



Security Assertion Markup Language (SAML) V2.0 Technical Overview v14

Draft

26 Sept 2007

Specification URIs:

This Version:

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-draft-14.html>

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-draft-14.pdf>

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-draft-14.odt>

Previous Version:

<http://www.oasis-open.org/committees/download.php/22555/sstc-saml-tech-overview-2.0-draft-13.odt>

Latest Version:

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html>

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.pdf>

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.odt>

Latest Approved Version:

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-01.html>

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-01.pdf>

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-01.odt>

Technical Committee:

OASIS Security Services TC

Chairs:

Hal Lockhart, BEA

Prateek Mishra, Oracle

Editors:

Nick Ragouzis, Enosis Group LLC

John Hughes, PA Consulting

Rob Philpott, EMC Corporation

Eve Maler, Sun Microsystems

Paul Madsen, NTT

Tom Scavo, NCSA/University of Illinois

Related Work:

N/A

Abstract:

The Security Assertion Markup Language (SAML) standard defines a framework for exchanging security information between online business partners. This document provides a technical

39 description of SAML V2.0.

40 **Status:**

41 The level of approval of this document is listed above. Check the "Latest Version" or "Latest
42 Approved Version" location noted above for possible later revisions of this document.

43 TC members should send comments on this specification to the TC's email list. Others should
44 send comments to the TC by using the "Send A Comment" button on the TC's web page at
45 <http://www.oasis-open.org/committees/security>.

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

Notices

50

51 Copyright © OASIS® 1993–2007. All Rights Reserved. OASIS trademark, IPR and other policies apply.

52

53 All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual
54 Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

55

56 This document and translations of it may be copied and furnished to others, and derivative works that
57 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published,
58 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice
59 and this section are included on all such copies and derivative works. However, this document itself may
60 not be modified in any way, including by removing the copyright notice or references to OASIS, except as
61 needed for the purpose of developing any document or deliverable produced by an OASIS Technical
62 Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be
63 followed) or as required to translate it into languages other than English.

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

58

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

60

61 OASIS requests that any OASIS Party or any other party that believes it has patent claims that would
62 necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to
63 notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such
64 patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced
65 this specification.

62

63 OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any
64 patent claims that would necessarily be infringed by implementations of this specification by a patent
65 holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR
66 Mode of the OASIS Technical Committee that produced this specification. OASIS may include such
67 claims on its website, but disclaims any obligation to do so.

64

65 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
66 might be claimed to pertain to the implementation or use of the technology described in this document or
67 the extent to which any license under such rights might or might not be available; neither does it represent
68 that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to
69 rights in any document or deliverable produced by an OASIS Technical Committee can be found on the
66 OASIS website. Copies of claims of rights made available for publication and any assurances of licenses
67 to be made available, or the result of an attempt made to obtain a general license or permission for the
68 use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS
69 Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any
70 information or list of intellectual property rights will at any time be complete, or that any claims in such list
71 are, in fact, Essential Claims.

67

68 The names "OASIS", [insert specific trademarked names, abbreviations, etc. here] are trademarks of
69 OASIS, the owner and developer of this specification, and should be used only to refer to the organization
70 and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications,
71 while reserving the right to enforce its marks against misleading uses. Please see [http://www.oasis-](http://www.oasis-open.org/who/trademark.php)
72 [open.org/who/trademark.php](http://www.oasis-open.org/who/trademark.php) for above guidance.

69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113

Table of Contents

- 1 Introduction.....6
- 1.1 References.....6
- 2 Overview.....8
- 2.1 Drivers of SAML Adoption.....8
- 2.2 Documentation Roadmap.....8
- 3 High-Level SAML Use Cases.....11
- 3.1 SAML Participants.....11
- 3.2 Web Single Sign-On Use Case.....11
- 3.3 Identity Federation Use Case.....12
- 4 SAML Architecture.....16
- 4.1 Basic Concepts.....16
- 4.2 Advanced Concepts.....17
- 4.2.1 Subject Confirmation.....17
- 4.3 SAML Components.....17
- 4.4 SAML XML Constructs and Examples.....19
- 4.4.1 Relationship of SAML Components.....19
- 4.4.2 Assertion, Subject, and Statement Structure.....20
- 4.4.3 Attribute Statement Structure.....21
- 4.4.4 Message Structure and the SOAP Binding.....22
- 4.5 Privacy in SAML.....24
- 4.6 Security in SAML.....25
- 5 Major Profiles and Federation Use Cases.....26
- 5.1 Web Browser SSO Profile.....26
- 5.1.1 Introduction.....26
- 5.1.2 SP-Initiated SSO: Redirect/POST Bindings.....27
- 5.1.3 SP-Initiated SSO: POST/Artifact Bindings.....30
- 5.1.4 IdP-Initiated SSO: POST Binding.....33
- 5.2 ECP Profile.....35
- 5.2.1 Introduction.....35
- 5.2.2 ECP Profile Using PAOS Binding.....35
- 5.3 Single Logout Profile.....36
- 5.3.1 Introduction.....37
- 5.3.2 SP-Initiated Single Logout with Multiple SPs.....37
- 5.4 Establishing and Managing Federated Identities.....38
- 5.4.1 Introduction.....38
- 5.4.2 Federation Using Out-of-Band Account Linking.....38
- 5.4.3 Federation Using Persistent Pseudonym Identifiers.....40
- 5.4.4 Federation Using Transient Pseudonym Identifiers.....42
- 5.4.5 Federation Termination.....44
- 5.5 Use of Attributes.....45
- 6 Extending and Profiling SAML for Use in Other Frameworks.....46
- 6.1 Web Services Security (WS-Security).....46
- 6.2 eXtensible Access Control Markup Language (XACML).....48

Table of Figures

Figure 1: SAML V2.0 Document Set.....	6
Figure 2: General Single Sign-On Use Case.....	9
Figure 3: General Identity Federation Use Case.....	11
Figure 4: Basic SAML Concepts.....	13
Figure 5: Relationship of SAML Components.....	17
Figure 6: Assertion with Subject, Conditions, and Authentication Statement.....	17
Figure 7: Attribute Statement.....	19
Figure 8: Protocol Messages Carried by SOAP Over HTTP.....	20
Figure 9: Authentication Request in SOAP Envelope.....	20
Figure 10: Response in SOAP Envelope.....	21
Figure 11: Differences in Initiation of Web Browser SSO.....	24
Figure 12: SP-Initiated SSO with Redirect and POST Bindings.....	25
Figure 13: IdP-Initiated SSO with POST Binding.....	31
Figure 14: Enhanced Client/Proxy Use Cases.....	32
Figure 15: SSO Using ECP with the PAOS Binding.....	33
Figure 16: SP-Initiated Single Logout with Multiple SPs.....	34
Figure 17: Identity Federation with Out-of-Band Account Linking.....	36
Figure 18: SP-Initiated Identity Federation with Persistent Pseudonym.....	38
Figure 19: SP-Initiated Identity Federation with Transient Pseudonym.....	40
Figure 20: Identity Federation Termination.....	41
Figure 21: WS-Security with a SAML Token.....	44
Figure 22: Typical Use of WS-Security with SAML Token.....	45
Figure 23: SAML and XACML Integration.....	46

1 Introduction

The Security Assertion Markup Language (SAML) standard defines a framework for exchanging security information between online business partners. It was developed by the Security Services Technical Committee (SSTC) of the standards organization OASIS (the Organization for the Advancement of Structured Information Standards). This document provides a technical description of SAML V2.0.

1.1 References

- [SAMLAuthnCxt]** J. Kemp et al. *Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-authn-context-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf>.
- [SAMLBind]** S. Cantor et al. *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf>.
- [SAMLConform]** P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-conformance-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-conformance-2.0-os.pdf>.
- [SAMLCore]** S. Cantor et al. *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-core-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>.
- [SAMLErrata]** J. Moreh. Errata for the *OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, May, 2006. Document ID sstc-saml-errata-2.0-draft-nn. See <http://www.oasis-open.org/committees/security/>.
- [SAMLExecOvr]** P. Madsen, et al. *SAML V2.0 Executive Overview*. OASIS SSTC, April, 2005. Document ID sstc-saml-exec-overview-2.0-cd-01. See <http://www.oasis-open.org/committees/security/>.
- [SAMLGloss]** J. Hodges et al. *Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-glossary-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-glossary-2.0-os.pdf>.
- [SAMLMDExtQ]** T. Scavo, et al. *SAML Metadata Extension for Query Requesters*. OASIS SSTC, March 2006. Document ID sstc-saml-metadata-ext-query-cd-01. See <http://www.oasis-open.org/committees/security/>.
- [SAMLMDV1x]** G. Whitehead et al. *Metadata Profile for the OASIS Security Assertion Markup Language (SAML) V1.x*. OASIS SSTC, March 2005. Document ID sstc-saml1x-metadata-cd-01. See <http://www.oasis-open.org/committees/security/>.
- [SAMLMeta]** S. Cantor et al. *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-metadata-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf>.
- [SAMLProf]** S. Cantor et al. *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-profiles-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf>.
- [SAMLProt3P]** S. Cantor. *SAML Protocol Extension for Third-Party Requests*. OASIS SSTC, March 2006. Document ID sstc-saml-protocol-ext-thirdparty-cd-01. See <http://www.oasis-open.org/committees/security/>.
- [SAMLSec]** F. Hirsch et al. *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-sec-consider-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf>.
- [SAMLWeb]** OASIS Security Services Technical Committee web site, <http://www.oasis-open.org/committees/security>.

167 **[SAMLX509Attr]** R. Randall et al. *SAML Attribute Sharing Profile for X.509 Authentication-Based*
168 *Systems*. OASIS SSTC, March 2006. Document ID sstc-saml-x509-authn-attr-
169 profile-cd-02. See <http://www.oasis-open.org/committees/security/>.

170 **[SAMLXPathAttr]** C. Morris et al. *SAML XPath Attribute Profile*. OASIS SSTC, August, 2005. Document
171 ID sstc-saml-xpath-attribute-profile-cd-01. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
172 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).

173 **[ShibReqs]** S. Carmody. *Shibboleth Overview and Requirements*. Shibboleth project of Internet2.
174 See [http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-](http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-requirements-01.html)
175 [requirements-01.html](http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-requirements-01.html).

176 **[WSS]** A. Nadalin et al. *Web Services Security: SOAP Message Security 1.1 (WS-Security*
177 *2004)*. OASIS WSS-TC, February 2006. Document ID wss-v1.1-spec-os-
178 SOAPMessageSecurity. See <http://www.oasis-open.org/committees/wss/>.

179 **[WSSSAML]** R. Monzillo et al. *Web Services Security: SAML Token Profile 1.1*. OASIS WSS-TC,
180 February 2006. Document ID wss-v1.1-spec-os-SAMLSAMLTokenProfile. See
181 <http://www.oasis-open.org/committees/wss/>.

182 **[XACML]** T. Moses, et al. *OASIS eXtensible Access Control Markup Language (XACML)*
183 *Version 2.0*. OASIS XACML-TC, February 2005. Document ID oasis-access_control-
184 xacml-2.0-core-spec-os. See <http://www.oasis-open.org/committees/xacml>.

185 **[XMLEnc]** D. Eastlake et al. *XML Encryption Syntax and Processing*. World Wide Web
186 Consortium. See <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>.

187 2 Overview

188 The OASIS Security Assertion Markup Language (SAML) standard defines an XML-based framework for
189 describing and exchanging security information between on-line business partners. This security
190 information is expressed in the form of portable SAML assertions that applications working across security
191 domain boundaries can trust. The OASIS SAML standard defines precise syntax and rules for requesting,
192 creating, communicating, and using these SAML assertions.

193 The OASIS Security Services Technical Committee (SSTC) develops and maintains the SAML standard.
194 The SSTC has produced this technical overview to assist those wanting to know more about SAML by
195 explaining the business use cases it addresses, the high-level technical components that make up a
196 SAML deployment, details of message exchanges for common use cases, and where to go for additional
197 information.

198 2.1 Drivers of SAML Adoption

199 Why is SAML needed for exchanging security information? There are several drivers behind the adoption
200 of the SAML standard, including:

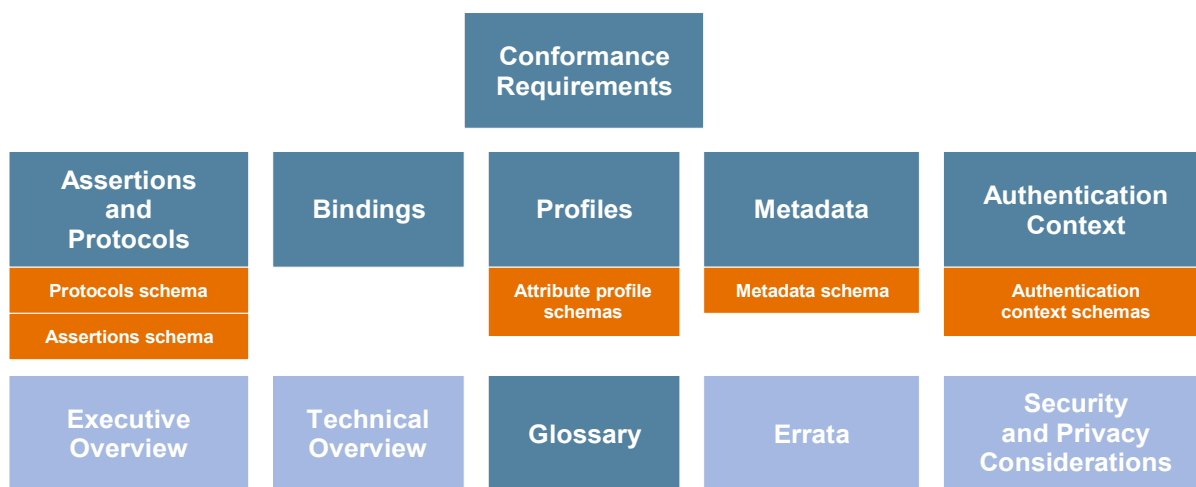
- 201 • **Single Sign-On:** Over the years, various products have been marketed with the claim of providing
202 support for web-based SSO. These products have typically relied on browser cookies to maintain
203 user authentication state information so that re-authentication is not required each time the web user
204 accesses the system. However, since browser cookies are never transmitted between DNS
205 domains, the authentication state information in the cookies from one domain is never available to
206 another domain. Therefore, these products have typically supported multi-domain SSO (MDSSO)
207 through the use of proprietary mechanisms to pass the authentication state information between the
208 domains. While the use of a single vendor's product may sometimes be viable within a single
209 enterprise, business partners usually have heterogeneous environments that make the use of
210 proprietary protocols impractical for MDSSO. SAML solves the MDSSO problem by providing a
211 standard vendor-independent grammar and protocol for transferring information about a user from
212 one web server to another independent of the server DNS domains.
- 213 • **Federated identity:** When online services wish to establish a collaborative application environment
214 for their mutual users, not only must the systems be able to understand the protocol syntax and
215 semantics involved in the exchange of information; they must also have a common understanding of
216 who the user is that is referred to in the exchange. Users often have individual local user identities
217 within the security domains of each partner with which they interact. Identity federation provides a
218 means for these partner services to agree on and establish a common, shared name identifier to
219 refer to the user in order to share information about the user across the organizational boundaries.
220 The user is said to have a **federated identity** when partners have established such an agreement
221 on how to refer to the user. From an administrative perspective, this type of sharing can help reduce
222 identity management costs as multiple services do not need to independently collect and maintain
223 identity-related data (e.g. passwords, identity attributes). In addition, administrators of these services
224 usually do not have to manually establish and maintain the shared identifiers; rather control for this
225 can reside with the user.
- 226 • **Web services and other industry standards:** SAML allows for its security assertion format to be
227 used outside of a "native" SAML-based protocol context. This modularity has proved useful to other
228 industry efforts addressing authorization services (IETF, OASIS), identity frameworks, web services
229 (OASIS, Liberty Alliance), etc. The OASIS WS-Security Technical Committee has defined a **profile**
230 for how to use SAML's rich assertion constructs within a WS-Security **security token** that can be
231 used, for example, to secure web service SOAP message exchanges. In particular, the advantage
232 offered by the use of a SAML assertion is that it provides a standards-based approach to the
233 exchange of information, including attributes, that are not easily conveyed using other WS-Security
234 token formats.

235 2.2 Documentation Roadmap

236 The OASIS SSTC has produced numerous documents related to SAML V2.0. This includes documents
237 that make up the official OASIS standard itself, outreach material intended to help the public better

238 understand SAML V2.0, and several extensions to SAML to facilitate its use in specific environments or to
239 integrate it with other technologies.

240 The documents that define and support the SAML V2.0 OASIS Standard are shown in Figure 1. The
241 lighter-colored boxes represent non-normative information.



SAML-docset

Figure 1: SAML V2.0 Document Set

- 243 • **Conformance Requirements** documents the technical requirements for SAML conformance, a
244 status that software vendors typically care about because it is one measure of cross-product
245 compatibility. If you need to make a formal reference to SAML V2.0 from another document, you
246 simply need to point to this one.
- 247 • **Assertions and Protocol** defines the syntax and semantics for creating XML-encoded assertions
248 to describe authentication, attribute, and authorization information, and for the protocol messages to
249 carry this information between systems. It has associated schemas, one for assertions and one for
250 protocols.
- 251 • **Bindings** defines how SAML assertions and request-response protocol messages can be
252 exchanged between systems using common underlying communication protocols and frameworks.
- 253 • **Profiles** defines specific sets of rules for using and restricting SAML's rich and flexible syntax for
254 conveying security information to solve specific business problems (for example, to perform a web
255 SSO exchange). It has several associated small schemas covering syntax aspects of attribute
256 profiles.
- 257 • **Metadata** defines how a SAML entity can describe its configuration data (e.g. service endpoint
258 URLs, key material for verifying signatures) in a standard way for consumption by partner entities. It
259 has an associated schema.
- 260 • **Authentication Context** defines a syntax for describing authentication context declarations which
261 describe various authentication mechanisms. It has an associated set of schemas.
- 262 • **Executive Overview** provides a brief executive-level overview of SAML and its primary benefits.
263 This is a non-normative document.
- 264 • **Technical Overview** is the document you are reading.
- 265 • **Glossary** normatively defines terms used throughout the SAML specifications. Where possible,
266 terms are aligned with those defined in other security glossaries.
- 267 • **Errata** clarifies interpretation of the SAML V2.0 standard where information in the final published
268 version was conflicting or unclear. Although the advice offered in this document is non-normative, it
269 is useful as a guide to the likely interpretations used by implementors of SAML-conforming software,
270 and is likely to be incorporated in any future revision to the standard. This document is updated on

271 an ongoing basis.

272 • **Security and Privacy Considerations** describes and analyzes the security and privacy properties
273 of SAML.

