



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

## Committee Draft 02

25 March 2008

### Specification URIs:

#### This Version:

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.html>

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.pdf>

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.cd-02.odt>

#### Previous Version:

N/A

#### 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>

#### Technical Committee:

OASIS Security Services TC

#### Chairs:

Hal Lockhart, BEA

Brian Campbell, Ping Identity

#### 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

39 security information between online business partners. This document provides a technical  
40 description of SAML V2.0.

41 **Status:**

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

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

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

---

# Notices

51

52 Copyright © OASIS Open 2008. All Rights Reserved.

53

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

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.

64

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

67

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

73

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

79

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

85

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

97

98 The name "OASIS" is a trademark of [OASIS](#), the owner and developer of this specification, and should be  
99 used only to refer to the organization and its official outputs. OASIS welcomes reference to, and  
100 implementation and use of, specifications, while reserving the right to enforce its marks against  
101 misleading uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

# Table of Contents

102		
103	1 Introduction.....	6
104	1.1 References.....	6
105	2 Overview.....	8
106	2.1 Drivers of SAML Adoption.....	8
107	2.2 Documentation Roadmap.....	8
108	3 High-Level SAML Use Cases.....	11
109	3.1 SAML Participants.....	11
110	3.2 Web Single Sign-On Use Case.....	11
111	3.3 Identity Federation Use Case.....	12
112	4 SAML Architecture.....	16
113	4.1 Basic Concepts.....	16
114	4.2 Advanced Concepts.....	17
115	4.2.1 Subject Confirmation.....	17
116	4.3 SAML Components.....	17
117	4.4 SAML XML Constructs and Examples.....	19
118	4.4.1 Relationship of SAML Components.....	19
119	4.4.2 Assertion, Subject, and Statement Structure.....	20
120	4.4.3 Attribute Statement Structure.....	21
121	4.4.4 Message Structure and the SOAP Binding.....	22
122	4.5 Privacy in SAML.....	24
123	4.6 Security in SAML.....	25
124	5 Major Profiles and Federation Use Cases.....	26
125	5.1 Web Browser SSO Profile.....	26
126	5.1.1 Introduction.....	26
127	5.1.2 SP-Initiated SSO: Redirect/POST Bindings.....	27
128	5.1.3 SP-Initiated SSO: POST/Artifact Bindings.....	30
129	5.1.4 IdP-Initiated SSO: POST Binding.....	33
130	5.2 ECP Profile.....	35
131	5.2.1 Introduction.....	35
132	5.2.2 ECP Profile Using PAOS Binding.....	35
133	5.3 Single Logout Profile.....	36
134	5.3.1 Introduction.....	37
135	5.3.2 SP-Initiated Single Logout with Multiple SPs.....	37
136	5.4 Establishing and Managing Federated Identities.....	38
137	5.4.1 Introduction.....	38
138	5.4.2 Federation Using Out-of-Band Account Linking.....	38
139	5.4.3 Federation Using Persistent Pseudonym Identifiers.....	40
140	5.4.4 Federation Using Transient Pseudonym Identifiers.....	42
141	5.4.5 Federation Termination.....	44
142	5.5 Use of Attributes.....	45
143	6 Extending and Profiling SAML for Use in Other Frameworks.....	46
144	6.1 Web Services Security (WS-Security).....	46
145	6.2 eXtensible Access Control Markup Language (XACML).....	48
146		

## 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>.

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

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

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

209 **[WSS]** A. Nadalin et al. *Web Services Security: SOAP Message Security 1.1 (WS-Security*  
210 *2004)*. OASIS WSS-TC, February 2006. Document ID wss-v1.1-spec-os-  
211 SOAPMessageSecurity. See <http://www.oasis-open.org/committees/wss/>.

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

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

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

220 **[XMLSig]** D. Eastlake et al. *XML-Signature Syntax and Processing*. World Wide Web  
221 Consortium. See <http://www.w3.org/TR/xmlsig-core/>



