



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

Committee Draft 02, 24 September 2004

Document identifier:

sstc-saml-profiles-2.0-cd-02

Location:

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

Editors:

John Hughes, Atos Origin
Scott Cantor, Internet2
Prateek Mishra, Netegrity
Frederick Hirsch, Nokia
Rob Philpott, RSA Security
Jeff Hodges, Sun Microsystems
Eve Maler, Sun Microsystems

SAML V2.0 Contributors:

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

45 Jeff Hodges, Sun Microsystems
46 Eve Maler, Sun Microsystems
47 Ron Monzillo, Sun Microsystems
48 Greg Whitehead, Trustgenix

49 **Abstract:**

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

53 **Status:**

54 This is a **second Committee Draft** approved by the Security Services Technical Committee on
55 21 September 2004.

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

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

Table of Contents

| | | |
|-----|---|----|
| 65 | | |
| 66 | 1 Introduction..... | 7 |
| 67 | 1.1 Profile Concepts..... | 7 |
| 68 | 1.2 Notation..... | 7 |
| 69 | 2 Specification of Additional Profiles..... | 9 |
| 70 | 2.1 Guidelines for Specifying Profiles..... | 9 |
| 71 | 2.2 Guidelines for Specifying Attribute Profiles..... | 9 |
| 72 | 3 Confirmation Method Identifiers..... | 11 |
| 73 | 3.1 Holder of Key..... | 11 |
| 74 | 3.2 Sender Vouches..... | 11 |
| 75 | 3.3 Bearer..... | 12 |
| 76 | 4 SSO Profiles of SAML..... | 13 |
| 77 | 4.1 Web Browser SSO Profile..... | 13 |
| 78 | 4.1.1 Required Information..... | 13 |
| 79 | 4.1.2 Profile Overview..... | 13 |
| 80 | 4.1.3 Profile Description..... | 15 |
| 81 | 4.1.3.1 HTTP Request to Service Provider..... | 15 |
| 82 | 4.1.3.2 Service Provider Determines Identity Provider..... | 15 |
| 83 | 4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider..... | 15 |
| 84 | 4.1.3.4 Identity Provider Identifies Principal..... | 16 |
| 85 | 4.1.3.5 Identity Provider Issues <Response> to Service Provider..... | 16 |
| 86 | 4.1.3.6 Service Provider Grants or Denies Access to User Agent..... | 16 |
| 87 | 4.1.4 Use of Authentication Request Protocol..... | 16 |
| 88 | 4.1.4.1 <AuthnRequest> Usage..... | 17 |
| 89 | 4.1.4.2 <Response> Usage..... | 17 |
| 90 | 4.1.4.3 <Response> Message Processing Rules..... | 18 |
| 91 | 4.1.4.4 Artifact-Specific <Response> Message Processing Rules..... | 18 |
| 92 | 4.1.4.5 POST-Specific Processing Rules..... | 18 |
| 93 | 4.1.5 Unsolicited Responses..... | 18 |
| 94 | 4.1.6 Use of Metadata..... | 19 |
| 95 | 4.2 Enhanced Client or Proxy (ECP) Profile..... | 19 |
| 96 | 4.2.1 Required Information..... | 20 |
| 97 | 4.2.2 Profile Overview..... | 20 |
| 98 | 4.2.3 Profile Description..... | 22 |
| 99 | 4.2.3.1 ECP issues HTTP Request to Service Provider..... | 22 |
| 100 | 4.2.3.2 Service Provider Issues <AuthnRequest> to ECP..... | 22 |
| 101 | 4.2.3.3 ECP Determines Identity Provider..... | 23 |
| 102 | 4.2.3.4 ECP issues <AuthnRequest> to Identity Provider..... | 23 |
| 103 | 4.2.3.5 Identity Provider Identifies Principal..... | 23 |
| 104 | 4.2.3.6 Identity Provider issues <Response> to ECP, targeted at service provider..... | 23 |
| 105 | 4.2.3.7 ECP Conveys <Response> Message to Service Provider..... | 23 |
| 106 | 4.2.3.8 Service Provider Grants or Denies Access to Principal..... | 24 |
| 107 | 4.2.4 ECP Profile Schema Usage..... | 24 |
| 108 | 4.2.4.1 PAOS Request Header Block: SP to ECP..... | 25 |

| | | |
|-----|---|----|
| 109 | 4.2.4.2 ECP Request Header Block : SP to ECP..... | 25 |
| 110 | 4.2.4.3 ECP RelayState Header Block: SP to ECP..... | 26 |
| 111 | 4.2.4.4 ECP Response Header Block : IdP to ECP..... | 27 |
| 112 | 4.2.4.5 PAOS Response Header Block : ECP to SP..... | 28 |
| 113 | 4.2.5 Security Considerations..... | 28 |
| 114 | 4.3 Identity Provider Discovery Profile..... | 29 |
| 115 | 4.3.1 Common Domain Cookie..... | 29 |
| 116 | 4.3.2 Setting the Common Domain Cookie..... | 29 |
| 117 | 4.3.3 Obtaining the Common Domain Cookie..... | 29 |
| 118 | 4.4 Single Logout Profile..... | 30 |
| 119 | 4.4.1 Required Information..... | 30 |
| 120 | 4.4.2 Profile Overview..... | 30 |
| 121 | 4.4.3 Profile Description..... | 32 |
| 122 | 4.4.3.1 <LogoutRequest> Issued by Session Participant to Identity Provider..... | 32 |
| 123 | 4.4.3.2 Identity Provider Determines Session Participants..... | 33 |
| 124 | 4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session Participant/Authority..... | 33 |
| 125 | 4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity Provider..... | 33 |
| 126 | 4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant..... | 34 |
| 127 | 4.4.4 Use of Single Logout Protocol..... | 34 |
| 128 | 4.4.4.1 <LogoutRequest> Usage..... | 34 |
| 129 | 4.4.4.2 <LogoutResponse> Usage..... | 34 |
| 130 | 4.4.5 Use of Metadata..... | 34 |
| 131 | 4.5 Name Identifier Management Profile..... | 34 |
| 132 | 4.5.1 Required Information..... | 35 |
| 133 | 4.5.2 Profile Overview..... | 35 |
| 134 | 4.5.3 Profile Description..... | 36 |
| 135 | 4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider..... | 36 |
| 136 | 4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service Provider..... | 36 |
| 137 | 4.5.4 Use of Name Identifier Management Protocol..... | 37 |
| 138 | 4.5.4.1 <ManageNameIDRequest> Usage..... | 37 |
| 139 | 4.5.4.2 <ManageNameIDResponse> Usage..... | 37 |
| 140 | 4.5.5 Use of Metadata..... | 37 |
| 141 | 5 Artifact Resolution Profile..... | 38 |
| 142 | 5.1 Required Information..... | 38 |
| 143 | 5.2 Profile Overview..... | 38 |
| 144 | 5.3 Profile Description..... | 39 |
| 145 | 5.3.1 <ArtifactResolve> issued by Requesting Entity..... | 39 |
| 146 | 5.3.2 <ArtifactResponse> issued by Responding Entity..... | 39 |
| 147 | 5.4 Use of Artifact Resolution Protocol..... | 39 |
| 148 | 5.4.1 <ArtifactResolve> Usage..... | 39 |
| 149 | 5.4.2 <ArtifactResponse> Usage..... | 39 |
| 150 | 5.5 Use of Metadata..... | 40 |
| 151 | 6 Assertion Query/Request Profile..... | 41 |
| 152 | 6.1 Required Information..... | 41 |
| 153 | 6.2 Profile Overview..... | 41 |
| 154 | 6.3 Profile Description..... | 42 |

| | | |
|-----|--|----|
| 155 | 6.3.1 Query/Request issued by Requesting Entity..... | 42 |
| 156 | 6.3.2 <Response> issued by SAML Authority..... | 42 |
| 157 | 6.4 Use of Query/Request Protocol..... | 42 |
| 158 | 6.4.1 Query/Request Usage..... | 42 |
| 159 | 6.4.2 <Response> Usage..... | 42 |
| 160 | 6.5 Use of Metadata..... | 42 |
| 161 | 7 Name Identifier Mapping Profile..... | 44 |
| 162 | 7.1 Required Information..... | 44 |
| 163 | 7.2 Profile Overview..... | 44 |
| 164 | 7.3 Profile Description..... | 45 |
| 165 | 7.3.1 <NameIDMappingRequest> issued by Requesting Entity..... | 45 |
| 166 | 7.3.2 <NameIDMappingResponse> issued by Identity Provider..... | 45 |
| 167 | 7.4 Use of Name Identifier Mapping Protocol..... | 45 |
| 168 | 7.4.1 <NameIDMappingRequest> Usage..... | 45 |
| 169 | 7.4.2 <NameIDMappingResponse> Usage..... | 45 |
| 170 | 7.4.2.1 Limiting Use of Mapped Identifier..... | 46 |
| 171 | 7.5 Use of Metadata..... | 46 |
| 172 | 8 SAML Attribute Profiles..... | 47 |
| 173 | 8.1 Basic Attribute Profile..... | 47 |
| 174 | 8.1.1 Required Information..... | 47 |
| 175 | 8.1.2 SAML Attribute Naming..... | 47 |
| 176 | 8.1.2.1 Attribute Name Comparison..... | 47 |
| 177 | 8.1.3 Profile-Specific XML Attributes..... | 47 |
| 178 | 8.1.4 SAML Attribute Values..... | 47 |
| 179 | 8.1.5 Example..... | 47 |
| 180 | 8.2 X.500/LDAP Attribute Profile..... | 47 |
| 181 | 8.2.1 Required Information..... | 48 |
| 182 | 8.2.2 SAML Attribute Naming..... | 48 |
| 183 | 8.2.2.1 Attribute Name Comparison..... | 48 |
| 184 | 8.2.3 Profile-Specific XML Attributes..... | 48 |
| 185 | 8.2.4 SAML Attribute Values..... | 48 |
| 186 | 8.2.5 Profile-Specific Schema..... | 49 |
| 187 | 8.2.6 Example..... | 49 |
| 188 | 8.3 UUID Attribute Profile..... | 50 |
| 189 | 8.3.1 Required Information..... | 50 |
| 190 | 8.3.2 UUID and GUID Background..... | 50 |
| 191 | 8.3.3 SAML Attribute Naming..... | 50 |
| 192 | 8.3.3.1 Attribute Name Comparison..... | 50 |
| 193 | 8.3.4 Profile-Specific XML Attributes..... | 51 |
| 194 | 8.3.5 SAML Attribute Values..... | 51 |
| 195 | 8.3.6 Example..... | 51 |
| 196 | 8.4 DCE PAC Attribute Profile..... | 51 |
| 197 | 8.4.1 Required Information..... | 51 |
| 198 | 8.4.2 PAC Description..... | 51 |

| | | |
|-----|--|----|
| 199 | 8.4.3 SAML Attribute Naming..... | 52 |
| 200 | 8.4.3.1 Attribute Name Comparison..... | 52 |
| 201 | 8.4.4 Profile-Specific XML Attributes..... | 52 |
| 202 | 8.4.5 SAML Attribute Values..... | 52 |
| 203 | 8.4.6 Attribute Definitions..... | 52 |
| 204 | 8.4.6.1 Realm..... | 53 |
| 205 | 8.4.6.2 Principal..... | 53 |
| 206 | 8.4.6.3 Primary Group..... | 53 |
| 207 | 8.4.6.4 Groups..... | 53 |
| 208 | 8.4.6.5 Foreign Groups..... | 53 |
| 209 | 8.4.7 Example..... | 54 |
| 210 | 8.5 XACML Attribute Profile..... | 54 |
| 211 | 8.5.1 Required Information..... | 55 |
| 212 | 8.5.2 SAML Attribute Naming..... | 55 |
| 213 | 8.5.2.1 Attribute Name Comparison..... | 55 |
| 214 | 8.5.3 Profile-Specific XML Attributes..... | 55 |
| 215 | 8.5.4 SAML Attribute Values..... | 55 |
| 216 | 8.5.5 Profile-Specific Schema..... | 55 |
| 217 | 8.5.6 Example..... | 56 |
| 218 | 9 References..... | 57 |

219

1 Introduction

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

223 A separate specification ([SAMLCore]) defines the SAML assertions and request-response protocol
224 messages themselves, and another ([SAMLBind]) defines bindings of SAML protocol messages to
225 underlying communications and messaging protocols.

1.1 Profile Concepts

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

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

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

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

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

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

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

1.2 Notation

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

258 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD
259 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as
260 described in IETF RFC 2119 [RFC2119].

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

262 `Example code listings appear like this.`

263 **Note:** Non-normative notes and explanations appear like this.

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

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

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

269

2 Specification of Additional Profiles

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

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

2.1 Guidelines for Specifying Profiles

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

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

2.2 Guidelines for Specifying Attribute Profiles

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

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

310 use when processing attributes, queries, etc.

311 5. Syntax and restrictions on values acceptable in the SAML <AttributeValue> element, including
312 whether the `xsi:type` XML attribute can or should be used.

313

3 Confirmation Method Identifiers

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

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

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

3.1 Holder of Key

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

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

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

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

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

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

3.2 Sender Vouches

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

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

357 **3.3 Bearer**

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

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

362 **Example:** The bearer of the assertion can confirm itself as the subject, provided the assertion is delivered
363 in a message sent to "<https://www.serviceprovider.com/saml/consumer>" before 1:37 PM GMT on March
364 19th, 2004, in response to a request with ID "_1234567890".

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

371 4 SSO Profiles of SAML

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

379 4.1 Web Browser SSO Profile

380 In the scenario supported by the web browser SSO profile, a web user either accesses a resource at a

381 service provider, or accesses an identity provider such that the service provider and desired resource are

382 understood or implicit. The web user authenticates (or has already authenticated) to the identity provider,

383 which then produces an authentication assertion (possibly with input from the service provider) and the

384 service provider consumes the assertion to establish a security context for the web user. During this

385 process, a name identifier might also be established between the providers for the principal, subject to the

386 parameters of the interaction and the consent of the parties.

387 To implement this scenario, a profile of the SAML Authentication Request protocol is used, in conjunction

388 with the HTTP Redirect, HTTP POST and HTTP Artifact bindings.

389 It is assumed that the user is using a standard commercial browser and can authenticate to the identity

390 provider by some means outside the scope of SAML.

391 4.1.1 Required Information

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

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

394 **SAML Confirmation Method Identifiers:** The SAML V2.0 "bearer" confirmation method identifier,

395 urn:oasis:names:tc:SAML:2.0:cm:bearer, is used by this profile.

396 **Description:** Given below.

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

398 4.1.2 Profile Overview

399 Figure 1 illustrates the basic template for achieving SSO. The following steps are described by the profile.

400 Within an individual step, there may be one or more actual message exchanges depending on the binding

401 used for that step and other implementation-dependent behavior.

