xsi:type="dce:DCEValueType"  
1980 dce:FriendlyName="example.com">  
1981         urn:uuid:003c6cc1-9ff8-10f9-990f-004005b13a2b  
1982       </saml:AttributeValue>  
1983     </saml:Attribute>  
1984     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1985       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal">  
1986       <saml:AttributeValue xsi:type="dce:DCEValueType" dce:FriendlyName="jdoe">  
1987         urn:uuid:00305ed1-a1bd-10f9-a2d0-004005b13a2b  
1988       </saml:AttributeValue>  
1989     </saml:Attribute>  
1990     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1991       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group">  
1992       <saml:AttributeValue xsi:type="dce:DCEValueType"  
1993         dce:FriendlyName="cubicle-dwellers">  
1994         urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b  
1995       </saml:AttributeValue>  
1996     </saml:Attribute>  
1997     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1998       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups">  
1999       <saml:AttributeValue xsi:type="dce:DCEValueType"  
2000         dce:FriendlyName="cubicle-dwellers">  
2001         urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b  
2002       </saml:AttributeValue>  
2003       <saml:AttributeValue xsi:type="dce:DCEValueType"  
2004         dce:FriendlyName="underpaid">  
2005         urn:uuid:006a5a91-a2b7-10f9-824d-004005b13a2b  
2006       </saml:AttributeValue>  
2007     </saml:Attribute>  
2008     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
2009       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-groups">  
2010       <saml:AttributeValue xsi:type="dce:DCEValueType"  
2011         dce:FriendlyName="engineers"  
2012         dce:Realm="urn:uuid:00583221-a35f-10f9-8b6e-004005b13a2b">  
2013         urn:uuid:00099cf1-a355-10f9-9e95-004005b13a2b  
2014       </saml:AttributeValue>  
2015     </saml:Attribute>  
2016   </saml:AttributeStatement>  
2017 </saml:Assertion>
```

2018 8.5 XACML Attribute Profile

2019 SAML attribute assertions may be used as input to authorization decisions made according to the OASIS
2020 eXtensible Access Control Markup Language [XACML] standard specification. Since the SAML attribute
2021 format differs from the XACML attribute format, there is a mapping that must be performed. The XACML
2022 attribute profile facilitates this mapping by standardizing naming, value syntax, and additional attribute
2023 metadata. SAML attributes generated in conformance with this profile can be mapped automatically into
2024 XACML attributes and used as input to XACML authorization decisions.

2025 **8.5.1 Required Information**

2026 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML (this is also the target namespace
2027 assigned in the corresponding XACML profile schema document [SAMLXAC-xsd])

2028 **Contact information:** security-services-comment@lists.oasis-open.org

2029 **Description:** Given below.

2030 **Updates:** None.

2031 **8.5.2 SAML Attribute Naming**

2032 The `NameFormat` XML attribute in `<Attribute>` elements MUST be

2033 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

2034 The `Name` XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

2035 For purposes of human readability, there may also be a requirement for some applications to carry an
2036 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
2037 [SAMLCore]) MAY be used for this purpose, but is not translatable into an XACML attribute equivalent.

2038 **8.5.2.1 Attribute Name Comparison**

2039 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
2040 values are equal in a binary comparison. The `FriendlyName` attribute plays no role in the comparison.

2041 **8.5.3 Profile-Specific XML Attributes**

2042 XACML requires each attribute to carry an explicit data type. To supply this data type value, a new URI-
2043 valued XML attribute called `DataType` is defined in the XML namespace

2044 `urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML`.

2045 SAML `<Attribute>` elements conforming to this profile MUST include the namespace-qualified
2046 `DataType` attribute, or the value is presumed to be <http://www.w3.org/2001/XMLSchema#string>.

2047 While in principle any URI reference can be used as a data type, the standard values to be used are
2048 specified in Appendix A of the XACML 2.0 Specification [XACML]. If non-standard values are used, then
2049 each XACML PDP that will be consuming mapped SAML attributes with non-standard `DataType` values
2050 must be extended to support the new data types.

