



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

Committee Draft 04, 15 January 2005

Document identifier:

sstc-saml-profiles-2.0-cd-04

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
Thomas Wisniewski, 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
Paul Madsen, NTT
Charles Knouse, Oblix
Steve Anderson, OpenNetwork
Prateek Mishra, Principal Identity
John Linn, RSA Security
Rob Philpott, RSA Security

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

51 **Abstract:**

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

55 **Status:**

56 This is a **Committee Draft** approved by the Security Services Technical Committee on 15
57 January 2005.

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

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

Table of Contents

67		
68	1 Introduction.....	7
69	1.1 Profile Concepts.....	7
70	1.2 Notation.....	7
71	2 Specification of Additional Profiles.....	10
72	2.1 Guidelines for Specifying Profiles.....	10
73	2.2 Guidelines for Specifying Attribute Profiles.....	10
74	3 Confirmation Method Identifiers.....	12
75	3.1 Holder of Key.....	12
76	3.2 Sender Vouches.....	12
77	3.3 Bearer.....	13
78	4 SSO Profiles of SAML.....	14
79	4.1 Web Browser SSO Profile.....	14
80	4.1.1 Required Information.....	14
81	4.1.2 Profile Overview.....	14
82	4.1.3 Profile Description.....	16
83	4.1.3.1 HTTP Request to Service Provider.....	16
84	4.1.3.2 Service Provider Determines Identity Provider.....	16
85	4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider.....	16
86	4.1.3.4 Identity Provider Identifies Principal.....	17
87	4.1.3.5 Identity Provider Issues <Response> to Service Provider.....	17
88	4.1.3.6 Service Provider Grants or Denies Access to User Agent.....	17
89	4.1.4 Use of Authentication Request Protocol.....	18
90	4.1.4.1 <AuthnRequest> Usage.....	18
91	4.1.4.2 <Response> Usage.....	18
92	4.1.4.3 <Response> Message Processing Rules.....	19
93	4.1.4.4 Artifact-Specific <Response> Message Processing Rules.....	19
94	4.1.4.5 POST-Specific Processing Rules.....	20
95	4.1.5 Unsolicited Responses.....	20
96	4.1.6 Use of Metadata.....	20
97	4.2 Enhanced Client or Proxy (ECP) Profile.....	21
98	4.2.1 Required Information.....	21
99	4.2.2 Profile Overview.....	21
100	4.2.3 Profile Description.....	24
101	4.2.3.1 ECP issues HTTP Request to Service Provider.....	24
102	4.2.3.2 Service Provider Issues <AuthnRequest> to ECP.....	24
103	4.2.3.3 ECP Determines Identity Provider.....	25
104	4.2.3.4 ECP issues <AuthnRequest> to Identity Provider.....	25
105	4.2.3.5 Identity Provider Identifies Principal.....	25
106	4.2.3.6 Identity Provider issues <Response> to ECP, targeted at service provider.....	25
107	4.2.3.7 ECP Conveys <Response> Message to Service Provider.....	26
108	4.2.3.8 Service Provider Grants or Denies Access to Principal.....	26
109	4.2.4 ECP Profile Schema Usage.....	26
110	4.2.4.1 PAOS Request Header Block: SP to ECP.....	27

111	4.2.4.2 ECP Request Header Block: SP to ECP.....	28
112	4.2.4.3 ECP RelayState Header Block: SP to ECP.....	28
113	4.2.4.4 ECP Response Header Block: IdP to ECP.....	29
114	4.2.4.5 PAOS Response Header Block: ECP to SP.....	30
115	4.2.5 Security Considerations.....	31
116	4.3 Identity Provider Discovery Profile.....	31
117	4.3.1 Common Domain Cookie.....	31
118	4.3.2 Setting the Common Domain Cookie.....	31
119	4.3.3 Obtaining the Common Domain Cookie.....	32
120	4.4 Single Logout Profile.....	32
121	4.4.1 Required Information.....	33
122	4.4.2 Profile Overview.....	33
123	4.4.3 Profile Description.....	34
124	4.4.3.1 <LogoutRequest> Issued by Session Participant to Identity Provider.....	34
125	4.4.3.2 Identity Provider Determines Session Participants.....	35
126	4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session Participant/Authority.....	35
127	4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity Provider.....	36
128	4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant.....	36
129	4.4.4 Use of Single Logout Protocol.....	36
130	4.4.4.1 <LogoutRequest> Usage.....	36
131	4.4.4.2 <LogoutResponse> Usage.....	37
132	4.4.5 Use of Metadata.....	37
133	4.5 Name Identifier Management Profile.....	37
134	4.5.1 Required Information.....	37
135	4.5.2 Profile Overview.....	38
136	4.5.3 Profile Description.....	38
137	4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider.....	38
138	4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service Provider.....	39
139	4.5.4 Use of Name Identifier Management Protocol.....	40
140	4.5.4.1 <ManageNameIDRequest> Usage.....	40
141	4.5.4.2 <ManageNameIDResponse> Usage.....	40
142	4.5.5 Use of Metadata.....	40
143	5 Artifact Resolution Profile.....	41
144	5.1 Required Information.....	41
145	5.2 Profile Overview.....	41
146	5.3 Profile Description.....	42
147	5.3.1 <ArtifactResolve> issued by Requesting Entity.....	42
148	5.3.2 <ArtifactResponse> issued by Responding Entity.....	42
149	5.4 Use of Artifact Resolution Protocol.....	42
150	5.4.1 <ArtifactResolve> Usage.....	42
151	5.4.2 <ArtifactResponse> Usage.....	43
152	5.5 Use of Metadata.....	43
153	6 Assertion Query/Request Profile.....	44
154	6.1 Required Information.....	44
155	6.2 Profile Overview.....	44
156	6.3 Profile Description.....	45

157	6.3.1 Query/Request issued by SAML Requester.....	45
158	6.3.2 <Response> issued by SAML Authority.....	45
159	6.4 Use of Query/Request Protocol.....	45
160	6.4.1 Query/Request Usage.....	45
161	6.4.2 <Response> Usage.....	45
162	6.5 Use of Metadata.....	46
163	7 Name Identifier Mapping Profile.....	47
164	7.1 Required Information.....	47
165	7.2 Profile Overview.....	47
166	7.3 Profile Description.....	48
167	7.3.1 <NameIDMappingRequest> issued by Requesting Entity.....	48
168	7.3.2 <NameIDMappingResponse> issued by Identity Provider.....	48
169	7.4 Use of Name Identifier Mapping Protocol.....	48
170	7.4.1 <NameIDMappingRequest> Usage.....	48
171	7.4.2 <NameIDMappingResponse> Usage.....	48
172	7.4.2.1 Limiting Use of Mapped Identifier.....	49
173	7.5 Use of Metadata.....	49
174	8 SAML Attribute Profiles.....	50
175	8.1 Basic Attribute Profile.....	50
176	8.1.1 Required Information.....	50
177	8.1.2 SAML Attribute Naming.....	50
178	8.1.2.1 Attribute Name Comparison.....	50
179	8.1.3 Profile-Specific XML Attributes.....	50
180	8.1.4 SAML Attribute Values.....	50
181	8.1.5 Example.....	50
182	8.2 X.500/LDAP Attribute Profile.....	50
183	8.2.1 Required Information.....	51
184	8.2.2 SAML Attribute Naming.....	51
185	8.2.2.1 Attribute Name Comparison.....	51
186	8.2.3 Profile-Specific XML Attributes.....	51
187	8.2.4 SAML Attribute Values.....	51
188	8.2.5 Profile-Specific Schema.....	52
189	8.2.6 Example.....	53
190	8.3 UUID Attribute Profile.....	53
191	8.3.1 Required Information.....	53
192	8.3.2 UUID and GUID Background.....	53
193	8.3.3 SAML Attribute Naming.....	54
194	8.3.3.1 Attribute Name Comparison.....	54
195	8.3.4 Profile-Specific XML Attributes.....	54
196	8.3.5 SAML Attribute Values.....	54
197	8.3.6 Example.....	54
198	8.4 DCE PAC Attribute Profile.....	55
199	8.4.1 Required Information.....	55
200	8.4.2 PAC Description.....	55

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

223 1 Introduction

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

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

231 1.1 Profile Concepts

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

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

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

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

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

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

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

257 1.2 Notation

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

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

265 described in IETF RFC 2119 [RFC2119].

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

267 Example code listings appear like this.

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

269 Conventional XML namespace prefixes are used throughout this specification to stand for their respective
270 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].
dse:	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.

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

274

2 Specification of Additional Profiles

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

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

2.1 Guidelines for Specifying Profiles

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

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

2.2 Guidelines for Specifying Attribute Profiles

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

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

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

318

3 Confirmation Method Identifiers

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

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

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

3.1 Holder of Key

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

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

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

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

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

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

3.2 Sender Vouches

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

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

362 **3.3 Bearer**

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

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

367 **Example:** The bearer of the assertion can confirm itself as the subject, provided the assertion is delivered
368 in a message sent to "<https://www.serviceprovider.com/saml/consumer>" before 1:37 PM GMT on March
369 19th, 2004, in response to a request with ID "_1234567890".

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

376 4 SSO Profiles of SAML

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

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

384 4.1 Web Browser SSO Profile

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

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

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

396 4.1.1 Required Information

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

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

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

401 **Description:** Given below.

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

403 4.1.2 Profile Overview

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

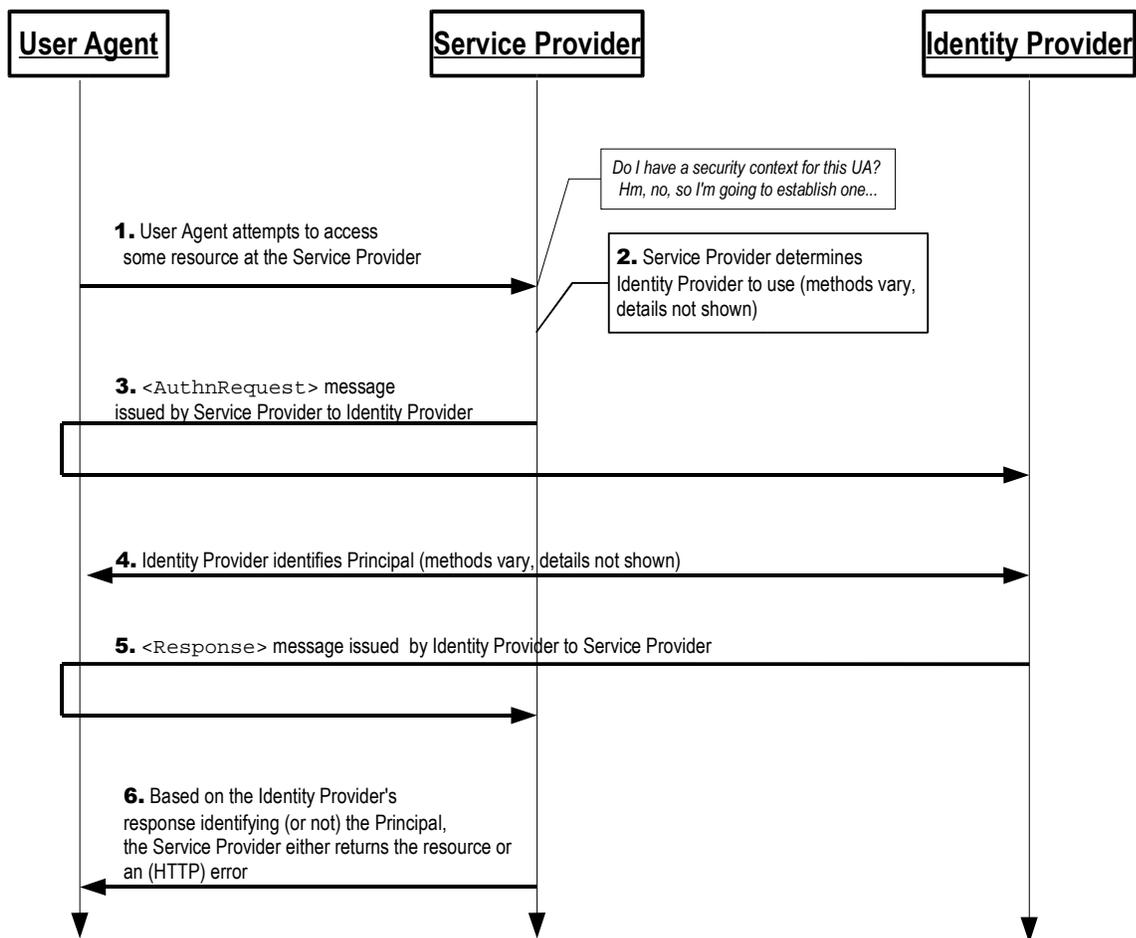


Figure 1

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

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

410 **2. Service Provider Determines Identity Provider**

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

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

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

419 **4. Identity Provider identifies Principal**

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

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

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

429 6. Service Provider grants or denies access to Principal

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

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

435 4.1.3 Profile Description

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

438 Single Sign-On Service

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

441 Assertion Consumer Service

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

444 4.1.3.1 HTTP Request to Service Provider

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

451 4.1.3.2 Service Provider Determines Identity Provider

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

458 4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider

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

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

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

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

478 **4.1.3.4 Identity Provider Identifies Principal**

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

486 **4.1.3.5 Identity Provider Issues <Response> to Service Provider**

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

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

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

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

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

506 **4.1.3.6 Service Provider Grants or Denies Access to User Agent**

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

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

512 **4.1.4 Use of Authentication Request Protocol**

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

517 **4.1.4.1 <AuthnRequest> Usage**

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

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

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

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

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

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

540 **4.1.4.2 <Response> Usage**

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

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

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

576 4.1.4.3 `<Response>` Message Processing Rules

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

- 578 • Verify any signatures present on the assertion(s) or the response
- 579 • Verify that the `Recipient` attribute in any bearer `<SubjectConfirmationData>` matches the
580 assertion consumer service URL to which the `<Response>` or artifact was delivered
- 581 • Verify that the `NotOnOrAfter` attribute in any bearer `<SubjectConfirmationData>` has not
582 passed, subject to allowable clock skew between the providers
- 583 • Verify that the `InResponseTo` attribute in the bearer `<SubjectConfirmationData>` equals the ID
584 of its original `<AuthnRequest>` message, unless the response is unsolicited (see Section 4.1.5), in
585 which case the attribute MUST NOT be present
- 586 • Verify that any assertions relied upon are valid in other respects

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

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

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

594 4.1.4.4 Artifact-Specific `<Response>` Message Processing Rules

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

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

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

601 **4.1.4.5 POST-Specific Processing Rules**

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

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

607 **4.1.5 Unsolicited Responses**

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

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

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

619 **4.1.6 Use of Metadata**

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

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

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

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

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

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

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

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

653 4.2 Enhanced Client or Proxy (ECP) Profile

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

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

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

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

668 4.2.1 Required Information

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

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

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

674 **Description:** Given below.