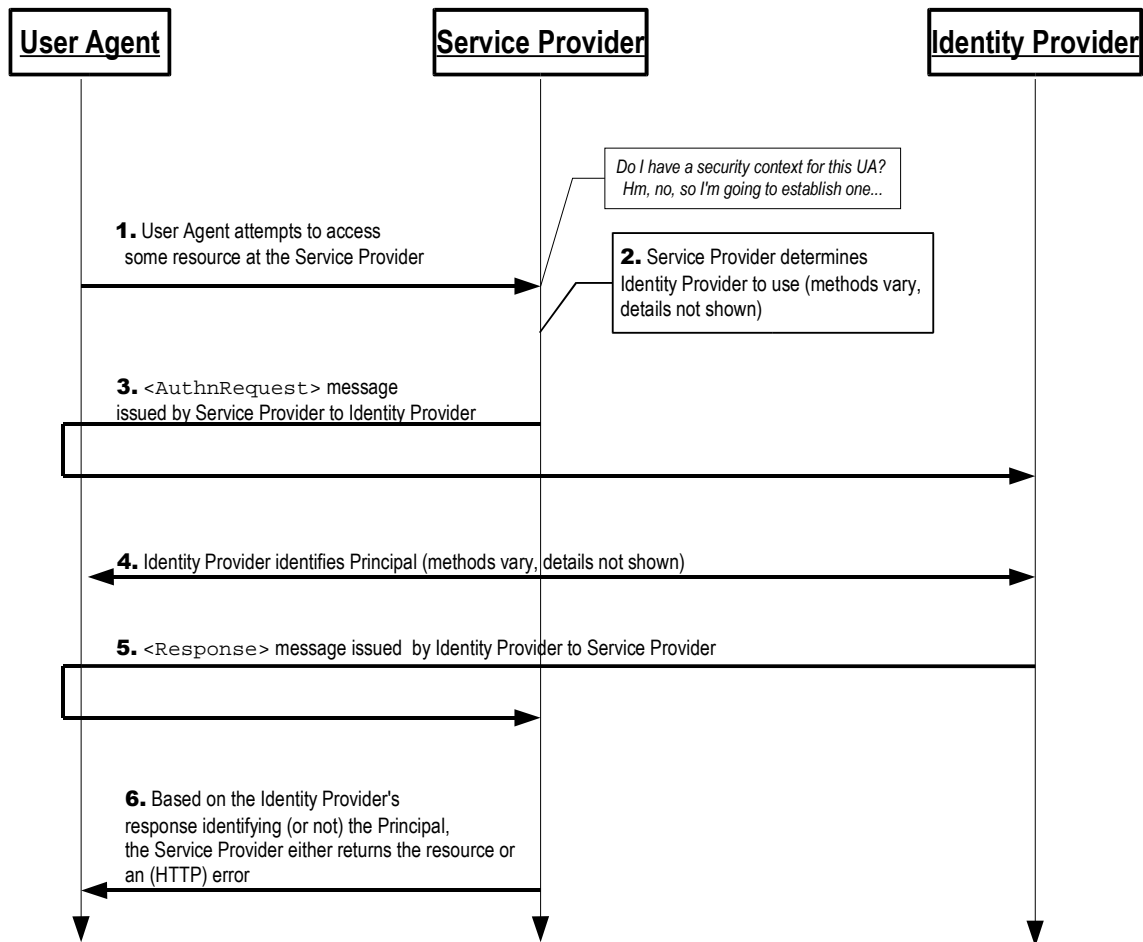


Figure 1

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

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

405 **2. Service Provider Determines Identity Provider**

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

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

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

414 **4. Identity Provider identifies Principal**

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

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

419 In step 5, the identity provider issues a <Response> message to be delivered by the user agent
420 to the service provider. Either the HTTP POST, or HTTP Artifact binding can be used to transfer

421 the message to the service provider through the user agent. The message may indicate an error,
422 or will include (at least) an authentication assertion. The HTTP Redirect binding MUST NOT be
423 used, as the response will typically exceed the URL length permitted by most user agents.

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

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

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

430 **4.1.3 Profile Description**

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

433 **Single Sign-On Service**

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

436 **Assertion Consumer Service**

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

439 **4.1.3.1 HTTP Request to Service Provider**

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

446 **4.1.3.2 Service Provider Determines Identity Provider**

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

453 **4.1.3.3 <AuthnRequest> Is Issued by Service Provider to Identity Provider**

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

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

466 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or TLS
467 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <AuthnRequest> message MAY
468 be signed, if authentication of the request issuer is required. The HTTP Artifact binding, if used, also
469 provides for an alternate means of authenticating the request issuer when the artifact is dereferenced.
470 The identity provider MUST process the <AuthnRequest> message as described in [SAMLCore]. This
471 may constrain the subsequent interactions with the user agent, for example if the `IsPassive` attribute is
472 included.

473 **4.1.3.4 Identity Provider Identifies Principal**

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

481 **4.1.3.5 Identity Provider Issues <Response> to Service Provider**

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

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

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

495 It is RECOMMENDED that the HTTP requests in this step be made over either SSL 3.0 ([SSL3]) or TLS
496 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <Assertion> element(s) in the
497 <Response> MUST be signed, if the HTTP POST binding is used, and MAY be signed if the HTTP-
498 Artifact binding is used.

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

501 **4.1.3.6 Service Provider Grants or Denies Access to User Agent**

502 To complete the profile, the service provider processes the <Response> and <Assertion>(s) and
503 grants or denies access to the resource. The service provider MAY establish a security context with the
504 user agent using any session mechanism it chooses. Any subsequent use of the <Assertion>(s)
505 provided are at the discretion of the service provider and other relying parties, subject to any restrictions
506 on use contained within them.

507 **4.1.4 Use of Authentication Request Protocol**

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

512 4.1.4.1 <AuthnRequest> Usage

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

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

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

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

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

531 4.1.4.2 <Response> Usage

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

- 535 • The <Issuer> element MAY be omitted, but if present it MUST contain the unique identifier of the
536 issuing identity provider; the Format attribute MUST be omitted or have a value of
537 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.
- 538 • It MUST contain at least one <Assertion>. Each assertion's <Issuer> element MUST contain the
539 unique identifier of the issuing identity provider; the Format attribute MUST be omitted or have a value
540 of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.
- 541 • The set of one or more assertions MUST contain at least one <AuthnStatement> that reflects the
542 authentication of the principal to the identity provider.
- 543 • At least one assertion containing an <AuthnStatement> MUST contain a <Subject> element with
544 at least one <SubjectConfirmation> element containing a Method of
545 urn:oasis:names:tc:SAML:2.0:cm:bearer. If the identity provider supports the Single Logout
546 profile, defined in Section 4.4, any such authentication statements MUST include a SessionIndex
547 attribute to enable per-session logout requests by the service provider.
- 548 • Any bearer <SubjectConfirmationData> elements MUST contain a Recipient attribute
549 containing the service provider's assertion consumer service URL and a NotOnOrAfter attribute that
550 limits the window during which the assertion can be delivered. It MAY contain an Address attribute
551 limiting the client address from which the assertion can be delivered. It MUST NOT contain a
552 NotBefore attribute. If the containing message is in response to an <AuthnRequest>, then the
553 InResponseTo attribute MUST match the request's ID.
- 554 • Other statements and confirmation methods MAY be included in the assertion(s) at the discretion of
555 the identity provider. In particular, <AttributeStatement> elements MAY be included. The
556 <AuthnRequest> MAY contain an AttributeConsumingServiceIndex XML attribute
557 referencing information about desired or required attributes in [SAMLMeta]. The identity provider MAY
558 ignore this, or send other attributes at its discretion.
- 559 • The assertion(s) containing a bearer subject confirmation MUST contain an
560 <AudienceRestriction> including the service provider's unique identifier as an <Audience>.

- 561 • Other conditions (and other <Audience> elements) MAY be included as requested by the service
562 provider or at the discretion of the identity provider. (Of course, all such conditions MUST be
563 understood by and accepted by the service provider in order for the assertion to be considered valid.)
564 The identity provider is NOT obligated to honor the requested set of <Conditions> in the
565 <AuthnRequest>, if any.

566 **4.1.4.3 <Response> Message Processing Rules**

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

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

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

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

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

584 **4.1.4.4 Artifact-Specific <Response> Message Processing Rules**

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

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

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

591 **4.1.4.5 POST-Specific Processing Rules**

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

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

597 **4.1.5 Unsolicited Responses**

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

600 An unsolicited <Response> MUST NOT contain an `InResponseTo` attribute, nor should any bearer
601 <SubjectConfirmationData> elements contain one. If metadata as specified in [SAMLMeta] is used,
602 the <Response> or artifact SHOULD be delivered to the <md:AssertionConsumerService> endpoint

603 of the service provider designated as the default.
604 Of special mention is that the identity provider SHOULD include a binding-specific "RelayState" parameter
605 that indicates, based on mutual agreement with the service provider, how to handle subsequent
606 interactions with the user agent. This MAY be the URL of a resource at the service provider.

607 4.1.6 Use of Metadata

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

611 The `<md:IDPDescriptor>` element's `WantAuthnRequestsSigned` attribute MAY be used by an
612 identity provider to document a requirement that requests be signed. The `<md:SPDescriptor>`
613 element's `AuthnRequestsSigned` attribute MAY be used by a service provider to document the
614 intention to sign all of its requests.

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

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

624 The `<md:SPDescriptor>` element's `WantAssertionsSigned` attribute MAY be used by a service
625 provider to document a requirement that assertions delivered with this profile be signed. This is in addition
626 to any requirements for signing imposed by the use of a particular binding.

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

629 The `<md:AttributeConsumerDescriptor>` element MAY be used to document the service provider's
630 need or desire for SAML attributes to be delivered along with authentication information. The actual
631 inclusion of attributes is of course at the discretion of the identity provider. One or more
632 `<md:AttributeConsumingService>` elements MAY be included in its metadata, each with an `index`
633 attribute to distinguish different services that MAY be specified by reference in the `<AuthnRequest>`
634 message. The `isDefault` attribute is used to specify a default set of attribute requirements.

635 4.2 Enhanced Client or Proxy (ECP) Profile

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

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

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

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

650 4.2.1 Required Information

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

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

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

656 **Description:** Given below.

657 **Updates:** None.

658 4.2.2 Profile Overview

659 As introduced above, the ECP profile specifies interactions between enhanced clients or proxies and
660 service providers and identity providers. It is a specific application of the SSO profile described in Section
661 4.1. If not otherwise specified by this profile, and if not specific to the use of browser-based bindings, the
662 rules specified in Section 4.1 MUST be observed.

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

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

666 This allows a service provider to make an authentication request to the ECP without the need to know
667 or discover the appropriate identity provider (effectively bypassing step 2 of the SSO profile in Section
668 4.1).

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

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

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

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

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

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

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

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

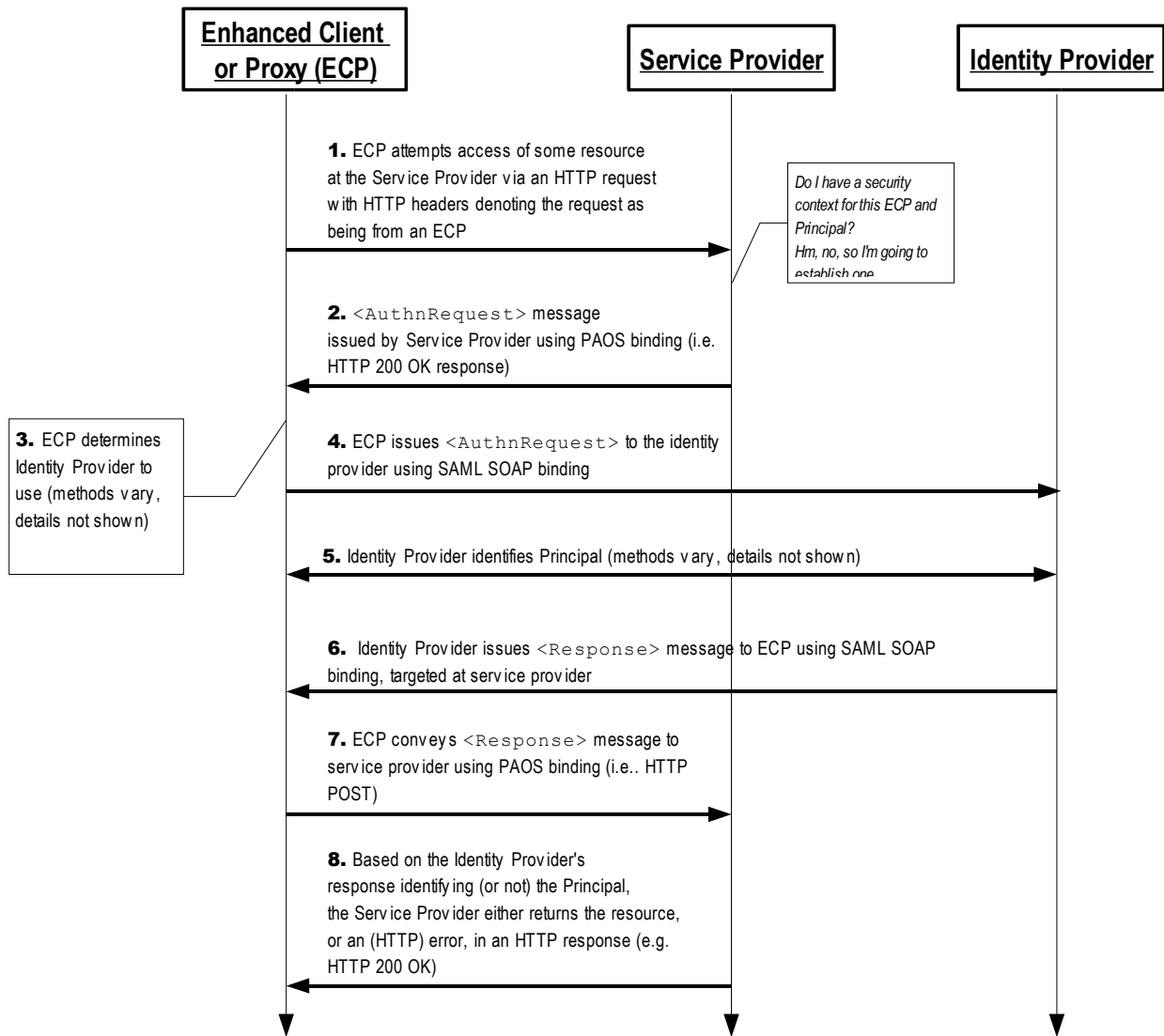


Figure 2

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

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

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

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

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

706 **3. ECP Determines Identity Provider**

707 In step 3, the ECP obtains the location of an endpoint at an identity provider for the authentication
 708 request protocol that supports its preferred binding. The means by which this is accomplished is

709 implementation-dependent. The ECP MAY use the SAML identity provider discovery profile
710 described in Section 4.3.

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

712 In step 4, the ECP conveys the <AuthnRequest> to the identity provider identified in step 3
713 using the SAML SOAP binding [SAMLBind].