274 Following the release of the SAML V2.0 OASIS Standard, the OASIS SSTC has continued work on
275 several enhancements. As of this writing, the documents for the following enhancements have been
276 approved as OASIS Committee Draft specifications and are available from the OASIS SSTC web site:

277 • **SAML Metadata Extension for Query Requesters** . Defines role descriptor types that describe a
278 standalone SAML V1.x or V2.0 query requester for each of the three predefined query types.

279 • **SAML Attribute Sharing Profile for X.509 Authentication-Based Systems** . Describes a SAML
280 profile enabling an attribute requester entity to make SAML attribute queries about users that have
281 authenticated at the requester entity using an X.509 client certificate.

282 • **SAML V1.x Metadata** . Describes the use of the SAML V2.0 metadata constructs to describe
283 SAML entities that support the SAML V1.x OASIS Standard.

284 • **SAML XPath Attribute Profile** . Profiles the use of SAML attributes for using XPath URI's as
285 attribute names.

286 • **SAML Protocol Extension for Third-Party Requests** . Defines an extension to the SAML protocol
287 to facilitate requests made by entities other than the intended response recipient.

288 3 High-Level SAML Use Cases

289 Prior to examining details of the SAML standard, it's useful to describe some of the high-level use cases it
290 addresses. More detailed use cases are described later in this document along with specific SAML
291 profiles.

292 3.1 SAML Participants

293 Who are the participants involved in a SAML interaction? At a minimum, SAML exchanges take place
294 between system entities referred to as a SAML *asserting party* and a SAML *relying party*. In many SAML
295 use cases, a user, perhaps running a web browser or executing a SAML-enabled application, is also a
296 participant, and may even be the asserting party.

297 An asserting party is a system entity that makes SAML assertions. It is also sometimes called a *SAML*
298 *authority*. A relying party is a system entity that uses assertions it has received. When a SAML asserting
299 or relying party makes a direct request to another SAML entity, the party making the request is called a
300 *SAML requester*, and the other party is referred to as a *SAML responder*. A replying party's willingness to
301 rely on information from an asserting party depends on the existence of a trust relationship with the
302 asserting party.

303 SAML system entities can operate in a variety of SAML *roles* which define the SAML services and protocol
304 messages they will use and the types of assertions they will generate or consume. For example, to
305 support Multi-Domain Single Sign-On (MDSSO, or often just SSO), SAML defines the roles called *identity*
306 *provider (IdP)* and *service provider (SP)*. Another example is the *attribute authority* role where a SAML
307 entity produces assertions in response to identity attribute queries from an entity acting as an *attribute*
308 *requester*.

309 At the heart of most SAML assertions is a *subject* (a principal – an entity that can be authenticated – within
310 the context of a particular security domain) about which something is being asserted. The subject could be
311 a human but could also be some other kind of entity, such as a company or a computer. The terms
312 subject and principal tend to be used interchangeably in this document.

313 A typical assertion from an identity provider might convey information such as “This user is John Doe, he
314 has an email address of john.doe@example.com, and he was authenticated into this system using a
315 password mechanism.” A service provider could choose to use this information, depending on its access
316 policies, to grant John Doe web SSO access to local resources.

317 3.2 Web Single Sign-On Use Case

318 Multi-domain web single sign-on is arguably the most important use case for which SAML is applied. In
319 this use case, a user has a login session (that is, a *security context*) on a web site (airline.example.com)
320 and is accessing resources on that site. At some point, either explicitly or transparently, he is directed over
321 to a partner's web site (cars.example.co.uk). In this case, we assume that a federated identity for the user
322 has been previously established between airline.example.com and cars.example.co.uk based on a
323 business agreement between them. The identity provider site (airline.example.com) asserts to the service
324 provider site (cars.example.co.uk) that the user is known (by referring to the user by their federated
325 identity), has authenticated to it, and has certain identity attributes (e.g. has a “Gold membership”). Since
326 cars.example.co.uk trusts airline.example.com, it trusts that the user is valid and properly authenticated
327 and thus creates a local session for the user. This use case is shown in Figure 2, which illustrates the fact
328 that the user is not required to re-authenticate when directed over to the cars.example.co.uk site.

329

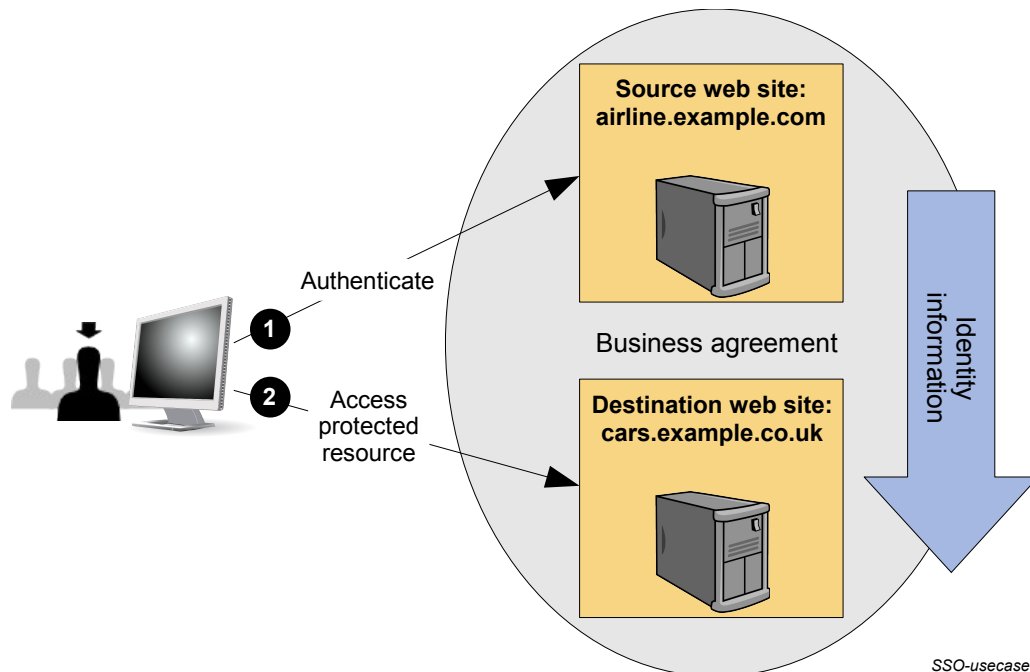


Figure 2: General Single Sign-On Use Case

330 This high-level description indicated that the user had first authenticated at the IdP before accessing a
 331 protected resource at the SP. This scenario is commonly referred to as an IdP-initiated web SSO
 332 scenario. While IdP-initiated SSO is useful in certain cases, a more common scenario starts with a user
 333 visiting an SP site through a browser bookmark, possibly first accessing resources that require no special
 334 authentication or authorization. In a SAML-enabled deployment, when they subsequently attempt to
 335 access a protected resource at the SP, the SP will send the user to the IdP with an authentication request
 336 in order to have the user log in. Thus this scenario is referred to as SP-initiated web SSO. Once logged in,
 337 the IdP can produce an assertion that can be used by the SP to validate the user's access rights to the
 338 protected resource. SAML V2.0 supports both the IdP-initiated and SP-initiated flows.

339 SAML supports numerous variations on these two primary flows that deal with requirements for using
 340 various types and strengths of user authentication methods, alternative formats for expressing federated
 341 identities, use of different bindings for transporting the protocol messages, inclusion of identity attributes,
 342 etc. Many of these options are looked at in more detail in later sections of this document.

343 3.3 Identity Federation Use Case

344 As mentioned earlier, a user's identity is said to be federated between a set of providers when there is an
 345 agreement between the providers on a set of identifiers and/or identity attributes by which the sites will
 346 refer to the user.

347 There are many questions that must be considered when business partners decide to use federated
 348 identities to share security and identity information about users. For example:

- 349 • Do the users have existing local identities at the sites that must be linked together through the
 350 federated identifiers?
- 351 • Will the establishment and termination of federated identifiers for the users be done dynamically or
 352 will the sites use pre-established federated identifiers?
- 353 • Do users need to explicitly consent to establishment of the federated identity?
- 354 • Do identity attributes about the users need to be exchanged?
- 355 • Should the identity federation rely on transient identifiers that are destroyed at the end of the user
 356 session?
- 357 • Is the privacy of information to be exchanged of high concern such that the information should be

358 encrypted?

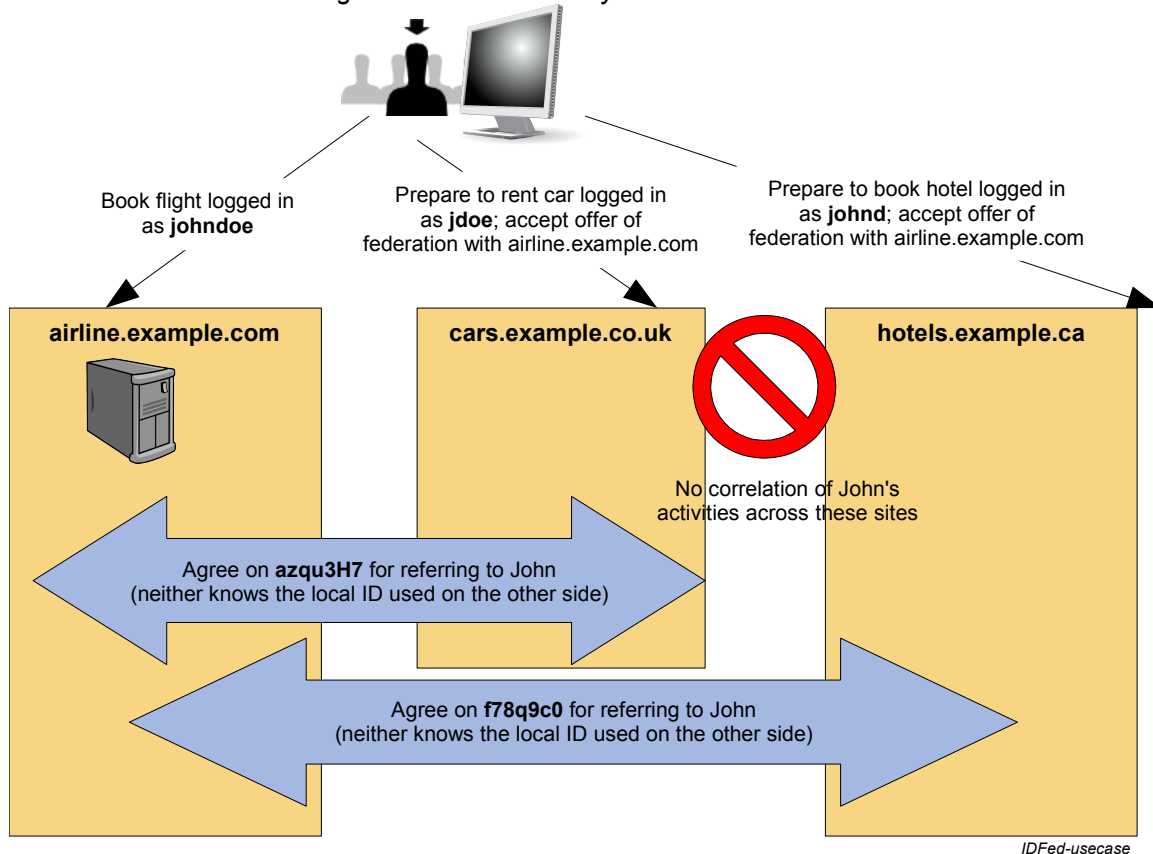
359 Previous versions of the SAML standard relied on out-of-band agreement on the types of identifiers that
360 would be used to represent a federated identity between partners (e.g. the use of X.509 subject names).
361 While it supported the use of federated identities, it provided no means to directly establish the identifiers
362 for those identities using SAML message exchanges. SAML V2.0 introduced two features to enhance its
363 federated identity capabilities. First, new constructs and messages were added to support the dynamic
364 establishment and management of federated name identifiers. Second, two new types of name identifiers
365 were introduced with privacy-preserving characteristics.

366 In some cases, exchanges of identity-related federation information may take place outside of the SAML
367 V2.0 message exchanges. For example, providers may choose to share information about registered
368 users via batch or off-line "identity feeds" that are driven by data sources (for example, human resources
369 databases) at the identity provider and then propagated to service providers. Subsequently, the user's
370 federated identity may be used in a SAML assertion and propagated between providers to implement
371 single sign-on or to exchange identity attributes about the user. Alternatively, identity federation may be
372 achieved purely by a business agreement that states that an identity provider will refer to a user based on
373 certain attribute names and values, with no additional flows required for maintaining and updating user
374 information between providers.

375 The high-level identity federation use case described here demonstrates how SAML can use the new
376 features to dynamically establish a federated identity for a user during a web SSO exchange. Most
377 identity management systems maintain *local identities* for users. These local identities might be
378 represented by the user's local login account or some other locally identifiable user profile. These local
379 identities must be linked to the federated identity that will be used to represent the user when the provider
380 interacts with a partner. The process of associating a federated identifier with the local identity at a partner
381 (or partners) where the federated identity will be used is often called *account linking*.

382 This use case, shown in , demonstrates how, during web SSO, the sites can dynamically establish the
383 federated name identifiers used in the account linking process. One identity provider,
384 airline.example.com, and two service providers exist in this example: cars.example.co.uk for car rentals
385 and hotels.example.ca for hotel bookings. The example assumes a user is registered on all three provider
386 sites (i.e. they have pre-existing local login accounts), but the local accounts all have different account
387 identifiers. At airline.example.com, user John is registered as **johndoe**, on cars.example.co.uk his
388 account is **jdoue**, and on hotels.example.ca it is **johnd**. The sites have established an agreement to use
389 **persistent** SAML privacy-preserving pseudonyms for the user's federated name identifiers. John has not
390 previously federated his identities between these sites.

Figure 3: General Identity Federation Use Case



392 The processing sequence is as follows:

- 393 1. John books a flight at airline.example.com using his **johndoe** user account.
- 394 2. John then uses a browser bookmark or clicks on a link to visit cars.example.co.uk to reserve a car.
- 395 This site sees that the browser user is not logged in locally but that he has previously visited their IdP
- 396 partner site airline.example.com (optionally using the new IdP discovery feature of SAML V2.0). So
- 397 cars.example.co.uk asks John if he would like to consent to federate his local cars.example.co.uk
- 398 identity with airline.example.com.
- 399 3. John consents to the federation and his browser is redirected back to airline.example.com where the
- 400 site creates a new pseudonym, **azqu3H7** for John's use when he visits cars.example.co.uk. The
- 401 pseudonym is linked to his **johndoe** account. Both providers agree to use this identifier to refer to John
- 402 in subsequent transactions.
- 403 4. John is then redirected back to cars.example.co.uk with a SAML assertion indicating that the user
- 404 represented by the federated persistent identifier **azqu3H7** is logged in at the IdP. Since this is the first
- 405 time that cars.example.co.uk has seen this identifier, it does not know which local user account to
- 406 which it applies.
- 407 5. Thus, John must log in at cars.example.co.uk using his **jdoe** account. Then cars.example.co.uk
- 408 attaches the identity **azqu3H7** to the local **jdoe** account for future use with the IdP airline.example.com.
- 409 The user accounts at the IdP and this SP are now *linked* using the federated name identifier **azqu3H7**.
- 410 6. After reserving a car, John selects a browser bookmark or clicks on a link to visit hotels.example.ca in
- 411 order to book a hotel room.
- 412 7. The federation process is repeated with the IdP airline.example.com, creating a new pseudonym,
- 413 **f78q9C0**, for IdP user **johndoe** that will be used when visiting hotels.example.ca.

414 8. John is redirected back to the hotels.example.ca SP with a new SAML assertion. The SP requires John
415 to log into his local **johnd** user account and adds the pseudonym as the federated name identifier for
416 future use with the IdP airline.example.com. The user accounts at the IdP and this SP are now *linked*
417 using the federated name identifier **f78q9C0**.

418 In the future, whenever John needs to books a flight, car, and hotel, he will only need to log in once to
419 airline.example.com before visiting cars.example.co.uk and hotels.example.ca. The airline.example.com
420 IdP will identify John as **azqu3H7** to cars.example.co.uk and as **f78q9C0** to hotels.example.ca. Each SP
421 will locate John's local user account through the linked persistent pseudonyms and allow John to conduct
422 business after the SSO exchange.

423 4 SAML Architecture

424 This section provides a brief description of the key SAML concepts and the components defined in the
425 standard.

426 4.1 Basic Concepts

427 SAML consists of building-block components that, when put together, allow a number of use cases to be
428 supported. The components primarily permit transfer of identity, authentication, attribute, and
429 authorization information between autonomous organizations that have an established trust relationship.
430 The **core** SAML specification defines the structure and content of both *assertions* and *protocol messages*
431 used to transfer this information.

432 SAML assertions carry statements about a principal that an asserting party claims to be true. The valid
433 structure and contents of an assertion are defined by the SAML assertion XML schema. Assertions are
434 usually created by an asserting party based on a request of some sort from a relying party, although under
435 certain circumstances, the assertions can be delivered to a relying party in an unsolicited manner. SAML
436 protocol messages are used to make the SAML-defined requests and return appropriate responses. The
437 structure and contents of these messages are defined by the SAML-defined protocol XML schema.

438 The means by which lower-level communication or messaging protocols (such as HTTP or SOAP) are
439 used to transport SAML protocol messages between participants is defined by the SAML *bindings*.

440 Next, SAML *profiles* are defined to satisfy a particular business use case, for example the Web Browser
441 SSO profile. Profiles typically define constraints on the contents of SAML assertions, protocols, and
442 bindings in order to solve the business use case in an interoperable fashion. There are also Attribute
443 Profiles, which do not refer to any protocol messages and bindings, that define how to exchange attribute
444 information using assertions in ways that align with a number of common usage environments (e.g. X.
445 500/LDAP directories, DCE).