675 **Updates:** None.

676 4.2.2 Profile Overview

677 As introduced above, the ECP profile specifies interactions between enhanced clients or proxies and

678 service providers and identity providers. It is a specific application of the SSO profile described in Section
679 4.1. If not otherwise specified by this profile, and if not specific to the use of browser-based bindings, the
680 rules specified in Section 4.1 MUST be observed.

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

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

684 This allows a service provider to make an authentication request to the ECP without the need to know
685 or discover the appropriate identity provider (effectively bypassing step 2 of the SSO profile in Section
686 4.1).

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

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

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

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

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

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

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

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

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

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

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

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

721 In step 2, the service provider issues an <AuthnRequest> message to the ECP, which is to be

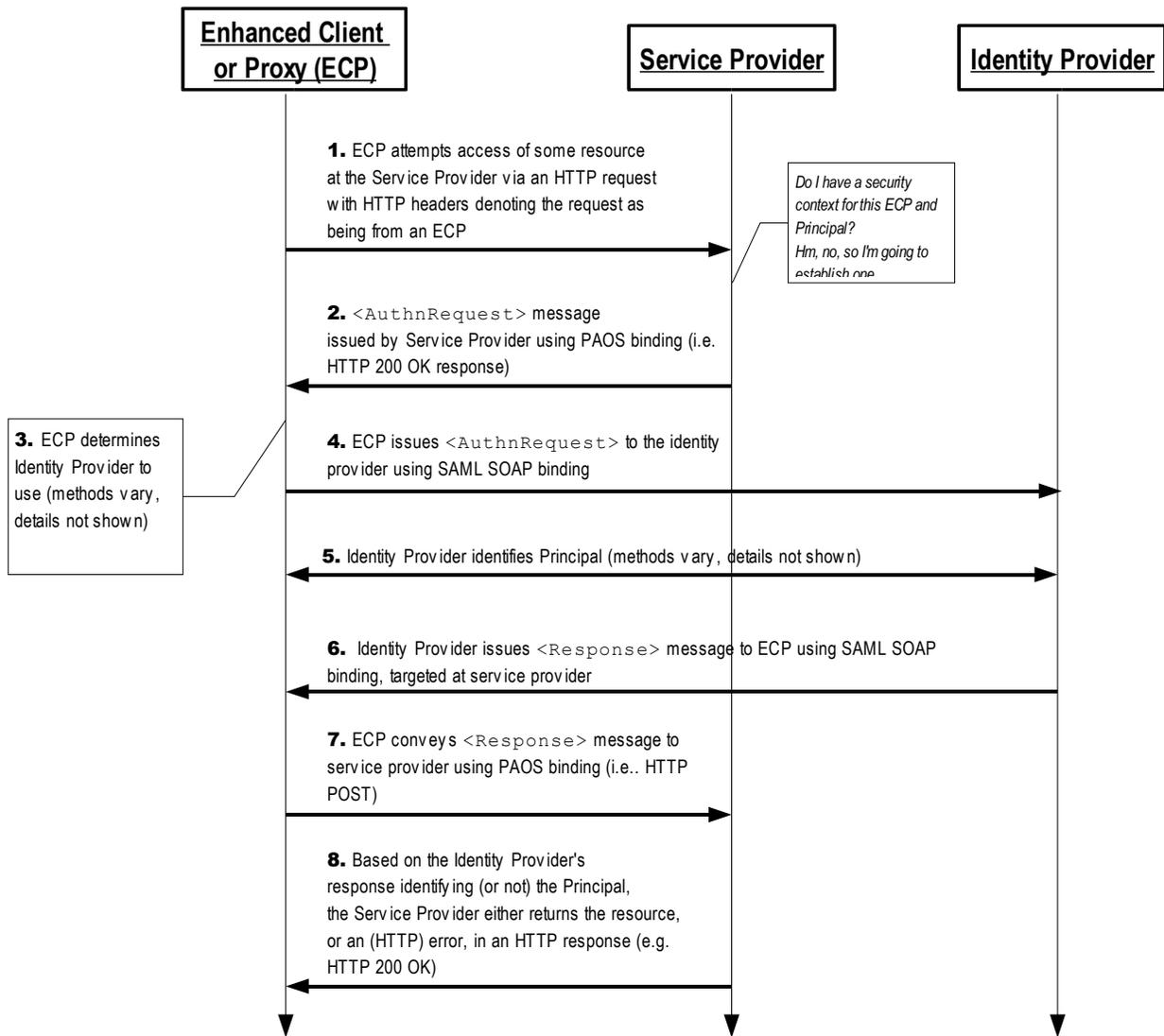


Figure 2

723 delivered by the ECP to the appropriate identity provider. The Reverse SOAP (PAOS) binding
 724 [SAMLBind] is used here.

725 **3. ECP Determines Identity Provider**

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

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

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

735 **5. Identity Provider identifies Principal**

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

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

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

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

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

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

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

750 **4.2.3 Profile Description**

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

752 **4.2.3.1 ECP issues HTTP Request to Service Provider**

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

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

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

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

768 **4.2.3.2 Service Provider Issues <AuthnRequest> to ECP**

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

773 The SOAP envelope MUST contain:

- 774 1. An <AuthnRequest> element in the SOAP body, intended for the ultimate SOAP recipient, the
775 identity provider.
- 776 2. A PAOS SOAP header block targeted at the ECP using the SOAP actor value of

778 `http://schemas.xmlsoap.org/soap/actor/next`. This header block provides control
779 information such as the URL to which to send the response in this solicit-response message
780 exchange pattern.

781 3. An ECP profile-specific Request SOAP header block targeted at the ECP using the SOAP actor
782 `http://schemas.xmlsoap.org/soap/actor/next`. The ECP Request header block defines
783 information related to the authentication request that the ECP may need to process it, such as a list
784 of identity providers acceptable to the service provider, whether the ECP may interact with the
785 principal through the client, and the service provider's human-readable name that may be displayed
786 to the principal.

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

790 **4.2.3.3 ECP Determines Identity Provider**

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

792 **4.2.3.4 ECP issues <AuthnRequest> to Identity Provider**

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

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

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

804 **4.2.3.5 Identity Provider Identifies Principal**

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

812 **4.2.3.6 Identity Provider issues <Response> to ECP, targeted at service provider**

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

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

820 4.2.3.7 ECP Conveys <Response> Message to Service Provider

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

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

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

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

838 4.2.3.8 Service Provider Grants or Denies Access to Principal

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

844 4.2.4 ECP Profile Schema Usage

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

```
847 <schema
848   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
849   xmlns="http://www.w3.org/2001/XMLSchema"
850   xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
851   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
852   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
853   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
854   elementFormDefault="unqualified"
855   attributeFormDefault="unqualified"
856   blockDefault="substitution"
857   version="2.0">
858   <import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
859     schemaLocation="sstc-saml-schema-protocol-2.0.xsd"/>
860   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
861     schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
862   <import namespace="http://schemas.xmlsoap.org/soap/envelope/"
863     schemaLocation="http://schemas.xmlsoap.org/soap/envelope/" />
864   <annotation>
865     <documentation>
866       Document identifier: sstc-saml-schema-ecp-2.0
867       Location: http://www.oasis-
868 open.org/committees/documents.php?wg_abbrev=security
869       Revision history:
870         V2.0 CD-04 (January, 2005):
871         Custom schema for ECP profile, first published in SAML 2.0.
```

```

872     </documentation>
873 </annotation>

874 <element name="Request" type="ecp:RequestType"/>
875 <complexType name="RequestType">
876   <sequence>
877     <element ref="saml:Issuer"/>
878     <element ref="samlp:IDPList" minOccurs="0"/>
879   </sequence>
880   <attribute ref="S:mustUnderstand" use="required"/>
881   <attribute ref="S:actor" use="required"/>
882   <attribute name="ProviderName" type="string" use="optional"/>
883   <attribute name="IsPassive" type="boolean" use="optional"/>
884 </complexType>
885
886 <element name="Response" type="ecp:ResponseType"/>
887 <complexType name="ResponseType">
888   <attribute ref="S:mustUnderstand" use="required"/>
889   <attribute ref="S:actor" use="required"/>
890   <attribute name="AssertionConsumerServiceURL" type="anyURI"
891 use="required"/>
892 </complexType>
893
894 <element name="RelayState" type="ecp:RelayStateType"/>
895 <complexType name="RelayStateType">
896   <simpleContent>
897     <extension base="string">
898       <attribute ref="S:mustUnderstand" use="required"/>
899       <attribute ref="S:actor" use="required"/>
900     </extension>
901   </simpleContent>
902 </complexType>
903 </schema>

```

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

905 **4.2.4.1 PAOS Request Header Block: SP to ECP**

906 The PAOS Request header block signals the use of PAOS processing and includes the following
907 attributes:

908 `responseConsumerURL` [Required]

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

914 `service` [Required]

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

917 `SOAP-ENV:mustUnderstand` [Required]

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

920 `SOAP-ENV:actor` [Required]

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

922 `messageID` [Optional]

923 Allows optional response correlation. It **MAY** be used in this profile, but is **NOT** required, since this

924 functionality is provided by the SAML protocol layer, via the `ID` attribute in the `<AuthnRequest>` and
925 the `InResponseTo` attribute in the `<Response>`.

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

927 **4.2.4.2 ECP Request Header Block: SP to ECP**

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

931 `SOAP-ENV:mustUnderstand` [Required]

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

934 `SOAP-ENV:actor` [Required]

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

936 `ProviderName` [Optional]

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

938 `IsPassive` [Optional]

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

942 `<saml:Issuer>` [Required]

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

946 `<samlp:IDPList>` [Optional]

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

949 **4.2.4.3 ECP RelayState Header Block: SP to ECP**

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

953 `SOAP-ENV:mustUnderstand` [Required]

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

955 `SOAP-ENV:actor` [Required]

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

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

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

```

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

```

999 As noted above, the PAOS and ECP header blocks are removed from the SOAP message by the ECP
1000 before the authentication request is forwarded to the identity provider. An example authentication request
1001 from the ECP to the identity provider is as follows:

```

1002 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
1003   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1004   <SOAP-ENV:Body>
1005     <samlp:AuthnRequest> ... </samlp:AuthnRequest>
1006   </SOAP-ENV:Body>
1007 </SOAP-ENV:Envelope>

```

1008 4.2.4.4 ECP Response Header Block: IdP to ECP

1009 The ECP response SOAP header block **MUST** be used on the response from the identity provider to the
1010 ECP. It contains the following attributes:

1011 SOAP-ENV:mustUnderstand [Required]

1012 The value **MUST** be 1 (true). A SOAP fault **MUST** be generated if the ECP header block is not
1013 understood.

1014 SOAP-ENV:actor [Required]

1015 The value **MUST** be http://schemas.xmlsoap.org/soap/actor/next.

1016 AssertionConsumerServiceURL [Required]

1017 Set by the identity provider based on the <AuthnRequest> message or the service provider's
1018 metadata obtained by the identity provider.

1019 The ECP MUST confirm that this value corresponds to the value the ECP obtained in the
1020 responseConsumerURL in the PAOS Request SOAP header block it received from the service
1021 provider. Since the responseConsumerURL MAY be relative and the
1022 AssertionConsumerServiceURL is absolute, some processing/normalization may be required.

1023 This mechanism is used for security purposes to confirm the correct response destination. If the
1024 values do not match, then the ECP MUST generate a SOAP fault response to the service provider
1025 and MUST NOT return the SAML response.