714 **5. Identity Provider identifies Principal**

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

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

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

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

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

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

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

729 **4.2.3 Profile Description**

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

731 **4.2.3.1 ECP issues HTTP Request to Service Provider**

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

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

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

```
742 GET /index HTTP/1.1  
743 Host: identity-service.example.com  
744 Accept: text/html; application/vnd.paos+xml  
745 PAOS: ver='urn:liberty:paos:2003-08' ;  
746 'urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp'
```

747 **4.2.3.2 Service Provider Issues <AuthnRequest> to ECP**

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

752 The SOAP envelope MUST contain:

- 753 1. An <AuthnRequest> element in the SOAP body, intended for the ultimate SOAP recipient, the

- 754 identity provider.
- 755 2. A PAOS SOAP header block targeted at the ECP using the SOAP `actor` value of
756 `http://schemas.xmlsoap.org/soap/actor/next`. This header block provides control
757 information such as the URL to which to send the response in this solicit-response message
758 exchange pattern.
 - 759 3. An ECP profile-specific Request SOAP header block targeted at the ECP using the SOAP actor
760 `http://schemas.xmlsoap.org/soap/actor/next`. The ECP Request header block defines
761 information related to the authentication request that the ECP may need to process it, such as a list
762 of identity providers acceptable to the service provider, whether the ECP may interact with the
763 principal through the client, and the service provider's human-readable name that may be displayed
764 to the principal.

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

768 **4.2.3.3 ECP Determines Identity Provider**

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

770 **4.2.3.4 ECP issues <AuthnRequest> to Identity Provider**

771 The ECP MUST remove the PAOS, ECP RelayState, and ECP Request header blocks before passing the
772 <AuthnRequest> message on to the identity provider, using the SAML SOAP binding.

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

775 Prior to or subsequent to this step, the identity provider MUST establish the identity of the principal by
776 some means, or it MUST return an error <Response> in step 4, described below.

777 **4.2.3.5 Identity Provider Identifies Principal**

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

785 **4.2.3.6 Identity Provider issues <Response> to ECP, targeted at service provider**

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

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

793 **4.2.3.7 ECP Conveys <Response> Message to Service Provider**

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

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

802 The RelayState header block value is typically provided by the service provider to the ECP with its request,
803 but if the identity provider is producing an unsolicited response (without having received a corresponding
804 SAML request), then it SHOULD include a RelayState header block that indicates, based on mutual
805 agreement with the service provider, how to handle subsequent interactions with the ECP. This MAY be
806 the URL of a resource at the service provider.

807 If the service provider included a RelayState SOAP header block in its request to the ECP, or if the identity
808 provider included a RelayState SOAP header block with its response, then the ECP MUST include an
809 identical header block with the SAML response sent to the service provider. The service provider's value
810 for this header block (if any) MUST take precedence.

811 4.2.3.8 Service Provider Grants or Denies Access to Principal

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

817 4.2.4 ECP Profile Schema Usage

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

```
820 <schema
821   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
822   xmlns="http://www.w3.org/2001/XMLSchema"
823   xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
824   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
825   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
826   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
827   elementFormDefault="unqualified"
828   attributeFormDefault="unqualified"
829   blockDefault="substitution"
830   version="2.0">
831   <import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
832     schemaLocation="sstc-saml-schema-protocol-2.0.xsd"/>
833   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
834     schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
835   <import namespace="http://schemas.xmlsoap.org/soap/envelope/"
836     schemaLocation="http://schemas.xmlsoap.org/soap/envelope/">
837
838   <element name="Request" type="ecp:RequestType"/>
839   <complexType name="RequestType">
840     <sequence>
841       <element ref="saml:Issuer"/>
842       <element ref="samlp:IDPList" minOccurs="0"/>
843     </sequence>
844     <attribute ref="SOAP-ENV:mustUnderstand" use="required"/>
845     <attribute ref="SOAP-ENV:actor" use="required"/>
846     <attribute name="ProviderName" type="string" use="optional"/>
847     <attribute name="IsPassive" type="boolean" use="optional"/>
848   </complexType>
849
850   <element name="Response" type="ecp:ResponseType"/>
851   <complexType name="ResponseType">
852     <attribute ref="SOAP-ENV:mustUnderstand" use="required"/>
```

```

851         <attribute ref="SOAP-ENV:actor" use="required"/>
852         <attribute name="AssertionConsumerServiceURL" type="anyURI"
853 use="required"/>
854     </complexType>

855     <element name="RelayState" type="ecp:RelayStateType"/>
856     <complexType name="RelayStateType">
857         <simpleContent>
858             <extension base="string">
859                 <attribute ref="SOAP-ENV:mustUnderstand"
860 use="required"/>
861                 <attribute ref="SOAP-ENV:actor" use="required"/>
862             </extension>
863         </simpleContent>
864     </complexType>
865 </schema>

```

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

867 **4.2.4.1 PAOS Request Header Block: SP to ECP**

868 The PAOS Request header block signals the use of PAOS processing and includes the following
869 attributes:

870 responseConsumerURL [Required]

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

876 service [Required]

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

879 SOAP-ENV:mustUnderstand [Required]

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

882 SOAP-ENV:actor [Required]

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

884 messageID [Optional]

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

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

889 **4.2.4.2 ECP Request Header Block : SP to ECP**

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

893 SOAP-ENV:mustUnderstand [Required]

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

896 SOAP-ENV:actor [Required]

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

898 `ProviderName` [Optional]

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

900 `IsPassive` [Optional]

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

904 `<saml:Issuer>` [Required]

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

908 `<samlp:IDPList>` [Optional]

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

911 4.2.4.3 ECP RelayState Header Block: SP to ECP

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

915 `SOAP-ENV:mustUnderstand` [Required]

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

917 `SOAP-ENV:actor` [Required]

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

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

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

```
925 <SOAP-ENV:Envelope
926     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
927     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
928     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
929   <SOAP-ENV:Header>
930     <paos:Request xmlns:paos="urn:liberty:paos:2003-08"
931         responseConsumerURL="http://identity-service.example.com/abc"
932         messageID="6c3a4f8b9c2d" SOAP-
933     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next" SOAP-
934     ENV:mustUnderstand="1"
935         service="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp">
936     </paos:Request>
937     <ecp:Request xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
938         SOAP-ENV:mustUnderstand="1" SOAP-
939     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
940         ProviderName="Service Provider X" IsPassive="0">
941     <saml:Issuer>https://ServiceProvider.example.com</saml:Issuer>
942     <samlp:IDPList>
943       <samlp:IDPEntry ProviderID="https://IdentityProvider.example.com"
944         Name="Identity Provider X"
945         Loc="https://IdentityProvider.example.com/saml2/sso"
```

```

946     </samlp:IDPEntry>
947     <samlp:GetComplete>
948     https://ServiceProvider.example.com/idplist?id=604be136-fe91-441e-afb8
949     </samlp:GetComplete>
950   </samlp:IDPList>
951 </ecp:Request>
952 <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
953   SOAP-ENV:mustUnderstand="1" SOAP-
954   ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
955   ...
956 </ecp:RelayState>
957 </SOAP-ENV:Header>
958 <SOAP-ENV:Body>
959   <samlp:AuthnRequest> ... </samlp:AuthnRequest>
960 </SOAP-ENV:Body>
961 </SOAP-ENV:Envelope>

```

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

```

965 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
966   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
967   <SOAP-ENV:Body>
968     <samlp:AuthnRequest> ... </samlp:AuthnRequest>
969   </SOAP-ENV:Body>
970 </SOAP-ENV:Envelope>

```

971 4.2.4.4 ECP Response Header Block : IdP to ECP

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

974 SOAP-ENV:mustUnderstand [Required]

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

977 SOAP-ENV:actor [Required]

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

979 AssertionConsumerServiceURL [Required]

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

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

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

989 The ECP Response SOAP header has no element content.

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

```

991 <SOAP-ENV:Envelope
992   xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
993   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
994   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
995   <SOAP-ENV:Header>

```

```

996     <ecp:Response SOAP-ENV:mustUnderstand="1" SOAP-
997     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
998     AssertionConsumerServiceURL="https://ServiceProvider.example.com/ecp_assertion_
999     consumer"/>
1000   </SOAP-ENV:Header>
1001   <SOAP-ENV:Body>
1002     <samlp:Response> ... </samlp:Response>
1003   </SOAP-ENV:Body>
1004 </SOAP-ENV:Envelope>

```

1005 4.2.4.5 PAOS Response Header Block : ECP to SP

1006 The PAOS Response header block includes the following attributes:

1007 SOAP-ENV:mustUnderstand [Required]

1008 The value MUST be 1 (true). A SOAP fault MUST be generated if the PAOS header block is not
1009 understood.

1010 SOAP-ENV:actor [Required]

1011 The value MUST be http://schemas.xmlsoap.org/soap/actor/next.

1012 refToMessageID [Optional]

1013 Allows correlation with the PAOS request. This optional attribute (and the header block as a whole)
1014 MUST be added by the ECP if the corresponding PAOS request specified the messageID attribute.
1015 Note that the equivalent functionality is provided in SAML using <AuthnRequest> and <Response>
1016 correlation.

1017 The PAOS Response SOAP header has no element content.

1018 Following is an example of an ECP-to-SP response.

```

1019 <SOAP-ENV:Envelope
1020   xmlns:paos="urn:liberty:paos:2003-08"
1021   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1022   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
1023   <SOAP-ENV:Header>
1024     <paos:Response refToMessageID="6c3a4f8b9c2d" SOAP-
1025     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next/" SOAP-
1026     ENV:mustUnderstand="1"/>
1027     <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
1028     SOAP-ENV:mustUnderstand="1" SOAP-
1029     ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">
1030     ...
1031   </ecp:RelayState>
1032   </SOAP-ENV:Header>
1033   <SOAP-ENV:Body>
1034     <samlp:Response> ... </samlp:Response>
1035   </SOAP-ENV:Body>
1036 </SOAP-ENV:Envelope>

```

1037 4.2.5 Security Considerations

1038 The <AuthnRequest> message SHOULD be signed. Per the rules specified by the browser SSO profile,
1039 the assertions enclosed in the <Response> MUST be signed. The delivery of the response in the SOAP
1040 envelope via PAOS is essentially analogous to the use of the HTTP POST binding and security
1041 countermeasures appropriate to that binding are used.

1042 The SOAP headers SHOULD be integrity protected, such as with SOAP Message Security or through the
1043 use of SSL/TLS over every HTTP exchange with the client.

1044 The service provider SHOULD be authenticated to the ECP, for example with server-side TLS
1045 authentication.

1046 The ECP SHOULD be authenticated to the identity provider, such as by maintaining an authenticated
1047 session.

1048 **4.3 Identity Provider Discovery Profile**

1049 This section defines a profile by which a service provider can discover which identity providers a principal
1050 is using with the Web Browser SSO profile. In deployments having more than one identity provider,
1051 service providers need a means to discover which identity provider(s) a principal uses. The discovery
1052 profile relies on a cookie that is written in a domain that is common between identity providers and service
1053 providers in a deployment. The domain that the deployment predetermines is known as the common
1054 domain in this profile, and the cookie containing the list of identity providers is known as the common
1055 domain cookie.

1056 Which entities host web servers in the common domain is a deployment issue and is outside the scope of
1057 this profile.

1058 **4.3.1 Common Domain Cookie**

1059 The name of the cookie MUST be `_saml_idp`. The format of the cookie value MUST be a set of one or
1060 more base-64 encoded URI values separated by a single space character. Each URI is the unique
1061 identifier of an identity provider, as defined in Section 8.3.6 of [SAMLCore]. The final set of values is then
1062 URL encoded.

1063 The common domain cookie writing service (see below) SHOULD append the identity provider's unique
1064 identifier to the list. If the identifier is already present in the list, it MAY remove and append it when
1065 authentication of the principal occurs. The intent is that the most recently established identity provider
1066 session is the last one in the list.

1067 The cookie MUST be set with no Path prefix or a Path prefix of `"/`. The Domain MUST be set to
1068 `"[common-domain]"` where `[common-domain]` is the common domain established within the deployment
1069 for use with this profile. The cookie MUST be marked as secure.

1070 Cookie syntax should be in accordance with IETF RFC 2965 [RFC2965] or [NSCookie]. The cookie MAY
1071 be either session-only or persistent. This choice may be made within a deployment, but should apply
1072 uniformly to all identity providers in the deployment.

1073 **4.3.2 Setting the Common Domain Cookie**

1074 After the identity provider authenticates a principal, it MAY set the common domain cookie. The means by
1075 which the identity provider sets the cookie are implementation-specific so long as the cookie is
1076 successfully set with the parameters given above. One possible implementation strategy follows and
1077 should be considered non-normative. The identity provider may:

- 1078 • Have previously established a DNS and IP alias for itself in the common domain.
- 1079 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
1080 scheme. The structure of the URL is private to the implementation and may include session
1081 information needed to identify the user-agent.
- 1082 • Set the cookie on the redirected user agent using the parameters specified above.
- 1083 • Redirect the user agent back to itself, or, if appropriate, to the service provider.

1084 **4.3.3 Obtaining the Common Domain Cookie**

1085 When a service provider needs to discover which identity providers a principal uses, it invokes an
1086 exchange designed to present the common domain cookie to the service provider after it is read by an
1087 HTTP server in the common domain.

1088 If the HTTP server in the common domain is operated by the service provider or if other arrangements are
1089 in place, the service provider MAY utilize the HTTP server in the common domain to relay its
1090 `<AuthnRequest>` to the identity provider for an optimized single sign-on process.

- 1091 The specific means by which the service provider reads the cookie are implementation-specific so long as
1092 it is able to cause the user agent to present cookies that have been set with the parameters given in
1093 Section 4.3.1. One possible implementation strategy is described as follows and should be considered
1094 non-normative. Additionally, it may be sub-optimal for some applications.
1095
- Have previously established a DNS and IP alias for itself in the common domain.
 - Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
1096 scheme. The structure of the URL is private to the implementation and may include session
1097 information needed to identify the user-agent.
1098
 - Set the cookie on the redirected user agent using the parameters specified above.
 - Redirect the user agent back to itself, or, if appropriate, to the identity provider.

1101 4.4 Single Logout Profile

1102 Once a principal has authenticated to an identity provider, the authenticating entity may establish a
1103 session with the principal (typically by means of a cookie, URL re-writing, or some other implementation-
1104 specific means). The identity provider may subsequently issue assertions to service providers or other
1105 relying parties, based on this authentication event; a relying party may use this to establish *its own* session
1106 with the user.

1107 In such a situation, the identity provider can act as a session authority and the relying parties as session
1108 participants. At some later time, the principal may wish to terminate his or her session either with an
1109 individual session participant, or with all session participants in a given session managed by the session
1110 authority. The former case is considered out of scope of this specification. The latter case, however, may
1111 be satisfied using this profile of the SAML Single Logout protocol ([SAMLCore] Section 3.7).

1112 Note that a principal (or an administrator terminating a principal's session) may choose to terminate this
1113 "global" session either by contacting the session authority, or an individual session participant. Also note
1114 that an identity provider acting as a session authority may *itself* act as a session participant in situations in
1115 which it is the relying party for another identity provider's assertions regarding that principal.

1116 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1117 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1118 front-channel binding may be required, for example, in cases in which a principal's session state exists
1119 solely in a user agent in the form of a cookie and a direct interaction between the user agent and the
1120 session participant or session authority is required.

1121 4.4.1 Required Information

1122 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:logout

1123 **Contact information:** security-services-comment@lists.oasis-open.org

1124 **Description:** Given below.