446 Figure 4 illustrates the relationship between these basic SAML concepts.

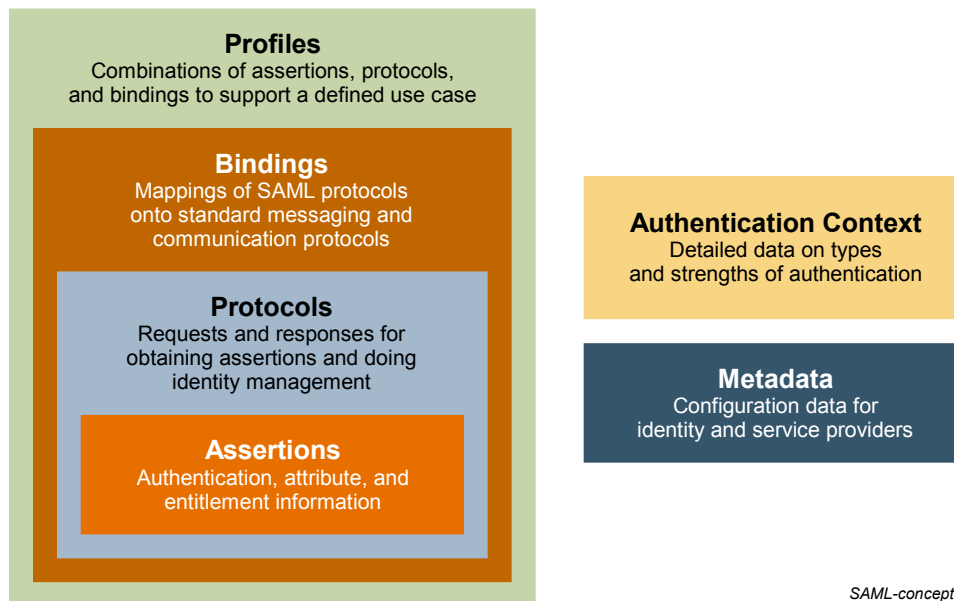


Figure 4: Basic SAML Concepts

448 Two other SAML concepts are useful for building and deploying a SAML environment:

- 449 • **Metadata** defines a way to express and share configuration information between SAML parties. For
450 instance, an entity's supported SAML bindings, operational roles (IDP, SP, etc), identifier
451 information, supporting identity attributes, and key information for encryption and signing can be

452 expressed using SAML metadata XML documents. SAML Metadata is defined by its own XML
453 schema.

454 • In a number of situations, a service provider may need to have detailed information regarding the
455 type and strength of authentication that a user employed when they authenticated at an identity
456 provider. A SAML *authentication context* is used in (or referred to from) an assertion's
457 authentication statement to carry this information. An SP can also include an authentication context
458 in a request to an IdP to request that the user be authenticated using a specific set of authentication
459 requirements, such as a multi-factor authentication. There is a general XML schema that defines the
460 mechanisms for creating authentication context declarations and a set of SAML-defined
461 Authentication Context Classes, each with their own XML schema, that describe commonly used
462 methods of authentication.

463 This document does not go into further detail about Metadata and Authentication Context; for more
464 information, see the specifications that focus on them (and , respectively).

465 It should be noted that the story of SAML need not end with its published set of assertions, protocols,
466 bindings, and profiles. It is designed to be highly flexible, and thus it comes with extensibility points in its
467 XML schemas, as well as guidelines for custom-designing new bindings and profiles in such a way as to
468 ensure maximum interoperability.

469 **4.2 Advanced Concepts**

470 **4.2.1 Subject Confirmation**

471 A SAML Assertion may contain an element called `SubjectConfirmation`. In practical terms, what
472 `SubjectConfirmation` says is "these are the conditions under which an attesting entity (somebody
473 trying to use the assertion) is permitted to do so". The "wielder" is attesting to its right to use the assertion,
474 usually by implying a relationship with the subject. An assertion can have any number of
475 `SubjectConfirmation` elements, but an attesting entity only has to satisfy one of them.

476 The `SubjectConfirmation` element provides the means for a relying party to verify the
477 correspondence of the subject of the assertion with the party with whom the relying party is
478 communicating. The `Method` attribute indicates the specific method that the relying party should use to
479 make this determination.
480

481 SAML 2.0 accounts for three different security scenarios by defining three values for the `Method` attribute
482 of the `SubjectConfirmation` element, these are

```
483 urn:oasis:names:tc:SAML:2.0:cm:holder-of-key  
484 urn:oasis:names:tc:SAML:2.0:cm:sender-vouches  
485 urn:oasis:names:tc:SAML:2.0:cm:bearer
```

486 In the `holder-of-key` model, the relying party will allow any party capable of demonstrating knowledge
487 of specific key information contained with the `SubjectConfirmation` element's
488 `SubjectConfirmationData` element to use the assertion (and thereby lay claim to some relationship
489 with the subject within).

490 In the `bearer` model, the relying party will allow any party that bears the Assertion (assuming any
491 other constraints are also met) to use the assertion (and thereby lay claim to some relationship with the
492 subject within).

493 In the `sender-vouches` model, the relying party will use other criteria in determining which parties should
494 be allowed to use the assertion (and thereby lay claim to some relationship with the subject within).

495 **4.3 SAML Components**

496 This section takes a more detailed look at each of the components that represent the assertion, protocol,
497 binding, and profile concepts in a SAML environment.

- 498 • **Assertions:** SAML allows for one party to assert security information in the form of **statements**
499 about a **subject**. For instance, a SAML assertion could state that the subject is named “John Doe”,
500 has an email address of john.doe@example.com, and is a member of the “engineering” group. An
501 assertion contains some basic required and optional information that applies all assertions, and
502 usually contains a *subject* of the assertion, *conditions* used to validate the assertion, and assertion
503 statements. SAML defines three kinds of statements that can be carried within an assertion:
- 504 • **Authentication statements:** These are created by the party that successfully authenticated a
505 user. At a minimum, they describe the particular means used to authenticate the user and the
506 specific time at which the authentication took place.
 - 507 • **Attribute statements:** These contain specific identifying attributes about the subject (for
508 example, that user “John Doe” has “Gold” card status).
 - 509 • **Authorization decision statements:** These define something that the subject is entitled to do
510 (for example, whether “John Doe” is permitted to buy a specified item).
- 511 • **Protocols:** SAML defines a number of generalized request/response protocols:
- 512 • **Authentication Request Protocol:** Defines a means by which a principal (or an agent acting on
513 behalf of the principal) can request assertions containing authentication statements and,
514 optionally, attribute statements. The Web Browser SSO Profile uses this protocol when
515 redirecting a user from an SP to an IdP when it needs to obtain an assertion in order to establish
516 a security context for the user at the SP.
 - 517 • **Single Logout Protocol:** Defines a mechanism to allow near-simultaneous logout of active
518 sessions associated with a principal. The logout can be directly initiated by the user, or initiated
519 by an IdP or SP because of a session timeout, administrator command, etc.
 - 520 • **Assertion Query and Request Protocol:** Defines a set of queries by which SAML assertions
521 may be obtained. The *Request* form of this protocol can ask an asserting party for an existing
522 assertion by referring to its assertion ID. The *Query* form of this protocol defines how a relying
523 party can ask for assertions (new or existing) on the basis of a specific subject and the desired
524 statement type.
 - 525 • **Artifact Resolution Protocol:** Provides a mechanism by which SAML protocol messages may
526 be passed by reference using a small, fixed-length value called an *artifact*. The artifact receiver
527 uses the Artifact Resolution Protocol to ask the message creator to dereference the artifact and
528 return the actual protocol message. The artifact is typically passed to a message recipient using
529 one SAML binding (e.g. HTTP Redirect) while the resolution request and response take place
530 over a synchronous binding, such as SOAP.
 - 531 • **Name Identifier Management Protocol:** Provides mechanisms to change the value or format
532 of the name identifier used to refer to a principal. The issuer of the request can be either the
533 service provider or the identity provider. The protocol also provides a mechanism to terminate an
534 association of a name identifier between an identity provider and service provider.
 - 535 • **Name Identifier Mapping Protocol:** Provides a mechanism to programmatically map one
536 SAML name identifier into another, subject to appropriate policy controls. It permits, for example,
537 one SP to request from an IdP an identifier for a user that the SP can use at another SP in an
538 application integration scenario.
- 539 • **Bindings:** SAML bindings detail exactly how the various SAML protocol messages can be carried
540 over underlying transport protocols. The bindings defined by SAML V2.0 are:
- 541 • **HTTP Redirect Binding:** Defines how SAML protocol messages can be transported using
542 HTTP redirect messages (302 status code responses).
 - 543 • **HTTP POST Binding:** Defines how SAML protocol messages can be transported within the
544 base64-encoded content of an HTML form control.
 - 545 • **HTTP Artifact Binding:** Defines how an artifact (described above in the Artifact Resolution
546 Protocol) is transported from a message sender to a message receiver using HTTP. Two
547 mechanisms are provided: either an HTML form control or a query string in the URL.

- 548
- 549
- **SAML SOAP Binding:** Defines how SAML protocol messages are transported within SOAP 1.1 messages, with details about using SOAP over HTTP.
- 550
- 551
- 552
- **Reverse SOAP (PAOS) Binding:** Defines a multi-stage SOAP/HTTP message exchange that permits an HTTP client to be a SOAP responder. Used in the Enhanced Client and Proxy Profile and particularly designed to support WAP gateways.
- 553
- 554
- **SAML URI Binding:** Defines a means for retrieving an existing SAML assertion by resolving a URI (uniform resource identifier).
- 555
- 556
- 557
- **Profiles:** SAML profiles define how the SAML assertions, protocols, and bindings are combined and constrained to provide greater interoperability in particular usage scenarios. Some of these profiles are examined in detail later in this document. The profiles defined by SAML V2.0 are:
- 558
- **Web Browser SSO Profile:** Defines how SAML entities use the Authentication Request Protocol and SAML Response messages and assertions to achieve single sign-on with standard web browsers. It defines how the messages are used in combination with the HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
- 559
- 560
- 561
- **Enhanced Client and Proxy (ECP) Profile:** Defines a specialized SSO profile where specialized clients or gateway proxies can use the Reverse-SOAP (PAOS) and SOAP bindings.
- 562
- **Identity Provider Discovery Profile:** Defines one possible mechanism for service providers to learn about the identity providers that a user has previously visited.
- 564
- 565
- **Single Logout Profile:** Defines how the SAML Single Logout Protocol can be used with SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
- 566
- 567
- **Assertion Query/Request Profile:** Defines how SAML entities can use the SAML Query and Request Protocol to obtain SAML assertions over a synchronous binding, such as SOAP.
- 568
- **Artifact Resolution Profile:** Defines how SAML entities can use the Artifact Resolution Protocol over a synchronous binding, such as SOAP, to obtain the protocol message referred to by an artifact.
- 570
- 571
- 572
- **Name Identifier Management Profile:** Defines how the Name Identifier Management Protocol may be used with SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
- 573
- **Name Identifier Mapping Profile:** Defines how the Name Identifier Mapping Protocol uses a synchronous binding such as SOAP.
- 574
- 575
- 576

577 **4.4 SAML XML Constructs and Examples**

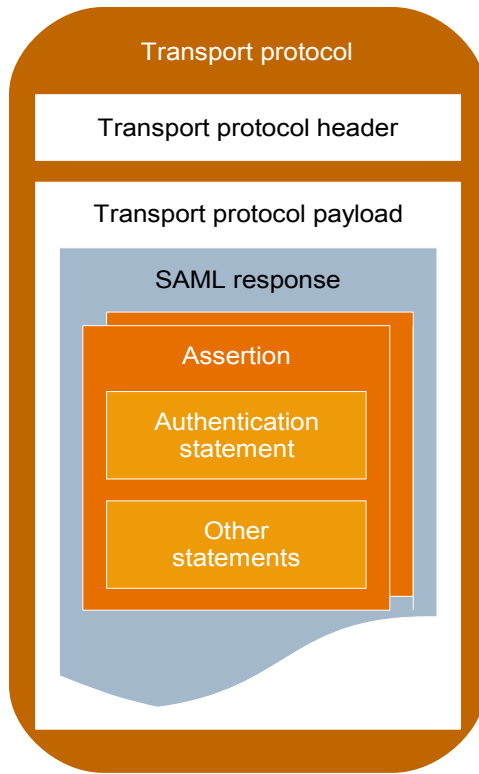
578 This section provides descriptions and examples of some of the key SAML XML constructs.

579 **4.4.1 Relationship of SAML Components**

580 An assertion contains one or more statements and some common information that applies to all contained
581 statements or to the assertion as a whole. A SAML assertion is typically carried between parties in a
582 SAML protocol response message, which itself must be transmitted using some sort of transport or
583 messaging protocol.

584 Figure 5 shows a typical example of containment: a SAML assertion containing a series of statements, the
585 whole being contained within a SAML response, which itself is carried by some kind of protocol.

586



SAML-component-nesting

Figure 5: Relationship of SAML Components

587 **4.4.2 Assertion, Subject, and Statement Structure**

```

1: <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
2:   Version="2.0"
3:   IssueInstant="2005-01-31T12:00:00Z">
4:   <saml:Issuer Format=urn:oasis:names:SAML:2.0:nameid-format:entity>
5:     http://idp.example.org
6:   </saml:Issuer>
7:   <saml:Subject>
8:     <saml:NameID
9:       Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
10:      j.doe@example.com
11:     </saml:NameID>
12:   </saml:Subject>
13:   <saml:Conditions
14:     NotBefore="2005-01-31T12:00:00Z"
15:     NotOnOrAfter="2005-01-31T12:10:00Z">
16:   </saml:Conditions>
17:   <saml:AuthnStatement
18:     AuthnInstant="2005-01-31T12:00:00Z" SessionIndex="6777527772">
19:     <saml:AuthnContext>
20:       <saml:AuthnContextClassRef>
21:         urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
22:       </saml:AuthnContextClassRef>
23:     </saml:AuthnContext>
24:   </saml:AuthnStatement>
25: </saml:Assertion>

```

Figure 6: Assertion with Subject, Conditions, and Authentication Statement

588

589 Figure shows an XML fragment containing an example assertion with a single authentication statement.
590 Note that the XML text in the figure (and elsewhere in this document) has been formatted for presentation
591 purposes. Specifically, while line breaks and extra spaces are ignored between XML attributes within an
592 XML element tag, when they appear between XML element start/end tags, they technically become part of
593 the element value. They are inserted in the example only for readability.

- 594 • Line 1 begins the assertion and contains the declaration of the SAML assertion namespace, which is
595 conventionally represented in the specifications with the `saml:` prefix.
- 596 • Lines 2 through 6 provide information about the nature of the assertion: which version of SAML is
597 being used, when the assertion was created, and who issued it.
- 597 • Lines 7 through 12 provide information about the subject of the assertion, to which all of the
598 contained statements apply. The subject has a name identifier (line 10) whose value is
599 "j.doe@example.com", provided in the format described on line 9 (email address). SAML defines
600 various name identifier formats, and you can also define your own.
- 601 • The assertion as a whole has a validity period indicated by lines 14 and 15. Additional conditions on
602 the use of the assertion can be provided inside this element; SAML predefines some and you can
603 define your own. Timestamps in SAML use the XML Schema **dateTime** data type.
- 604 • The authentication statement appearing on lines 17 through 24 shows that this subject was originally
605 authenticated using a password-protected transport mechanism (e.g. entering a username and
606 password submitted over an SSL-protected browser session) at the time and date shown. SAML
607 predefines numerous authentication context mechanisms (called classes), and you can also define
608 your own mechanisms.

609 The `<NameID>` element within a `<Subject>` offers the ability to provide name identifiers in a number of
610 different formats. SAML's predefined formats include:

- 611 • Email address
- 612 • X.509 subject name
- 613 • Windows domain qualified name
- 614 • Kerberos principal name
- 615 • Entity identifier
- 616 • Persistent identifier
- 617 • Transient identifier

618 Of these, persistent and transient name identifiers utilize privacy-preserving pseudonyms to represent the
619 principal. **Persistent identifiers** provide a permanent privacy-preserving federation since they remain
620 associated with the local identities until they are explicitly removed. **Transient identifiers** support
621 "anonymity" at an SP since they correspond to a "one-time use" identifier created at the IdP. They are not
622 associated with a specific local user identity at the SP and are destroyed once the user session
623 terminates.

624 When persistent identifiers are created by an IdP, they are usually established for use only with a single
625 SP. That is, an SP will only know about the persistent identifier that the IdP created for a principal for use
626 when visiting that SP. The SP does not know about identifiers for the same principal that the IdP may
627 have created for the user at other service providers. SAML does, however, also provide support for the
628 concept of an **affiliation** of service providers which can share a single persistent identifier to identify a
629 principal. This provides a means for one SP to directly utilize services of another SP in the affiliation on
630 behalf of the principal. Without an affiliation, service providers must rely on the Name Identifier Mapping
631 protocol and always interact with the IdP to obtain an identifier that can be used at some other specific SP.

632 4.4.3 Attribute Statement Structure

633 Attribute information about a principal is often provided as an adjunct to authentication information in
634 single sign-on or can be returned in response to attribute queries from a relying party. SAML's attribute
635 structure does not presume that any particular type of data store or data types are being used for the
636 attributes; it has an attribute type-agnostic structure.

634 Figure 7 shows an XML fragment containing an example attribute statement.

635

- 636 •

```

1: <saml:AttributeStatement>
2:   <saml:Attribute
3:     xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
4:     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
5:     Name="urn:oid:2.5.4.42"
6:     FriendlyName="givenName">
7:     <saml:AttributeValue xsi:type="xs:string"
8:       x500:Encoding="LDAP">John</saml:AttributeValue>
9:   </saml:Attribute>
10:  <saml:Attribute
11:    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
12:    Name="LastName">
13:    <saml:AttributeValue
14:      xsi:type="xs:string">Doe</saml:AttributeValue>
15:  </saml:Attribute>
16:  <saml:Attribute
17:    NameFormat="http://smithco.com/attr-formats"
18:    Name="CreditLimit">
19:    xmlns:smithco="http://www.smithco.com/smithco-schema.xsd"
20:    <saml:AttributeValue xsi:type="smithco:type">
21:      <smithco:amount currency="USD">500.00</smithco:amount>
22:    </saml:AttributeValue>
23:  </saml:Attribute>
24: </saml:AttributeStatement>

```

Figure 7: Attribute Statement

637

638

639 Note the following:

- 640 • A single statement can contain multiple attributes. In this example, there are three attributes (starting
641 on lines 2, 10, and 16) within the statement.
- 642 • Attribute names are qualified with a name format (lines 4, 11, and 17) which indicates how the
643 attribute name is to be interpreted. This example takes advantage of two of the SAML-defined
644 **attribute profiles** and defines a third custom attribute as well. The first attribute uses the SAML **X.**
645 **500/LDAP Attribute Profile** to define a value for the LDAP attribute identified by the OID "2.5.4.42".
646 This attribute in an LDAP directory has a friendly name of "givenName" and the attribute's value is
647 "John". The second attribute utilizes the SAML **Basic Attribute Profile**, refers to an attribute named
648 "LastName" which has the value "Doe". The name format of the third attribute indicates the name is
649 not of a format defined by SAML, but is rather defined by a third party, SmithCo. Note that the use of
650 private formats and attribute profiles can create significant interoperability issues. See the SAML
Profiles specification for more information and examples.
- 642 • The value of an attribute can be defined by simple data types, as on lines 7 and 14, or can be
643 structured XML, as on lines 20 through 22.

643 4.4.4 Message Structure and the SOAP Binding

644 In environments where communicating SAML parties are SOAP-enabled, the SOAP-over-HTTP binding
645 can be used to exchange SAML request/response protocol messages. Figure 8 shows the structure of a
646 SAML response message being carried within the SOAP body of a SOAP envelope, which itself has an
647 HTTP response wrapper. Note that SAML itself does not make use of the SOAP header of a SOAP
648 envelope but it does not prevent SAML-based application environments from doing so if needed.