1026 The ECP Response SOAP header has no element content.

1027 Following is an example of an IdP-to-ECP response.

```
1028 <SOAP-ENV:Envelope
1029     xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
1030     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1031     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
1032   <SOAP-ENV:Header>
1033     <ecp:Response SOAP-ENV:mustUnderstand="1" SOAP-
1034 ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
1035 AssertionConsumerServiceURL="https://ServiceProvider.example.com/ecp_assertion_cons
1036 mer"/>
1037   </SOAP-ENV:Header>
1038   <SOAP-ENV:Body>
1039     <samlp:Response> ... </samlp:Response>
1040   </SOAP-ENV:Body>
1041 </SOAP-ENV:Envelope>
```

1042 4.2.4.5 PAOS Response Header Block: ECP to SP

1043 The PAOS Response header block includes the following attributes:

1044 SOAP-ENV:mustUnderstand [Required]

1045 The value MUST be 1 (true). A SOAP fault MUST be generated if the PAOS header block is not
1046 understood.

1047 SOAP-ENV:actor [Required]

1048 The value MUST be http://schemas.xmlsoap.org/soap/actor/next.

1049 refToMessageID [Optional]

1050 Allows correlation with the PAOS request. This optional attribute (and the header block as a whole)
1051 MUST be added by the ECP if the corresponding PAOS request specified the messageID attribute.
1052 Note that the equivalent functionality is provided in SAML using <AuthnRequest> and <Response>
1053 correlation.

1054 The PAOS Response SOAP header has no element content.

1055 Following is an example of an ECP-to-SP response.

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

```
1070 <SOAP-ENV:Body>
1071 <samlp:Response> ... </samlp:Response>
1072 </SOAP-ENV:Body>
1073 </SOAP-ENV:Envelope>
```

1074 4.2.5 Security Considerations

1075 The <AuthnRequest> message SHOULD be signed. Per the rules specified by the browser SSO profile,
1076 the assertions enclosed in the <Response> MUST be signed. The delivery of the response in the SOAP
1077 envelope via PAOS is essentially analogous to the use of the HTTP POST binding and security
1078 countermeasures appropriate to that binding are used.

1079 The SOAP headers SHOULD be integrity protected, such as with SOAP Message Security or through the
1080 use of SSL/TLS over every HTTP exchange with the client.

1081 The service provider SHOULD be authenticated to the ECP, for example with server-side TLS
1082 authentication.

1083 The ECP SHOULD be authenticated to the identity provider, such as by maintaining an authenticated
1084 session. Any HTTP exchanges subsequent to the delivery of the <AuthnRequest> message and before
1085 the identity provider returns a <Response> MUST be securely associated with the original request.

1086 4.3 Identity Provider Discovery Profile

1087 This section defines a profile by which a service provider can discover which identity providers a principal
1088 is using with the Web Browser SSO profile. In deployments having more than one identity provider,
1089 service providers need a means to discover which identity provider(s) a principal uses. The discovery
1090 profile relies on a cookie that is written in a domain that is common between identity providers and service
1091 providers in a deployment. The domain that the deployment predetermines is known as the common
1092 domain in this profile, and the cookie containing the list of identity providers is known as the common
1093 domain cookie.

1094 Which entities host web servers in the common domain is a deployment issue and is outside the scope of
1095 this profile.

1096 4.3.1 Common Domain Cookie

1097 The name of the cookie MUST be "_saml_idp". The format of the cookie value MUST be a set of one or
1098 more base-64 encoded URI values separated by a single space character. Each URI is the unique
1099 identifier of an identity provider, as defined in Section 8.3.6 of [SAMLCore]. The final set of values is then
1100 URL encoded.

1101 The common domain cookie writing service (see below) SHOULD append the identity provider's unique
1102 identifier to the list. If the identifier is already present in the list, it MAY remove and append it. The intent is
1103 that the most recently established identity provider session is the last one in the list.

1104 The cookie MUST be set with a Path prefix of "/". The Domain MUST be set to ".[common-domain]" where
1105 [common-domain] is the common domain established within the deployment for use with this profile.
1106 There MUST be a leading period. The cookie MUST be marked as secure.

1107 Cookie syntax should be in accordance with IETF RFC 2965 [RFC2965] or [NSCookie]. The cookie MAY
1108 be either session-only or persistent. This choice may be made within a deployment, but should apply
1109 uniformly to all identity providers in the deployment.

1110 4.3.2 Setting the Common Domain Cookie

1111 After the identity provider authenticates a principal, it MAY set the common domain cookie. The means by
1112 which the identity provider sets the cookie are implementation-specific so long as the cookie is

1113 successfully set with the parameters given above. One possible implementation strategy follows and
1114 should be considered non-normative. The identity provider may:

- 1115 • Have previously established a DNS and IP alias for itself in the common domain.
- 1116 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
1117 scheme. The structure of the URL is private to the implementation and may include session
1118 information needed to identify the user agent.
- 1119 • Set the cookie on the redirected user agent using the parameters specified above.
- 1120 • Redirect the user agent back to itself, or, if appropriate, to the service provider.

1121 **4.3.3 Obtaining the Common Domain Cookie**

1122 When a service provider needs to discover which identity providers a principal uses, it invokes an
1123 exchange designed to present the common domain cookie to the service provider after it is read by an
1124 HTTP server in the common domain.

1125 If the HTTP server in the common domain is operated by the service provider or if other arrangements are
1126 in place, the service provider MAY utilize the HTTP server in the common domain to relay its
1127 <AuthnRequest> to the identity provider for an optimized single sign-on process.

1128 The specific means by which the service provider reads the cookie are implementation-specific so long as
1129 it is able to cause the user agent to present cookies that have been set with the parameters given in
1130 Section 4.3.1. One possible implementation strategy is described as follows and should be considered
1131 non-normative. Additionally, it may be sub-optimal for some applications.

- 1132 • Have previously established a DNS and IP alias for itself in the common domain.
- 1133 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
1134 scheme. The structure of the URL is private to the implementation and may include session
1135 information needed to identify the user agent.
- 1136 • Redirect the user agent back to itself, or, if appropriate, to the identity provider.

1137 **4.4 Single Logout Profile**

1138 Once a principal has authenticated to an identity provider, the authenticating entity may establish a
1139 session with the principal (typically by means of a cookie, URL re-writing, or some other implementation-
1140 specific means). The identity provider may subsequently issue assertions to service providers or other
1141 relying parties, based on this authentication event; a relying party may use this to establish *its own* session
1142 with the principal.

1143 In such a situation, the identity provider can act as a session authority and the relying parties as session
1144 participants. At some later time, the principal may wish to terminate his or her session either with an
1145 individual session participant, or with all session participants in a given session managed by the session
1146 authority. The former case is considered out of scope of this specification. The latter case, however, may
1147 be satisfied using this profile of the SAML Single Logout protocol ([SAMLCore] Section 3.7).

1148 Note that a principal (or an administrator terminating a principal's session) may choose to terminate this
1149 "global" session either by contacting the session authority, or an individual session participant. Also note
1150 that an identity provider acting as a session authority may *itself* act as a session participant in situations in
1151 which it is the relying party for another identity provider's assertions regarding that principal.

1152 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1153 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1154 front-channel binding may be required, for example, in cases in which a principal's session state exists
1155 solely in a user agent in the form of a cookie and a direct interaction between the user agent and the
1156 session participant or session authority is required. As will be discussed below, session participants
1157 should if possible use a "front-channel" binding when initiating this profile to maximize the likelihood that

1158 the session authority can propagate the logout successfully to all participants.

1159 4.4.1 Required Information

1160 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:logout

1161 **Contact information:** security-services-comment@lists.oasis-open.org

1162 **Description:** Given below.

1163 **Updates:** None

1164 4.4.2 Profile Overview

1165 Figure 3 illustrates the basic template for achieving single logout:

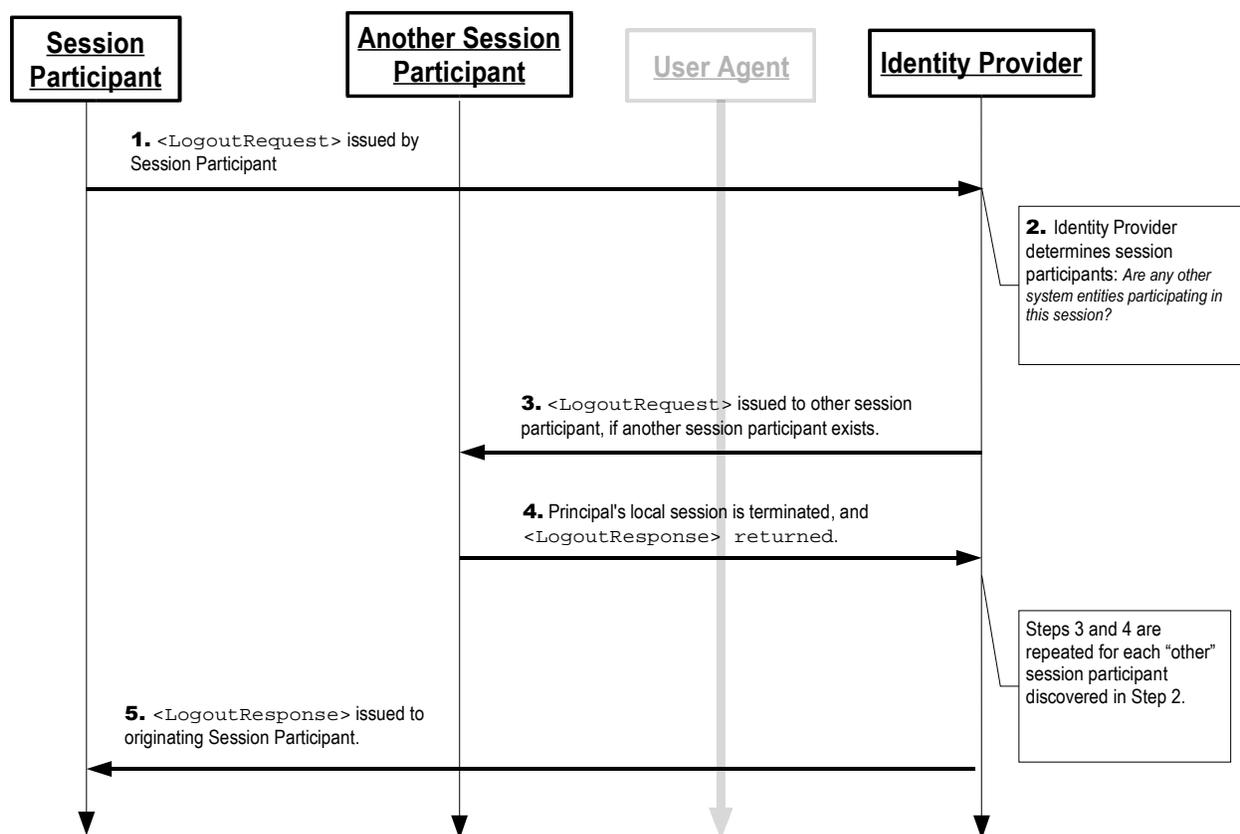


Figure 3

1166 The grayed-out user agent illustrates that the message exchange may pass through the user agent or
1167 may be a direct exchange between system entities, depending on the SAML binding used to implement
1168 the profile.

1169 The following steps are described by the profile. Within an individual step, there may be one or more
1170 actual message exchanges depending on the binding used for that step and other implementation-
1171 dependent behavior.

1172 1. <LogoutRequest> issued by Session Participant to Identity Provider

1173 In step 1, the session participant initiates single logout and terminates a principal's session(s) by
1174 sending a `<LogoutRequest>` message to the identity provider from whom it received the
1175 corresponding authentication assertion. The request may be sent directly to the identity provider
1176 or sent indirectly through the user agent.

1177 **2. Identity Provider determines Session Participants**

1178 In step 2, the identity provider uses the contents of the `<LogoutRequest>` message (or if
1179 initiating logout itself, some other mechanism) to determine the session(s) being terminated. If
1180 there are no other session participants, the profile proceeds with step 5. Otherwise, steps 3 and 4
1181 are repeated for each session participant identified.

1182 **3. `<LogoutRequest>` issued by Identity Provider to Session Participant/Authority**

1183 In step 3, the identity provider issues a `<LogoutRequest>` message to a session participant or
1184 session authority related to one or more of the session(s) being terminated. The request may be
1185 sent directly to the entity or sent indirectly through the user agent (if consistent with the form of the
1186 request in step 1).

1187 **4. Session Participant/Authority issues `<LogoutResponse>` to Identity Provider**

1188 In step 4, a session participant or session authority terminates the principal's session(s) as
1189 directed by the request (if possible) and returns a `<LogoutResponse>` to the identity provider.
1190 The response may be returned directly to the identity provider or indirectly through the user agent
1191 (if consistent with the form of the request in step 3).

1192 **5. Identity Provider issues `<LogoutResponse>` to Session Participant**

1193 In step 5, the identity provider issues a `<LogoutResponse>` message to the original requesting
1194 session participant. The response may be returned directly to the session participant or indirectly
1195 through the user agent (if consistent with the form of the request in step 1).

1196 Note that an identity provider (acting as session authority) can initiate this profile at step 2 and issue a
1197 `<LogoutRequest>` to all session participants, also skipping step 5.

1198 **4.4.3 Profile Description**

1199 If the profile is initiated by a session participant, start with Section 4.4.3.1. If initiated by the identity
1200 provider, start with Section 4.4.3.2. In the descriptions below, the following is referred to:

1201 **Single Logout Service**

1202 This is the single logout protocol endpoint at an identity provider or session participant to which the
1203 `<LogoutRequest>` or `<LogoutResponse>` messages (or an artifact representing them) are
1204 delivered. The same or different endpoints MAY be used for requests and responses.

1205 **4.4.3.1 `<LogoutRequest>` Issued by Session Participant to Identity Provider**

1206 If the logout profile is initiated by a session participant, it examines the authentication assertion(s) it
1207 received pertaining to the session(s) being terminated, and collects the `SessionIndex` value(s) it
1208 received from the identity provider. If multiple identity providers are involved, then the profile MUST be
1209 repeated independently for each one.