1125 **Updates:** None

1126 4.4.2 Profile Overview

1127 Figure 3 illustrates the basic template for achieving single logout:

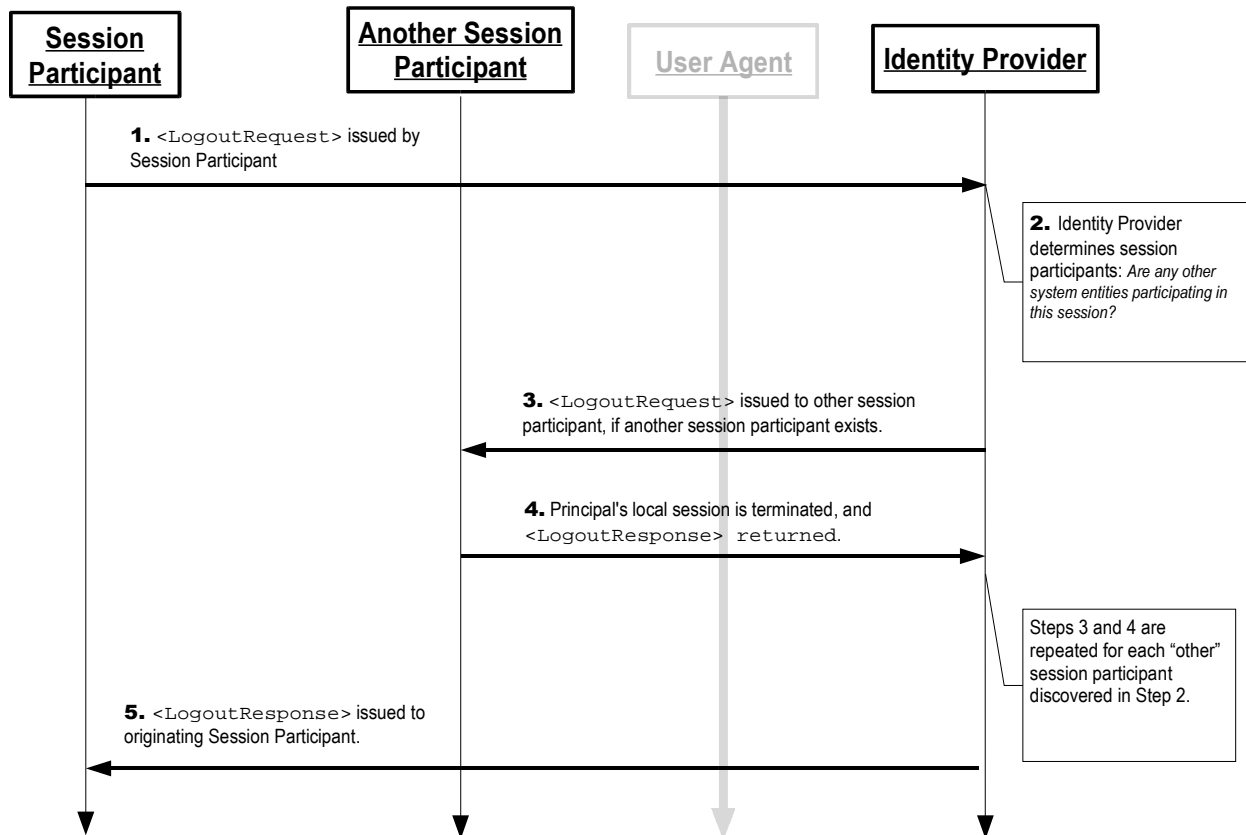


Figure 3

1128 The grayed-out user agent illustrates that the message exchange may pass through the user agent or
 1129 may be a direct exchange between system entities, depending on the SAML binding used to implement
 1130 the profile.

1131 The following steps are described by the profile. Within an individual step, there may be one or more
 1132 actual message exchanges depending on the binding used for that step and other implementation-
 1133 dependent behavior.

1134 **1. <LogoutRequest> issued by Session Participant to Identity Provider**

1135 In step 1, the session participant initiates single logout and terminates a principal's session(s) by
 1136 sending a <LogoutRequest> message to the identity provider from whom it received the
 1137 corresponding authentication assertion. The request may be sent directly to the identity provider
 1138 or sent indirectly through the user agent.

1139 **2. Identity Provider determines Session Participants**

1140 In step 2, the identity provider uses the contents of the <LogoutRequest> message (or if
 1141 initiating logout itself, some other mechanism) to determine the session(s) being terminated. If
 1142 there are no other session participants, the profile proceeds with step 5. Otherwise, steps 3 and 4
 1143 are repeated for each session participant identified.

1144 **3. <LogoutRequest> issued by Identity Provider to Session Participant/Authority**

1145 In step 3, the identity provider issues a <LogoutRequest> message to a session participant or
 1146 session authority related to one or more of the session(s) being terminated. The request may be
 1147 sent directly to the entity or sent indirectly through the user agent (if consistent with the form of the
 1148 request in step 1).

1149 **4. Session Participant/Authority issues <LogoutResponse> to Identity Provider**

1150 In step 4, a session participant or session authority terminates the principal's session(s) as
 1151 directed by the request (if possible) and returns a <LogoutResponse> to the identity provider.

1152 The response may be returned directly to the identity provider or indirectly through the user agent
1153 (if consistent with the form of the request in step 3).

1154 **5. Identity Provider issues <LogoutResponse> to Session Participant**

1155 In step 5, the identity provider issues a <LogoutResponse> message to the original requesting
1156 session participant. The response may be returned directly to the session participant or indirectly
1157 through the user agent (if consistent with the form of the request in step 1).

1158 Note that an identity provider (acting as session authority) can initiate this profile at step 2 and issue a
1159 <LogoutRequest> to all session participants, also skipping step 5.

1160 **4.4.3 Profile Description**

1161 If the profile is initiated by a session participant, start with Section 4.4.3.1. If initiated by the identity
1162 provider, start with Section 4.4.3.2. In the descriptions below, the following is referred to:

1163 **Single Logout Service**

1164 This is the single logout protocol endpoint at an identity provider or session participant to which the
1165 <LogoutRequest> or <LogoutResponse> messages (or an artifact representing them) are
1166 delivered. The same or different endpoints MAY be used for requests and responses.

1167 **4.4.3.1 <LogoutRequest> Issued by Session Participant to Identity Provider**

1168 If the logout profile is initiated by a session participant, it examines the authentication assertion(s) it
1169 received pertaining to the session(s) being terminated, and collects the `SessionIndex` value(s) it
1170 received from the identity provider. If multiple identity providers are involved, then the profile MUST be
1171 repeated independently for each one.

1172 To initiate the profile, the session participant issues a <LogoutRequest> message to the identity
1173 provider's single logout service request endpoint containing one or more applicable <SessionIndex>
1174 elements. At least one element MUST be included. Metadata (as in [SAMLMeta]) MAY be used to
1175 determine the location of this endpoint and the bindings supported by the identity provider.

1176 **Synchronous Bindings (Back-Channel)**

1177 The session participant MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to
1178 send the request directly to the identity provider. The identity provider would then propagate any
1179 required logout messages to additional session participants as required using a synchronous binding.
1180 The requester MUST authenticate itself to the identity provider, either by signing the
1181 <LogoutRequest> or using any other binding-supported mechanism.

1182 **Asynchronous Bindings (Front-Channel)**

1183 Alternatively, the session participant MAY (if the principal's user agent is present) use an
1184 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to send the
1185 request to the identity provider through the user agent.

1186 If the HTTP Redirect or POST binding is used, then the <LogoutRequest> message is delivered to
1187 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile
1188 defined in Section 5 is used by the identity provider, which makes a callback to the session participant
1189 to retrieve the <LogoutRequest> message, using for example the SOAP binding.

1190 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1191 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <LogoutRequest>
1192 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,
1193 if used, also provides for an alternate means of authenticating the request issuer when the artifact is
1194 dereferenced.

1195 Each of these bindings provide a RelayState mechanism that the session participant MAY use to
1196 associate the profile exchange with the original request. The session participant SHOULD reveal as
1197 little information as possible in the RelayState value unless the use of the profile does not require such
1198 privacy measures.

1199 Profile-specific rules for the contents of the <LogoutRequest> message are included in Section 4.4.4.1.

1200 **4.4.3.2 Identity Provider Determines Session Participants**

1201 If the logout profile is initiated by an identity provider, or upon receiving a valid <LogoutRequest>
1202 message, the identity provider processes the request as defined in [SAMLCore]. It MUST examine the
1203 principal identifier and <SessionIndex> elements and determine the set of sessions to be terminated.

1204 The identity provider then follows steps 3 and 4 for each entity participating in the session(s) being
1205 terminated, other than the original requesting session participant (if any), as described in Section 3.7.3.2
1206 of [SAMLCore].

1207 **4.4.3.3 <LogoutRequest> Issued by Identity Provider to Session 1208 Participant/Authority**

1209 To propagate the logout, the identity provider issues its own <LogoutRequest> to a session authority or
1210 participant in a session being terminated. The request is sent in the same fashion as described in step 1
1211 using a SAML binding consistent with the capability of the responder and the availability of the user agent
1212 at the identity provider.

1213 Profile-specific rules for the contents of the <LogoutRequest> message are included in Section 4.4.4.1.

1214 **4.4.3.4 Session Participant/Authority Issues <LogoutResponse> to Identity 1215 Provider**

1216 The session participant/authority MUST process the <LogoutRequest> message as defined in
1217 [SAMLCore]. After processing the message or upon encountering an error, the entity MUST issue a
1218 <LogoutResponse> message containing an appropriate status code to the requesting identity provider
1219 to complete the SAML protocol exchange.

1220 **Synchronous Bindings (Back-Channel)**

1221 If the identity provider used a synchronous binding, such as the SOAP binding [SAMLBind], the
1222 response is returned directly to complete the synchronous communication. The responder MUST
1223 authenticate itself to the requesting identity provider, either by signing the <LogoutResponse> or
1224 using any other binding-supported mechanism.

1225 **Asynchronous Bindings (Front-Channel)**

1226 If the identity provider used an asynchronous binding, such as the HTTP Redirect, POST, or Artifact
1227 bindings [SAMLBind], then the <LogoutResponse> (or artifact) is returned through the user agent to
1228 the identity provider's single logout service response endpoint. Metadata (as in [SAMLMeta]) MAY be
1229 used to determine the location of this endpoint and the bindings supported by the identity provider.

1230 If the HTTP Redirect or POST binding is used, then the <LogoutResponse> message is delivered to
1231 the identity provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution profile
1232 defined in Section 5 is used by the identity provider, which makes a callback to the responding entity
1233 to retrieve the <LogoutResponse> message, using for example the SOAP binding.

1234 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1235 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <LogoutResponse>
1236 message MUST be signed if the HTTP POST or Redirect binding is used. The HTTP Artifact binding,
1237 if used, also provides for an alternate means of authenticating the response issuer when the artifact is
1238 dereferenced.

1239 Profile-specific rules for the contents of the <LogoutResponse> message are included in Section
1240 4.4.4.2.

1241 4.4.3.5 Identity Provider Issues <LogoutResponse> to Session Participant

1242 After processing the original session participant's <LogoutRequest> in step 1, or upon encountering an
1243 error, the identity provider MUST respond to the original request with a <LogoutResponse> containing
1244 an appropriate status code to complete the SAML protocol exchange.

1245 The response is sent to the original session participant in the same fashion as described in step 4, using a
1246 SAML binding consistent with the binding used in the request, the capability of the responder, and the
1247 availability of the user agent at the identity provider.

1248 Profile-specific rules for the contents of the <LogoutResponse> message are included in Section
1249 4.4.4.2.

1250 4.4.4 Use of Single Logout Protocol

1251 4.4.4.1 <LogoutRequest> Usage

1252 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1253 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1254 format:entity.

1255 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1256 the message or using a binding-specific mechanism.

1257 The principal MUST be identified in the request using an identifier that **strongly matches** the identifier in
1258 the authentication assertion the requester issued or received regarding the session being terminated, per
1259 the matching rules defined in Section 3.3.4 of [SAMLCore].

1260 If the requester is a session participant, it MUST include at least one <SessionIndex> element in the
1261 request. If the requester is a session authority (or acting on its behalf), then it MAY omit any such
1262 elements to indicate the termination of all of the principal's applicable sessions.

1263 4.4.4.2 <LogoutResponse> Usage

1264 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1265 entity; the Format attribute MUST be omitted or have a value of
1266 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1267 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1268 the message or using a binding-specific mechanism.

1269 4.4.5 Use of Metadata

1270 [SAMLMeta] defines an endpoint element, <md:SingleLogoutService>, to describe supported
1271 bindings and location(s) to which an entity may send requests and responses using this profile.

1272 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>
1273 element with a use attribute of encryption to determine an appropriate encryption algorithm and
1274 settings to use, along with a public key to use in delivering a bulk encryption key.

1275 4.5 Name Identifier Management Profile

1276 In the scenario supported by the Name Identifier Management profile, an identity provider has exchanged
1277 some form of persistent identifier for a principal with a service provider, allowing them to share a common
1278 identifier for some length of time. Subsequently, the identity provider may wish to notify the service
1279 provider of a change in the format and/or value that it will use to identify the same principal in the future.
1280 Alternatively the service provider may wish to attach its own "alias" for the principal in order to insure that
1281 the identity provider will include it when communicating with it in the future about the principal. Finally, one
1282 of the providers may wish to inform the other that it will no longer issue or accept messages using a
1283 particular identifier. To implement these scenarios, a profile of the SAML Name Identifier Management

1284 protocol is used.

1285 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1286 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1287 front-channel binding may be required, for example, in cases in which direct interaction between the user
1288 agent and the responding provider is required in order to effect the change.

1289 4.5.1 Required Information

1290 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:nameid-mgmt

1291 **Contact information:** security-services-comment@lists.oasis-open.org

1292 **Description:** Given below.

1293 **Updates:** None.

1294 4.5.2 Profile Overview

1295 Figure 4 illustrates the basic template for the name identifier management profile.

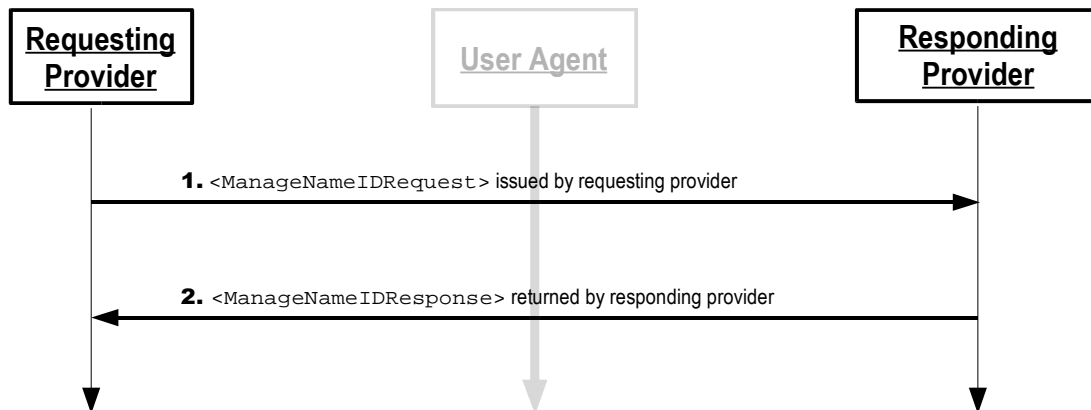


Figure 4

1296 The grayed-out user agent illustrates that the message exchange may pass through the user agent or
1297 may be a direct exchange between system entities, depending on the SAML binding used to implement
1298 the profile.