645

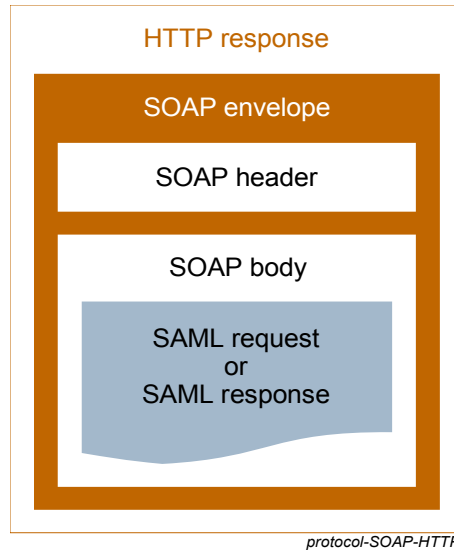


Figure 8: Protocol Messages Carried by SOAP Over HTTP

646 Figure 9 shows an XML document containing an example SAML authentication request message being
 647 transported within a SOAP envelope.

```

1: <?xml version="1.0" encoding="UTF-8"?>
2: <env:Envelope
3:   xmlns:env="http://www.w3.org/2003/05/soap/envelope/">
4:   <env:Body>
5:     <samlp:AuthnRequest
6:       xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
7:       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
8:       Version="2.0"
9:       ID="f0485a7ce95939c093e3de7b2e2984c0"
10:      IssueInstant="2005-01-31T12:00:00Z"
11:      Destination="https://idp.example.org/IdP/" >
12:      AssertionConsumerServiceIndex="1"
13:      AttributeConsumingServiceIndex="0" >
14:      <saml:Issuer>http://sp.example.com</saml:Issuer>
15:      <samlp:RequestedAuthnContext>
16:        <saml:AuthnContextClassRef>
17:          urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
18:        </saml:AuthnContextClassRef>
19:        <samlp:NameIDPolicy
20:          Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"
21:        </samlp:NameIDPolicy>
22:      </samlp:RequestedAuthnContext>
23:    </env:Body>
24:  </env:Envelope>
  
```

Figure 9: Authentication Request in SOAP Envelope

647
 648
 649

650 Note the following:

- 651 • The SOAP envelope starts at line 2.
- 652 • The SAML authentication request starting on line 5 is embedded in a SOAP body element starting
 653 on line 4.
- 654 • The authentication request contains, from lines 6 through 13, various required and optional XML
 655 attributes including declarations of the SAML V2.0 assertion and protocol namespaces, the
 656 message ID, and the index of an assertion consumer service at the SP at which the IdP should
 return the response message.

- 654 • The request specifies a number of optional elements, from lines 15 through 21, that govern the type
655 of assertion the requester expects back. This includes, for example, the requested type of name
656 identifier (email address) and the authentication method with which the user must authenticate at the
657 IdP (username/password over a protected transport).

655 An example XML fragment containing a SAML protocol Response message being transported in a SOAP
656 message is shown in Figure 10.

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
3:   <env:Body>
4:     <samlp:Response
5:       xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
6:       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
7:       Version="2.0"
8:       ID="i92f8b5230dc04d73e93095719d191915fdc67d5e"
9:       IssueInstant="2005-11-10T06:47:42.000Z"
10:      InResponseTo="f0485a7ce95939c093e3de7b2e2984c0">
11:       <saml:Issuer>http://idp.example.org</saml:Issuer>
12:       <samlp:Status>
13:         <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
14:       </samlp:Status>
15:       ...SAML assertion...
16:     </samlp:Response>
17:   </env:Body>
18: </env:Envelope>
```

Figure 10: Response in SOAP Envelope

656
657 Note the following:

- 658 • On line 10, the Response header `InResponseTo` XML attribute references the request to which the
659 asserting party is responding, and specifies additional information (lines 7 through 14) needed to
660 process the response, including status information. SAML defines a number of status codes and, in
661 many cases, dictates the circumstances under which they must be used.
- 659 • Within the response (line 15; detail elided) is a SAML assertion, typically containing one or more
660 statements as discussed earlier.

660 4.5 Privacy in SAML

661 In an information technology context, privacy generally refers to both a user's ability to control how their
662 identity data is shared and used, and to mechanisms that inhibit their actions at multiple service providers
663 from being inappropriately correlated.

662 SAML is often deployed in scenarios where such privacy requirements must be accounted for (as it is also
663 often deployed in scenarios where such privacy need not be explicitly addressed, the assumption being
664 that appropriate protections are enabled through other means and/or layers).

663 SAML has a number of mechanisms that support deployment in privacy .

- 664 • SAML supports the establishment of pseudonyms established between an identity provider and a
665 service provider. Such pseudonyms do not themselves enable inappropriate correlation between
666 service providers (as would be possible if the identity provider asserted the same identifier for a
667 user to every service provider, a so-called *global* identifier).
- 665 • SAML supports *one-time* or transient identifiers – such identifiers ensure that every time a certain
666 user accesses a given service provider through a single sign-on operation from an identity
667 provider, that service provider will be unable to recognize them as the same individual as might
668 have previously visited (based solely on the identifier, correlation may be possible through non-
669 SAML handles).
- 666 • SAML's Authentication Context mechanisms allow a user to be authenticated at a sufficient (but
667 not more than necessary) assurance level, appropriate to the resource they may be attempting to
668 access at some service provider.

- 667 • SAML allows the claimed fact of a user consenting to certain operations (e.g. the act of
668 federation) to be expressed between providers. How, when or where such consent is obtained is
669 out of scope for SAML.

668 **4.6 Security in SAML**

669 Just providing assertions from an asserting party to a relying party may not be adequate to ensure a
670 secure system. How does the relying party trust what is being asserted to it? In addition, what prevents a
671 “man-in-the-middle” attack that might grab assertions to be illicitly “replayed” at a later date? These and
672 many more security considerations are discussed in detail in the SAML Security and Privacy
673 Considerations specification .

670 SAML defines a number of security mechanisms to detect and protect against such attacks. The primary
671 mechanism is for the relying party and asserting party to have a pre-existing trust relationship which
672 typically relies on a Public Key Infrastructure (PKI). While use of a PKI is not mandated by SAML, it is
673 recommended.

671 Use of particular security mechanisms are described for each SAML binding. A general overview of what
672 is recommended is provided below:

- 672 • Where message integrity and message confidentiality are required, then HTTP over SSL 3.0 or TLS
673 1.0 is recommended.
- 673 • When a relying party requests an assertion from an asserting party, bi-lateral authentication is
674 required and the use of SSL 3.0 or TLS 1.0 using mutual authentication or authentication via digital
675 signatures is recommended.
- 674 • When a response message containing an assertion is delivered to a relying party via a user's web
675 browser (for example using the HTTP POST binding), then to ensure message integrity, it is
676 mandated that the response message be digitally signed using XML signature .

675 **5 Major Profiles and Federation Use Cases**

676 As mentioned earlier, SAML defines a number of profiles to describe and constrain the use of SAML
677 protocol messages and assertions to solve specific business use cases. This section provides greater
678 detail on some of the most important SAML profiles and identity federation use cases.

677 **5.1 Web Browser SSO Profile**

678 This section describes the typical flows likely to be used with the web browser SSO profile of SAML V2.0.

679 **5.1.1 Introduction**

680 The Web Browser SSO Profile defines how to use SAML messages and bindings to support the web SSO
681 use case described in section 3.2. This profile provides a wide variety of options, primarily having to do
682 with two dimensions of choice: first whether the message flows are IdP-initiated or SP-initiated, and
683 second, which bindings are used to deliver messages between the IdP and the SP.

681 The first choice has to do with where the user starts the process of a web SSO exchange. SAML supports
682 two general message flows to support the processes. The most common scenario for starting a web SSO
683 exchange is the SP-initiated web SSO model which begins with the user choosing a browser bookmark or
684 clicking a link that takes them directly to an SP application resource they need to access. However, since
685 the user is not logged in at the SP, before it allows access to the resource, the SP sends the user to an
686 IdP to authenticate. The IdP builds an assertion representing the user's authentication at the IdP and then
687 sends the user back to the SP with the assertion. The SP processes the assertion and determines
688 whether to grant the user access to the resource.

682 In an IdP-initiated scenario, the user is visiting an IdP where they are already authenticated and they click
683 on a link to a partner SP. The IdP builds an assertion representing the user's authentication state at the
684 IdP and sends the user's browser over to the SP's assertion consumer service, which processes the
685 assertion and creates a local security context for the user at the SP. This approach is useful in certain
686 environments, but requires the IdP to be configured with inter-site transfer links to the SP's site.
687 Sometimes a binding-specific field called `RelayState` is used to coordinate messages and actions of
688 IdPs and SPs, for example, to allow an IdP (with which SSO was initiated) to indicate the URL of a desired
689 resource when communicating with an SP.

683 Figure compares the IdP-initiated and SP-initiated models.

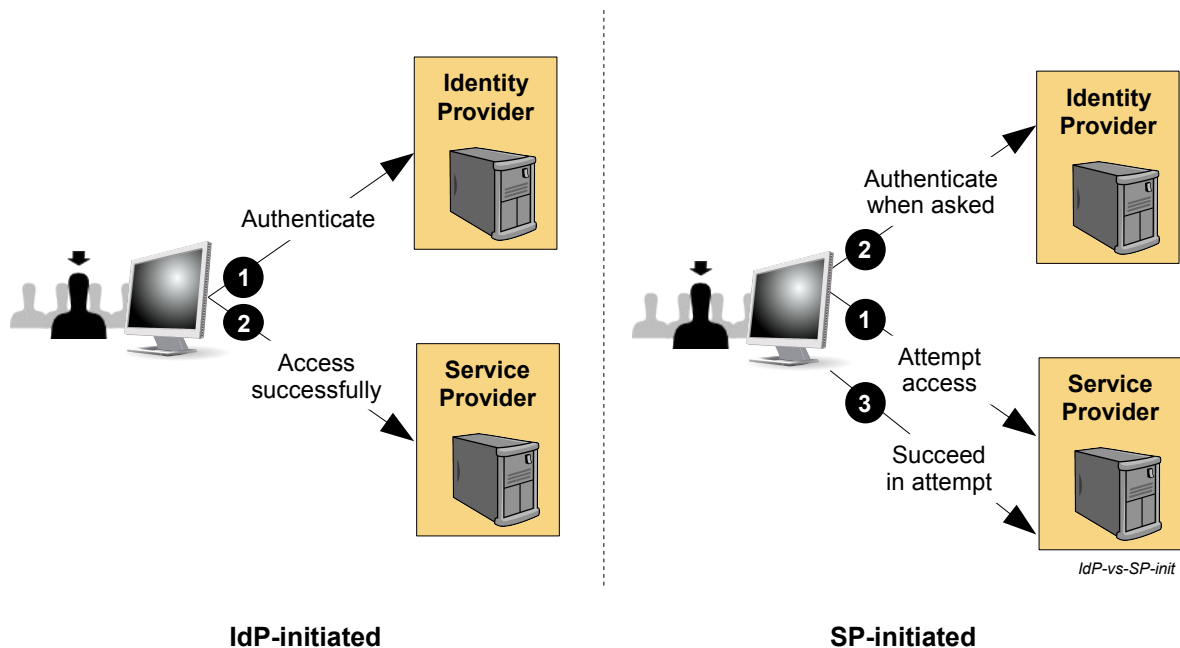


Figure 11: Differences in Initiation of Web Browser SSO

685 The second choice to be made when using the SAML profiles centers around which SAML bindings will be
 686 used when sending messages back and forth between the IdP and SP. There are many combinations of
 687 message flows and bindings that are possible, many of which are discussed in the following subsections.
 688 For the web SSO profile, we are mainly concerned with two SAML messages; namely an Authentication
 689 Request message sent from an SP to an IdP, and a Response message containing a SAML assertion that
 690 is sent from the IdP to the SP (and then, secondarily, with messages related to artifact resolution if that
 691 binding is chosen).

686 The SAML Conformance and Profiles specifications identify the SAML bindings that can legally be used
 687 with these two messages. Specifically, an Authentication Request message can be sent from an SP to an
 688 IdP using either the HTTP Redirect Binding, HTTP POST Binding, or HTTP Artifact Binding. The
 689 Response message can be sent from an IdP to an SP using either the HTTP POST Binding or the HTTP
 690 Artifact Binding. For this pair of messages, SAML permits asymmetry in the choice of bindings used. That
 691 is, a request can be sent using one binding and the response can be returned using a different binding.
 692 The decision of which bindings to use is typically driven by configuration settings at the IdP and SP
 693 systems. Factors such as potential message sizes, whether identity information is allowed to transit
 694 through the browser, etc. must be considered in the choice of bindings.

687 The following subsections describe the detailed message flows involved in web SSO exchanges for the
 688 following use case scenarios:

- 688 • SP-initiated SSO using a Redirect Binding for the SP-to-IdP <AuthnRequest> message and a POST
 689 Binding for the IdP-to-SP <Response> message
- 689 • SP-initiated SSO using a POST Binding for the <AuthnRequest> message and an Artifact Binding for
 690 the <Response> message
- 690 • IDP-initiated SSO using a POST Binding for the IdP-to-SP <Response> message; no SP-to-IdP
 691 <AuthnRequest> message is involved.

691 5.1.2 SP-Initiated SSO: Redirect/POST Bindings

692 This first example describes an SP-initiated SSO exchange. In such an exchange, the user attempts to
 693 access a resource on the SP, sp.example.com. However they do not have a current logon session on this

693 site and their federated identity is managed by their IdP, idp.example.org. They are sent to the IdP to log
 694 on and the IdP provides a SAML web SSO assertion for the user's federated identity back to the SP.

694 For this specific use case, the HTTP Redirect Binding is used to deliver the SAML <AuthnRequest>
 695 message to the IdP and the HTTP POST Binding is used to return the SAML <Response> message
 696 containing the assertion to the SP. Figure 12 illustrates the message flow.

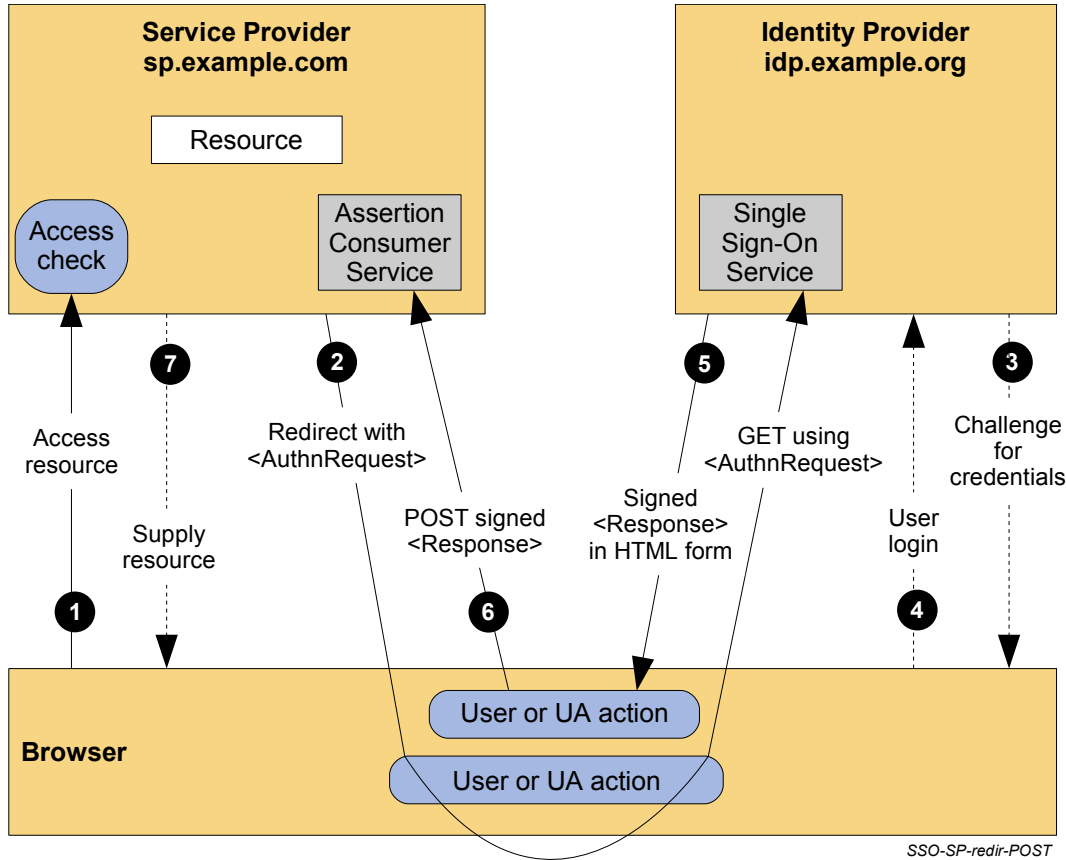


Figure 12: SP-Initiated SSO with Redirect and POST Bindings

696 The processing is as follows:

- 697 1. The user attempts to access a resource on sp.example.com. The user does not have a valid login
 698 session (i.e. security context) on this site. The SP saves the requested resource URL in local state
 699 information that can be saved across the web SSO exchange.
- 698 2. The SP sends an HTTP redirect response to the browser (HTTP status 302 or 303). The Location
 699 HTTP header contains the destination URI of the Sign-On Service at the identity provider together with
 700 an <AuthnRequest> message encoded as a URL query variable named SAMLRequest.

```

699 <saml:AuthnRequest
700   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
701   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
702   ID="identifier_1"
703   Version="2.0"
704   IssueInstant="2004-12-05T09:21:59Z"
705   AssertionConsumerServiceIndex="1">
706   <saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>
707   <samlp:NameIDPolicy
708     AllowCreate="true"
709     Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>
710 </saml:AuthnRequest>
  
```

711 The query string is encoded using the DEFLATE encoding. The browser processes the redirect
 712 response and issues an HTTP GET request to the IdP's Single Sign-On Service with the

712 SAMLRequest query parameter. The local state information (or a reference to it) is also included in the
713 HTTP response encoded in a RelayState query string parameter.

714 `https://idp.example.org/SAML2/SSO/Redirect?SAMLRequest=request&RelayState=token`

- 714 3. The Single Sign-On Service determines whether the user has an existing logon security context at the
715 identity provider that meets the default or requested (in the <AuthnRequest>) authentication policy
716 requirements. If not, the IdP interacts with the browser to challenge the user to provide valid
717 credentials.
- 715 4. The user provides valid credentials and a local logon security context is created for the user at the IdP.
- 716 5. The IdP Single Sign-On Service builds a SAML assertion representing the user's logon security
717 context. Since a POST binding is going to be used, the assertion is digitally signed and then placed
718 within a SAML <Response> message. The <Response> message is then placed within an HTML
719 FORM as a hidden form control named SAMLResponse. If the IdP received a RelayState value
720 from the SP, it must return it unmodified to the SP in a hidden form control named RelayState. The
721 Single Sign-On Service sends the HTML form back to the browser in the HTTP response. For ease of
722 use purposes, the HTML FORM typically will be accompanied by script code that will automatically post
723 the form to the destination site.