2051 **8.5.4 SAML Attribute Values**

2052 The syntax of the `<AttributeValue>` element's content MUST correspond to the data type expressed
2053 in the profile-specific `DataType` XML attribute appearing in the parent `<Attribute>` element. For data
2054 types corresponding to the types defined in Section 3.3 of [Schema2], the `xsi:type` XML attribute
2055 SHOULD also be used on the `<AttributeValue>` element(s).

2056 **8.5.5 Profile-Specific Schema**

2057 The following schema listing shows how the profile-specific `DataType` XML attribute is defined
2058 [SAMLXAC-xsd]:

```
2059 <schema  
2060     targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"  
2061     xmlns="http://www.w3.org/2001/XMLSchema"  
2062     elementFormDefault="unqualified"
```

```

2063     attributeFormDefault="unqualified"
2064     blockDefault="substitution"
2065     version="2.0">
2066     <annotation>
2067         <documentation>
2068             Document identifier: sstc-saml-schema-xacml-2.0
2069             Location: http://www.oasis-
2070 open.org/committees/documents.php?wg_abbrev=security
2071             Revision history:
2072             V2.0 CD-03 (December, 2004):
2073             Custom schema for XACML attribute profile, first published in
2074 SAML 2.0.
2075         </documentation>
2076     </annotation>
2077     <attribute name="DataType" type="anyURI"/>
2078 </schema>

```

2079 8.5.6 Example

2080 The following is an example of a mapping of the "givenName" LDAP/X.500 attribute, representing the
2081 SAML assertion subject's first name. It also illustrates that a single SAML attribute can conform to multiple
2082 attribute profiles when they are compatible with each other.

```

2083 <saml:Attribute
2084 xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
2085     xmlns:ldaprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP"
2086     xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string"
2087     ldaprof:Encoding="LDAP"
2088     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
2089     Name="urn:oid:2.5.4.42" FriendlyName="givenName">
2090     <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>
2091 </saml:Attribute>

```

9 References

- 2093 **[AES]** FIPS-197, Advanced Encryption Standard (AES), available from <http://www.nist.gov/>.
- 2094 **[Anders]** A suggestion on how to implement SAML browser bindings without using “Artifacts”,
2095 <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- 2096 **[ASN.1]** Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic
2097 notation, ITU-T Recommendation X.680, July 2002. See
2098 [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)
2099 [X.680](http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.680).
- 2100 **[eduPerson]** eduPerson.Idif. See <http://www.educase.edu/eduperson>.
- 2101 **[LDAP]** J. Hodges et al., Lightweight Directory Access Protocol (v3): Technical Specification,
2102 IETF RFC 3377, September 2002. See <http://www.ietf.org/rfc/rfc3377.txt>.
- 2103 **[Mealling]** P Leach et al, A UUID URN Namespace. Internet-Draft, draft-mealling-uuid-urn-03.
2104 January 2004
- 2105 **[MSURL]** Microsoft technical support article,
2106 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- 2107 **[NSCookie]** Persistent Client State HTTP Cookies, Netscape documentation. See
2108 http://wp.netscape.com/newsref/std/cookie_spec.html.
- 2109 **[PAOS]** Aarts, R., “Liberty Reverse HTTP Binding for SOAP Specification”, Version: 1.0,
2110 <https://www.projectliberty.org/specs/liberty-paos-v1.0.pdf>
- 2111 **[Rescorla-Sec]** E. Rescorla et al., Guidelines for Writing RFC Text on Security Considerations,
2112 <http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt>.
- 2113 **[RFC1738]** Uniform Resource Locators (URL), <http://www.ietf.org/rfc/rfc1738.txt>
- 2114 **[RFC1750]** Randomness Recommendations for Security. <http://www.ietf.org/rfc/rfc1750.txt>
- 2115 **[RFC1945]** Hypertext Transfer Protocol -- HTTP/1.0, <http://www.ietf.org/rfc/rfc1945.txt>.
- 2116 **[RFC2045]** Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message
2117 Bodies, <http://www.ietf.org/rfc/rfc2045.txt>
- 2118 **[RFC2119]** S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, IETF RFC
2119 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.
- 2120 **[RFC2246]** The TLS Protocol Version 1.0, <http://www.ietf.org/rfc/rfc2246.txt>.
- 2121 **[RFC2256]** M. Wahl, RFC 2256 - A Summary of the X.500(96) User Schema for use with LDAPv3,
2122 December 1997
- 2123 **[RFC2279]** UTF-8, a transformation format of ISO 10646, <http://www.ietf.org/rfc/rfc2279.txt>.
- 2124 **[RFC2616]** Hypertext Transfer Protocol -- HTTP/1.1, <http://www.ietf.org/rfc/rfc2616.txt>.
- 2125 **[RFC2617]** HTTP Authentication: Basic and Digest Access Authentication, IETF RFC 2617,
2126 <http://www.ietf.org/rfc/rfc2617.txt>.
- 2127 **[RFC2798]** M. Smith, Definition of the inetOrgPerson LDAP Object Class, IETF RFC 2798, April
2128 200. See <http://www.ietf.org/rfc/rfc2798.txt>.
- 2129 **[RFC2965]** D. Cristol et al., HTTP State Management Mechanism, IETF RFC 2965, October 2000.
2130 See <http://www.ietf.org/rfc/rfc2965.txt>.
- 2131 **[RFC3061]** M. Mealling, A URN Namespace of Object Identifiers, IETF RFC 3061, February 2001.