1299 The following steps are described by the profile. Within an individual step, there may be one or more
1300 actual message exchanges depending on the binding used for that step and other implementation-
1301 dependent behavior.

1302 1. <ManageNameIDRequest> issued by Requesting Identity/Service Provider

1303 In step 1, an identity or service provider initiates the profile by sending a
1304 <ManageNameIDRequest> message to another provider that it wishes to inform of a change.
1305 The request may be sent directly to the responding provider or sent indirectly through the user
1306 agent.

1307 2. <ManageNameIDResponse> issued by Responding Identity/Service Provider

1308 In step 2, the responding provider (after processing the request) issues a
1309 <ManageNameIDResponse> message to the original requesting provider. The response may be
1310 returned directly to the requesting provider or indirectly through the user agent (if consistent with
1311 the form of the request in step 1).

1312 4.5.3 Profile Description

1313 In the descriptions below, the following is referred to:

1314 Name Identifier Management Service

1315 This is the name identifier management protocol endpoint at an identity or service provider to which
1316 the <ManageNameIDRequest> or <ManageNameIDResponse> messages (or an artifact
1317 representing them) are delivered. The same or different endpoints MAY be used for requests and
1318 responses.

1319 4.5.3.1 <ManageNameIDRequest> Issued by Requesting Identity/Service Provider

1320 To initiate the profile, the requesting provider issues a <ManageNameIDRequest> message to another
1321 provider's name identifier management service request endpoint. Metadata (as in [SAMLMeta]) MAY be
1322 used to determine the location of this endpoint and the bindings supported by the responding provider.

1323 Synchronous Bindings (Back-Channel)

1324 The requesting provider MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to
1325 send the request directly to the other provider. The requester MUST authenticate itself to the other
1326 provider, either by signing the <ManageNameIDRequest> or using any other binding-supported
1327 mechanism.

1328 Asynchronous Bindings (Front-Channel)

1329 Alternatively, the requesting provider MAY (if the principal's user agent is present) use an
1330 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to send the
1331 request to the other provider through the user agent.

1332 If the HTTP Redirect or POST binding is used, then the <ManageNameIDRequest> message is
1333 delivered to the other provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution
1334 profile defined in Section 55 is used by the other provider, which makes a callback to the requesting
1335 provider to retrieve the <ManageNameIDRequest> message, using for example the SOAP binding.

1336 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or
1337 TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1338 <ManageNameIDRequest> message MUST be signed if the HTTP POST or Redirect binding is
1339 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1340 request issuer when the artifact is dereferenced.

1341 Each of these bindings provide a RelayState mechanism that the requesting provider MAY use to
1342 associate the profile exchange with the original request. The requesting provider SHOULD reveal as
1343 little information as possible in the RelayState value unless the use of the profile does not require such
1344 privacy measures.

1345 Profile-specific rules for the contents of the <ManageNameIDRequest> message are included in Section
1346 4.5.4.1.

1347 4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service 1348 Provider

1349 The recipient MUST process the <ManageNameIDRequest> message as defined in [SAMLCore]. After
1350 processing the message or upon encountering an error, the recipient MUST issue a
1351 <ManageNameIDResponse> message containing an appropriate status code to the requesting provider
1352 to complete the SAML protocol exchange.

1353 Synchronous Bindings (Back-Channel)

1354 If the requesting provider used a synchronous binding, such as the SOAP binding [SAMLBind], the
1355 response is returned directly to complete the synchronous communication. The responder MUST
1356 authenticate itself to the requesting provider, either by signing the <ManageNameIDResponse> or

1357 using any other binding-supported mechanism.

1358 **Asynchronous Bindings (Front-Channel)**

1359 If the requesting provider used an asynchronous binding, such as the HTTP Redirect, POST, or
1360 Artifact bindings [SAMLBind], then the <ManageNameIDResponse> (or artifact) is returned through
1361 the user agent to the requesting provider's name identifier management service response endpoint.
1362 Metadata (as in [SAMLMeta]) MAY be used to determine the location of this endpoint and the bindings
1363 supported by the requesting provider.

1364 If the HTTP Redirect or POST binding is used, then the <ManageNameIDResponse> message is
1365 delivered to the requesting provider in this step. If the HTTP Artifact binding is used, the Artifact
1366 Resolution profile defined in Section 55 is used by the requesting provider, which makes a callback to
1367 the responding provider to retrieve the <ManageNameIDResponse> message, using for example the
1368 SOAP binding.

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

1374 Profile-specific rules for the contents of the <ManageNameIDResponse> message are included in
1375 Section 4.5.4.2.

1376 **4.5.4 Use of Name Identifier Management Protocol**

1377 **4.5.4.1 <ManageNameIDRequest> Usage**

1378 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1379 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1380 format:entity.

1381 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1382 the message or using a binding-specific mechanism.

1383 **4.5.4.2 <ManageNameIDResponse> Usage**

1384 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1385 entity; the Format attribute MUST be omitted or have a value of
1386 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1387 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1388 the message or using a binding-specific mechanism.

1389 **4.5.5 Use of Metadata**

1390 [SAMLMeta] defines an endpoint element, <md:ManageNameIDService>, to describe supported
1391 bindings and location(s) to which an entity may send requests and responses using this profile.

1392 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>
1393 element with a use attribute of encryption to determine an appropriate encryption algorithm and
1394 settings to use, along with a public key to use in delivering a bulk encryption key.

1395

5 Artifact Resolution Profile

1396 [SAMLCore] defines an Artifact Resolution protocol for dereferencing a SAML artifact into a corresponding
1397 protocol message. The HTTP Artifact binding in [SAMLBind] leverages this mechanism to pass SAML
1398 protocol messages by reference. This profile describes the use of this protocol with a synchronous
1399 binding, such as the SOAP binding defined in [SAMLBind].

5.1 Required Information

1401 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:artifact

1402 **Contact information:** security-services-comment@lists.oasis-open.org

1403 **Description:** Given below.

1404 **Updates:** None

5.2 Profile Overview

1406 The message exchange and basic processing rules that govern this profile are largely defined by Section
1407 3.5 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1408 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1409 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1410 Figure 5 illustrates the basic template for the artifact resolution profile.

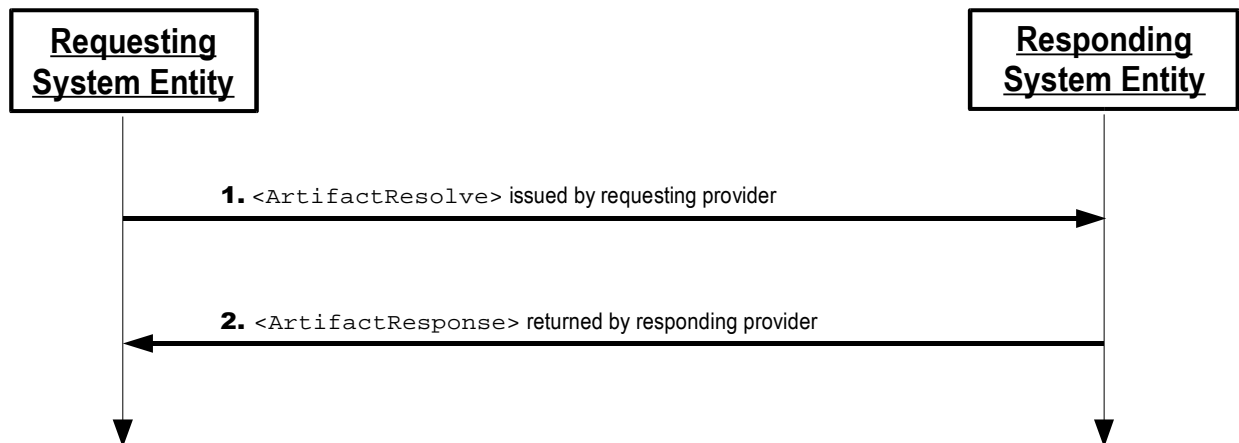


Figure 5

1411 The following steps are described by the profile.

1412 1. <ArtifactResolve> issued by Requesting Entity

1413 In step 1, a requester initiates the profile by sending an <ArtifactResolve> message to an
1414 artifact issuer.

1415 2. <ArtifactResponse> issued by Responding Entity

1416 In step 2, the responder (after processing the request) issues an <ArtifactResponse>
1417 message to the requester.

1418 **5.3 Profile Description**

1419 In the descriptions below, the following is referred to:

1420 **Artifact Resolution Service**

1421 This is the artifact resolution protocol endpoint at an artifact issuer to which `<ArtifactResolve>`
1422 messages are delivered.

1423 **5.3.1 `<ArtifactResolve>` issued by Requesting Entity**

1424 To initiate the profile, a requester, having received an artifact and determined the issuer using the
1425 `SourceID`, sends an `<ArtifactResolve>` message containing the artifact to an artifact issuer's artifact
1426 resolution service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of this
1427 endpoint and the bindings supported by the artifact issuer

1428 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1429 request directly to the artifact issuer. The requester SHOULD authenticate itself to the responder, either by
1430 signing the `<ArtifactResolve>` message or using any other binding-supported mechanism. Specific
1431 profiles that use the HTTP Artifact binding MAY impose additional requirements such that authentication is
1432 mandatory.

1433 Profile-specific rules for the contents of the `<ArtifactResolve>` message are included in Section 5.4.1.

1434 **5.3.2 `<ArtifactResponse>` issued by Responding Entity**

1435 The artifact issuer MUST process the `<ArtifactResolve>` message as defined in [SAMLCore]. After
1436 processing the message or upon encountering an error, the artifact issuer MUST return an
1437 `<ArtifactResponse>` message containing an appropriate status code to the requester to complete the
1438 SAML protocol exchange. If successful, the dereferenced SAML protocol message corresponding to the
1439 artifact will also be included.

1440 The responder MUST authenticate itself to the requester, either by signing the `<ArtifactResponse>` or
1441 using any other binding-supported mechanism.

1442 Profile-specific rules for the contents of the `<ArtifactResponse>` message are included in Section
1443 5.4.2.

1444 **5.4 Use of Artifact Resolution Protocol**

1445 **5.4.1 `<ArtifactResolve>` Usage**

1446 The `<Issuer>` element MUST be present and MUST contain the unique identifier of the requesting entity;
1447 the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`
1448 `format:entity`.

1449 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by
1450 signing the message or using a binding-specific mechanism. Specific profiles that use the HTTP Artifact
1451 binding MAY impose additional requirements such that authentication is mandatory.

1452 **5.4.2 `<ArtifactResponse>` Usage**

1453 The `<Issuer>` element MUST be present and MUST contain the unique identifier of the artifact issuer;
1454 the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`
1455 `format:entity`.

1456 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1457 the message or using a binding-specific mechanism.

1458 **5.5 Use of Metadata**

1459 [SAMLMeta] defines an indexed endpoint element, `<md:ArtifactResolutionService>`, to describe
1460 supported bindings and location(s) to which a requester may send requests using this profile. The `index`
1461 attribute is used to distinguish the possible endpoints that may be specified by reference in the artifact's
1462 `EndpointIndex` field.

1463

6 Assertion Query/Request Profile

1464 [SAMLCore] defines a protocol for requesting existing assertions by reference or by querying on the basis
1465 of a subject and additional statement-specific criteria. This profile describes the use of this protocol with a
1466 synchronous binding, such as the SOAP binding defined in [SAMLBind].

6.1 Required Information

1468 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:query

1469 **Contact information:** security-services-comment@lists.oasis-open.org

1470 **Description:** Given below.

1471 **Updates:** None.

6.2 Profile Overview

1473 The message exchange and basic processing rules that govern this profile are largely defined by Section
1474 3.3 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1475 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1476 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1477 Figure 6 illustrates the basic template for the query/request profile.

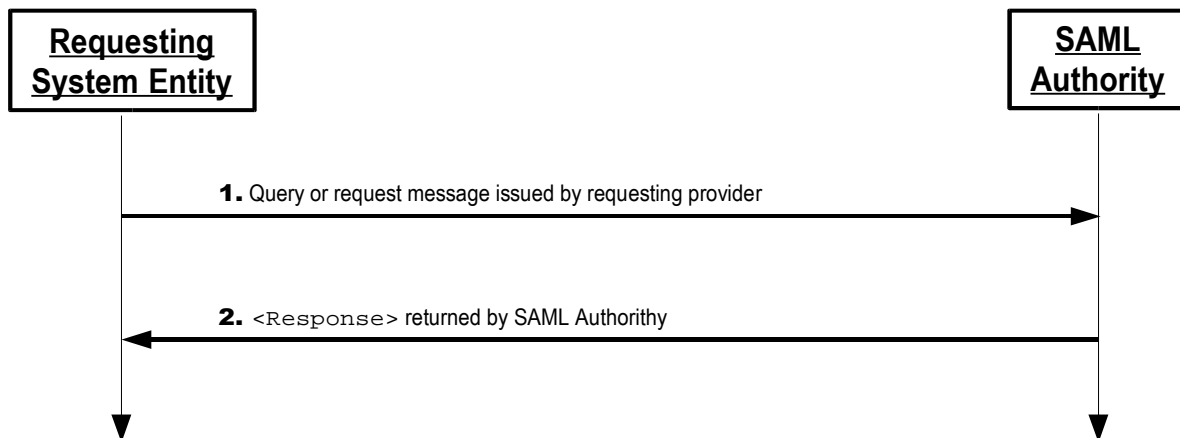


Figure 6

1478 The following steps are described by the profile.

1479 1. Query/Request issued by Requesting Entity

1480 In step 1, a requester initiates the profile by sending an `<AssertionIDRequest>`,
1481 `<SubjectQuery>`, `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>`
1482 message to a SAML authority.

1483 2. <Response> issued by SAML Authority

1484 In step 2, the responding SAML authority (after processing the query or request) issues a
1485 `<Response>` message to the requester.

1486 **6.3 Profile Description**

1487 In the descriptions below, the following are referred to:

1488 **Query/Request Service**

1489 This is the query/request protocol endpoint at a SAML authority to which query or
1490 `<AssertionIDRequest>` messages are delivered.

1491 **6.3.1 Query/Request issued by Requesting Entity**

1492 To initiate the profile, a requester issues an `<AssertionIDRequest>`, `<SubjectQuery>`,
1493 `<AuthnQuery>`, `<AttributeQuery>`, or `<AuthzDecisionQuery>` message to a SAML authority's
1494 query/request service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of
1495 this endpoint and the bindings supported by the SAML authority.

1496 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1497 request directly to the identity provider. The requester SHOULD authenticate itself to the SAML authority
1498 either by signing the message or using any other binding-supported mechanism.