1210 To initiate the profile, the session participant issues a `<LogoutRequest>` message to the identity
1211 provider's single logout service request endpoint containing one or more applicable `<SessionIndex>`
1212 elements. At least one element MUST be included. Metadata (as in [SAMLMeta]) MAY be used to
1213 determine the location of this endpoint and the bindings supported by the identity provider.

1214 **Asynchronous Bindings (Front-Channel)**

1215 The session participant SHOULD (if the principal's user agent is present) use an asynchronous
1216 binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind], to send the request to
1217 the identity provider through the user agent. The identity provider SHOULD then propagate any
1218 required logout messages to additional session participants as required using either a synchronous or
1219 asynchronous binding. The use of an asynchronous binding for the original request is preferred
1220 because it gives the identity provider the best chance of successfully propagating the logout to the
1221 other session participants during step 3.

1222 If the HTTP Redirect or POST binding is used, then the <LogoutRequest> message is delivered to
1223 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile
1224 defined in Section 5 is used by the identity provider, which makes a callback to the session participant
1225 to retrieve the <LogoutRequest> message, using for example the SOAP binding.

1226 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1227 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <LogoutRequest>
1228 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,
1229 if used, also provides for an alternate means of authenticating the request issuer when the artifact is
1230 dereferenced.

1231 Each of these bindings provide a RelayState mechanism that the session participant MAY use to
1232 associate the profile exchange with the original request. The session participant SHOULD reveal as
1233 little information as possible in the RelayState value unless the use of the profile does not require such
1234 privacy measures.

1235 **Synchronous Bindings (Back-Channel)**

1236 Alternatively, the session participant MAY use a synchronous binding, such as the SOAP binding
1237 [SAMLBind], to send the request directly to the identity provider. The identity provider SHOULD then
1238 propagate any required logout messages to additional session participants as required using a
1239 synchronous binding. The requester MUST authenticate itself to the identity provider, either by signing
1240 the <LogoutRequest> or using any other binding-supported mechanism.

1241 Profile-specific rules for the contents of the <LogoutRequest> message are included in Section 4.4.4.1.

1242 **4.4.3.2 Identity Provider Determines Session Participants**

1243 If the logout profile is initiated by an identity provider, or upon receiving a valid <LogoutRequest>
1244 message, the identity provider processes the request as defined in [SAMLCore]. It MUST examine the
1245 identifier and <SessionIndex> elements and determine the set of sessions to be terminated.

1246 The identity provider then follows steps 3 and 4 for each entity participating in the session(s) being
1247 terminated, other than the original requesting session participant (if any), as described in Section 3.7.3.2
1248 of [SAMLCore].

1249 **4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session 1250 Participant/Authority**

1251 To propagate the logout, the identity provider issues its own <LogoutRequest> to a session authority or
1252 participant in a session being terminated. The request is sent using a SAML binding consistent with the
1253 capability of the responder and the availability of the user agent at the identity provider.

1254 In general, the binding with which the original request was received in step 1 does not dictate the binding
1255 that may be used in this step except that as noted in step 1, using a synchronous binding that bypasses
1256 the user agent constrains the identity provider to use a similar binding to propagate additional requests.

1257 Profile-specific rules for the contents of the <LogoutRequest> message are included in Section 4.4.4.1.

1258 **4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity**
1259 **Provider**

1260 The session participant/authority MUST process the <LogoutRequest> message as defined in
1261 [SAMLCore]. After processing the message or upon encountering an error, the entity MUST issue a
1262 <LogoutResponse> message containing an appropriate status code to the requesting identity provider
1263 to complete the SAML protocol exchange.

1264 **Synchronous Bindings (Back-Channel)**

1265 If the identity provider used a synchronous binding, such as the SOAP binding [SAMLBind], the
1266 response is returned directly to complete the synchronous communication. The responder MUST
1267 authenticate itself to the requesting identity provider, either by signing the <LogoutResponse> or
1268 using any other binding-supported mechanism.

1269 **Asynchronous Bindings (Front-Channel)**

1270 If the identity provider used an asynchronous binding, such as the HTTP Redirect, POST, or Artifact
1271 bindings [SAMLBind], then the <LogoutResponse> (or artifact) is returned through the user agent to
1272 the identity provider's single logout service response endpoint. Metadata (as in [SAMLMeta]) MAY be
1273 used to determine the location of this endpoint and the bindings supported by the identity provider.
1274 Any asynchronous binding supported by both entities MAY be used.

1275 If the HTTP Redirect or POST binding is used, then the <LogoutResponse> message is delivered to
1276 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile
1277 defined in Section 5 is used by the identity provider, which makes a callback to the responding entity
1278 to retrieve the <LogoutResponse> message, using for example the SOAP binding.

1279 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1280 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <LogoutResponse>
1281 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,
1282 if used, also provides for an alternate means of authenticating the response issuer when the artifact is
1283 dereferenced.

1284 Profile-specific rules for the contents of the <LogoutResponse> message are included in Section
1285 4.4.4.2.

1286 **4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant**

1287 After processing the original session participant's <LogoutRequest> as described in the previous steps
1288 the identity provider MUST respond to the original request with a <LogoutResponse> containing an
1289 appropriate status code to complete the SAML protocol exchange.

1290 The response is sent to the original session participant, using a SAML binding consistent with the binding
1291 used in the original request, the capability of the responder, and the availability of the user agent at the
1292 identity provider. Assuming an asynchronous binding was used in step 1, then any binding supported by
1293 both entities MAY be used.

1294 Profile-specific rules for the contents of the <LogoutResponse> message are included in Section
1295 4.4.4.2.

1296 **4.4.4 Use of Single Logout Protocol**

1297 **4.4.4.1 <LogoutRequest> Usage**

1298 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1299 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1300 format:entity.

1301 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1302 the message or using a binding-specific mechanism.

1303 The principal MUST be identified in the request using an identifier that **strongly matches** the identifier in
1304 the authentication assertion the requester issued or received regarding the session being terminated, per
1305 the matching rules defined in Section 3.3.4 of [SAMLCore].

1306 If the requester is a session participant, it MUST include at least one `<SessionIndex>` element in the
1307 request. If the requester is a session authority (or acting on its behalf), then it MAY omit any such
1308 elements to indicate the termination of all of the principal's applicable sessions.

1309 **4.4.4.2 <LogoutResponse> Usage**

1310 The `<Issuer>` element MUST be present and MUST contain the unique identifier of the responding
1311 entity; the `Format` attribute MUST be omitted or have a value of
1312 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

1313 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1314 the message or using a binding-specific mechanism.

1315 **4.4.5 Use of Metadata**

1316 [SAMLMeta] defines an endpoint element, `<md:SingleLogoutService>`, to describe supported
1317 bindings and location(s) to which an entity may send requests and responses using this profile.

1318 A requester, if encrypting the principal's identifier, can use the responder's `<md:KeyDescriptor>`
1319 element with a `use` attribute of `encryption` to determine an appropriate encryption algorithm and
1320 settings to use, along with a public key to use in delivering a bulk encryption key.

1321 **4.5 Name Identifier Management Profile**

1322 In the scenario supported by the Name Identifier Management profile, an identity provider has exchanged
1323 some form of persistent identifier for a principal with a service provider, allowing them to share a common
1324 identifier for some length of time. Subsequently, the identity provider may wish to notify the service
1325 provider of a change in the format and/or value that it will use to identify the same principal in the future.
1326 Alternatively the service provider may wish to attach its own "alias" for the principal in order to ensure that
1327 the identity provider will include it when communicating with it in the future about the principal. Finally, one
1328 of the providers may wish to inform the other that it will no longer issue or accept messages using a
1329 particular identifier. To implement these scenarios, a profile of the SAML Name Identifier Management
1330 protocol is used.

1331 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1332 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1333 front-channel binding may be required, for example, in cases in which direct interaction between the user
1334 agent and the responding provider is required in order to effect the change.

1335 **4.5.1 Required Information**

1336 **Identification:** `urn:oasis:names:tc:SAML:2.0:profiles:SSO:nameid-mgmt`

1337 **Contact information:** security-services-comment@lists.oasis-open.org

1338 **Description:** Given below.

1339 **Updates:** None.

1340 4.5.2 Profile Overview

1341 Figure 4 illustrates the basic template for the name identifier management profile.

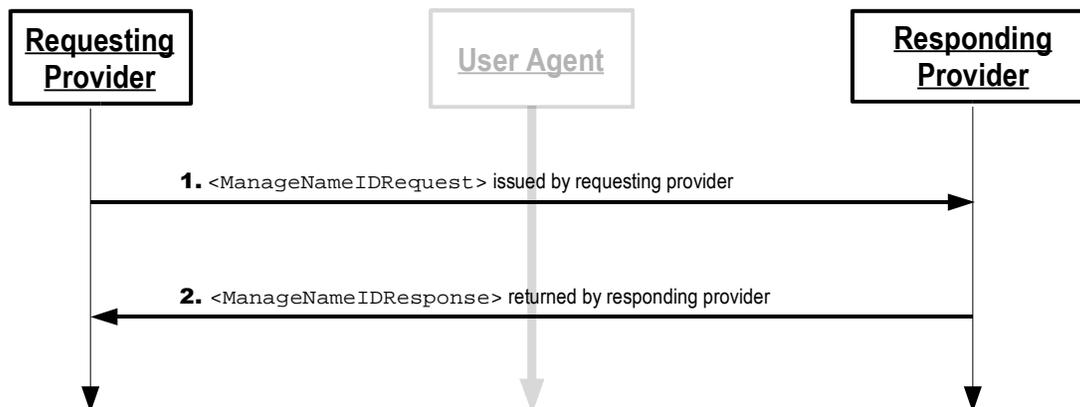


Figure 4

1342 The grayed-out user agent illustrates that the message exchange may pass through the user agent or
1343 may be a direct exchange between system entities, depending on the SAML binding used to implement
1344 the profile.

1345 The following steps are described by the profile. Within an individual step, there may be one or more
1346 actual message exchanges depending on the binding used for that step and other implementation-
1347 dependent behavior.

1348 1. <ManageNameIDRequest> issued by Requesting Identity/Service Provider

1349 In step 1, an identity or service provider initiates the profile by sending a
1350 <ManageNameIDRequest> message to another provider that it wishes to inform of a change.
1351 The request may be sent directly to the responding provider or sent indirectly through the user
1352 agent.

1353 2. <ManageNameIDResponse> issued by Responding Identity/Service Provider

1354 In step 2, the responding provider (after processing the request) issues a
1355 <ManageNameIDResponse> message to the original requesting provider. The response may be
1356 returned directly to the requesting provider or indirectly through the user agent (if consistent with
1357 the form of the request in step 1).

1358 4.5.3 Profile Description

1359 In the descriptions below, the following is referred to:

1360 Name Identifier Management Service

1361 This is the name identifier management protocol endpoint at an identity or service provider to which
1362 the <ManageNameIDRequest> or <ManageNameIDResponse> messages (or an artifact
1363 representing them) are delivered. The same or different endpoints MAY be used for requests and
1364 responses.

1365 4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider

1366 To initiate the profile, the requesting provider issues a <ManageNameIDRequest> message to another

1367 provider's name identifier management service request endpoint. Metadata (as in [SAMLMeta]) MAY be
1368 used to determine the location of this endpoint and the bindings supported by the responding provider.

1369 **Synchronous Bindings (Back-Channel)**

1370 The requesting provider MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to
1371 send the request directly to the other provider. The requester MUST authenticate itself to the other
1372 provider, either by signing the <ManageNameIDRequest> or using any other binding-supported
1373 mechanism.

1374 **Asynchronous Bindings (Front-Channel)**

1375 Alternatively, the requesting provider MAY (if the principal's user agent is present) use an
1376 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to send the
1377 request to the other provider through the user agent.

1378 If the HTTP Redirect or POST binding is used, then the <ManageNameIDRequest> message is
1379 delivered to the other provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution
1380 profile defined in Section 5 is used by the other provider, which makes a callback to the requesting
1381 provider to retrieve the <ManageNameIDRequest> message, using for example the SOAP binding.

1382 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1383 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1384 <ManageNameIDRequest> message MUST be signed if the HTTP POST or Redirect binding is
1385 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1386 request issuer when the artifact is dereferenced.

1387 Each of these bindings provide a RelayState mechanism that the requesting provider MAY use to
1388 associate the profile exchange with the original request. The requesting provider SHOULD reveal as
1389 little information as possible in the RelayState value unless the use of the profile does not require such
1390 privacy measures.

1391 Profile-specific rules for the contents of the <ManageNameIDRequest> message are included in Section
1392 4.5.4.1.

1393 **4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service 1394 Provider**

1395 The recipient MUST process the <ManageNameIDRequest> message as defined in [SAMLCore]. After
1396 processing the message or upon encountering an error, the recipient MUST issue a
1397 <ManageNameIDResponse> message containing an appropriate status code to the requesting provider
1398 to complete the SAML protocol exchange.

1399 **Synchronous Bindings (Back-Channel)**

1400 If the requesting provider used a synchronous binding, such as the SOAP binding [SAMLBind], the
1401 response is returned directly to complete the synchronous communication. The responder MUST
1402 authenticate itself to the requesting provider, either by signing the <ManageNameIDResponse> or
1403 using any other binding-supported mechanism.

1404 **Asynchronous Bindings (Front-Channel)**

1405 If the requesting provider used an asynchronous binding, such as the HTTP Redirect, POST, or
1406 Artifact bindings [SAMLBind], then the <ManageNameIDResponse> (or artifact) is returned through
1407 the user agent to the requesting provider's name identifier management service response endpoint.
1408 Metadata (as in [SAMLMeta]) MAY be used to determine the location of this endpoint and the bindings
1409 supported by the requesting provider. Any binding supported by both entities MAY be used.