2132 See <http://www.ietf.org/rfc/rfc3061.txt>.
- 2133 **[SAMLBind]** S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language (SAML)*
2134 *V2.0*. OASIS SSTC, December 2004. Document ID sstc-saml-bindings-2.0-cd-03. See
2135 <http://www.oasis-open.org/committees/security/>.

- 2136 **[SAMLConform]** P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, December 2004. Document ID sstc-saml-conformance-2.0-cd-03. <http://www.oasis-open.org/committees/security/>.
- 2137
- 2138
- 2139 **[SAMLCore]** S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, December 2004. Document ID sstc-saml-core-2.0-cd-03. See <http://www.oasis-open.org/committees/security/>.
- 2140
- 2141
- 2142 **[SAML DCE-xsd]** S. Cantor et al., SAML DCE PAC attribute profile schema. OASIS SSTC, December 2004. Document ID sstc-saml-schema-dce-2.0. See <http://www.oasis-open.org/committees/security/>.
- 2143
- 2144
- 2145 **[SAML ECP-xsd]** S. Cantor et al., SAML ECP profile schema. OASIS SSTC, December 2004. Document ID sstc-saml-schema-ecp-2.0. See <http://www.oasis-open.org/committees/security/>.
- 2146
- 2147 **[SAML Gloss]** J. Hodges et al., *Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, December 2004. Document ID sstc-saml-glossary-2.0-cd-03. See <http://www.oasis-open.org/committees/security/>.
- 2148
- 2149
- 2150 **[SAML LDAP-xsd]** S. Cantor et al., SAML LDAP attribute profile schema. OASIS SSTC, December 2004. Document ID sstc-saml-schema-ldap-2.0. See <http://www.oasis-open.org/committees/security/>.
- 2151
- 2152
- 2153 **[SAML Meta]** S. Cantor et al., *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, December 2004. Document ID sstc-saml-metadata-2.0-cd-03. See <http://www.oasis-open.org/committees/security/>.
- 2154
- 2155
- 2156 **[SAML Reqs]** Darren Platt et al., SAML Requirements and Use Cases, OASIS, April 2002, <http://www.oasis-open.org/committees/security/>.
- 2157
- 2158 **[SAML Sec]** F. Hirsch et al., *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, December 2004. Document ID sstc-saml-sec-consider-2.0-cd-03. See <http://www.oasis-open.org/committees/security/>.
- 2159
- 2160
- 2161 **[SAML Web]** OASIS Security Services Technical Committee website, <http://www.oasis-open.org/committees/security/>.
- 2162
- 2163 **[SAML XAC-xsd]** S. Cantor et al., SAML XACML attribute profile schema. OASIS SSTC, December 2004. Document ID sstc-saml-schema-xacml-2.0. See <http://www.oasis-open.org/committees/security/>.
- 2164
- 2165
- 2166 **[Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web Consortium Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-1/>. Note that this specification normatively references [Schema2], listed below.
- 2167
- 2168
- 2169 **[Schema2]** Paul V. Biron, Ashok Malhotra, XML Schema Part 2: Datatypes, W3C Recommendation 02 May 2001, <http://www.w3.org/TR/xmlschema-2/>
- 2170
- 2171 **[SESSION]** RL "Bob" Morgan, Support of target web server sessions in Shibboleth, <http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt>
- 2172
- 2173 **[ShibMarlena]** Marlena Erdos, Shibboleth Architecture DRAFT v1.1, <http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html> .
- 2174
- 2175 **[SOAP1.1]** D. Box et al., Simple Object Access Protocol (SOAP) 1.1, World Wide Web Consortium Note, May 2000, <http://www.w3.org/TR/SOAP>.
- 2176
- 2177 **[SSL3]** A. Frier et al., The SSL 3.0 Protocol, Netscape Communications Corp, November 1996.
- 2178
- 2179 **[WEBSSO]** RL "Bob" Morgan, Interactions between Shibboleth and local-site web sign-on services, <http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-websso-00.txt>
- 2180
- 2181 **[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>.
- 2182
- 2183
- 2184
- 2185 **[XML Enc]** D. Eastlake et al., XML Encryption Syntax and Processing, <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>, World Wide Web
- 2186

2187		Consortium.
2188	[XMLSig]	D. Eastlake et al., XML-Signature Syntax and Processing, World Wide Web Consortium, http://www.w3.org/TR/xmlsig-core/ .
2189		
2190	[XACML]	T. Moses, ed., <i>OASIS eXtensible Access Control Markup Language (XACML) Versions 1.0, 1.1, and 2.0</i> . Available on the OASIS XACML TC web page at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml .
2191		
2192		

Appendix A. Acknowledgments

2194 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
2195 Committee, whose voting members at the time of publication were:

- 2196 • Conor Cahill, AOL
- 2197 • John Hughes, Atos Origin
- 2198 • Hal Lockhart, BEA Systems
- 2199 • Mike Beach, Boeing
- 2200 • Rebekah Metz, Booz Allen Hamilton
- 2201 • Rick Randall, Booz Allen Hamilton
- 2202 • Ronald Jacobson, Computer Associates
- 2203 • Paul Madsen, Entrust
- 2204 • Dana Kaufman, Forum Systems
- 2205 • Paula Austel, IBM
- 2206 • Michael McIntosh, IBM
- 2207 • Anthony Nadalin, IBM
- 2208 • Nick Ragouzis, Individual
- 2209 • Scott Cantor, Internet2
- 2210 • Bob Morgan, Internet2
- 2211 • Peter Davis, Neustar
- 2212 • Jeff Hodges, Neustar
- 2213 • Frederick Hirsch, Nokia
- 2214 • John Kemp, Nokia
- 2215 • Abbie Barbir, Nortel Networks
- 2216 • Scott Kiestler, Novell
- 2217 • Cameron Morris, Novell
- 2218 • Charles Knouse, Oblix
- 2219 • Steve Anderson, OpenNetwork
- 2220 • Ari Kermaier, Oracle
- 2221 • Vamsi Motukuru, Oracle
- 2222 • Darren Platt, Ping Identity
- 2223 • Prateek Mishra, Principal Identity
- 2224 • Jim Lien, RSA Security
- 2225 • Rob Philpott, RSA Security
- 2226 • Dipak Chopra, SAP
- 2227 • Jahan Moreh, Sigaba
- 2228 • Bhavna Bhatnagar, Sun Microsystems
- 2229 • Eve Maler, Sun Microsystems
- 2230 • Ronald Monzillo, Sun Microsystems
- 2231 • Emily Xu, Sun Microsystems
- 2232 • Greg Whitehead, Trustgenix

2233 The editors also would like to acknowledge the following people for their contributions to previous versions
2234 of the OASIS Security Assertions Markup Language Standard:

- 2235 • Stephen Farrell, Baltimore Technologies
- 2236 • David Orchard, BEA Systems
- 2237 • Krishna Sankar, Cisco Systems
- 2238 • Zahid Ahmed, CommerceOne
- 2239 • Carlisle Adams, Entrust
- 2240 • Tim Moses, Entrust
- 2241 • Nigel Edwards, Hewlett-Packard
- 2242 • Joe Pato, Hewlett-Packard
- 2243 • Bob Blakley, IBM
- 2244 • Marlena Erdos, IBM
- 2245 • Marc Chanliau, Netegrity
- 2246 • Chris McLaren, Netegrity
- 2247 • Lynne Rosenthal, NIST
- 2248 • Mark Skall, NIST
- 2249 • Simon Godik, Overxeer
- 2250 • Charles Norwood, SAIC
- 2251 • Evan Prodromou, Securant
- 2252 • Robert Griffin, RSA Security (former editor)
- 2253 • Sai Allarvarpu, Sun Microsystems
- 2254 • Chris Ferris, Sun Microsystems
- 2255 • Emily Xu, Sun Microsystems
- 2256 • Mike Myers, Traceroute Security
- 2257 • Phillip Hallam-Baker, VeriSign (former editor)
- 2258 • James Vanderbeek, Vodafone
- 2259 • Mark O'Neill, Vordel
- 2260 • Tony Palmer, Vordel

2261 Finally, the editors wish to acknowledge the following people for their contributions of material used as
2262 input to the OASIS Security Assertions Markup Language specifications:

- 2263 • Thomas Gross, IBM
- 2264 • Birgit Pfitzmann, IBM

2265 **Appendix B. Notices**

2266 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
2267 might be claimed to pertain to the implementation or use of the technology described in this document or
2268 the extent to which any license under such rights might or might not be available; neither does it represent
2269 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
2270 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
2271 available for publication and any assurances of licenses to be made available, or the result of an attempt
2272 made to obtain a general license or permission for the use of such proprietary rights by implementors or
2273 users of this specification, can be obtained from the OASIS Executive Director.

2274 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
2275 other proprietary rights which may cover technology that may be required to implement this specification.
2276 Please address the information to the OASIS Executive Director.

2277 **Copyright © OASIS Open 2004. All Rights Reserved.**

2278 This document and translations of it may be copied and furnished to others, and derivative works that
2279 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
2280 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
2281 this paragraph are included on all such copies and derivative works. However, this document itself may
2282 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
2283 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
2284 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
2285 into languages other than English.

2286 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
2287 or assigns.

2288 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2289 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
2290 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
2291 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.