```
717 <form method="post" action="https://sp.example.com/SAML2/SSO/POST" ...>  
718   <input type="hidden" name="SAMLResponse" value="response" />  
719   <input type="hidden" name="RelayState" value="token" />  
720   ...  
721   <input type="submit" value="Submit" />  
722 </form>
```

723 The value of the SAMLResponse parameter is the base64 encoding of the following
724 <saml:Response> element:

```
724 <saml:Response  
725   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
726   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
727   ID="identifier_2"  
728   InResponseTo="identifier_1"  
729   Version="2.0"  
730   IssueInstant="2004-12-05T09:22:05Z"  
731   Destination="https://sp.example.com/SAML2/SSO/POST">  
732   <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>  
733   <samlp:Status>  
734     <samlp:StatusCode  
735       Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>  
736   </samlp:Status>  
737   <saml:Assertion  
738     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
739     ID="identifier_3"  
740     Version="2.0"  
741     IssueInstant="2004-12-05T09:22:05Z">  
742     <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>  
743     <!-- a POSTed assertion MUST be signed -->  
744     <ds:Signature  
745       xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>  
746     <saml:Subject>  
747       <saml:NameID  
748         Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">  
749         3f7b3dcf-1674-4ecd-92c8-1544f346baf8  
750       </saml:NameID>  
751       <saml:SubjectConfirmation  
752         Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">  
753         <saml:SubjectConfirmationData  
754           InResponseTo="identifier_1"  
755           Recipient="https://sp.example.com/SAML2/SSO/POST"  
756           NotOnOrAfter="2004-12-05T09:27:05Z"/>  
757         </saml:SubjectConfirmation>  
758       </saml:Subject>  
759       <saml:Conditions  
760         NotBefore="2004-12-05T09:17:05Z"  
761         NotOnOrAfter="2004-12-05T09:27:05Z">  
762         <saml:AudienceRestriction>  
763           <saml:Audience>https://sp.example.com/SAML2</saml:Audience>  
764         </saml:AudienceRestriction>  
765       </saml:Conditions>
```

```
766 <saml:AuthnStatement
767   AuthnInstant="2004-12-05T09:22:00Z"
768   SessionIndex="identifier_3">
769   <saml:AuthnContext>
770     <saml:AuthnContextClassRef>
771       urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
772     </saml:AuthnContextClassRef>
773   </saml:AuthnContext>
774 </saml:AuthnStatement>
775 </saml:Assertion>
776 </samlp:Response>
```

777 6. The browser, due either to a user action or execution of an “auto-submit” script, issues an HTTP POST
778 request to send the form to the SP's Assertion Consumer Service.

```
778 POST /SAML2/SSO/POST HTTP/1.1
779 Host: sp.example.com
780 Content-Type: application/x-www-form-urlencoded
781 Content-Length: nnn
782
783 SAMLResponse=response&RelayState=token
```

784 where the values of the SAMLResponse and RelayState parameters are taken from the HTML
785 form of Step 5.

786 The service provider's Assertion Consumer Service obtains the <Response> message from the
787 HTML FORM for processing. The digital signature on the SAML assertion must first be validated
788 and then the assertion contents are processed in order to create a local logon security context for
789 the user at the SP. Once this completes, the SP retrieves the local state information indicated by
790 the RelayState data to recall the originally-requested resource URL. It then sends an HTTP
791 redirect response to the browser directing it to access the originally requested resource (not
shown).

786 7. An access check is made to establish whether the user has the correct authorization to access the
787 resource. If the access check passes, the resource is then returned to the browser.

787 5.1.3 SP-Initiated SSO: POST/Artifact Bindings

788 This use case again describes an SP-initiated SSO exchange.

789 However, for this use case, the HTTP POST binding is used to deliver the SAML <AuthnRequest> to
790 the IdP and the SAML <Response> message is returned using the Artifact binding. The HTTP POST
791 binding may be necessary for an <AuthnRequest> message in cases where its length precludes the use
792 of the HTTP Redirect binding (which is typical). The message may be long enough to require a POST
793 binding when, for example, it includes many of its optional elements and attributes, or when it must be
794 digitally signed.

790 When using the HTTP Artifact binding for the SAML <Response> message, SAML permits the artifact to
791 be delivered via the browser using either an HTTP POST or HTTP Redirect response (not to be confused
792 with the SAML HTTP POST and Redirect Bindings). In this example, the artifact is delivered using an
793 HTTP redirect.

791 Once the SP is in possession of the artifact, it contacts the IdP's Artifact Resolution Service using the
792 synchronous SOAP binding to obtain the SAML message that corresponds to the artifact. Figure illustrates
793 the message flow.

792

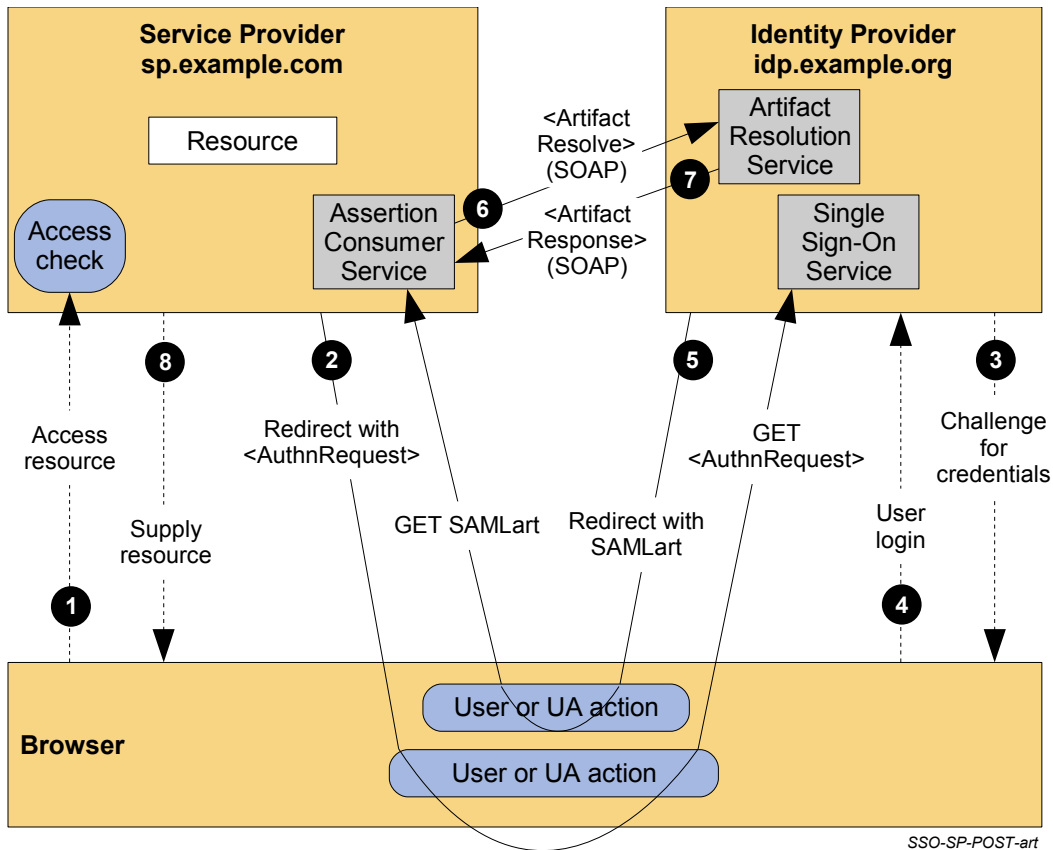


Figure 13: SP-Initiated SSO with Binding

SSO-SP-POST-art

794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810

The processing is as follows:

1. The user attempts to access a resource on sp.example.com. The user does not have a valid logon session (i.e. security context) on this site. The SP saves the requested resource URL in local state information that can be saved across the web SSO exchange.
2. The SP sends an HTML form back to the browser in the HTTP response (HTTP status 200). The HTML FORM contains a SAML <AuthnRequest> message encoded as the value of a hidden form control named SAMLRequest.

```

<form method="post" action="https://idp.example.org/SAML2/SSO/POST" ...>
  <input type="hidden" name="SAMLRequest" value="request" />
  <input type="hidden" name="RelayState" value="token" />
  ...
  <input type="submit" value="Submit" />
</form>

```

811 The RelayState token is an opaque reference to state information maintained at the service
812 provider. (The RelayState mechanism can leak details of the user's activities at the SP to the IdP
813 so care should be taken in its implementation.) The value of the SAMLRequest parameter is the
814 base64 encoding of the following <samlp:AuthnRequest> element:

```
812 <samlp:AuthnRequest
813   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
814   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
815   ID="identifier_1"
816   Version="2.0"
817   IssueInstant="2004-12-05T09:21:59Z"
818   AssertionConsumerServiceIndex="1">
819   <saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>
820   <samlp:NameIDPolicy
821     AllowCreate="true"
822     Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>
823 </samlp:AuthnRequest>
```

824 1. For ease-of-use purposes, the HTML FORM typically will be accompanied by script code that will
825 automatically post the form to the destination site (which is the IdP in this case). The browser, due
826 either to a user action or execution of an "auto-submit" script, issues an HTTP POST request to send
827 the form to the identity provider's Single Sign-On Service.

```
825 POST /SAML2/SSO/POST HTTP/1.1
826 Host: idp.example.org
827 Content-Type: application/x-www-form-urlencoded
828 Content-Length: nnn
829
830 SAMLRequest=request&RelayState=token
```

831 3. The Single Sign-On Service determines whether the user has an existing logon security context at the
832 identity provider that meets the default or requested authentication policy requirements. If not, the IdP
833 interacts with the browser to challenge the user to provide valid credentials.

832 4. The user provides valid credentials and a local logon security context is created for the user at the IdP.

833 5. The IdP Single Sign-On Service issues a SAML assertion representing the user's logon security
834 context and places the assertion within a SAML <Response> message. Since the HTTP Artifact
835 binding will be used to deliver the SAML Response message, it is not mandated that the assertion be
836 digitally signed. The IdP creates an artifact containing the source ID for the idp.example.org site and a
837 reference to the <Response> message (the MessageHandle). The HTTP Artifact binding allows the
838 choice of either HTTP redirection or an HTML form POST as the mechanism to deliver the artifact to
839 the partner. The figure shows the use of redirection.

834 6. The SP's Assertion Consumer Service now sends a SAML <ArtifactResolve> message containing
835 the artifact to the IdP's Artifact Resolution Service endpoint. This exchange is performed using a
836 synchronous SOAP message exchange.

```
835 <samlp:ArtifactResolve
836   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
837   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
838   ID="identifier_2"
839   Version="2.0"
840   IssueInstant="2004-12-05T09:22:04Z"
841   Destination="https://idp.example.org/SAML2/ArtifactResolution">
842   <saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>
843   <!-- an ArtifactResolve message SHOULD be signed -->
844   <ds:Signature
845     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
846   <samlp:Artifact>artifact</samlp:Artifact>
847 </samlp:ArtifactResolve>
```

848 7. The IdP's Artifact Resolution Service extracts the MessageHandle from the artifact and locates the
849 original SAML <Response> message associated with it. This message is then placed inside a SAML
850 <ArtifactResponse> message, which is returned to the SP over the SOAP channel.

```
849
850 <samlp:ArtifactResponse
851   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
852   ID="identifier_3"
853   InResponseTo="identifier_2">
```