1410 If the HTTP Redirect or POST binding is used, then the <ManageNameIDResponse> message is
1411 delivered to the requesting provider in this step. If the HTTP Artifact binding is used, the Artifact
1412 Resolution profile defined in Section 5 is used by the requesting provider, which makes a callback to
1413 the responding provider to retrieve the <ManageNameIDResponse> message, using for example the

1414 SOAP binding.

1415 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1416 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1417 <ManageNameIDResponse> message MUST be signed if the HTTP POST or Redirect binding is
1418 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1419 response issuer when the artifact is dereferenced.

1420 Profile-specific rules for the contents of the <ManageNameIDResponse> message are included in
1421 Section 4.5.4.2.

1422 **4.5.4 Use of Name Identifier Management Protocol**

1423 **4.5.4.1 <ManageNameIDRequest> Usage**

1424 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1425 the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`
1426 `format:entity`.

1427 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1428 the message or using a binding-specific mechanism.

1429 **4.5.4.2 <ManageNameIDResponse> Usage**

1430 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1431 entity; the `Format` attribute MUST be omitted or have a value of
1432 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

1433 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1434 the message or using a binding-specific mechanism.

1435 **4.5.5 Use of Metadata**

1436 [SAMLMeta] defines an endpoint element, <md:ManageNameIDService>, to describe supported
1437 bindings and location(s) to which an entity may send requests and responses using this profile.

1438 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>
1439 element with a `use` attribute of `encryption` to determine an appropriate encryption algorithm and
1440 settings to use, along with a public key to use in delivering a bulk encryption key.

1441

5 Artifact Resolution Profile

1442 [SAMLCore] defines an Artifact Resolution protocol for dereferencing a SAML artifact into a corresponding
1443 protocol message. The HTTP Artifact binding in [SAMLBind] leverages this mechanism to pass SAML
1444 protocol messages by reference. This profile describes the use of this protocol with a synchronous
1445 binding, such as the SOAP binding defined in [SAMLBind].

5.1 Required Information

1447 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:artifact

1448 **Contact information:** security-services-comment@lists.oasis-open.org

1449 **Description:** Given below.

1450 **Updates:** None

5.2 Profile Overview

1452 The message exchange and basic processing rules that govern this profile are largely defined by Section
1453 3.5 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1454 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1455 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1456 Figure 5 illustrates the basic template for the artifact resolution profile.

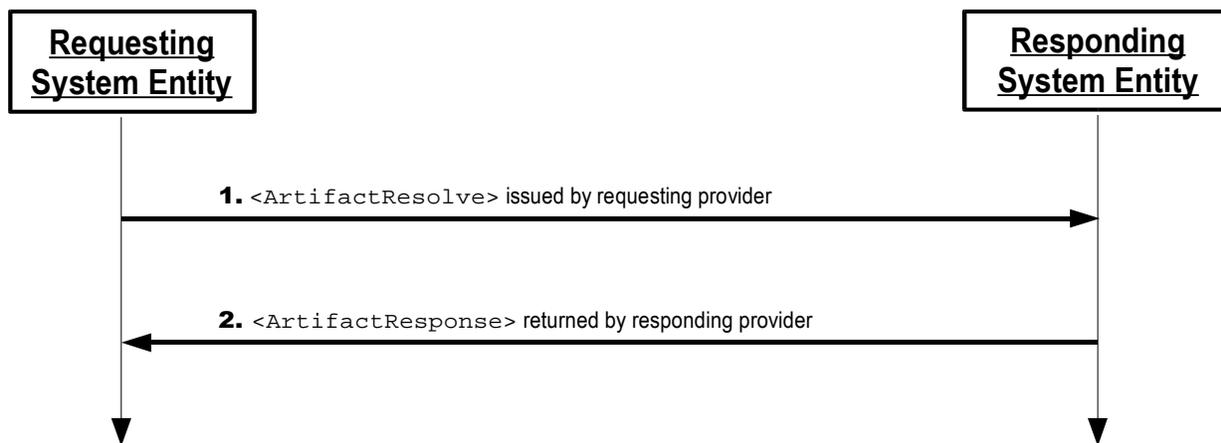


Figure 5

1457 The following steps are described by the profile.

1458 1. <ArtifactResolve> issued by Requesting Entity

1459 In step 1, a requester initiates the profile by sending an <ArtifactResolve> message to an
1460 artifact issuer.

1461 2. <ArtifactResponse> issued by Responding Entity

1462 In step 2, the responder (after processing the request) issues an <ArtifactResponse>
1463 message to the requester.

1464 5.3 Profile Description

1465 In the descriptions below, the following is referred to:

1466 Artifact Resolution Service

1467 This is the artifact resolution protocol endpoint at an artifact issuer to which <ArtifactResolve>
1468 messages are delivered.

1469 5.3.1 <ArtifactResolve> issued by Requesting Entity

1470 To initiate the profile, a requester, having received an artifact and determined the issuer using the
1471 SourceID, sends an <ArtifactResolve> message containing the artifact to an artifact issuer's artifact
1472 resolution service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of this
1473 endpoint and the bindings supported by the artifact issuer.

1474 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1475 request directly to the artifact issuer. The requester SHOULD authenticate itself to the responder, either by
1476 signing the <ArtifactResolve> message or using any other binding-supported mechanism. Specific
1477 profiles that use the HTTP Artifact binding MAY impose additional requirements such that authentication is
1478 mandatory.

1479 Profile-specific rules for the contents of the <ArtifactResolve> message are included in Section 5.4.1.

1480 5.3.2 <ArtifactResponse> issued by Responding Entity

1481 The artifact issuer MUST process the <ArtifactResolve> message as defined in [SAMLCore]. After
1482 processing the message or upon encountering an error, the artifact issuer MUST return an
1483 <ArtifactResponse> message containing an appropriate status code to the requester to complete the
1484 SAML protocol exchange. If successful, the dereferenced SAML protocol message corresponding to the
1485 artifact will also be included.

1486 The responder MUST authenticate itself to the requester, either by signing the <ArtifactResponse> or
1487 using any other binding-supported mechanism.

1488 Profile-specific rules for the contents of the <ArtifactResponse> message are included in Section
1489 5.4.2.

1490 5.4 Use of Artifact Resolution Protocol

1491 5.4.1 <ArtifactResolve> Usage

1492 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1493 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1494 format:entity.

1495 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by
1496 signing the message or using a binding-specific mechanism. Specific profiles that use the HTTP Artifact
1497 binding MAY impose additional requirements such that authentication is mandatory.

1498 **5.4.2 <ArtifactResponse> Usage**

1499 The `<Issuer>` element MUST be present and MUST contain the unique identifier of the artifact issuer;
1500 the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`
1501 `format:entity`.

1502 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1503 the message or using a binding-specific mechanism.

1504 **5.5 Use of Metadata**

1505 [SAMLMeta] defines an indexed endpoint element, `<md:ArtifactResolutionService>`, to describe
1506 supported bindings and location(s) to which a requester may send requests using this profile. The `index`
1507 attribute is used to distinguish the possible endpoints that may be specified by reference in the artifact's
1508 `EndpointIndex` field.

1509 6 Assertion Query/Request Profile

1510 [SAMLCore] defines a protocol for requesting existing assertions by reference or by querying on the basis
1511 of a subject and additional statement-specific criteria. This profile describes the use of this protocol with a
1512 synchronous binding, such as the SOAP binding defined in [SAMLBind].

1513 6.1 Required Information

1514 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:query

1515 **Contact information:** security-services-comment@lists.oasis-open.org

1516 **Description:** Given below.

1517 **Updates:** None.

1518 6.2 Profile Overview

1519 The message exchange and basic processing rules that govern this profile are largely defined by Section
1520 3.3 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1521 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1522 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1523 Figure 6 illustrates the basic template for the query/request profile.

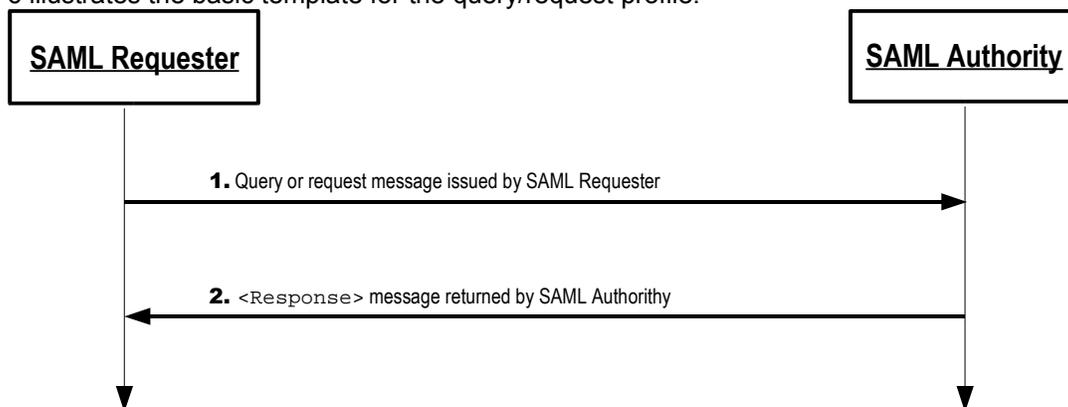


Figure 6

1524 The following steps are described by the profile.

1525 1. Query/Request issued by SAML Requester

1526 In step 1, a SAML requester initiates the profile by sending an `<AssertionIDRequest>`,
1527 `<SubjectQuery>`, `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>`
1528 message to a SAML authority.

1529 2. <Response> issued by SAML Authority

1530 In step 2, the responding SAML authority (after processing the query or request) issues a
1531 `<Response>` message to the SAML requester.

1532 **6.3 Profile Description**

1533 In the descriptions below, the following are referred to:

1534 **Query/Request Service**

1535 This is the query/request protocol endpoint at a SAML authority to which query or
1536 `<AssertionIDRequest>` messages are delivered.

1537 **6.3.1 Query/Request issued by SAML Requester**

1538 To initiate the profile, a SAML requester issues an `<AssertionIDRequest>`, `<SubjectQuery>`,
1539 `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>` message to a SAML authority's
1540 query/request service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of
1541 this endpoint and the bindings supported by the SAML authority.

1542 The SAML requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send
1543 the request directly to the identity provider. The requester SHOULD authenticate itself to the SAML
1544 authority either by signing the message or using any other binding-supported mechanism.

1545 Profile-specific rules for the contents of the various messages are included in Section 6.4.1.

1546 **6.3.2 `<Response>` issued by SAML Authority**

1547 The SAML authority MUST process the query or request message as defined in [SAMLCore]. After
1548 processing the message or upon encountering an error, the SAML authority MUST return a `<Response>`
1549 message containing an appropriate status code to the SAML requester to complete the SAML protocol
1550 exchange. If the request is successful in locating one or more matching assertions, they will also be
1551 included in the response.

1552 The responder SHOULD authenticate itself to the requester, either by signing the `<Response>` or using
1553 any other binding-supported mechanism.

1554 Profile-specific rules for the contents of the `<Response>` message are included in Section 6.4.2.

1555 **6.4 Use of Query/Request Protocol**

1556 **6.4.1 Query/Request Usage**

1557 The `<Issuer>` element MUST be present.

1558 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by
1559 signing the message or using a binding-specific mechanism.

1560 **6.4.2 `<Response>` Usage**

1561 The `<Issuer>` element MUST be present and MUST contain the unique identifier of the responding
1562 SAML authority; the `Format` attribute MUST be omitted or have a value of
1563 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`. Note that this need not necessarily
1564 match the `<Issuer>` element in the returned assertion(s).

1565 The responder SHOULD authenticate itself to the requester and ensure message integrity, either by
1566 signing the message or using a binding-specific mechanism.

1567 **6.5 Use of Metadata**

1568 [SAMLMeta] defines several endpoint elements, `<md:AssertionIDRequestService>`,
1569 `<md:AuthnQueryService>`, `<md:AttributeService>`, and `<md:AuthzService>`, to describe
1570 supported bindings and location(s) to which a requester may send requests or queries using this profile.

1571 The SAML authority, if encrypting the resulting assertions or assertion contents for a particular entity, can
1572 use that entity's `<md:KeyDescriptor>` element with a `use` attribute of `encryption` to determine an
1573 appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk
1574 encryption key.

1575 The various role descriptors MAY contain `<md:NameIDFormat>`, `<md:AttributeProfile>`, and
1576 `<saml:Attribute>` elements (as applicable) to indicate the general ability to support particular name
1577 identifier formats, attribute profiles, or specific attributes and values. The ability to support any such
1578 features during a given request is dependent on policy and the discretion of the authority.

1579

7 Name Identifier Mapping Profile

1580 [SAMLCore] defines a Name Identifier Mapping protocol for mapping a principal's name identifier into a
1581 different name identifier for the same principal. This profile describes the use of this protocol with a
1582 synchronous binding, such as the SOAP binding defined in [SAMLBind], and additional guidelines for
1583 protecting the privacy of the principal with encryption and limiting the use of the mapped identifier.

7.1 Required Information

1585 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:nameidmapping

1586 **Contact information:** security-services-comment@lists.oasis-open.org

1587 **Description:** Given below.

1588 **Updates:** None.

7.2 Profile Overview

1590 The message exchange and basic processing rules that govern this profile are largely defined by Section
1591 3.8 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1592 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1593 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1594 Figure 7 illustrates the basic template for the name identifier mapping profile.