1499 Profile-specific rules for the contents of the various messages are included in Section 6.4.1.

1500 **6.3.2 `<Response>` issued by SAML Authority**

1501 The SAML authority MUST process the query or request message as defined in [SAMLCore]. After
1502 processing the message or upon encountering an error, the SAML authority MUST return a `<Response>`
1503 message containing an appropriate status code to the requester to complete the SAML protocol
1504 exchange. If the request is successful in locating one or more matching assertions, they will also be
1505 included in the response.

1506 The responder SHOULD authenticate itself to the requester, either by signing the `<Response>` or using
1507 any other binding-supported mechanism.

1508 Profile-specific rules for the contents of the `<Response>` message are included in Section 6.4.2.

1509 **6.4 Use of Query/Request Protocol**

1510 **6.4.1 Query/Request Usage**

1511 The `<Issuer>` element MUST be present.

1512 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by
1513 signing the message or using a binding-specific mechanism.

1514 **6.4.2 `<Response>` Usage**

1515 The `<Issuer>` element MUST be present and MUST contain the unique identifier of the responding
1516 SAML authority; the `Format` attribute MUST be omitted or have a value of
1517 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`. Note that this need not necessarily
1518 match the `<Issuer>` element in the returned assertion(s).

1519 The responder SHOULD authenticate itself to the requester and ensure message integrity, either by
1520 signing the message or using a binding-specific mechanism.

1521 **6.5 Use of Metadata**

1522 [SAMLMeta] defines several endpoint elements, `<md:AssertionIDRequestService>`,
1523 `<md:AuthnQueryService>`, `<md:AttributeService>`, and `<md:AuthzService>`, to describe
1524 supported bindings and location(s) to which a requester may send requests or queries using this profile.

1525 The SAML authority, if encrypting the resulting assertions or assertion contents for a particular entity, can
1526 use that entity's `<md:KeyDescriptor>` element with a `use` attribute of `encryption` to determine an
1527 appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk
1528 encryption key.

1529

7 Name Identifier Mapping Profile

1530 [SAMLCore] defines a Name Identifier Mapping protocol for mapping a principal's name identifier into a
1531 different name identifier for the same principal. This profile describes the use of this protocol with a
1532 synchronous binding, such as the SOAP binding defined in [SAMLBind], and additional guidelines for
1533 protecting the privacy of the principal with encryption and limiting the use of the mapped identifier.

7.1 Required Information

1535 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:nameidmapping

1536 **Contact information:** security-services-comment@lists.oasis-open.org

1537 **Description:** Given below.

1538 **Updates:** None.

7.2 Profile Overview

1540 The message exchange and basic processing rules that govern this profile are largely defined by Section
1541 3.8 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1542 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1543 SOAP V1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1544 Figure 7 illustrates the basic template for the name identifier mapping profile.

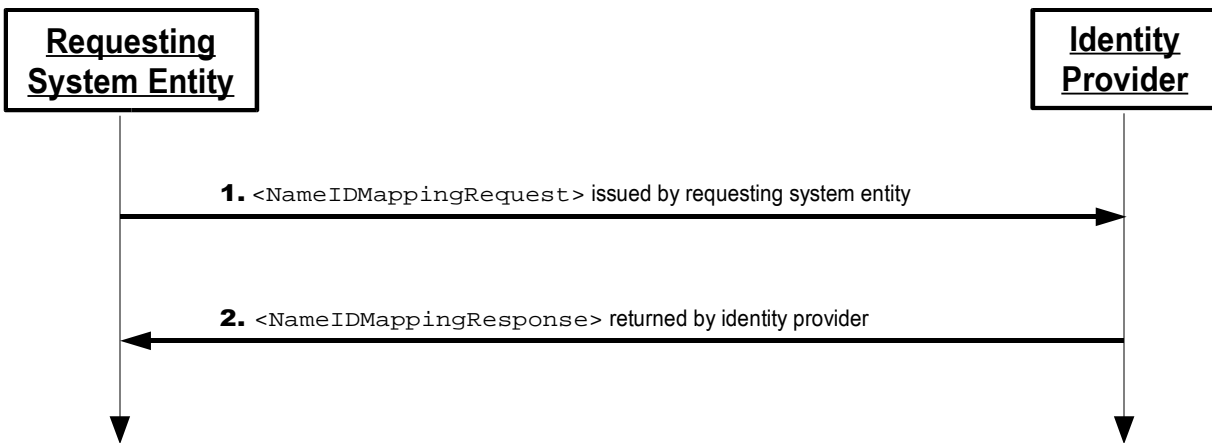


Figure 7

1545 The following steps are described by the profile.

1546 1. <NameIDMappingRequest> issued by Requesting Entity

1547 In step 1, a requester initiates the profile by sending a <NameIDMappingRequest> message to
1548 an identity provider.

1549 2. <NameIDMappingResponse> issued by Identity Provider

1550 In step 2, the responding identity provider (after processing the request) issues a
1551 <NameIDMappingResponse> message to the requester.

1552 **7.3 Profile Description**

1553 In the descriptions below, the following is referred to:

1554 **Name Identifier Mapping Service**

1555 This is the name identifier mapping protocol endpoint at an identity provider to which
1556 <NameIDMappingRequest> messages are delivered.

1557 **7.3.1 <NameIDMappingRequest> issued by Requesting Entity**

1558 To initiate the profile, a requester issues a <NameIDMappingRequest> message to an identity provider's
1559 name identifier mapping service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the
1560 location of this endpoint and the bindings supported by the identity provider.

1561 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1562 request directly to the identity provider. The requester MUST authenticate itself to the identity provider,
1563 either by signing the <NameIDMappingRequest> or using any other binding-supported mechanism.

1564 Profile-specific rules for the contents of the <NameIDMappingRequest> message are included in
1565 Section 7.4.1.

1566 **7.3.2 <NameIDMappingResponse> issued by Identity Provider**

1567 The identity provider MUST process the <ManageNameIDRequest> message as defined in [SAMLCore].
1568 After processing the message or upon encountering an error, the identity provider MUST return a
1569 <NameIDMappingResponse> message containing an appropriate status code to the requester to
1570 complete the SAML protocol exchange.

1571 The responder MUST authenticate itself to the requester, either by signing the
1572 <NameIDMappingResponse> or using any other binding-supported mechanism.

1573 Profile-specific rules for the contents of the <NameIDMappingResponse> message are included in
1574 Section 7.4.2.

1575 **7.4 Use of Name Identifier Mapping Protocol**

1576 **7.4.1 <NameIDMappingRequest> Usage**

1577 The <Issuer> element MUST be present.

1578 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1579 the message or using a binding-specific mechanism.

1580 **7.4.2 <NameIDMappingResponse> Usage**

1581 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1582 identity provider; the `Format` attribute MUST be omitted or have a value of
1583 `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

1584 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1585 the message or using a binding-specific mechanism.

1586 Section 2.2.3 of [SAMLCore] defines the use of encryption to apply confidentiality to a name identifier. In
1587 most cases, the identity provider SHOULD encrypt the mapped name identifier it returns to the requester
1588 to protect the privacy of the principal. The requester can extract the <EncryptedID> element and place it
1589 in subsequent protocol messages or assertions.

1590 **7.4.2.1 Limiting Use of Mapped Identifier**

1591 Additional limits on the use of the resulting identifier MAY be applied by the identity provider by returning
1592 the mapped name identifier in the form of an <Assertion> containing the identifier in its <Subject> but
1593 without any statements. The assertion is then encrypted and the result used as the <EncryptedData>
1594 element in the <EncryptedID> returned to the requester. The assertion MAY include a <Conditions>
1595 element to limit use, as defined by [SAMLCore], such as time-based constraints or use by specific relying
1596 parties, and MUST be signed for integrity protection.

1597 **7.5 Use of Metadata**

1598 [SAMLMeta] defines an endpoint element, <md:NameIDMappingService>, to describe supported
1599 bindings and location(s) to which a requester may send requests using this profile.

1600 The identity provider, if encrypting the resulting identifier for a particular entity, can use that entity's
1601 <md:KeyDescriptor> element with a use attribute of encryption to determine an appropriate
1602 encryption algorithm and settings to use, along with a public key to use in delivering a bulk encryption key.

1603 8 SAML Attribute Profiles

1604 8.1 Basic Attribute Profile

1605 The Basic attribute profile specifies simplified, but non-unique, naming of SAML attributes together with
1606 attribute values based on the built-in XML Schema data types, eliminating the need for extension schemas
1607 to validate syntax.

1608 8.1.1 Required Information

1609 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:basic

1610 **Contact information:** security-services-comment@lists.oasis-open.org

1611 **Description:** Given below.

1612 **Updates:** None.

1613 8.1.2 SAML Attribute Naming

1614 The `NameFormat` XML attribute in `<Attribute>` elements MUST be
1615 `urn:oasis:names:tc:SAML:2.0:attrname-format:basic`.

1616 The `Name` XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

1617 8.1.2.1 Attribute Name Comparison

1618 Two `<Attribute>` elements refer to the same SAML attribute if and only if the values of their `Name` XML
1619 attributes are equal in the sense of Section 3.3.6 of [Schema2].

1620 8.1.3 Profile-Specific XML Attributes

1621 No additional XML attributes are defined for use with the `<Attribute>` element.

1622 8.1.4 SAML Attribute Values

1623 The schema type of the contents of the `<AttributeValue>` element MUST be drawn from one of the
1624 types defined in Section 3.3 of [Schema2]. The `xsi:type` attribute MUST be present and be given the
1625 appropriate value.

1626 8.1.5 Example

```
1627 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"  
1628     Name="FirstName">  
1629     <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>  
1630 </saml:Attribute>
```

1631 8.2 X.500/LDAP Attribute Profile

1632 Directories based on the ITU-T X.500 specifications [X.500] and the related IETF Lightweight Directory
1633 Access Protocol specifications [LDAP] are widely deployed. Directory schema is used to model
1634 information to be stored in these directories. In particular, in X.500, attribute type definitions are used to
1635 specify the syntax and other features of attributes, the basic information storage unit in a directory (this
1636 document refers to these as “directory attributes”). Directory attribute types are defined in schema in the
1637 X.500 and LDAP specifications themselves, schema in other public documents (such as the
1638 Internet2/Educause EduPerson schema [eduPerson], or the inetOrgperson schema [RFC2798]), and

1639 schema defined for private purposes. In any of these cases, it is useful for deployers to take advantage of
1640 these directory attribute types in the context of SAML attribute statements, without having to manually
1641 create SAML-specific attribute definitions for them, and to do this in an interoperable fashion.
1642 The X.500/LDAP attribute profile defines a common convention for the naming and representation of such
1643 attributes when expressed as SAML attributes.

1644 8.2.1 Required Information

1645 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500 (this is also the target namespace
1646 assigned in the corresponding X.500/LDAP profile schema document [SAMLX500-xsd])

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

1648 **Description:** Given below.

1649 **Updates:** None.

1650 8.2.2 SAML Attribute Naming

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

1653 To construct attribute names, the URN `oid` namespace described in IETF RFC 3061 [RFC3061] is used.
1654 In this approach the `Name` XML attribute is based on the OBJECT IDENTIFIER assigned to the directory
1655 attribute type.

1656 Example:

```
1657 urn:oid:2.5.4.3
```

1658 Since X.500 procedures require that every attribute type be identified with a unique OBJECT IDENTIFIER,
1659 this naming scheme ensures that the derived SAML attribute names are unambiguous.

1660 For purposes of human readability, there may also be a requirement for some applications to carry an
1661 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
1662 [SAMLCore]) MAY be used for this purpose. If the definition of the directory attribute type includes one or
1663 more descriptors (short names) for the attribute type, the `FriendlyName` value, if present, SHOULD be
1664 one of the defined descriptors.

1665 8.2.2.1 Attribute Name Comparison

1666 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
1667 values are equal in the sense of [RFC3061]. The `FriendlyName` attribute plays no role in the
1668 comparison.

1669 8.2.3 Profile-Specific XML Attributes

1670 No additional XML attributes are defined for use with the `<Attribute>` element.

1671 8.2.4 SAML Attribute Values

1672 Directory attribute type definitions for use in native X.500 directories specify the syntax of the attribute
1673 using ASN.1 [ASN.1]. For use in LDAP, directory attribute definitions additionally include an LDAP syntax
1674 which specifies how attribute or assertion values conforming to the syntax are to be represented when
1675 transferred in the LDAP protocol (known as an LDAP-specific encoding). The LDAP-specific encoding
1676 commonly produces Unicode characters in UTF-8 form. This SAML attribute profile specifies the form of
1677 SAML attribute values only for those directory attributes which have LDAP syntaxes. Future extensions to
1678 this profile may define attribute value formats for directory attributes whose syntaxes specify other
1679 encodings.

1680 To represent the encoding rules in use for a particular attribute value, the `<AttributeValue>` element
1681 MUST contain an XML attribute named `Encoding` defined in the XML namespace

1682 urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500.

1683 For any directory attribute with a syntax whose LDAP-specific encoding exclusively produces UTF-8
1684 character strings as values, the SAML attribute value is encoded as simply the UTF-8 string itself, as the
1685 content of the <AttributeValue> element, with no additional whitespace. In such cases, the
1686 xsi:type XML attribute MUST be set to **xs:string**. The profile-specific Encoding XML attribute is
1687 provided, with a value of LDAP.

1688 A list of some LDAP attribute syntaxes to which this applies is:

| | | |
|------|-------------------------------|-------------------------------|
| 1689 | Attribute Type Description | 1.3.6.1.4.1.1466.115.121.1.3 |
| 1690 | Bit String | 1.3.6.1.4.1.1466.115.121.1.6 |
| 1691 | Boolean | 1.3.6.1.4.1.1466.115.121.1.7 |
| 1692 | Country String | 1.3.6.1.4.1.1466.115.121.1.11 |
| 1693 | DN | 1.3.6.1.4.1.1466.115.121.1.12 |
| 1694 | Directory String | 1.3.6.1.4.1.1466.115.121.1.15 |
| 1695 | Facsimile Telephone Number | 1.3.6.1.4.1.1466.115.121.1.22 |
| 1696 | Generalized Time | 1.3.6.1.4.1.1466.115.121.1.24 |
| 1697 | IA5 String | 1.3.6.1.4.1.1466.115.121.1.26 |
| 1698 | INTEGER | 1.3.6.1.4.1.1466.115.121.1.27 |
| 1699 | LDAP Syntax Description | 1.3.6.1.4.1.1466.115.121.1.54 |
| 1700 | Matching Rule Description | 1.3.6.1.4.1.1466.115.121.1.30 |
| 1701 | Matching Rule Use Description | 1.3.6.1.4.1.1466.115.121.1.31 |
| 1702 | Name And Optional UID | 1.3.6.1.4.1.1466.115.121.1.34 |
| 1703 | Name Form Description | 1.3.6.1.4.1.1466.115.121.1.35 |
| 1704 | Numeric String | 1.3.6.1.4.1.1466.115.121.1.36 |
| 1705 | Object Class Description | 1.3.6.1.4.1.1466.115.121.1.37 |
| 1706 | Octet String | 1.3.6.1.4.1.1466.115.121.1.40 |
| 1707 | OID | 1.3.6.1.4.1.1466.115.121.1.38 |
| 1708 | Other Mailbox | 1.3.6.1.4.1.1466.115.121.1.39 |
| 1709 | Postal Address | 1.3.6.1.4.1.1466.115.121.1.41 |
| 1710 | Presentation Address | 1.3.6.1.4.1.1466.115.121.1.43 |
| 1711 | Printable String | 1.3.6.1.4.1.1466.115.121.1.44 |
| 1712 | Substring Assertion | 1.3.6.1.4.1.1466.115.121.1.58 |
| 1713 | Telephone Number | 1.3.6.1.4.1.1466.115.121.1.50 |
| 1714 | UTC Time | 1.3.6.1.4.1.1466.115.121.1.53 |

1715 For all other LDAP syntaxes, the attribute value is encoded, as the content of the <AttributeValue>
1716 element, by base64-encoding [RFC2045] the encompassing ASN.1 OCTET STRING-encoded LDAP
1717 attribute value. The xsi:type XML attribute MUST be set to **xs:base64Binary**. The profile-specific
1718 Encoding XML attribute is provided, with a value of "LDAP".