```

854 Version="2.0"
855 IssueInstant="2004-12-05T09:22:05Z">
856 <!-- an ArtifactResponse message SHOULD be signed -->
857 <ds:Signature
858   xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
859 <samlp:Status>
860   <samlp:StatusCode
861     Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
862 </samlp:Status>
863 <samlp:Response
864   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
865   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
866   ID="identifier_4"
867   InResponseTo="identifier_1"
868   Version="2.0"
869   IssueInstant="2004-12-05T09:22:05Z"
870   Destination="https://sp.example.com/SAML2/SSO/Artifact">
871 <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
872 <ds:Signature
873   xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
874 <samlp:Status>
875   <samlp:StatusCode
876     Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
877 </samlp:Status>
878 <saml:Assertion
879   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
880   ID="identifier_5"
881   Version="2.0"
882   IssueInstant="2004-12-05T09:22:05Z">
883 <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
884 <!-- a Subject element is required -->
885 <saml:Subject>
886   <saml:NameID
887     Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
888     user@mail.example.org
889   </saml:NameID>
890   <saml:SubjectConfirmation
891     Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
892     <saml:SubjectConfirmationData
893       InResponseTo="identifier_1"
894       Recipient="https://sp.example.com/SAML2/SSO/Artifact"
895       NotOnOrAfter="2004-12-05T09:27:05Z"/>
896   </saml:SubjectConfirmation>
897 </saml:Subject>
898 <saml:Conditions
899   NotBefore="2004-12-05T09:17:05Z"
900   NotOnOrAfter="2004-12-05T09:27:05Z">
901 <saml:AudienceRestriction>
902   <saml:Audience>https://sp.example.com/SAML2</saml:Audience>
903 </saml:AudienceRestriction>
904 </saml:Conditions>
905 <saml:AuthnStatement
906   AuthnInstant="2004-12-05T09:22:00Z"
907   SessionIndex="identifier_5">
908 <saml:AuthnContext>
909   <saml:AuthnContextClassRef>
910     urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
911   </saml:AuthnContextClassRef>
912 </saml:AuthnContext>
913 </saml:AuthnStatement>
914 </saml:Assertion>
915 </samlp:Response>
916 </samlp:ArtifactResponse>

```

917 The SP extracts and processes the `<Response>` message and then processes the embedded
918 assertion in order to create a local logon security context for the user at the SP. Once this is
919 completed, the SP retrieves the local state information indicated by the `RelayState` data to recall the
920 originally-requested resource URL. It then sends an HTTP redirect response to the browser directing it
921 to access the originally requested resource (not shown).

918 8. An access check is made to establish whether the user has the correct authorization to access the
919 resource. If the access check passes, the resource is then returned to the browser.

919 **5.1.4 IdP-Initiated SSO: POST Binding**

920 In addition to supporting the new SP-Initiated web SSO use cases, SAML v2 continues to support the IdP-
 921 initiated web SSO use cases originally supported by SAML v1. In an IdP-initiated use case, the identity
 922 provider is configured with specialized links that refer to the desired service providers. These links actually
 923 refer to the local IdP's Single Sign-On Service and pass parameters to the service identifying the remote
 924 SP. So instead of visiting the SP directly, the user accesses the IdP site and clicks on one of the links to
 925 gain access to the remote SP. This triggers the creation of a SAML assertion that, in this example, will be
 926 transported to the service provider using the HTTP POST binding.

921 Figure 14 shows the process flow for an IdP-initiated web SSO exchange.

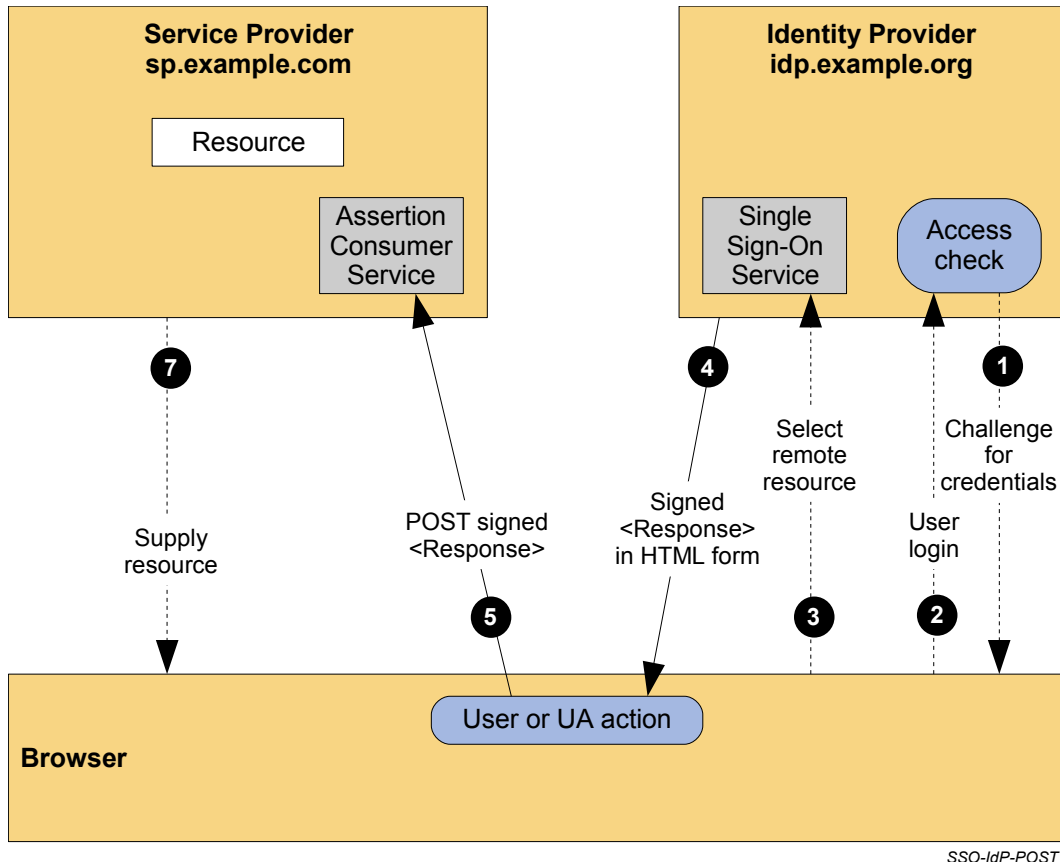


Figure 14: IdP-Initiated SSO with POST Binding

923 The processing is as follows:

- 924 1. If the user does not have a valid local security context at the IdP, at some point the user will be
 925 challenged to supply their credentials to the IdP site, idp.example.org.
- 926 2. The user provides valid credentials and a local logon security context is created for the user at the IdP.
- 927 3. The user selects a menu option or link on the IdP to request access to an SP web site,
 928 sp.example.com. This causes the IdP's Single Sign-On Service to be called.
- 929 4. The Single Sign-On Service builds a SAML assertion representing the user's logon security context.
 930 Since a POST binding is going to be used, the assertion is digitally signed before it is placed within a
 931 SAML <Response> message. The <Response> message is then placed within an HTML FORM as
 932 a hidden form control named SAMLResponse. (If the convention for identifying a specific application
 933 resource at the SP is supported at the IdP and SP, the resource URL at the SP is also encoded into
 the form using a hidden form control named RelayState.) The Single Sign-On Service sends the
 HTML form back to the browser in the HTTP response. For ease-of-use purposes, the HTML FORM

- 928 typically will contain script code that will automatically post the form to the destination site.
- 929 5. The browser, due either to a user action or execution of an “auto-submit” script, issues an HTTP POST
 930 request to send the form to the SP’s Assertion Consumer Service. The service provider’s Assertion
 931 Consumer Service obtains the <Response> message from the HTML FORM for processing. The
 932 digital signature on the SAML assertion must first be validated and then the assertion contents are
 933 processed in order to create a local logon security context for the user at the SP. Once this completes,
 934 the SP retrieves the `RelayState` data (if any) to determine the desired application resource URL and
 935 sends an HTTP redirect response to the browser directing it to access the requested resource (not
 936 shown).
- 930 6. An access check is made to establish whether the user has the correct authorization to access the
 931 resource. If the access check passes, the resource is then returned to the browser.

931 5.2 ECP Profile

932 The browser SSO profile discussed above works with commercial browsers that have no special
 933 capabilities. This section describes a SAML V2.0 profile that takes into account enhanced client devices
 934 and proxy servers.

933 5.2.1 Introduction

934 The Enhanced Client and Proxy (ECP) Profile supports several SSO use cases, in particular:

- 935 • Use of a proxy server, for example a WAP gateway in front of a mobile device which has limited
 936 functionality
- 936 • Clients where it is impossible to use redirects
- 937 • It is impossible for the identity provider and service provider to directly communicate (and hence the
 938 HTTP Artifact binding cannot be used)

938 Figure 15 illustrates two use cases for using the ECP Profile.

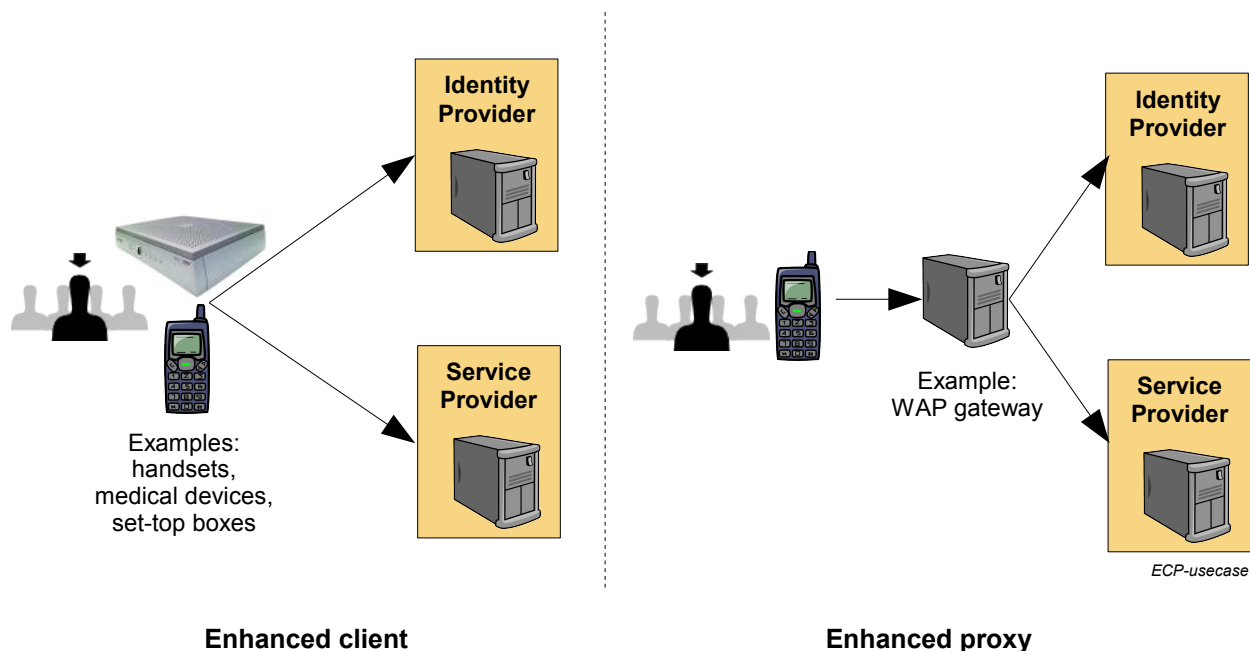


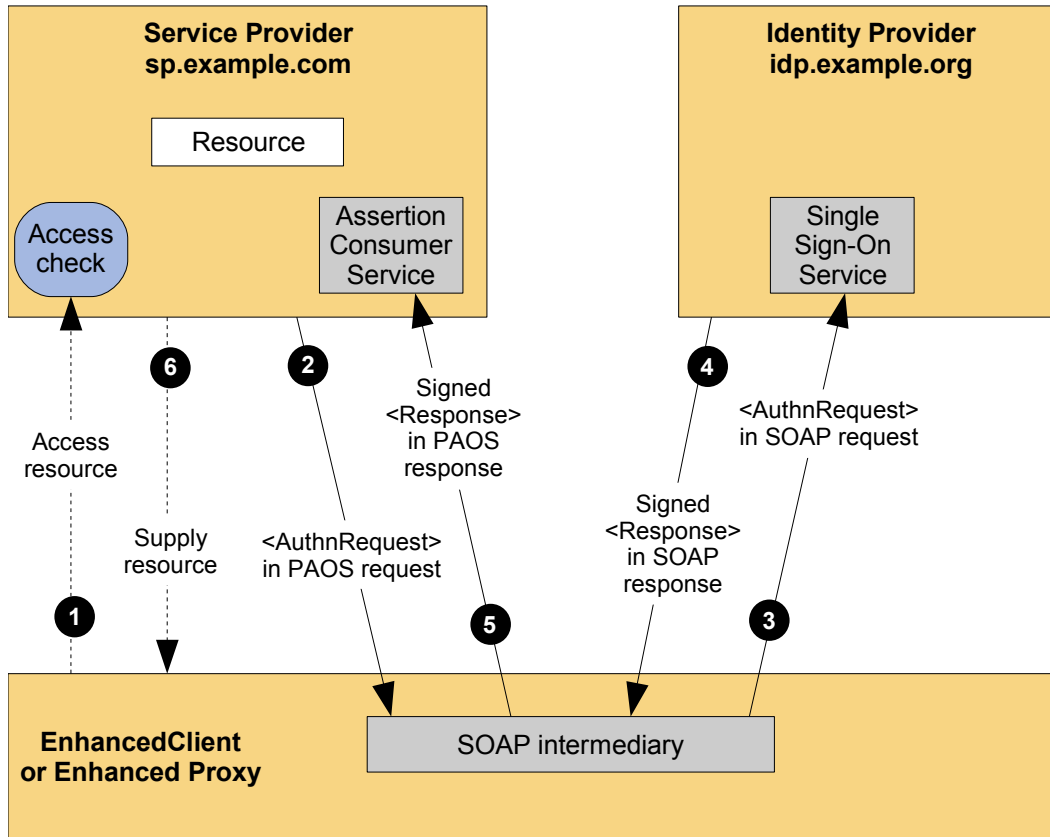
Figure 15: Enhanced Client/Proxy Use Cases

940 The ECP profile defines a single binding – PAOS (Reverse SOAP). The profile uses SOAP headers and
 941 SOAP bodies to transport SAML <AuthnRequest> and SAML <Response> messages between the

941 service provider and the identity provider.

942 5.2.2 ECP Profile Using PAOS Binding

943 Figure 16 shows the message flows between the ECP, service provider and identity provider. The ECP is
944 shown as a single logical entity.



SSO-ECP-PAOS

Figure 16: SSO Using ECP with the PAOS Binding

945 The processing is as follows:

- 946 1. The ECP wishes to gain access to a resource on the service provider, sp.example.com. The ECP will
947 issue an HTTP request for the resource. The HTTP request contains a PAOS HTTP header defining
948 that the ECP service is to be used.
- 947 2. Accessing the resource requires that the principal has a valid security context, and hence a SAML
948 assertion needs to be supplied to the service provider. In the HTTP response to the ECP an
949 <AuthnRequest> is carried within a SOAP body. Additional information, using the PAOS binding, is
950 provided back to the ECP
- 948 3. After some processing in the ECP the <AuthnRequest> is sent to the appropriate identity provider
949 using the SAML SOAP binding.
- 949 4. The identity provider validates the <AuthnRequest> and sends back to the ECP a SAML
950 <Response>, again using the SAML SOAP binding.
- 950 5. The ECP extracts the <Response> and forwards it to the service provider as a PAOS response.
- 951 6. The service provider sends to the ECP an HTTP response containing the resource originally
952 requested.

952 **5.3 Single Logout Profile**

953 Once single sign-on has been achieved, several individual sessions with service providers share a single
954 authentication context. This section discusses SAML's profile for single logout, which allows for reversing
955 the sign-on process with all of these providers at once.

954 One representative flow option is discussed in detail: single logout that is initiated at one SP and results in
955 logout from multiple SPs.

955 **5.3.1 Introduction**

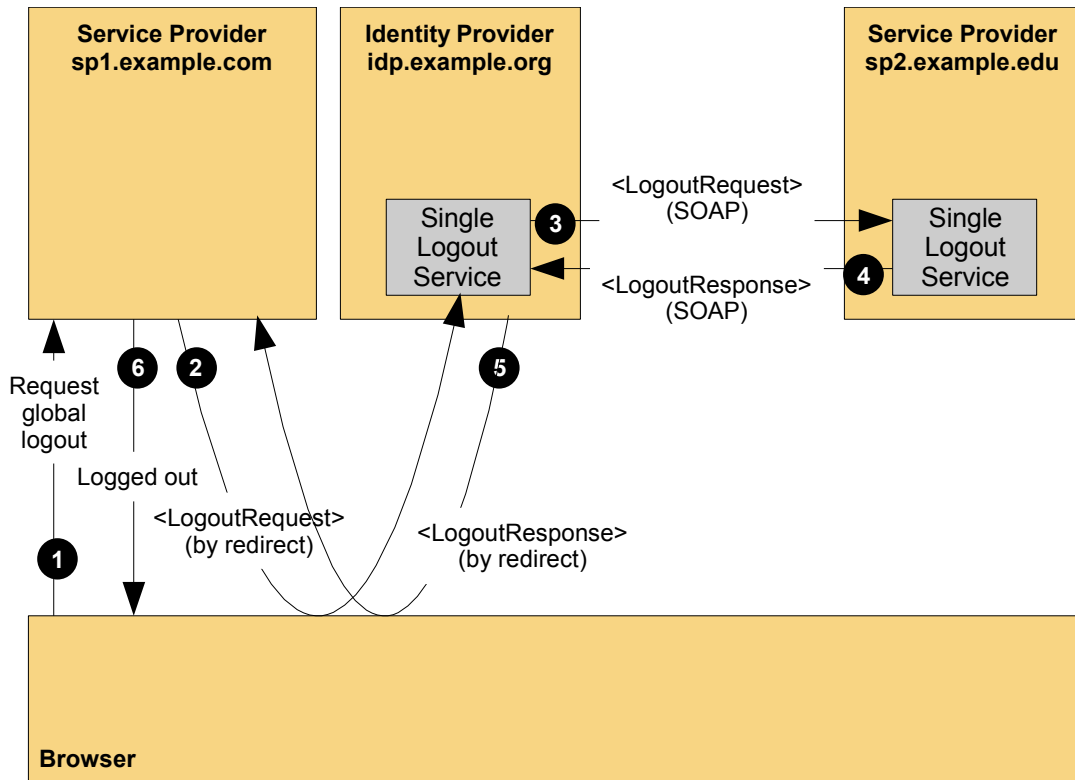
956 Single logout permits near real-time session logout of a user from all participants in a session. A request
957 can be issued by any session participant to request that the session is to be ended. As specified in the
958 SAML Conformance specification, the SAML logout messages can be exchanged over either the
959 synchronous SOAP over HTTP binding or using the asynchronous HTTP Redirect, HTTP POST, or HTTP
960 Artifact bindings. Note that a browser logout operation often requires access to local authentication
961 cookies stored in the user's browser. Thus, asynchronous front-channel bindings are typically preferred for
962 these exchanges in order to force the browser to visit each session participant to permit access to the
963 browser cookies. However, user interaction with the browser might interrupt the process of visiting each
964 participant and thus, the result of the logout process cannot be guaranteed.

957 **5.3.2 SP-Initiated Single Logout with Multiple SPs**

958 In the example shown in Figure 16, a user visiting the sp1.example.com service provider web site decides
959 that they wish to log out of their web SSO session. The identity provider idp.example.org determines that
960 other service providers are also participants in the web SSO session, and thus sends `<LogoutRequest>`
961 messages to each of the other SPs. In this example, different bindings are used for the exchanges
962 between the various pairs of session participants. The SP initiating the single logout uses the HTTP
963 Redirect binding with the IdP, while the IdP uses a back-channel SOAP over HTTP binding to
964 communicate with the other SP sp2.example.edu.

959

960



SLO-SP-init-mult

Figure 17: SP-initiated Single Logout with Multiple SPs

961 The processing is as follows:

- 962 1. A user was previously authenticated by the idp.example.org identity provider and is interacting with the
 963 sp1.example.com service provider through a web SSO session. The user decides to terminate their
 964 session and selects a link on the SP that requests a global logout.
- 963 2. The SP sp1.example.com destroys the local authentication session state for the user and then sends
 964 the idp.example.org identity provider a SAML `<LogoutRequest>` message requesting that the user's
 965 session be logged out. The request identifies the principal to be logged out using a `<NameID>`
 966 element, as well as providing a `<SessionIndex>` element to uniquely identify the session being
 967 closed. The `<LogoutRequest>` message is digitally signed and then transmitted using the HTTP
 968 Redirect binding. The identity provider verifies that the `<LogoutRequest>` originated from a known
 969 and trusted service provider. The identity provider processes the request and destroys any local
 970 session information for the user.
- 964 3. Having determined that other service providers are also participants in the web SSO session, the
 965 identity provider similar sends a `<LogoutRequest>` message to those providers. In this example,
 966 there is one other service provider, sp2.example.edu. The `<LogoutRequest>` message is sent using
 967 the SOAP over HTTP Binding.
- 965 4. The service provider sp2.example.edu returns a `<LogoutResponse>` message containing a suitable
 966 status code response to the identity provider. The response is digitally signed and returned (in this
 967 case) using the SOAP over HTTP binding.
- 966 5. The identity provider returns a `<LogoutResponse>` message containing a suitable status code
 967 response to the original requesting service provider, sp1.example.com. The response is digitally
 968 signed and returned (in this case) using the HTTP Redirect binding.