Figure 7

1595 The following steps are described by the profile.

1596 1. <NameIDMappingRequest> issued by Requesting Entity

1597 In step 1, a requester initiates the profile by sending a <NameIDMappingRequest> message to
1598 an identity provider.

1599 2. <NameIDMappingResponse> issued by Identity Provider

1600 In step 2, the responding identity provider (after processing the request) issues a
1601 <NameIDMappingResponse> message to the requester.

1602 **7.3 Profile Description**

1603 In the descriptions below, the following is referred to:

1604 **Name Identifier Mapping Service**

1605 This is the name identifier mapping protocol endpoint at an identity provider to which
1606 <NameIDMappingRequest> messages are delivered.

1607 **7.3.1 <NameIDMappingRequest> issued by Requesting Entity**

1608 To initiate the profile, a requester issues a <NameIDMappingRequest> message to an identity provider's
1609 name identifier mapping service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the
1610 location of this endpoint and the bindings supported by the identity provider.

1611 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1612 request directly to the identity provider. The requester MUST authenticate itself to the identity provider,
1613 either by signing the <NameIDMappingRequest> or using any other binding-supported mechanism.

1614 Profile-specific rules for the contents of the <NameIDMappingRequest> message are included in
1615 Section 7.4.1.

1616 **7.3.2 <NameIDMappingResponse> issued by Identity Provider**

1617 The identity provider MUST process the <ManageNameIDRequest> message as defined in [SAMLCore].
1618 After processing the message or upon encountering an error, the identity provider MUST return a
1619 <NameIDMappingResponse> message containing an appropriate status code to the requester to
1620 complete the SAML protocol exchange.

1621 The responder MUST authenticate itself to the requester, either by signing the
1622 <NameIDMappingResponse> or using any other binding-supported mechanism.

1623 Profile-specific rules for the contents of the <NameIDMappingResponse> message are included in
1624 Section 7.4.2.

1625 **7.4 Use of Name Identifier Mapping Protocol**

1626 **7.4.1 <NameIDMappingRequest> Usage**

1627 The <Issuer> element MUST be present.

1628 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1629 the message or using a binding-specific mechanism.

1630 **7.4.2 <NameIDMappingResponse> Usage**

1631 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1632 identity provider; the Format attribute MUST be omitted or have a value of
1633 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1634 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1635 the message or using a binding-specific mechanism.

1636 Section 2.2.3 of [SAMLCore] defines the use of encryption to apply confidentiality to a name identifier. In
1637 most cases, the identity provider SHOULD encrypt the mapped name identifier it returns to the requester
1638 to protect the privacy of the principal. The requester can extract the <EncryptedID> element and place it
1639 in subsequent protocol messages or assertions.

1640 **7.4.2.1 Limiting Use of Mapped Identifier**

1641 Additional limits on the use of the resulting identifier MAY be applied by the identity provider by returning
1642 the mapped name identifier in the form of an <Assertion> containing the identifier in its <Subject> but
1643 without any statements. The assertion is then encrypted and the result used as the <EncryptedData>
1644 element in the <EncryptedID> returned to the requester. The assertion MAY include a <Conditions>
1645 element to limit use, as defined by [SAMLCore], such as time-based constraints or use by specific relying
1646 parties, and MUST be signed for integrity protection.

1647 **7.5 Use of Metadata**

1648 [SAMLMeta] defines an endpoint element, <md:NameIDMappingService>, to describe supported
1649 bindings and location(s) to which a requester may send requests using this profile.

1650 The identity provider, if encrypting the resulting identifier for a particular entity, can use that entity's
1651 <md:KeyDescriptor> element with a use attribute of encryption to determine an appropriate
1652 encryption algorithm and settings to use, along with a public key to use in delivering a bulk encryption key.

1653 8 SAML Attribute Profiles

1654 8.1 Basic Attribute Profile

1655 The Basic attribute profile specifies simplified, but non-unique, naming of SAML attributes together with
1656 attribute values based on the built-in XML Schema data types, eliminating the need for extension schemas
1657 to validate syntax.

1658 8.1.1 Required Information

1659 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:basic

1660 **Contact information:** security-services-comment@lists.oasis-open.org

1661 **Description:** Given below.

1662 **Updates:** None.

1663 8.1.2 SAML Attribute Naming

1664 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
1665 `urn:oasis:names:tc:SAML:2.0:attrname-format:basic`.

1666 The `Name` XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

1667 8.1.2.1 Attribute Name Comparison

1668 Two `<Attribute>` elements refer to the same SAML attribute if and only if the values of their `Name` XML
1669 attributes are equal in the sense of Section 3.3.6 of [Schema2].

1670 8.1.3 Profile-Specific XML Attributes

1671 No additional XML attributes are defined for use with the `<Attribute>` element.

1672 8.1.4 SAML Attribute Values

1673 The schema type of the contents of the `<AttributeValue>` element MUST be drawn from one of the
1674 types defined in Section 3.3 of [Schema2]. The `xsi:type` attribute MUST be present and be given the
1675 appropriate value.

1676 8.1.5 Example

```
1677 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"  
1678     Name="FirstName">  
1679     <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>  
1680 </saml:Attribute>
```

1681 8.2 X.500/LDAP Attribute Profile

1682 Directories based on the ITU-T X.500 specifications [X.500] and the related IETF Lightweight Directory
1683 Access Protocol specifications [LDAP] are widely deployed. Directory schema is used to model
1684 information to be stored in these directories. In particular, in X.500, attribute type definitions are used to
1685 specify the syntax and other features of attributes, the basic information storage unit in a directory (this

1686 document refers to these as “directory attributes”). Directory attribute types are defined in schema in the
1687 X.500 and LDAP specifications themselves, schema in other public documents (such as the
1688 Internet2/Educause EduPerson schema [eduPerson], or the inetOrgperson schema [RFC2798]), and
1689 schema defined for private purposes. In any of these cases, it is useful for deployers to take advantage of
1690 these directory attribute types in the context of SAML attribute statements, without having to manually
1691 create SAML-specific attribute definitions for them, and to do this in an interoperable fashion.
1692 The X.500/LDAP attribute profile defines a common convention for the naming and representation of such
1693 attributes when expressed as SAML attributes.

1694 8.2.1 Required Information

1695 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500 (this is also the target namespace
1696 assigned in the corresponding X.500/LDAP profile schema document [SAMLX500-xsd])

1697 **Contact information:** security-services-comment@lists.oasis-open.org

1698 **Description:** Given below.

1699 **Updates:** None.

1700 8.2.2 SAML Attribute Naming

1701 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
1702 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1703 To construct attribute names, the URN `oid` namespace described in IETF RFC 3061 [RFC3061] is used.
1704 In this approach the `Name` XML attribute is based on the OBJECT IDENTIFIER assigned to the directory
1705 attribute type.

1706 Example:

```
1707 urn:oid:2.5.4.3
```

1708 Since X.500 procedures require that every attribute type be identified with a unique OBJECT IDENTIFIER,
1709 this naming scheme ensures that the derived SAML attribute names are unambiguous.

1710 For purposes of human readability, there may also be a requirement for some applications to carry an
1711 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
1712 [SAMLCore]) MAY be used for this purpose. If the definition of the directory attribute type includes one or
1713 more descriptors (short names) for the attribute type, the `FriendlyName` value, if present, SHOULD be
1714 one of the defined descriptors.

1715 8.2.2.1 Attribute Name Comparison

1716 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
1717 values are equal in the sense of [RFC3061]. The `FriendlyName` attribute plays no role in the
1718 comparison.

1719 8.2.3 Profile-Specific XML Attributes

1720 No additional XML attributes are defined for use with the `<Attribute>` element.

1721 8.2.4 SAML Attribute Values

1722 Directory attribute type definitions for use in native X.500 directories specify the syntax of the attribute
1723 using ASN.1 [ASN.1]. For use in LDAP, directory attribute definitions additionally include an LDAP syntax
1724 which specifies how attribute or assertion values conforming to the syntax are to be represented when
1725 transferred in the LDAP protocol (known as an LDAP-specific encoding). The LDAP-specific encoding

1726 commonly produces Unicode characters in UTF-8 form. This SAML attribute profile specifies the form of
1727 SAML attribute values only for those directory attributes which have LDAP syntaxes. Future extensions to
1728 this profile may define attribute value formats for directory attributes whose syntaxes specify other
1729 encodings.

1730 To represent the encoding rules in use for a particular attribute value, the <AttributeValue> element
1731 MUST contain an XML attribute named `Encoding` defined in the XML namespace
1732 `urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500`.

1733 For any directory attribute with a syntax whose LDAP-specific encoding exclusively produces UTF-8
1734 character strings as values, the SAML attribute value is encoded as simply the UTF-8 string itself, as the
1735 content of the <AttributeValue> element, with no additional whitespace. In such cases, the
1736 `xsi:type` XML attribute MUST be set to **xs:string**. The profile-specific `Encoding` XML attribute is
1737 provided, with a value of `LDAP`.

1738 A list of some LDAP attribute syntaxes to which this applies is:

1739	Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3
1740	Bit String	1.3.6.1.4.1.1466.115.121.1.6
1741	Boolean	1.3.6.1.4.1.1466.115.121.1.7
1742	Country String	1.3.6.1.4.1.1466.115.121.1.11
1743	DN	1.3.6.1.4.1.1466.115.121.1.12
1744	Directory String	1.3.6.1.4.1.1466.115.121.1.15
1745	Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22
1746	Generalized Time	1.3.6.1.4.1.1466.115.121.1.24
1747	IA5 String	1.3.6.1.4.1.1466.115.121.1.26
1748	INTEGER	1.3.6.1.4.1.1466.115.121.1.27
1749	LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54
1750	Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30
1751	Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31
1752	Name And Optional UID	1.3.6.1.4.1.1466.115.121.1.34
1753	Name Form Description	1.3.6.1.4.1.1466.115.121.1.35
1754	Numeric String	1.3.6.1.4.1.1466.115.121.1.36
1755	Object Class Description	1.3.6.1.4.1.1466.115.121.1.37
1756	Octet String	1.3.6.1.4.1.1466.115.121.1.40
1757	OID	1.3.6.1.4.1.1466.115.121.1.38
1758	Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39
1759	Postal Address	1.3.6.1.4.1.1466.115.121.1.41
1760	Presentation Address	1.3.6.1.4.1.1466.115.121.1.43
1761	Printable String	1.3.6.1.4.1.1466.115.121.1.44
1762	Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58
1763	Telephone Number	1.3.6.1.4.1.1466.115.121.1.50
1764	UTC Time	1.3.6.1.4.1.1466.115.121.1.53

1765 For all other LDAP syntaxes, the attribute value is encoded, as the content of the <AttributeValue>
1766 element, by base64-encoding [RFC2045] the encompassing ASN.1 OCTET STRING-encoded LDAP
1767 attribute value. The `xsi:type` XML attribute MUST be set to **xs:base64Binary**. The profile-specific
1768 `Encoding` XML attribute is provided, with a value of `"LDAP"`.

1769 When comparing SAML attribute values for equality, the matching rules specified for the corresponding
1770 directory attribute type MUST be observed (case sensitivity, for example).

1771 **8.2.5 Profile-Specific Schema**

1772 The following schema listing shows how the profile-specific `Encoding` XML attribute is defined
1773 [SAMLX500-xsd]:

```

1774 <schema
1775   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
1776   xmlns="http://www.w3.org/2001/XMLSchema"
1777   elementFormDefault="unqualified"
1778   attributeFormDefault="unqualified"
1779   blockDefault="substitution"
1780   version="2.0">
1781   <annotation>
1782     <documentation>
1783       Document identifier: sstc-saml-schema-x500-2.0
1784       Location: http://www.oasis-
1785 open.org/committees/documents.php?wg_abbrev=security
1786       Revision history:
1787         V2.0 CD-04 (January, 2005):
1788         Custom schema for X.500 attribute profile, first published in
1789 SAML 2.0.
1790     </documentation>
1791   </annotation>
1792   <attribute name="Encoding" type="string"/>
1793 </schema>

```

1794 8.2.6 Example

1795 The following is an example of a mapping of the "givenName" directory attribute, representing the SAML
1796 assertion subject's first name. It's OBJECT IDENTIFIER is 2.5.4.42 and its LDAP syntax is Directory
1797 String.

```

1798 <saml:Attribute xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
1799   NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1800   Name="urn:oid:2.5.4.42" FriendlyName="givenName">
1801   <saml:AttributeValue xsi:type="xs:string"
1802     x500:Encoding="LDAP">Steven</saml:AttributeValue>
1803 </saml:Attribute>

```

1804 8.3 UUID Attribute Profile

1805 The UUID attribute profile standardizes the expression of UUID values as SAML attribute names and
1806 values. It is applicable when the attribute's source system is one that identifies an attribute or its value with
1807 a UUID.

1808 8.3.1 Required Information

1809 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:UUID

1810 **Contact information:** security-services-comment@lists.oasis-open.org

1811 **Description:** Given below.

1812 **Updates:** None.

1813 8.3.2 UUID and GUID Background

1814 UUIDs (Universally Unique Identifiers), also known as GUIDs (Globally Unique Identifiers), are used to
1815 define objects and subjects such that they are guaranteed uniqueness across space and time. UUIDs
1816 were originally used in the Network Computing System (NCS), and then used in the Open Software
1817 Foundation's (OSF) Distributed Computing Environment (DCE). Recently GUIDs have been used in
1818 Microsoft's COM and Active Directory/Windows 2000/2003 platform.