1719 When comparing SAML attribute values for equality, the matching rules specified for the corresponding
1720 directory attribute type MUST be observed (case sensitivity, for example).

1721 8.2.5 Profile-Specific Schema

1722 The following schema defines the profile-specific Encoding XML attribute:

```
1723 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"  
1724         xmlns="http://www.w3.org/2001/XMLSchema"  
1725         version="2.0">  
1726     <attribute name="Encoding" type="string"/>  
1727 </schema>
```

1728 8.2.6 Example

1729 The following is an example of a mapping of the "givenName" directory attribute, representing the SAML
1730 assertion subject's first name. It's OBJECT IDENTIFIER is 2.5.4.42 and its LDAP syntax is Directory
1731 String.

```
1732 <saml:Attribute xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
1733     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
1734     Name="urn:oid:2.5.4.42" FriendlyName="givenName">
1735     <saml:AttributeValue xsi:type="xs:string"
1736         x500:Encoding="LDAP">Steven</saml:AttributeValue>
1737 </saml:Attribute>
```

1738 8.3 UUID Attribute Profile

1739 The UUID attribute profile standardizes the expression of UUID values as SAML attribute names and
1740 values. It is applicable when the attribute's source system is one that identifies an attribute or its value with
1741 a UUID.

1742 8.3.1 Required Information

1743 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:UUID

1744 **Contact information:** security-services-comment@lists.oasis-open.org

1745 **Description:** Given below.

1746 **Updates:** None.

1747 8.3.2 UUID and GUID Background

1748 UUIDs (Universally Unique Identifiers), also known as GUIDs (Globally Unique Identifiers), are used to
1749 define objects and subjects such that they are guaranteed uniqueness across space and time. UUIDs
1750 were originally used in the Network Computing System (NCS), and then used in the Open Software
1751 Foundation's (OSF) Distributed Computing Environment (DCE). Recently GUIDs have been used in
1752 Microsoft's COM and Active Directory/Windows 2000/2003 platform.

1753 A UUID is a 128 bit number, generated such that it should never be duplicated within the domain of
1754 interest. UUIDs are used to represent a wide range of objects including, but not limited to, subjects/users,
1755 groups of users and node names. A UUID, represented as a hexadecimal string, is as follows:

```
1756 f81d4fae-7dec-11d0-a765-00a0c91e6bf6
```

1757 In DCE and Microsoft Windows, the UUID is usually presented to the administrator in the form of a
1758 "friendly name". For instance the above UUID could represent the user john.doe@example.com.

1759 8.3.3 SAML Attribute Naming

1760 The NameFormat XML attribute in <Attribute> elements MUST be
1761 urn:oasis:names:tc:SAML:2.0:attrname-format:uri.

1762 If the underlying representation of the attribute's name is a UUID, then the URN uuid namespace
1763 described in [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt] is used. In this approach the
1764 Name XML attribute is based on the URN form of the underlying UUID that identifies the attribute.

1765 **Example:**

```
1766 urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
```

1767 If the underlying representation of the attribute's name is not a UUID, then any form of URI MAY be used
1768 in the Name XML attribute.

1769 For purposes of human readability, there may also be a requirement for some applications to carry an
1770 optional string name together with the URI. The optional XML attribute FriendlyName (defined in
1771 [SAMLCore]) MAY be used for this purpose.

1772 8.3.3.1 Attribute Name Comparison

1773 Two <Attribute> elements refer to the same SAML attribute if and only if their Name XML attribute

1774 values are equal in the sense of [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt]. The
1775 FriendlyName attribute plays no role in the comparison.

1776 8.3.4 Profile-Specific XML Attributes

1777 No additional XML attributes are defined for use with the <Attribute> element.

1778 8.3.5 SAML Attribute Values

1779 In cases in which the attribute's value is also a UUID, the same URN syntax described above MUST be
1780 used to express the value within the <AttributeValue> element. The xsi:type XML attribute MUST
1781 be set to xs:anyURI.

1782 If the attribute's value is not a UUID, then there are no restrictions on the use of the <AttributeValue>
1783 element.

1784 8.3.6 Example

1785 The following is an example of a DCE Extended Registry Attribute, the "pre_auth_req" setting, which has a
1786 well-known UUID of 6c9d0ec8-dd2d-11cc-abdd-080009353559 and is integer-valued.

```
1787 <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1788           Name="urn:uuid:6c9d0ec8-dd2d-11cc-abdd-080009353559"  
1789           FriendlyName="pre_auth_req">  
1790   <saml:AttributeValue xsi:type="xs:integer">1</saml:AttributeValue>  
1791 </saml:Attribute>
```

1792 8.4 DCE PAC Attribute Profile

1793 The DCE PAC attribute profile defines the expression of DCE PAC information as SAML attribute names
1794 and values. It is used to standardize a mapping between the primary information that makes up a DCE
1795 principal's identity and a set of SAML attributes. This profile builds on the UUID attribute profile defined in
1796 Section 8.3.

1797 8.4.1 Required Information

1798 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE (this is also the target namespace
1799 assigned in the corresponding DCE PAC attribute profile schema document [SAML DCE-xsd])

1800 **Contact information:** security-services-comment@lists.oasis-open.org

1801 **Description:** Given below.

1802 **Updates:** None.

1803 8.4.2 PAC Description

1804 A DCE PAC is an extensible structure that can carry arbitrary DCE registry attributes, but a core set of
1805 information is common across principals and makes up the bulk of a DCE identity:

- 1806 • The principal's DCE "realm" or "cell"
- 1807 • The principal's unique identifier
- 1808 • The principal's primary DCE local group membership
- 1809 • The principal's set of DCE local group memberships (multi-valued)
- 1810 • The principal's set of DCE foreign group memberships (multi-valued)

1811 The primary value(s) of each of these attributes is a UUID.

1812 8.4.3 SAML Attribute Naming

1813 This profile defines a mapping of specific DCE information into SAML attributes, and thus defines actual
1814 specific attribute names, rather than a naming convention.

1815 For all attributes defined by this profile, the `NameFormat` XML attribute in `<Attribute>` elements MUST
1816 have the value `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1817 For purposes of human readability, there may also be a requirement for some applications to carry an
1818 optional string name together with the URI. The optional XML attribute `FriendlyName` (defined in
1819 [SAMLCore]) MAY be used for this purpose.

1820 See Section 8.4.6 for the specific attribute names defined by this profile.

1821 8.4.3.1 Attribute Name Comparison

1822 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
1823 values are equal in the sense of [<http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt>]. The
1824 `FriendlyName` attribute plays no role in the comparison.

1825 8.4.4 Profile-Specific XML Attributes

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

1827 8.4.5 SAML Attribute Values

1828 The primary value(s) of each of the attributes defined by this profile is a UUID. The URN syntax described
1829 in Section 8.3.5 of the UUID profile is used to represent such values.

1830 However, additional information associated with the UUID value is permitted by this profile, consisting of a
1831 friendly, human-readable string, and an additional UUID representing a DCE cell or realm. The additional
1832 information is carried in the `<AttributeValue>` element in `FriendlyName` and `Realm` XML attributes
1833 defined in the XML namespace `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE`. Note
1834 that this is not the same as the `FriendlyName` XML attribute defined in [SAMLCore], although it has the
1835 same basic purpose.

1836 The following schema defines the profile-specific XML attributes and a complex type used in an
1837 `xsi:type` specification:

```
1838 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
1839         xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE"  
1840         xmlns="http://www.w3.org/2001/XMLSchema"  
1841         version="2.0">  
1842     <attribute name="Realm" type="anyURI"/>  
1843     <attribute name="FriendlyName" type="string"/>  
1844     <complexType name="DCEValueType">  
1845         <simpleContent>  
1846             <extension base="anyURI">  
1847                 <attribute ref="dce:Realm" use="optional"/>  
1848                 <attribute ref="dce:FriendlyName" use="optional"/>  
1849             </extension>  
1850         </simpleContent>  
1851     </complexType>  
1852 </schema>
```

1853 8.4.6 Attribute Definitions

1854 The following are the set of SAML attributes defined by this profile. In each case, an `xsi:type` XML
1855 attribute MAY be included in the `<AttributeValue>` element, but MUST have the value
1856 **dce:DCEValueType**, where the `dce` prefix is arbitrary and MUST be bound to the XML namespace
1857 `urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE`.

1858 Note that such use of `xsi:type` will require validating attribute consumers to include the extension
1859 schema defined by this profile.

1860 **8.4.6.1 Realm**

1861 This single-valued attribute represents the SAML assertion subject's DCE realm or cell.

1862 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm

1863 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion
1864 subject's DCE realm/cell, with an optional profile-specific `FriendlyName` XML attribute containing the
1865 realm's string name.

1866 **8.4.6.2 Principal**

1867 This single-valued attribute represents the SAML assertion subject's DCE principal identity.

1868 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal

1869 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion
1870 subject's DCE principal identity, with an optional profile-specific `FriendlyName` XML attribute containing
1871 the principal's string name.

1872 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
1873 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
1874 8.4.6.1).

1875 **8.4.6.3 Primary Group**

1876 This single-valued attribute represents the SAML assertion subject's primary DCE group membership.

1877 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group

1878 The single `<AttributeValue>` element contains a UUID in URN form identifying the SAML assertion
1879 subject's primary DCE group, with an optional profile-specific `FriendlyName` XML attribute containing
1880 the group's string name.

1881 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
1882 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
1883 8.4.6.1).

1884 **8.4.6.4 Groups**

1885 This multi-valued attribute represents the SAML assertion subject's DCE local group memberships.

1886 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups

1887 Each `<AttributeValue>` element contains a UUID in URN form identifying a DCE group membership
1888 of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute containing
1889 the group's string name.

1890 The profile-specific `Realm` XML attribute MAY be included and MUST contain a UUID in URN form
1891 identifying the SAML assertion subject's DCE realm/cell (the value of the attribute defined in Section
1892 8.4.6.1).

1893 **8.4.6.5 Foreign Groups**

1894 This multi-valued attribute represents the SAML assertion subject's DCE foreign group memberships.

1895 **Name:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-groups

1896 Each `<AttributeValue>` element contains a UUID in URN form identifying a DCE foreign group
1897 membership of the SAML assertion subject, with an optional profile-specific `FriendlyName` XML attribute
1898 containing the group's string name.

1899 The profile-specific Realm XML attribute MUST be included and MUST contain a UUID in URN form
1900 identifying the DCE realm/cell of the foreign group.

1901 8.4.7 Example

1902 The following is an example of the transformation of PAC data into SAML attributes belonging to a DCE
1903 principal named "jdoe" in realm "example.com", a member of the "cubicle-dwellers" and "underpaid" local
1904 groups and an "engineers" foreign group.

```
1905 <saml:Assertion  
1906 xmlns:dce="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE" ...>  
1907   <saml:Issuer>...</saml:Issuer>  
1908   <saml:Subject>...</saml:Subject>  
1909   <saml:AttributeStatement>  
1910     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1911       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:realm">  
1912       <saml:AttributeValue xsi:type="dce:DCEValueType"  
1913         dce:FriendlyName="example.com">  
1914         urn:uuid:003c6cc1-9ff8-10f9-990f-004005b13a2b  
1915         </saml:AttributeValue>  
1916       </saml:Attribute>  
1917     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1918       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:principal">  
1919       <saml:AttributeValue xsi:type="dce:DCEValueType" dce:FriendlyName="jdoe">  
1920       urn:uuid:00305ed1-albd-10f9-a2d0-004005b13a2b  
1921       </saml:AttributeValue>  
1922     </saml:Attribute>  
1923     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1924       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:primary-group">  
1925     <saml:AttributeValue xsi:type="dce:DCEValueType"  
1926       dce:FriendlyName="cubicle-dwellers">  
1927       urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b  
1928       </saml:AttributeValue>  
1929     </saml:Attribute>  
1930     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1931       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:groups">  
1932     <saml:AttributeValue xsi:type="dce:DCEValueType"  
1933       dce:FriendlyName="cubicle-dwellers">  
1934       urn:uuid:008c6181-a288-10f9-b6d6-004005b13a2b  
1935       </saml:AttributeValue>  
1936     <saml:AttributeValue xsi:type="dce:DCEValueType"  
1937       dce:FriendlyName="underpaid">  
1938       urn:uuid:006a5a91-a2b7-10f9-824d-004005b13a2b  
1939       </saml:AttributeValue>  
1940     </saml:Attribute>  
1941     <saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
1942       Name="urn:oasis:names:tc:SAML:2.0:profiles:attribute:DCE:foreign-groups">  
1943     <saml:AttributeValue xsi:type="dce:DCEValueType"  
1944       dce:FriendlyName="engineers"  
1945       dce:Realm="urn:uuid:00583221-a35f-10f9-8b6e-004005b13a2b">  
1946       urn:uuid:00099cf1-a355-10f9-9e95-004005b13a2b  
1947       </saml:AttributeValue>  
1948     </saml:Attribute>  
1949   </saml:AttributeStatement>  
1950 </saml:Assertion>
```

1951 8.5 XACML Attribute Profile

1952 SAML attribute assertions may be used as input to authorization decisions made according to the OASIS
1953 eXtensible Access Control Markup Language [XACML] standard specification. Since the SAML attribute
1954 format differs from the XACML attribute format, there is a mapping that must be performed. The XACML
1955 attribute profile facilitates this mapping by standardizing naming, value syntax, and additional attribute
1956 metadata. SAML attributes generated in conformance with this profile can be mapped automatically into
1957 XACML attributes and used as input to XACML authorization decisions.

1958 **8.5.1 Required Information**

1959 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML (this is also the target namespace
1960 assigned in the corresponding XACML profile schema document [SAMLXAC-xsd])

1961 **Contact information:** security-services-comment@lists.oasis-open.org

1962 **Description:** Given below.

1963 **Updates:** None.

1964 **8.5.2 SAML Attribute Naming**

1965 The `NameFormat` XML attribute in `<Attribute>` elements **MUST** be

1966 `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.