- 967 6. Finally, the service provider sp1.example.com informs the user that they are logged out of all the
 968 providers.

968 **5.4 Establishing and Managing Federated Identities**

969 Thus far, the use case examples that have been presented have focused on the SAML message
970 exchanges required to facilitate the implementation of web single sign-on solutions. This section
971 examines issues surrounding how these message exchanges are tied to individual local and federated
972 user identities shared between participants in the solution.

970 **5.4.1 Introduction**

971 The following sections describe mechanisms supported by SAML for establishing and managing federated
972 identities. The following use cases are described:

- 972 • **Federation via Out-of-Band Account Linking:** The establishment of federated identities for users
973 and the association of those identities to local user identities can be performed without the use of
974 SAML protocols and assertions. This was the only style of federation supported by SAML V1 and is
975 still supported in SAML v2.0.
- 973 • **Federation via Persistent Pseudonym Identifiers:** An identity provider federates the user's local
974 identity principal with the principal's identity at the service provider using a persistent SAML name
975 identifier.
- 974 • **Federation via Transient Pseudonym Identifiers:** A temporary identifier is used to federate
975 between the IdP and the SP for the life of the user's web SSO session.
- 975 • **Federation via Identity Attributes:** Attributes of the principal, as defined by the identity provider,
976 are used to link to the account used at the service provider.
- 976 • **Federation Termination:** termination of an existing federation.

977 To simplify the examples, not all possible SAML bindings are illustrated.

978 All the examples are based on the use case scenarios originally defined in Section 3.2, with
979 airline.example.com being the identity provider.

979 **5.4.2 Federation Using Out-of-Band Account Linking**

980 In this example, shown in Figure 18, the user John has accounts on both airline.example.com and
981 cars.example.co.uk each using the same local user ID (**john**). The identity data stores at both sites are
982 synchronized by some out-of-band means, for example using database synchronization or off-line batch
983 updates.

981

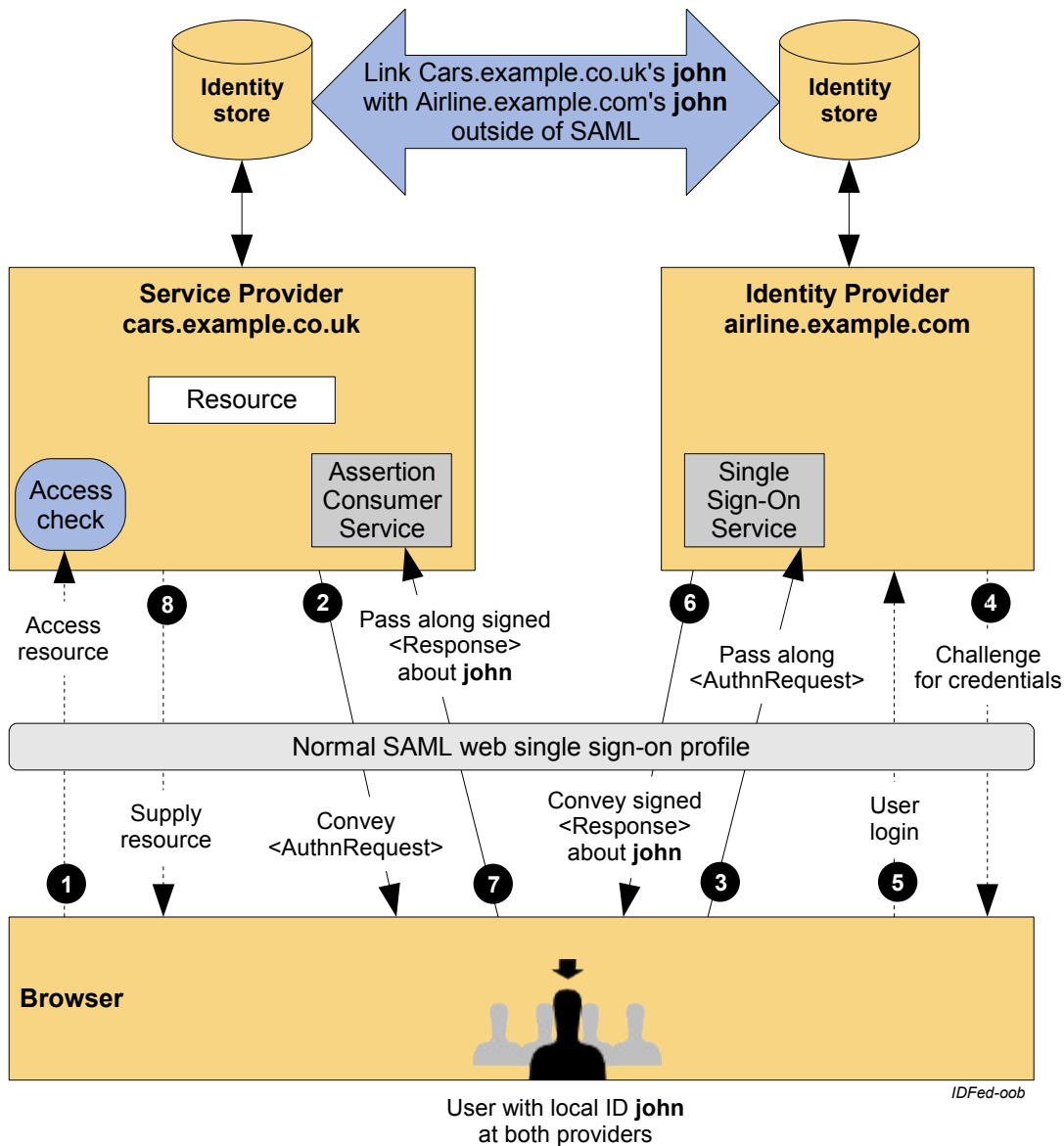


Figure 18: Identity Federation with Out-of-Band Account Linking

982 The processing is as follows:

- 983 1. The user is challenged to supply their credentials to the site airline.example.com.
- 984 2. The user successfully provides their credentials and has a security context with the
- 985 airline.example.com identity provider.
- 985 3. The user selects a menu option (or function) on the airline.example.com application that means the
- 986 user wants to access a resource or application on cars.example.co.uk. The airline.example.com
- 987 identity provider sends a HTML form back to the browser. The HTML FORM contains a SAML
- 988 response, within which is a SAML assertion about user john.
- 986 4. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing the
- 987 SAML response to be sent to the cars.example.co.uk Service provider.

987 The cars.example.co.uk service provider's Assertion Consumer Service validates the digital signature on

988 the SAML Response. If this, and the assertion validate correctly it creates a local session for user john,

989 based on the local john account. It then sends an HTTP redirect to the browser causing it to access the

990 TARGET resource, with a cookie that identifies the local session. An access check is then made to

988 establish whether the user john has the correct authorization to access the cars.example.co.uk web site
 989 and the TARGET resource. The TARGET resource is then returned to the browser.

989 5.4.3 Federation Using Persistent Pseudonym Identifiers

990 In this use case scenario, the partner sites take advantage of SAML V2.0's ability to dynamically establish
 991 a federated identity for a user as part of the web SSO message exchange. SAML V2.0 provides the
 992 NameIDPolicy element on the AuthnRequest to allow the SP to constrain such dynamic behaviour. The
 993 user **jd**oe on *cars.example.co.uk* wishes to federate this account with his **john** account on the IdP,
 994 *airline.example.com*. Figure 19 illustrates dynamic identity federation using persistent pseudonym
 995 identifiers in an SP-initiated web SSO exchange.

991

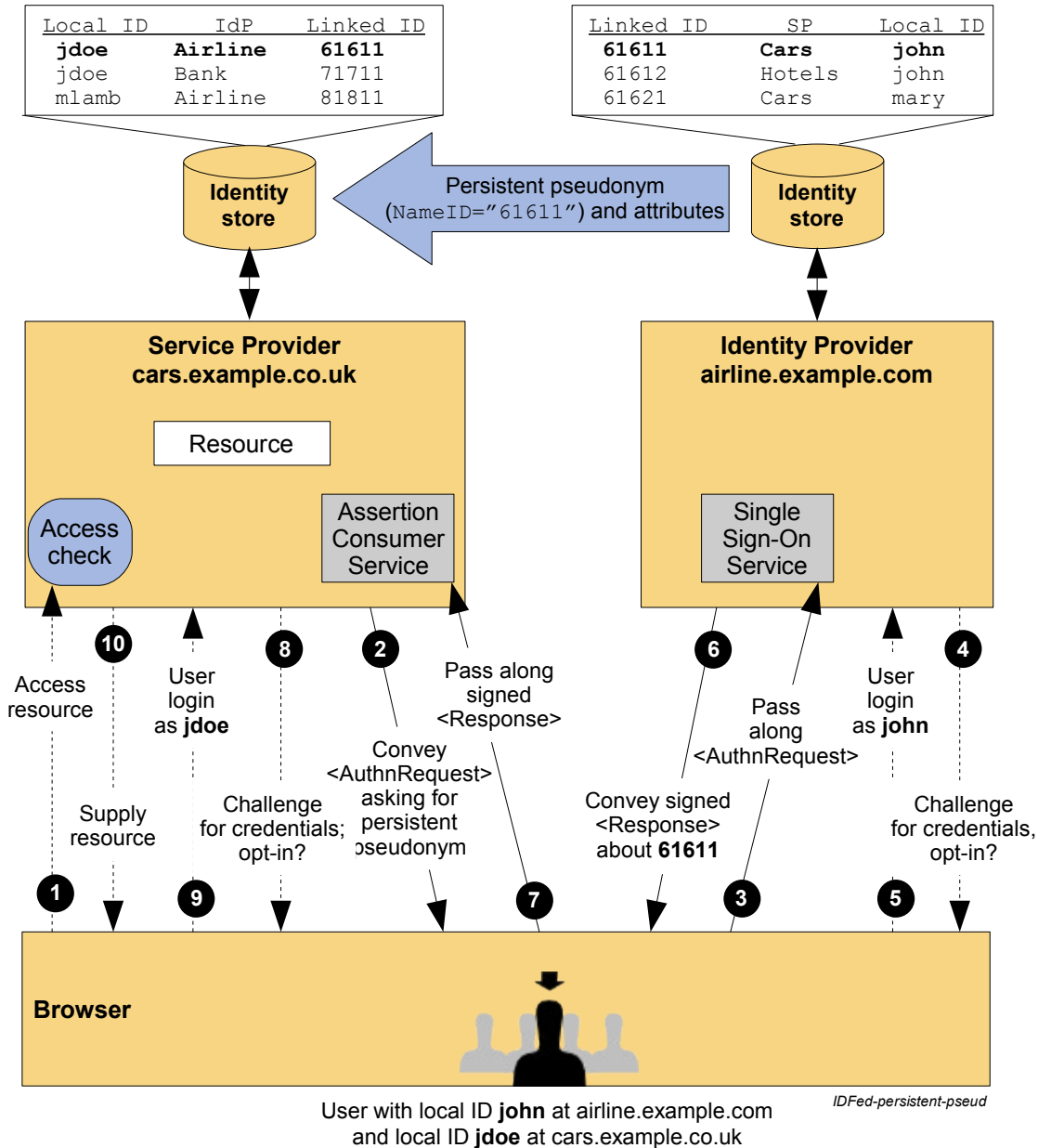


Figure 19: SP-Initiated Identity Federation with Persistent Pseudonym

993 The processing is as follows:

- 994 1. The user attempts to access a resource on cars.example.co.uk. The user does not have any current
995 logon session (i.e. security context) on this site, and is unknown to it. The resource that the user
996 attempted to access is saved as `RelayState` information.
- 995 2. The service provider uses the HTTP Redirect Binding to send the user to the Single Sign-On Service at
996 the identity provider (airline.example.com). The HTTP redirect includes a SAML `<AuthnRequest>`
997 message requesting that the identity provider provide an assertion using a persistent name identifier
998 for the user. As the service provider desires the IdP have the flexibility to generate a new identifier for
999 the user should one not already exist, the SP sets the `AllowCreate` attribute on the `NameIDPolicy`
1000 element to 'true'.
- 996 3. The user will be challenged to provide valid credentials.
- 997 4. The user provides valid credentials identifying himself as **john** and a local security context is created
998 for the user at the IdP.
- 998 5. The Single Sign-On Service looks up user **john** in its identity store and, seeing that the `AllowCreate`
999 attribute allows it to, creates a persistent name identifier (`61611`) to be used for the session at the
1000 service provider. It then builds a signed SAML web SSO assertion where the subject uses a transient
1001 name identifier format. The name **john** is not contained anywhere in the assertion. Note that
1002 depending on the partner agreements, the assertion might also contain an attribute statement
1003 describing identity attributes about the user (e.g. their membership level).
- 999 6. The browser, due either to a user action or execution of an "auto-submit" script, issues an HTTP POST
1000 request to send the form to the service provider's Assertion Consumer Service.
- 1000 7. The cars.example.co.uk service provider's Assertion Consumer service validates the digital signature
1001 on the SAML Response and validates the SAML assertion. The supplied name identifier is then used
1002 to determine whether a previous federation has been established. If a previous federation has been
1003 established (because the name identifier maps to a local account) then go to step 9. If no federation
1004 exists for the persistent identifier in the assertion, then the SP needs to determine the local identity to
1005 which it should be assigned. The user will be challenged to provide local credentials at the SP.
1006 Optionally the user might first be asked whether he would like to federate the two accounts.
- 1001 8. The user provides valid credentials and identifies his account at the SP as **jdoe**. The persistent name
1002 identifier is then stored and registered with the **jdoe** account along with the name of the identity
1003 provider that created the name identifier.
- 1002 9. A local logon session is created for user **jdoe** and an access check is then made to establish whether
1003 the user **jdoe** has the correct authorization to access the desired resource at the cars.example.co.uk
1004 web site (the resource URL was retrieved from state information identified by the `RelayState`
1005 information).
- 1003 10. If the access check passes, the desired resource is returned to the browser.

1004 **5.4.4 Federation Using Transient Pseudonym Identifiers**

1005 The previous use case showed the use of persistent identifiers. So what if you do not want to establish a
1006 permanent federated identity between the partner sites? This is where the use of transient identifiers are
1007 useful. Transient identifiers allow you to:

- 1006 • Completely avoid having to manage user ID's and passwords at the service provider.
- 1007 • Have a scheme whereby the service provider does not have to manage specific user accounts, for
1008 instance it could be a site with a "group-like" access policy.
- 1008 • Support a truly anonymous service

1009 As with the Persistent Federation use cases, one can have SP and IdP-initiated variations. Figure 20
1010 shows the SP-initiated use case using transient pseudonym name identifiers.

1010

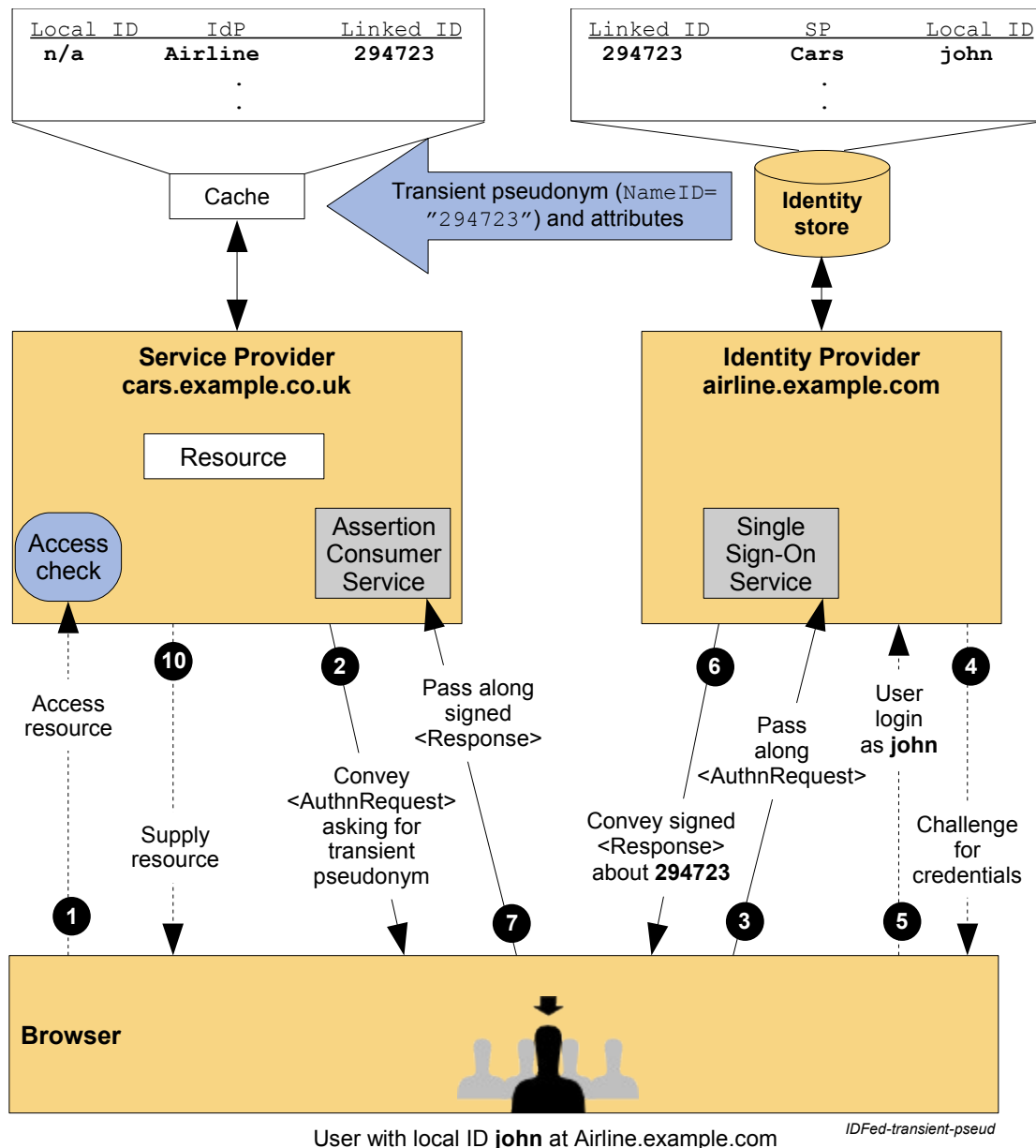


Figure 20: SP-Initiated Identity Federation with Transient Pseudonym

1011 The processing is as follows:

- 1012 1. The user attempts to access a resource on cars.example.co.uk. The user does not have any current
1013 logon session (i.e. security context) on this site, and is unknown to it. The resource that the user
1014 attempted to access is saved as `RelayState` information.
- 1013 2. The service provider uses the HTTP Redirect Binding to send the user to the Single Sign-On Service at
1014 the identity provider (airline.example.com). The HTTP redirect includes a SAML `<AuthnRequest>`
1015 message requesting that the identity provider provide an assertion using a transient name identifier for
1016 the user.
- 1014 3. The user will be challenged to provide valid credentials at the identity provider.
- 1015 4. The user provides valid credentials identifying himself as **john** and a local security context is created
1016 for the user at the IdP.
- 1016 5. The Single Sign-On Service looks up user **john** in its identity store and creates a transient name

1017 identifier (294723) to be used for the session at the service provider. It then builds a signed SAML web
 1018 SSO assertion where the subject uses a transient name identifier format. The name **john** is not
 1019 contained anywhere in the assertion. The assertion also contains an attribute statement with a
 1020 membership level attribute ("Gold" level). The assertion is placed in a SAML response message and
 1021 the IdP uses the HTTP POST Binding to send the Response message to the service provider.

1018 6. The browser, due either to a user action or execution of an "auto-submit" script, issues an HTTP POST
 1019 request to send the form to the service provider's Assertion Consumer Service.

1019 7. The cars.example.co.uk service provider's Assertion Consumer service validates the SAML Response
 1020 and SAML assertion. The supplied transient name identifier is then used to dynamically create a
 1021 session for the user at the SP. The membership level attribute might be used to perform an access
 1022 check on the requested resource and customize the content provided to the user.

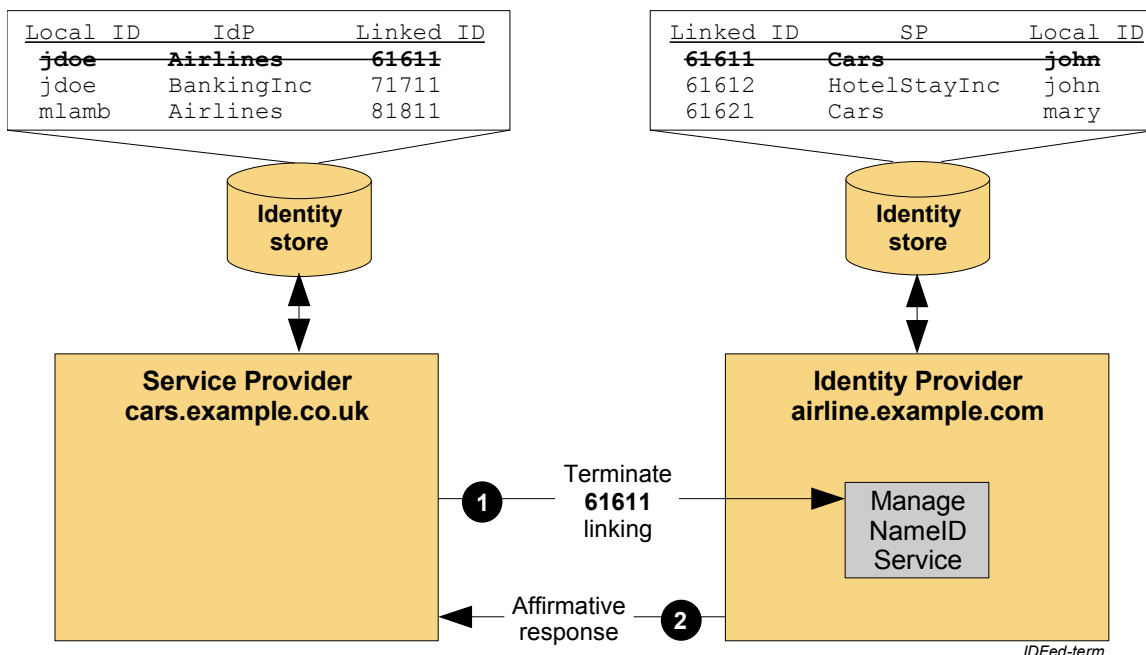
1020 8. If the access check passes, the requested resource is then returned to the browser.

1021 While not shown in the diagram, the transient identifier remains active for the life of the user authentication
 1022 session. If needed, the SP could use the identifier to make SAML attribute queries back to an attribute
 1023 authority at airline.example.com to obtain other identity attributes about the user in order to customize their
 1024 service provider content, etc.

1022 5.4.5 Federation Termination

1023 This example builds upon the previous example and shows how a federation can be terminated. In this
 1024 case the **jd**oe account on cars.example.co.uk service provider has been deleted, hence it wishes to
 1025 terminate the federation with airline.example.com for this user.

1024 The Terminate request is sent to the identity provider using the Name Identifier Management Protocol,
 1025 specifically using the <ManageNameIDRequest>. The example shown in Figure 21 uses the SOAP over
 1026 HTTP binding which demonstrates a use of the back channel. Bindings are also defined that permit the
 1027 request (and response) to be sent via the browser using asynchronous "front-channel" bindings, such as
 1028 the HTTP Redirect, HTTP POST, or Artifact bindings.



1026 In this example the processing is as follows:

1027 1. The service provider, cars.example.co.uk, determines that the local account, **jd**oe, should no longer be
 1028 federated. An example of this could be that the account has been deleted. The service provider sends

1028 to the airline.example.com identity provider a <ManageIDNameRequest> defining that the persistent
1029 identifier (previously established) must no longer be used. The request is carried in a SOAP message
1030 which is transported using HTTP, as defined by the SAML SOAP binding. The request is also digitally