1819 A UUID is a 128 bit number, generated such that it should never be duplicated within the domain of
1820 interest. UUIDs are used to represent a wide range of objects including, but not limited to, subjects/users,
1821 groups of users and node names. A UUID, represented as a hexadecimal string, is as follows:

1822 `f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1823 In DCE and Microsoft Windows, the UUID is usually presented to the administrator in the form of a
1824 "friendly name". For instance the above UUID could represent the user john.doe@example.com.

1825 **8.3.3 SAML Attribute Naming**

1826 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
1827 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1828 If the underlying representation of the attribute's name is a UUID, then the URN `uuid` namespace
1829 described in [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt] is used. In this approach the
1830 `Name` XML attribute is based on the URN form of the underlying UUID that identifies the attribute.

1831 Example:

1832 `urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1833 If the underlying representation of the attribute's name is not a UUID, then any form of URI MAY be used
1834 in the `Name` XML attribute.

1835 For purposes of human readability, there may also be a requirement for some applications to carry an
1836 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in
1837 [SAMLCORE]) MAY be used for this purpose.

1838 **8.3.3.1 Attribute Name Comparison**

1839 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
1840 values are equal in the sense of [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt]. The
1841 `FriendlyName` attribute plays no role in the comparison.

1842 **8.3.4 Profile-Specific XML Attributes**

1843 No additional XML attributes are defined for use with the `<Attribute>` element.

1844 **8.3.5 SAML Attribute Values**

1845 In cases in which the attribute's value is also a UUID, the same URN syntax described above MUST be
1846 used to express the value within the `<AttributeValue>` element. The `xsi:type` XML attribute MUST
1847 be set to `xs:anyURI`.

1848 If the attribute's value is not a UUID, then there are no restrictions on the use of the `<AttributeValue>`
1849 element.

1850 **8.3.6 Example**

1851 The following is an example of a DCE Extended Registry Attribute, the "pre_auth_req" setting, which has a
1852 well-known UUID of 6c9d0ec8-dd2d-11cc-abdd-080009353559 and is integer-valued.

```
1853 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1854     Name="urn:uuid:6c9d0ec8-dd2d-11cc-abdd-080009353559"  
1855     FriendlyName="pre_auth_req">  
1856     <saml:AttributeValue xsi:type="xs:integer">1</saml:AttributeValue>  
1857 </saml:Attribute>
```

1858 **8.4 DCE PAC Attribute Profile**

1859 The DCE PAC attribute profile defines the expression of DCE PAC information as SAML attribute names
1860 and values. It is used to standardize a mapping between the primary information that makes up a DCE
1861 principal's identity and a set of SAML attributes. This profile builds on the UUID attribute profile defined in
1862 Section 8.3.

1863 **8.4.1 Required Information**

1864 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE (this is also the target namespace
1865 assigned in the corresponding DCE PAC attribute profile schema document [SAML DCE-xsd])

1866 **Contact information:** security-services-comment@lists.oasis-open.org

1867 **Description:** Given below.

1868 **Updates:** None.

1869 **8.4.2 PAC Description**

1870 A DCE PAC is an extensible structure that can carry arbitrary DCE registry attributes, but a core set of
1871 information is common across principals and makes up the bulk of a DCE identity:

- 1872 • The principal's DCE "realm" or "cell"
- 1873 • The principal's unique identifier
- 1874 • The principal's primary DCE local group membership
- 1875 • The principal's set of DCE local group memberships (multi-valued)
- 1876 • The principal's set of DCE foreign group memberships (multi-valued)

1877 The primary value(s) of each of these attributes is a UUID.

1878 **8.4.3 SAML Attribute Naming**

1879 This profile defines a mapping of specific DCE information into SAML attributes, and thus defines actual
1880 specific attribute names, rather than a naming convention.

1881 For all attributes defined by this profile, the `NameFormat` XML attribute in `<Attribute>` elements MUST
1882 have the value `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1883 For purposes of human readability, there may also be a requirement for some applications to carry an
1884 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in
1885 [SAMLCore]) MAY be used for this purpose.

1886 See Section 8.4.6 for the specific attribute names defined by this profile.

1887 **8.4.3.1 Attribute Name Comparison**

1888 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
1889 values are equal in the sense of [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt]. The
1890 `FriendlyName` attribute plays no role in the comparison.

1891 **8.4.4 Profile-Specific XML Attributes**

1892 No additional XML attributes are defined for use with the `<Attribute>` element.

1893 8.4.5 SAML Attribute Values

1894 The primary value(s) of each of the attributes defined by this profile is a UUID. The URN syntax described
1895 in Section 8.3.5 of the UUID profile is used to represent such values.

1896 However, additional information associated with the UUID value is permitted by this profile, consisting of a
1897 friendly, human-readable string, and an additional UUID representing a DCE cell or realm. The additional
1898 information is carried in the <AttributeValue> element in FriendlyName and Realm XML attributes
1899 defined in the XML namespace urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE. Note
1900 that this is not the same as the FriendlyName XML attribute defined in [SAMLCore], although it has the
1901 same basic purpose.

1902 The following schema listing shows how the profile-specific XML attributes and complex type used in an
1903 xsi:type specification are defined [SAML DCE-xsd]:

```
1904 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
1905   xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
1906   xmlns="http://www.w3.org/2001/XMLSchema"  
1907   elementFormDefault="unqualified"  
1908   attributeFormDefault="unqualified"  
1909   blockDefault="substitution"  
1910   version="2.0">  
1911   <annotation>  
1912     <documentation>  
1913       Document identifier: sstc-saml-schema-dce-2.0  
1914       Location: http://www.oasis-  
1915 open.org/committees/documents.php?wg_abbrev=security  
1916       Revision history:  
1917         V2.0 CD-04 (January, 2005):  
1918         Custom schema for DCE attribute profile, first published in  
1919 SAML 2.0.  
1920     </documentation>  
1921   </annotation>  
1922   <complexType name="DCEValueType">  
1923     <simpleContent>  
1924       <extension base="anyURI">  
1925         <attribute ref="dce:Realm" use="optional"/>  
1926         <attribute ref="dce:FriendlyName" use="optional"/>  
1927       </extension>  
1928     </simpleContent>  
1929   </complexType>  
1930   <attribute name="Realm" type="anyURI"/>  
1931   <attribute name="FriendlyName" type="string"/>  
1932 </schema>
```

1933 8.4.6 Attribute Definitions

1934 The following are the set of SAML attributes defined by this profile. In each case, an xsi:type XML
1935 attribute MAY be included in the <AttributeValue> element, but MUST have the value
1936 **dce:DCEValueType**, where the dce prefix is arbitrary and MUST be bound to the XML namespace
1937 urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE.

1938 Note that such use of xsi:type will require validating attribute consumers to include the extension
1939 schema defined by this profile.

1940 8.4.6.1 Realm

1941 This single-valued attribute represents the SAML assertion subject's DCE realm or cell.

1942 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm

1943 The single <AttributeValue> element contains a UUID in URN form identifying the SAML assertion
1944 subject's DCE realm/cell, with an optional profile-specific `FriendlyName` XML attribute containing the
1945 realm's string name.

1946 **8.4.6.2 Principal**

1947 This single-valued attribute represents the SAML assertion subject's DCE principal identity.

1948 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal

1949 The single <AttributeValue> element contains a UUID in URN form identifying the SAML assertion
1950 subject's DCE principal identity, with an optional profile-specific `FriendlyName` XML attribute containing
1951 the principal's string name.

1952 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
1953 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
1954 8.4.6.1).

1955 **8.4.6.3 Primary Group**

1956 This single-valued attribute represents the SAML assertion subject's primary DCE group membership.

1957 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group

1958 The single <AttributeValue> element contains a UUID in URN form identifying the SAML assertion
1959 subject's primary DCE group, with an optional profile-specific `FriendlyName` XML attribute containing
1960 the group's string name.

1961 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
1962 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
1963 8.4.6.1).

1964 **8.4.6.4 Groups**

1965 This multi-valued attribute represents the SAML assertion subject's DCE local group memberships.

1966 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups

1967 Each <AttributeValue> element contains a UUID in URN form identifying a DCE group membership
1968 of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute containing
1969 the group's string name.

1970 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
1971 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
1972 8.4.6.1).

1973 **8.4.6.5 Foreign Groups**

1974 This multi-valued attribute represents the SAML assertion subject's DCE foreign group memberships.

1975 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-groups

1976 Each <AttributeValue> element contains a UUID in URN form identifying a DCE foreign group
1977 membership of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute
1978 containing the group's string name.

1979 The profile-specific `Realm` XML attribute MUST be included and MUST contain a UUID in URN form
1980 identifying the DCE realm/cell of the foreign group.

1981 8.4.7 Example

1982 The following is an example of the transformation of PAC data into SAML attributes belonging to a DCE
1983 principal named "jdoe" in realm "example.com", a member of the "cubicle-dwellers" and "underpaid" local
1984 groups and an "engineers" foreign group.

```
1985 <saml:Assertion xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE" ...>
1986   <saml:Issuer>...</saml:Issuer>
1987   <saml:Subject>...</saml:Subject>
1988   <saml:AttributeStatement>
1989     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1990       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm">
1991       <saml:AttributeValue xsi:type="dce:DCEValueType" dce:FriendlyName="example.com">
1992         urn:uuid:003c6cc1-9ff8-10f9-990f-004005b13a2b
1993       </saml:AttributeValue>
1994     </saml:Attribute>
1995     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1996       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal">
1997       <saml:AttributeValue xsi:type="dce:DCEValueType" dce:FriendlyName="jdoe">
1998         urn:uuid:00305ed1-a1bd-10f9-a2d0-004005b13a2b
1999       </saml:AttributeValue>
2000     </saml:Attribute>
2001     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
2002       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group">
2003       <saml:AttributeValue xsi:type="dce:DCEValueType"
2004         dce:FriendlyName="cubicle-dwellers">
2005         urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b
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:groups">
2010       <saml:AttributeValue xsi:type="dce:DCEValueType"
2011         dce:FriendlyName="cubicle-dwellers">
2012         urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b
2013       </saml:AttributeValue>
2014       <saml:AttributeValue xsi:type="dce:DCEValueType" dce:FriendlyName="underpaid">
2015         urn:uuid:006a5a91-a2b7-10f9-824d-004005b13a2b
2016       </saml:AttributeValue>
2017     </saml:Attribute>
2018     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
2019       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-groups">
2020       <saml:AttributeValue xsi:type="dce:DCEValueType" dce:FriendlyName="engineers"
2021         dce:Realm="urn:uuid:00583221-a35f-10f9-8b6e-004005b13a2b">
2022         urn:uuid:00099cf1-a355-10f9-9e95-004005b13a2b
2023       </saml:AttributeValue>
2024     </saml:Attribute>
2025   </saml:AttributeStatement>
2026 </saml:Assertion>
```

2027 8.5 XACML Attribute Profile

2028 SAML attribute assertions may be used as input to authorization decisions made according to the OASIS
2029 eXtensible Access Control Markup Language [XACML] standard specification. Since the SAML attribute
2030 format differs from the XACML attribute format, there is a mapping that must be performed. The XACML
2031 attribute profile facilitates this mapping by standardizing naming, value syntax, and additional attribute
2032 metadata. SAML attributes generated in conformance with this profile can be mapped automatically into
2033 XACML attributes and used as input to XACML authorization decisions.

2034 8.5.1 Required Information

2035 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML (this is also the target namespace
2036 assigned in the corresponding XACML profile schema document [SAMLXAC-xsd])

2037 **Contact information:** security-services-comment@lists.oasis-open.org

2038 **Description:** Given below.

2039 **Updates:** None.

2040 8.5.2 SAML Attribute Naming

2041 The `NameFormat` XML attribute in `<Attribute>` elements **MUST** be
2042 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

2043 The `Name` XML attribute **MUST** adhere to the rules specified for that format, as defined by [SAMLCore].

2044 For purposes of human readability, there may also be a requirement for some applications to carry an
2045 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
2046 [SAMLCore]) **MAY** be used for this purpose, but is not translatable into an XACML attribute equivalent.

2047 8.5.2.1 Attribute Name Comparison

2048 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
2049 values are equal in a binary comparison. The `FriendlyName` attribute plays no role in the comparison.

2050 8.5.3 Profile-Specific XML Attributes

2051 XACML requires each attribute to carry an explicit data type. To supply this data type value, a new URI-
2052 valued XML attribute called `DataType` is defined in the XML namespace
2053 `urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML`.

2054 SAML `<Attribute>` elements conforming to this profile **MUST** include the namespace-qualified
2055 `DataType` attribute, or the value is presumed to be <http://www.w3.org/2001/XMLSchema#string>.

2056 While in principle any URI reference can be used as a data type, the standard values to be used are
2057 specified in Appendix A of the XACML 2.0 Specification [XACML]. If non-standard values are used, then
2058 each XACML PDP that will be consuming mapped SAML attributes with non-standard `DataType` values
2059 must be extended to support the new data types.

2060 8.5.4 SAML Attribute Values

2061 The syntax of the `<AttributeValue>` element's content **MUST** correspond to the data type expressed
2062 in the profile-specific `DataType` XML attribute appearing in the parent `<Attribute>` element. For data
2063 types corresponding to the types defined in Section 3.3 of [Schema2], the `xsi:type` XML attribute
2064 **SHOULD** also be used on the `<AttributeValue>` element(s).