1967 The `Name` XML attribute **MUST** adhere to the rules specified for that format, as defined by [SAMLCore].

1968 For purposes of human readability, there may also be a requirement for some applications to carry an
1969 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
1970 [SAMLCore]) **MAY** be used for this purpose, but is not translatable into the XACML attribute equivalent.

1971 **8.5.2.1 Attribute Name Comparison**

1972 Two `<Attribute>` elements refer to the same SAML attribute if and only if their `Name` XML attribute
1973 values are equal in a binary comparison. The `FriendlyName` attribute plays no role in the comparison.

1974 **8.5.3 Profile-Specific XML Attributes**

1975 XACML requires each attribute to carry an explicit data type. To supply this data type value, a new URI-
1976 valued XML attribute called `DataType` is defined in the XML namespace

1977 `urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML`.

1978 SAML `<Attribute>` elements conforming to this profile **MUST** include the namespace-qualified

1979 `DataType` attribute, or the value is presumed to be <http://www.w3.org/2001/XMLSchema#string>.

1980 While in principle any URI reference can be used as a data type, the standard values to be used are
1981 specified in Appendix A of the XACML 2.0 Specification [XACML]. If non-standard values are used, then
1982 each XACML PDP that will be consuming mapped SAML attributes with non-standard `DataType` values
1983 must be extended to support the new data types.

1984 **8.5.4 SAML Attribute Values**

1985 The syntax of the `<AttributeValue>` element's content **MUST** correspond to the data type expressed
1986 in the profile-specific `DataType` XML attribute appearing in the parent `<Attribute>` element. For data
1987 types corresponding to the types defined in Section 3.3 of [Schema2], the `xsi:type` XML attribute
1988 **SHOULD** also be used.

1989 **8.5.5 Profile-Specific Schema**

1990 The following schema defines the profile-specific `DataType` XML attribute:

```
1991 <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"  
1992     xmlns="http://www.w3.org/2001/XMLSchema"  
1993     version="2.0">  
1994     <attribute name="DataType" type="anyURI"/>  
1995 </schema>
```

1996 8.5.6 Example

1997 The following is an example of a mapping of the "givenName" LDAP/X.500 attribute, representing the
1998 SAML assertion subject's first name. It also illustrates that a single SAML attribute can conform to multiple
1999 attribute profiles when they are compatible with each other.

```
2000 <saml:Attribute  
2001   xmlns:xacmlprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"  
2002     xmlns:ldapprof="urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP"  
2003      xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string"  
2004       ldapprof:Encoding="LDAP"  
2005       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"  
2006       Name="urn:oid:2.5.4.42" FriendlyName="givenName">  
2007   <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>  
2008 </saml:Attribute>
```

9 References

2009

- 2010 **[AES]** FIPS-197, Advanced Encryption Standard (AES), available from <http://www.nist.gov/>.
- 2011 **[Anders]** A suggestion on how to implement SAML browser bindings without using “Artifacts”,
2012 <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- 2013 **[ASN.1]** Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic
2014 notation, ITU-T Recommendation X.680, July 2002. See
2015 [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)
2016 [X.680](http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.680).
- 2017 **[eduPerson]** eduPerson.Idif. See <http://www.educase.edu/eduperson>.
- 2018 **[LDAP]** J. Hodges et al., Lightweight Directory Access Protocol (v3): Technical Specification,
2019 IETF RFC 3377, September 2002. See <http://www.ietf.org/rfc/rfc3377.txt>.
- 2020 **[Mealling]** P Leach et al, A UUID URN Namespace. Internet-Draft, draft-mealling-uuid-urn-03.
2021 January 2004
- 2022 **[MSURL]** Microsoft technical support article,
2023 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- 2024 **[NSCookie]** Persistent Client State HTTP Cookies, Netscape documentation. See
2025 http://wp.netscape.com/newsref/std/cookie_spec.html.
- 2026 **[Rescorla-Sec]** E. Rescorla et al., Guidelines for Writing RFC Text on Security Considerations,
2027 <http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt>.
- 2028 **[RFC1738]** Uniform Resource Locators (URL), <http://www.ietf.org/rfc/rfc1738.txt>
- 2029 **[RFC1750]** Randomness Recommendations for Security. <http://www.ietf.org/rfc/rfc1750.txt>
- 2030 **[RFC1945]** Hypertext Transfer Protocol -- HTTP/1.0, <http://www.ietf.org/rfc/rfc1945.txt>.
- 2031 **[RFC2045]** Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message
2032 Bodies, <http://www.ietf.org/rfc/rfc2045.txt>
- 2033 **[RFC2119]** S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, IETF RFC
2034 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.
- 2035 **[RFC2246]** The TLS Protocol Version 1.0, <http://www.ietf.org/rfc/rfc2246.txt>.
- 2036 **[RFC2256]** M. Wahl, RFC 2256 - A Summary of the X.500(96) User Schema for use with LDAPv3,
2037 December 1997
- 2038 **[RFC2279]** UTF-8, a transformation format of ISO 10646, <http://www.ietf.org/rfc/rfc2279.txt>.
- 2039 **[RFC2616]** Hypertext Transfer Protocol -- HTTP/1.1, <http://www.ietf.org/rfc/rfc2616.txt>.
- 2040 **[RFC2617]** HTTP Authentication: Basic and Digest Access Authentication, IETF RFC 2617,
2041 <http://www.ietf.org/rfc/rfc2617.txt>.
- 2042 **[RFC2798]** M. Smith, Definition of the inetOrgPerson LDAP Object Class, IETF RFC 2798, April
2043 200. See <http://www.ietf.org/rfc/rfc2798.txt>.
- 2044 **[RFC2965]** D. Cristol et al., HTTP State Management Mechanism, IETF RFC 2965, October 2000.
2045 See <http://www.ietf.org/rfc/rfc2965.txt>.
- 2046 **[RFC3061]** M. Mealling, A URN Namespace of Object Identifiers, IETF RFC 3061, February 2001.
2047 See <http://www.ietf.org/rfc/rfc3061.txt>.
- 2048 **[SAMLBind]** S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language (SAML)*
2049 *V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-bindings-2.0-cd-02. See
2050 <http://www.oasis-open.org/committees/security/>.
- 2051 **[SAMLCore]** S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion Markup*
2052 *Language (SAML) V2.0*. OASIS SSTC, September 2004. Document ID sstc-saml-core-
2053 2.0-cd-02. See <http://www.oasis-open.org/committees/security/>.

| | | |
|------|-----------------------|--|
| 2054 | [SAML DCE-xsd] | S. Cantor et al., SAML DCE PAC attribute profile schema. OASIS SSTC, September 2004. Document ID sstc-saml-schema-dce-2.0. See http://www.oasis-open.org/committees/security/ . |
| 2055 | | |
| 2056 | | |
| 2057 | [SAML ECP-xsd] | S. Cantor et al., SAML ECP profile schema. OASIS SSTC, September 2004. Document ID sstc-saml-schema-ecp-2.0. See http://www.oasis-open.org/committees/security/ . |
| 2058 | | |
| 2059 | | |
| 2060 | [SAML Gloss] | J. Hodges et al., <i>Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, September 2004. Document ID sstc-saml-glossary-2.0-cd-02. See http://www.oasis-open.org/committees/security/ . |
| 2061 | | |
| 2062 | | |
| 2063 | [SAMLX500-xsd] | S. Cantor et al., SAML LDAP attribute profile schema. OASIS SSTC, September 2004. Document ID sstc-saml-schema-ldap-2.0. See http://www.oasis-open.org/committees/security/ . |
| 2064 | | |
| 2065 | | |
| 2066 | [SAML Meta] | S. Cantor et al., <i>Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, September 2004. Document ID sstc-saml-metadata-2.0-cd-02. See http://www.oasis-open.org/committees/security/ . |
| 2067 | | |
| 2068 | | |
| 2069 | [SAML Reqs] | Darren Platt et al., SAML Requirements and Use Cases, OASIS, April 2002, http://www.oasis-open.org/committees/security/ . |
| 2070 | | |
| 2071 | [SAML Sec] | F. Hirsch et al., <i>Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, September 2004. Document ID sstc-saml-sec-consider-2.0-cd-02. See http://www.oasis-open.org/committees/security/ . |
| 2072 | | |
| 2073 | | |
| 2074 | [SAML Web] | OASIS Security Services Technical Committee website, http://www.oasis-open.org/committees/security/ . |
| 2075 | | |
| 2076 | [SAML XAC-xsd] | S. Cantor et al., SAML XACML attribute profile schema. OASIS SSTC, September 2004. Document ID sstc-saml-schema-xacml-2.0. See http://www.oasis-open.org/committees/security/ . |
| 2077 | | |
| 2078 | | |
| 2079 | [Schema1] | H. S. Thompson et al. <i>XML Schema Part 1: Structures</i> . World Wide Web Consortium Recommendation, May 2001. http://www.w3.org/TR/xmlschema-1/ . Note that this specification normatively references [Schema2], listed below. |
| 2080 | | |
| 2081 | | |
| 2082 | [Schema2] | Paul V. Biron, Ashok Malhotra, <i>XML Schema Part 2: Datatypes</i> , W3C Recommendation 02 May 2001, http://www.w3.org/TR/xmlschema-2/ |
| 2083 | | |
| 2084 | [SESSION] | RL "Bob" Morgan, Support of target web server sessions in Shibboleth, http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt |
| 2085 | | |
| 2086 | [ShibMarlena] | Marlena Erdos, Shibboleth Architecture DRAFT v1.1, http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html . |
| 2087 | | |
| 2088 | [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 . |
| 2089 | | |
| 2090 | [SSL3] | A. Frier et al., The SSL 3.0 Protocol, Netscape Communications Corp, November 1996. |
| 2091 | | |
| 2092 | [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 |
| 2093 | | |
| 2094 | [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 . |
| 2095 | | |
| 2096 | | |
| 2097 | | |
| 2098 | [XML Enc] | D. Eastlake et al., XML Encryption Syntax and Processing, http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/ , World Wide Web Consortium. |
| 2099 | | |
| 2100 | | |
| 2101 | [XML Sig] | D. Eastlake et al., XML-Signature Syntax and Processing, World Wide Web Consortium, http://www.w3.org/TR/xmlsig-core/ . |
| 2102 | | |
| 2103 | [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 |
| 2104 | | |

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

2106

Appendix A. Acknowledgments

2107 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
2108 Committee, whose voting members at the time of publication were:

- 2109 • Conor Cahill, AOL
- 2110 • John Hughes, ATOS Origin
- 2111 • Hal Lockhart, BEA Systems
- 2112 • Rick Randall, Booz Allen Hamilton
- 2113 • Ronald Jacobson, Computer Associates
- 2114 • Gavenraj Sodhi, Computer Associates
- 2115 • Tim Alsop, CyberSafe Limited
- 2116 • Paul Madsen, Entrust
- 2117 • Carolina Canales-Valenzuela, Ericsson
- 2118 • Dana Kaufman, Forum Systems
- 2119 • Irving Reid, Hewlett-Packard
- 2120 • Paula Austel, IBM
- 2121 • Maryann Hondo, IBM
- 2122 • Michael McIntosh, IBM
- 2123 • Anthony Nadalin, IBM
- 2124 • Nick Ragouzis, Individual
- 2125 • Scott Cantor, Internet2
- 2126 • Bob Morgan, Internet2
- 2127 • Prateek Mishra, Netegrity
- 2128 • Forest Yin, Netegrity
- 2129 • Peter Davis, Neustar
- 2130 • Frederick Hirsch, Nokia
- 2131 • John Kemp, Nokia
- 2132 • Senthil Sengodan, Nokia
- 2133 • Scott Kiestler, Novell
- 2134 • Cameron Morris, Novell
- 2135 • Charles Knouse, Oblix
- 2136 • Steve Anderson, OpenNetwork
- 2137 • Ari Kermaier, Oracle
- 2138 • Vamsi Motukuru, Oracle
- 2139 • Darren Platt, Ping Identity
- 2140 • Jim Lien, RSA Security
- 2141 • John Linn, RSA Security
- 2142 • Rob Philpott, RSA Security
- 2143 • Dipak Chopra, SAP
- 2144 • Jahan Moreh, Sigaba
- 2145 • Bhavna Bhatnagar, Sun Microsystems
- 2146 • Jeff Hodges, Sun Microsystems
- 2147 • Eve Maler, Sun Microsystems

- 2148 • Ronald Monzillo, Sun Microsystems
- 2149 • Emily Xu, Sun Microsystems
- 2150 • Mike Beach, Boeing
- 2151 • Greg Whitehead, Trustgenix

2152 •
2153 The editors also would like to acknowledge the following people for their contributions to previous versions of the OASIS Security Assertions Markup Language Standard:

- 2154 • Stephen Farrell, Baltimore Technologies
- 2155 • David Orchard, BEA Systems
- 2156 • Krishna Sankar, Cisco Systems
- 2157 • Zahid Ahmed, CommerceOne
- 2158 • Carlisle Adams, Entrust
- 2159 • Tim Moses, Entrust
- 2160 • Nigel Edwards, Hewlett-Packard
- 2161 • Joe Pato, Hewlett-Packard
- 2162 • Bob Blakley, IBM
- 2163 • Marlena Erdos, IBM
- 2164 • Marc Chanliau, Netegrity
- 2165 • Chris McLaren, Netegrity
- 2166 • Lynne Rosenthal, NIST
- 2167 • Mark Skall, NIST
- 2168 • Simon Godik, Overxeer
- 2169 • Charles Norwood, SAIC
- 2170 • Evan Prodromou, Securant
- 2171 • Robert Griffin, RSA Security (former editor)
- 2172 • Sai Allarvarpu, Sun Microsystems
- 2173 • Chris Ferris, Sun Microsystems
- 2174 • Emily Xu, Sun Microsystems
- 2175 • Mike Myers, Traceroute Security
- 2176 • Phillip Hallam-Baker, VeriSign (former editor)
- 2177 • James Vanderbeek, Vodafone
- 2178 • Mark O'Neill, Vordel
- 2179 • Tony Palmer, Vordel

2180 Finally, the editors wish to acknowledge the following people for their contributions of material used as
2181 input to the OASIS Security Assertions Markup Language specifications:

- 2182 • Thomas Gross, IBM
- 2183 • Birgit Pfitzmann, IBM

Appendix B. Notices

2185 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
2186 might be claimed to pertain to the implementation or use of the technology described in this document or
2187 the extent to which any license under such rights might or might not be available; neither does it represent
2188 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
2189 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
2190 available for publication and any assurances of licenses to be made available, or the result of an attempt
2191 made to obtain a general license or permission for the use of such proprietary rights by implementors or
2192 users of this specification, can be obtained from the OASIS Executive Director.

2193 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
2194 other proprietary rights which may cover technology that may be required to implement this specification.
2195 Please address the information to the OASIS Executive Director.

2196 **Copyright © OASIS Open 2004. All Rights Reserved.**

2197 This document and translations of it may be copied and furnished to others, and derivative works that
2198 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
2199 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
2200 this paragraph are included on all such copies and derivative works. However, this document itself may
2201 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
2202 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
2203 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
2204 into languages other than English.

2205 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
2206 or assigns.

2207 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2208 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
2209 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
2210 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.