1031 signed by the service provider.

1029 2. The identity provider verifies the digital signature ensuring that the <ManageIDNameRequest>
1030 originated from a known and trusted service provider. The identity Provider processes the request
1031 and returns a <ManageIDNameResponse> containing a suitable status code response. The response
1032 is carried within a SOAP over HTTP message and is digitally signed.

1030 **5.5 Use of Attributes**

1031 As explained in Section 3.2, in describing the web single sign-on use case, the SAML assertion
1032 transferred from an identity provider to a service provider may include attributes describing the user. The
1033 ability to transfer attributes within an assertion is a powerful SAML feature and it may also be combined
1034 with the forms of identity federation described above.

1032 The following are some typical use patterns:

- 1033 • Transfer of profile information

1034 Attributes may be used to convey user profile information from the identity provider to the service
1035 provider. This information may be used to provide personalized services at the service provider, or to
1036 augment or even create a new account for the user at the service provider. The user should be
1037 informed about the transfer of information, and, if required, user consent explicitly obtained.

- 1035 • Authorization based on attributes

1036 In this model, the attributes provided in the SAML assertion by the identity provider are used to
1037 authorize specific services at the service provider. The service provider and identity provider need
1038 prior agreement (out of band) on the attribute names and values included in the SAML assertion. An
1039 interesting use of this pattern which preserves user anonymity but allows for differential classes of
1040 service is found in Shibboleth : federation using transient pseudonyms combined with authorization
1041 based on attributes.

1037 **6 Extending and Profiling SAML for Use in Other** 1038 **Frameworks**

1038 SAML's components are modular and extensible. The SAML Assertions and Protocols specification has a
1039 section describing the basic extension features provided. The SAML Profiles specification provides
1040 guidelines on how to define new profiles and attribute profiles. The SAML Bindings specification likewise
1041 offers guidelines for defining new bindings.

1039 As a result of this flexibility, SAML has been adopted for use with several other standard frameworks.
1040 Following are some examples.

1040 **6.1 Web Services Security (WS-Security)**

1041 SAML assertions can be conveyed by means other than the SAML Request/Response protocols or
1042 profiles defined by the SAML specification set. One example of this is their use with Web Services
1043 Security (WS-Security), which is a set of specifications that define means for providing security protection
1044 of SOAP messages. The services provided by WS-Security are authentication, data integrity, and
1045 confidentiality.

1042 WS-Security defines a `<Security>` element that may be included in a SOAP message header. This
1043 element specifies how the message is protected. WS-Security makes use of mechanisms defined in the
1044 W3C XML Signature and XML Encryption specifications to sign and encrypt message data in both the
1045 SOAP header and body. The information in the `<Security>` element specifies what operations were
1046 performed and in what order, what keys were used for these operations, and what attributes and identity
1047 information are associated with that information. WS-Security also contains other features, such as the
1048 ability to timestamp the security information and to address it to a specified Role.

1043 In WS-Security, security data is specified using security *tokens*. Tokens can either be binary or structured
1044 XML. Binary tokens, such as X.509 certificates and Kerberos tickets, are carried in an XML wrapper. XML
1045 tokens, such as SAML assertions, are inserted directly as sub-elements of the `<Security>` element. A
1046 Security Token Reference may also be used to refer to a token in one of a number of ways.

1044 WS-Security consists of a core specification, which describes the mechanisms independent of the type of
1045 token being used, and a number of token profiles which describe the use of particular types of tokens.
1046 Token profiles cover considerations relating to that particular token type and methods of referencing the
1047 token using a Security Token Reference. The use of SAML assertions with WS-Security is described in
1048 the SAML Token Profile.

1045 Because the SAML protocols have a binding to SOAP, it is easy to get confused between that SAML-
1046 defined binding and the use of SAML assertions by WS-Security. They can be distinguished by their
1047 purpose, the message format, and the parties involved in processing the messages.

1046 The characteristics of the SAML Request/Response protocol binding over SOAP are as follows:

- 1047 • It is used to obtain SAML assertions for use external to the SOAP message exchange; they play no
1048 role in protecting the SOAP message.
- 1048 • The SAML assertions are contained within a SAML Response, which is carried in the body of the
1049 SOAP envelope.
- 1049 • The SAML assertions are provided by a trusted authority and may or may not pertain to the party
1050 requesting them.

1050 The characteristics of the use of SAML assertions as defined by WS-Security are as follows:

- 1051 • The SAML assertions are carried in a `<Security>` element within the header of the SOAP
1052 envelope as shown in Figure 22.
- 1052 • The SAML assertions usually play a role in the protection of the message they are carried in;
1053 typically they contain a key used for digitally signing data within the body of the SOAP message.
- 1053 • The SAML assertions will have been obtained previously and typically pertain to the identity of the
1054 sender of the SOAP message.

1054 Note that in principle, SAML assertions could be used in both ways in a single SOAP message. In this
1055 case the assertions in the header would refer to the identity of the Responder (and Requester) of the
1056 message. However, at this time, SAML has not profiled the use of WS-Security to secure the SOAP
1057 message exchanges that are made within a SAML deployment.

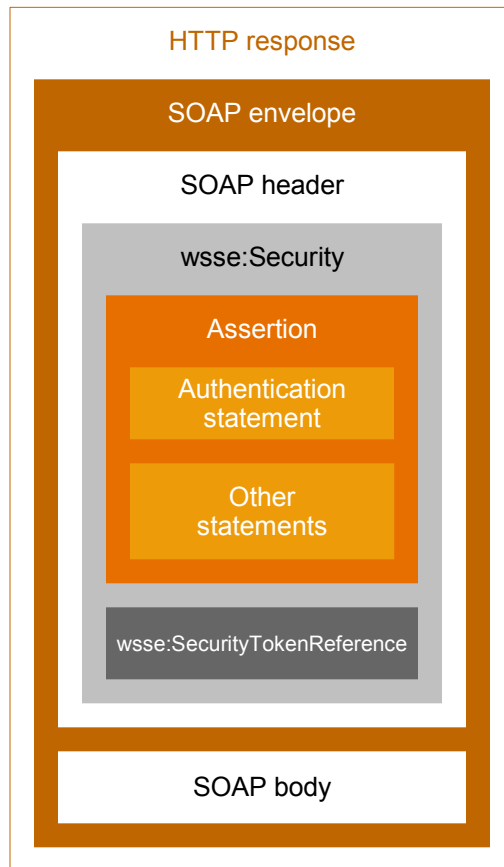


Figure 22: WS-Security with a SAML Token

1056 The following sequence of steps typifies the use of SAML assertions with WS-Security.

1057 A SOAP message sender obtains a SAML assertion by means of the SAML Request/Response protocol
1058 or other means. In this example, the assertion contains an attribute statement and a subject with a
1059 confirmation method called *Holder of Key*.

1058 To protect the SOAP message:

- 1059 1. The sender constructs the SOAP message, including a SOAP header with a WS-Security header.
1060 A SAML assertion is placed within a WS-Security token and included in the security header. The
1061 key referred to by the SAML assertion is used to construct a digital signature over data in the
1062 SOAP message body. Signature information is also included in the security header.
- 1060 2. The message receiver verifies the digital signature.
- 1061 3. The information in the SAML assertion is used for purposes such as Access Control and Audit
1062 logging.

1062 Figure 23 illustrates this usage scenario.

1063

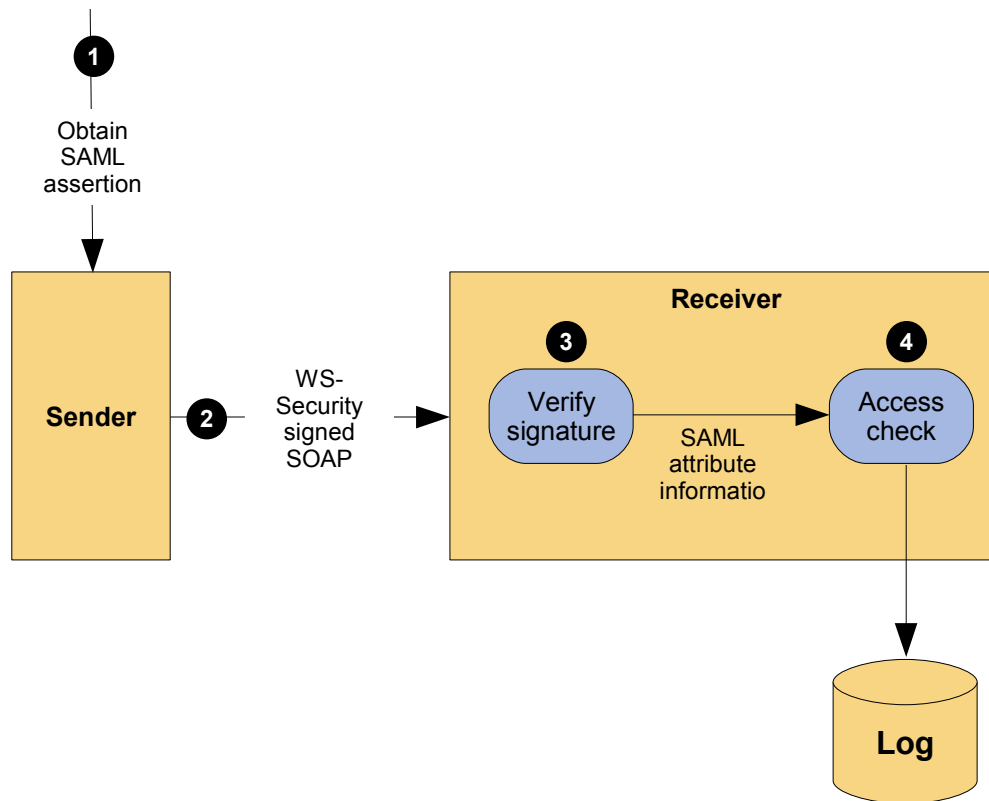


Figure 23: Typical Use of WS-Security with SAML Token

1064 6.2 eXtensible Access Control Markup Language (XACML)

1065 SAML assertions provide a means to distribute security-related information that may be used for a number
 1066 of purposes. One of the most important of these purposes is as input to Access Control decisions. For
 1067 example, it is common to consider when and how a user authenticated or what their attributes are in
 1068 deciding if a request should be allowed. SAML does not specify how this information should be used or
 1069 how access control policies should be addressed. This makes SAML suitable for use in a variety of
 1070 environments, including ones that existed prior to SAML.

1066 The eXtensible Access Control Markup Language (XACML) is an OASIS Standard that defines the syntax
 1067 and semantics of a language for expressing and evaluating access control policies. The work to define
 1068 XACML was started slightly after SAML began. From the beginning they were viewed as related efforts
 1069 and consideration was given to specifying both within the same Technical Committee. Ultimately, it was
 1070 decided to allow them to proceed independently but to align them. Compatibility with SAML was written in
 1071 to the charter of the XACML TC.

1067 As a result, SAML and XACML can each be used independently of the other, or both can be used
 1068 together. Figure 24 illustrates the typical use of SAML with XACML.

1068

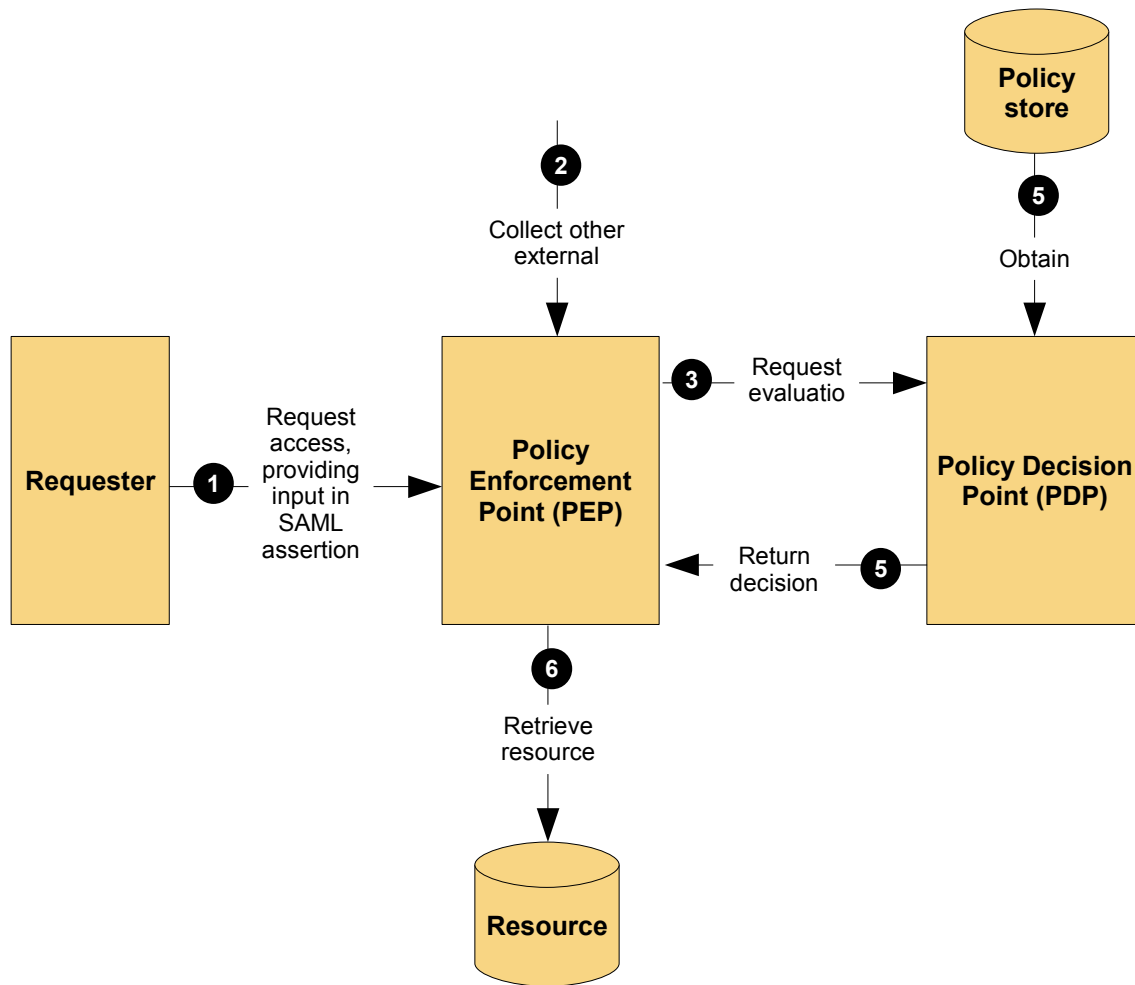


Figure 24: SAML and XACML Integration

1069 Using SAML and XACML in combination would typically involve the following steps.

- 1070 1. An XACML Policy Enforcement Point (PEP) receives a request to access some resource.
- 1071 2. The PEP obtains SAML assertions containing information about the parties to the request,
- 1072 such as the requester, the receiver (if different) or intermediaries. These assertions might
- 1073 accompany the request or be obtained directly from a SAML Authority, depending on the SAML
- 1074 profile used.
- 1072 3. The PEP obtains other information relevant to the request, such as time, date, location, and
- 1073 properties of the resource.
- 1073 4. The PEP presents all the information to a Policy Decision Point (PDP) to decide if the access
- 1074 should be allowed.
- 1074 5. The PDP obtains all the policies relevant to the request and evaluates them, combining
- 1075 conflicting results if necessary.
- 1075 6. The PDP informs the PEP of the decision result.
- 1076 7. The PEP enforces the decision, by either allowing the requested access or indicating that
- 1077 access is not allowed.

1077 The SAML and XACML specification sets contain some features specifically designed to facilitate their
1078 combined use.

1078 The XACML Attribute Profile in the SAML Profiles specification defines how attributes can be described
1079 using SAML syntax so that they may be automatically mapped to XACML Attributes. A schema is provided

1079 by SAML to facilitate this.

1080 A document that was produced by the XACML Technical Committee, SAML V2.0 profile of XACML v2.0,
1081 provides additional information on mapping SAML Attributes to XACML Attributes. This profile also defines
1082 a new type of Authorization decision query specifically designed for use in an XACML environment. It
1083 extends the SAML protocol schema and provides a request and response that contains exactly the inputs
1084 and outputs defined by XACML.

1081 That same document also contains two additional features that extend the SAML schemas. While they
1082 are not, strictly speaking, intended primarily to facilitate combining SAML and XACML, they are worth
1083 noting. The first is the XACML Policy Query. This extension to the SAML protocol schema allows the
1084 SAML protocol to be used to retrieve XACML policy which may be applicable to a given access decision.

1082 The second feature extends the SAML schema by allowing the SAML assertion envelope to be used to
1083 wrap an XACML policy. This makes available to XACML features such as Issuer, Validity interval and
1084 signature, without requiring the definition of a redundant or inconsistent scheme. This promotes code and
1085 knowledge reuse between SAML and XACML.

1083 **A. Acknowledgments**

1084 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
1085 Committee, whose voting members at the time of publication were:
1085

- 1086 ● Steve Anderson BMC Software
- 1087 ● Bhavna Bhatnagar Sun Microsystems
- 1088 ● Conor P. Cahill Intel
- 1089 ● Brian Campbell Ping Identity
- 1090 ● Scott Cantor Internet2
- 1091 ● Heather Hinton IBM
- 1092 ● Frederick Hirsch Nokia
- 1093 ● Jeff Hodges NeuStar
- 1094 ● Ari Kermaier Oracle
- 1095 ● Chris Laskowski Booz Allen Hamilton
- 1096 ● Hal Lockhart BEA Systems, Inc
- 1097 ● Paul Madsen NTT Corporation
- 1098 ● Eve Maler Sun Microsystems
- 1099 ● Prateek Mishra Oracle
- 1100 ● Bob Morgan Internet2
- 1101 ● Anthony Nadalin IBM
- 1102 ● Ashish Patel France Telecom
- 1103 ● Rob Philpott EMC Corporation
- 1104 ● Tom Scavo National Center for Supercomputing Applications
- 1105 ● David Staggs Veteran's Health Admin
- 1106 ● Eric Tiffany IEEE Industry Standards
- 1107 ● Greg Whitehead Hewlett-Packard Company
- 1108 ● Emily Xu Sun Microsystems

1109 Of particular note are the contributions from: Hal Lockhart BEA, Thomas Wisniewski Entrust, Scott Cantor
1110 Internet2, Prateek Mishra Oracle, and Jim Lien EMC Corporation.