2065 8.5.5 Profile-Specific Schema

2066 The following schema listing shows how the profile-specific `DataType` XML attribute is defined
2067 [SAMLXAC-xsd]:

```
2068 <schema  
2069   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"  
2070   xmlns="http://www.w3.org/2001/XMLSchema"  
2071   elementFormDefault="unqualified"  
2072   attributeFormDefault="unqualified"  
2073   blockDefault="substitution"  
2074   version="2.0">  
2075   <annotation>  
2076     <documentation>  
2077       Document identifier: sstc-saml-schema-xacml-2.0
```

```
2078         Location: http://www.oasis-
2079 open.org/committees/documents.php?wg_abbrev=security
2080         Revision history:
2081         V2.0 CD-04 (January, 2005):
2082         Custom schema for XACML attribute profile, first published in
2083 SAML 2.0.
2084         </documentation>
2085     </annotation>
2086     <attribute name="DataType" type="anyURI"/>
2087 </schema>
```

2088 8.5.6 Example

2089 The following is an example of a mapping of the "givenName" LDAP/X.500 attribute, representing the
2090 SAML assertion subject's first name. It also illustrates that a single SAML attribute can conform to multiple
2091 attribute profiles when they are compatible with each other.

```
2092 <saml:Attribute
2093 xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
2094     xmlns:ldapprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP"
2095     xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string"
2096     ldapprof:Encoding="LDAP"
2097     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
2098     Name="urn:oid:2.5.4.42" FriendlyName="givenName">
2099     <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>
2100 </saml:Attribute>
```

9 References

2101

- 2102 **[AES]** FIPS-197, Advanced Encryption Standard (AES), available from <http://www.nist.gov/>.
- 2103 **[Anders]** A suggestion on how to implement SAML browser bindings without using “Artifacts”,
2104 <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- 2105 **[ASN.1]** Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic
2106 notation, ITU-T Recommendation X.680, July 2002. See
2107 [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)
2108 [X.680](http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.680).
- 2109 **[eduPerson]** eduPerson.Idif. See <http://www.educase.edu/eduperson>.
- 2110 **[LDAP]** J. Hodges et al., Lightweight Directory Access Protocol (v3): Technical Specification,
2111 IETF RFC 3377, September 2002. See <http://www.ietf.org/rfc/rfc3377.txt>.
- 2112 **[Mealling]** P Leach et al, A UUID URN Namespace. Internet-Draft, draft-mealling-uuid-urn-03.
2113 January 2004
- 2114 **[MSURL]** Microsoft technical support article,
2115 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- 2116 **[NSCookie]** Persistent Client State HTTP Cookies, Netscape documentation. See
2117 http://wp.netscape.com/newsref/std/cookie_spec.html.
- 2118 **[PAOS]** Aarts, R., “Liberty Reverse HTTP Binding for SOAP Specification”, Version: 1.0,
2119 <https://www.projectliberty.org/specs/liberty-paos-v1.0.pdf>
- 2120 **[Rescorla-Sec]** E. Rescorla et al., Guidelines for Writing RFC Text on Security Considerations,
2121 <http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt>.
- 2122 **[RFC1738]** Uniform Resource Locators (URL), <http://www.ietf.org/rfc/rfc1738.txt>
- 2123 **[RFC1750]** Randomness Recommendations for Security. <http://www.ietf.org/rfc/rfc1750.txt>
- 2124 **[RFC1945]** Hypertext Transfer Protocol -- HTTP/1.0, <http://www.ietf.org/rfc/rfc1945.txt>.
- 2125 **[RFC2045]** Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message
2126 Bodies, <http://www.ietf.org/rfc/rfc2045.txt>
- 2127 **[RFC2119]** S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, IETF RFC
2128 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.
- 2129 **[RFC2246]** The TLS Protocol Version 1.0, <http://www.ietf.org/rfc/rfc2246.txt>.
- 2130 **[RFC2256]** M. Wahl, RFC 2256 - A Summary of the X.500(96) User Schema for use with LDAPv3,
2131 December 1997
- 2132 **[RFC2279]** UTF-8, a transformation format of ISO 10646, <http://www.ietf.org/rfc/rfc2279.txt>.
- 2133 **[RFC2616]** Hypertext Transfer Protocol -- HTTP/1.1, <http://www.ietf.org/rfc/rfc2616.txt>.
- 2134 **[RFC2617]** HTTP Authentication: Basic and Digest Access Authentication, IETF RFC 2617,
2135 <http://www.ietf.org/rfc/rfc2617.txt>.
- 2136 **[RFC2798]** M. Smith, Definition of the inetOrgPerson LDAP Object Class, IETF RFC 2798, April
2137 200. See <http://www.ietf.org/rfc/rfc2798.txt>.
- 2138 **[RFC2965]** D. Cristol et al., HTTP State Management Mechanism, IETF RFC 2965, October 2000.
2139 See <http://www.ietf.org/rfc/rfc2965.txt>.
- 2140 **[RFC3061]** M. Mealling, A URN Namespace of Object Identifiers, IETF RFC 3061, February 2001.
2141 See <http://www.ietf.org/rfc/rfc3061.txt>.
- 2142 **[SAMLBind]** S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language (SAML)*
2143 *V2.0*. OASIS SSTC, January 2005. Document ID sstc-saml-bindings-2.0-cd-04. See
2144 <http://www.oasis-open.org/committees/security/>.

- 2145 **[SAMLConform]** P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, January 2005. Document ID sstc-saml-conformance-2.0-cd-04. <http://www.oasis-open.org/committees/security/>.
- 2146
- 2147
- 2148 **[SAMLCore]** S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, January 2005. Document ID sstc-saml-core-2.0-cd-04. See <http://www.oasis-open.org/committees/security/>.
- 2149
- 2150
- 2151 **[SAML DCE-xsd]** S. Cantor et al., SAML DCE PAC attribute profile schema. OASIS SSTC, January 2005. Document ID sstc-saml-schema-dce-2.0. See <http://www.oasis-open.org/committees/security/>.
- 2152
- 2153
- 2154 **[SAML ECP-xsd]** S. Cantor et al., SAML ECP profile schema. OASIS SSTC, January 2005. Document ID sstc-saml-schema-ecp-2.0. See <http://www.oasis-open.org/committees/security/>.
- 2155
- 2156 **[SAML Gloss]** J. Hodges et al., *Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, January 2005. Document ID sstc-saml-glossary-2.0-cd-04. See <http://www.oasis-open.org/committees/security/>.
- 2157
- 2158
- 2159 **[SAML X500-xsd]** S. Cantor et al., SAML X.500/LDAP attribute profile schema. OASIS SSTC, January 2005. Document ID sstc-saml-schema-x500-2.0. See <http://www.oasis-open.org/committees/security/>.
- 2160
- 2161
- 2162 **[SAML Meta]** S. Cantor et al., *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, January 2005. Document ID sstc-saml-metadata-2.0-cd-04. See <http://www.oasis-open.org/committees/security/>.
- 2163
- 2164
- 2165 **[SAML Reqs]** Darren Platt et al., SAML Requirements and Use Cases, OASIS, April 2002, <http://www.oasis-open.org/committees/security/>.
- 2166
- 2167 **[SAML Sec]** F. Hirsch et al., *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, January 2005. Document ID sstc-saml-sec-consider-2.0-cd-04. See <http://www.oasis-open.org/committees/security/>.
- 2168
- 2169
- 2170 **[SAML Web]** OASIS Security Services Technical Committee website, <http://www.oasis-open.org/committees/security/>.
- 2171
- 2172 **[SAML XAC-xsd]** S. Cantor et al., SAML XACML attribute profile schema. OASIS SSTC, January 2005. Document ID sstc-saml-schema-xacml-2.0. See <http://www.oasis-open.org/committees/security/>.
- 2173
- 2174
- 2175 **[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.
- 2176
- 2177
- 2178 **[Schema2]** Paul V. Biron, Ashok Malhotra, XML Schema Part 2: Datatypes, W3C Recommendation 02 May 2001, <http://www.w3.org/TR/xmlschema-2/>
- 2179
- 2180 **[SESSION]** RL "Bob" Morgan, Support of target web server sessions in Shibboleth, <http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt>
- 2181
- 2182 **[ShibMarlena]** Marlena Erdos, Shibboleth Architecture DRAFT v1.1, <http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html> .
- 2183
- 2184 **[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> .
- 2185
- 2186 **[SSL3]** A. Frier et al., The SSL 3.0 Protocol, Netscape Communications Corp, November 1996.
- 2187
- 2188 **[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>
- 2189
- 2190 **[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>.
- 2191
- 2192
- 2193
- 2194 **[XML Enc]** D. Eastlake et al., XML Encryption Syntax and Processing, <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>, World Wide Web
- 2195

2196		Consortium.
2197	[XMLSig]	D. Eastlake et al., XML-Signature Syntax and Processing, World Wide Web Consortium, http://www.w3.org/TR/xmlsig-core/ .
2198		
2199	[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 .
2200		
2201		

2202 Appendix A. Acknowledgments

2203 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
2204 Committee, whose voting members at the time of publication were:

- 2205 • Conor Cahill, AOL
- 2206 • John Hughes, Atos Origin
- 2207 • Hal Lockhart, BEA Systems
- 2208 • Mike Beach, Boeing
- 2209 • Rebekah Metz, Booz Allen Hamilton
- 2210 • Rick Randall, Booz Allen Hamilton
- 2211 • Ronald Jacobson, Computer Associates
- 2212 • Carolina Canales-Valenzuela, Ericsson
- 2213 • Dana Kaufman, Forum Systems
- 2214 • Irving Reid, Hewlett-Packard
- 2215 • Paula Austel, IBM
- 2216 • Michael McIntosh, IBM
- 2217 • Anthony Nadalin, IBM
- 2218 • Nick Ragouzis, Individual
- 2219 • Scott Cantor, Internet2
- 2220 • Bob Morgan, Internet2
- 2221 • Peter Davis, Neustar
- 2222 • Jeff Hodges, Neustar
- 2223 • Frederick Hirsch, Nokia
- 2224 • Senthil Sengodan, Nokia
- 2225 • Abbie Barbir, Nortel Networks
- 2226 • Scott Kiestler, Novell
- 2227 • Cameron Morris, Novell
- 2228 • Paul Madsen, NTT
- 2229 • Steve Anderson, OpenNetwork
- 2230 • Ari Kermaier, Oracle
- 2231 • Vamsi Motukuru, Oracle
- 2232 • Darren Platt, Ping Identity
- 2233 • Prateek Mishra, Principal Identity
- 2234 • Jim Lien, RSA Security
- 2235 • John Linn, RSA Security
- 2236 • Rob Philpott, RSA Security
- 2237 • Dipak Chopra, SAP
- 2238 • Jahan Moreh, Sigaba
- 2239 • Bhavna Bhatnagar, Sun Microsystems
- 2240 • Eve Maler, Sun Microsystems
- 2241 • Ronald Monzillo, Sun Microsystems
- 2242 • Emily Xu, Sun Microsystems
- 2243 • Greg Whitehead, Trustgenix

2244 The editors also would like to acknowledge the following people for their contributions to previous versions
2245 of the OASIS Security Assertions Markup Language Standard:

- 2246 • Stephen Farrell, Baltimore Technologies
- 2247 • David Orchard, BEA Systems
- 2248 • Krishna Sankar, Cisco Systems
- 2249 • Zahid Ahmed, CommerceOne
- 2250 • Carlisle Adams, Entrust
- 2251 • Tim Moses, Entrust
- 2252 • Nigel Edwards, Hewlett-Packard
- 2253 • Joe Pato, Hewlett-Packard
- 2254 • Bob Blakley, IBM
- 2255 • Marlena Erdos, IBM
- 2256 • Marc Chanliau, Netegrity
- 2257 • Chris McLaren, Netegrity
- 2258 • Lynne Rosenthal, NIST
- 2259 • Mark Skall, NIST
- 2260 • Simon Godik, Overxeer
- 2261 • Charles Norwood, SAIC
- 2262 • Evan Prodromou, Securant
- 2263 • Robert Griffin, RSA Security (former editor)
- 2264 • Sai Allarvarpu, Sun Microsystems
- 2265 • Chris Ferris, Sun Microsystems
- 2266 • Mike Myers, Traceroute Security
- 2267 • Phillip Hallam-Baker, VeriSign (former editor)
- 2268 • James Vanderbeek, Vodafone
- 2269 • Mark O'Neill, Vordel
- 2270 • Tony Palmer, Vordel

2271 Finally, the editors wish to acknowledge the following people for their contributions of material used as
2272 input to the OASIS Security Assertions Markup Language specifications:

- 2273 • Thomas Gross, IBM
- 2274 • Birgit Pfitzmann, IBM

2275 **Appendix B. Notices**

2276 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
2277 might be claimed to pertain to the implementation or use of the technology described in this document or
2278 the extent to which any license under such rights might or might not be available; neither does it represent
2279 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
2280 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
2281 available for publication and any assurances of licenses to be made available, or the result of an attempt
2282 made to obtain a general license or permission for the use of such proprietary rights by implementors or
2283 users of this specification, can be obtained from the OASIS Executive Director.

2284 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
2285 other proprietary rights which may cover technology that may be required to implement this specification.
2286 Please address the information to the OASIS Executive Director.

2287 **Copyright © OASIS Open 2005. All Rights Reserved.**

2288 This document and translations of it may be copied and furnished to others, and derivative works that
2289 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
2290 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
2291 this paragraph are included on all such copies and derivative works. However, this document itself may
2292 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
2293 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
2294 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
2295 into languages other than English.

2296 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
2297 or assigns.

2298 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2299 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
2300 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
2301 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.