## 222 2 Overview

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

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

### 233 2.1 Drivers of SAML Adoption

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

- 236 • **Single Sign-On:** Over the years, various products have been marketed with the claim of providing  
237 support for web-based SSO. These products have typically relied on browser cookies to maintain  
238 user authentication state information so that re-authentication is not required each time the web user  
239 accesses the system. However, since browser cookies are never transmitted between DNS  
240 domains, the authentication state information in the cookies from one domain is never available to  
241 another domain. Therefore, these products have typically supported multi-domain SSO (MDSSO)  
242 through the use of proprietary mechanisms to pass the authentication state information between the  
243 domains. While the use of a single vendor's product may sometimes be viable within a single  
244 enterprise, business partners usually have heterogeneous environments that make the use of  
245 proprietary protocols impractical for MDSSO. SAML solves the MDSSO problem by providing a  
246 standard vendor-independent grammar and protocol for transferring information about a user from  
247 one web server to another independent of the server DNS domains.
- 248 • **Federated identity:** When online services wish to establish a collaborative application environment  
249 for their mutual users, not only must the systems be able to understand the protocol syntax and  
250 semantics involved in the exchange of information; they must also have a common understanding of  
251 who the user is that is referred to in the exchange. Users often have individual local user identities  
252 within the security domains of each partner with which they interact. Identity federation provides a  
253 means for these partner services to agree on and establish a common, shared name identifier to  
254 refer to the user in order to share information about the user across the organizational boundaries.  
255 The user is said to have a **federated identity** when partners have established such an agreement  
256 on how to refer to the user. From an administrative perspective, this type of sharing can help reduce  
257 identity management costs as multiple services do not need to independently collect and maintain  
258 identity-related data (e.g. passwords, identity attributes). In addition, administrators of these services  
259 usually do not have to manually establish and maintain the shared identifiers; rather control for this  
260 can reside with the user.
- 261 • **Web services and other industry standards:** SAML allows for its security assertion format to be  
262 used outside of a "native" SAML-based protocol context. This modularity has proved useful to other  
263 industry efforts addressing authorization services (IETF, OASIS), identity frameworks, web services  
264 (OASIS, Liberty Alliance), etc. The OASIS WS-Security Technical Committee has defined a **profile**  
265 for how to use SAML's rich assertion constructs within a WS-Security **security token** that can be  
266 used, for example, to secure web service SOAP message exchanges. In particular, the advantage  
267 offered by the use of a SAML assertion is that it provides a standards-based approach to the  
268 exchange of information, including attributes, that are not easily conveyed using other WS-Security  
269 token formats.

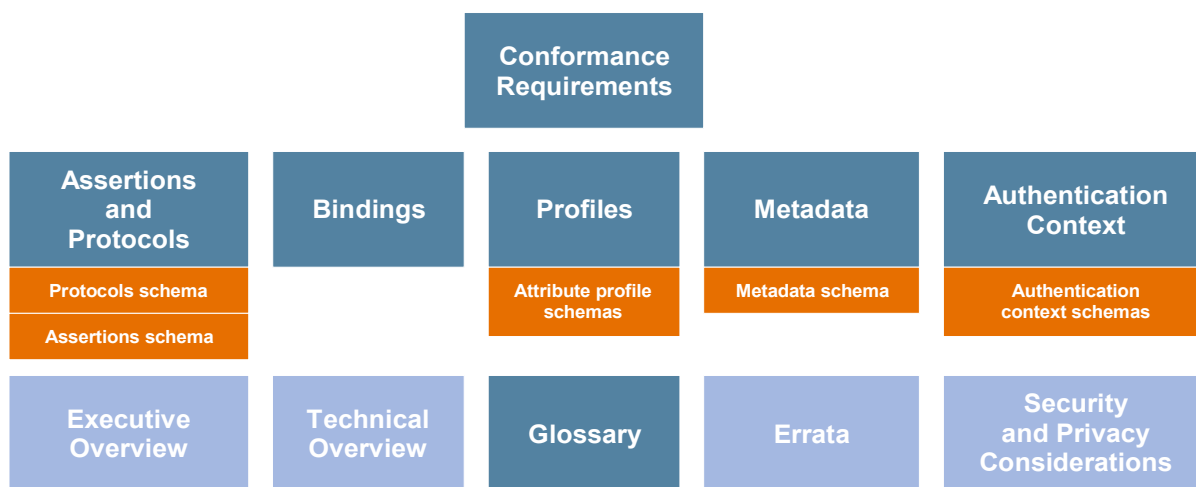
### 270 2.2 Documentation Roadmap

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



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

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



SAML-docset

Figure 1: SAML V2.0 Document Set

- 278 • **Conformance Requirements** documents the technical requirements for SAML conformance, a  
279 status that software vendors typically care about because it is one measure of cross-product  
280 compatibility. If you need to make a formal reference to SAML V2.0 from another document, you  
281 simply need to point to this one.
- 282 • **Assertions and Protocol** defines the syntax and semantics for creating XML-encoded assertions  
283 to describe authentication, attribute, and authorization information, and for the protocol messages to  
284 carry this information between systems. It has associated schemas, one for assertions and one for  
285 protocols.
- 286 • **Bindings** defines how SAML assertions and request-response protocol messages can be  
287 exchanged between systems using common underlying communication protocols and frameworks.
- 288 • **Profiles** defines specific sets of rules for using and restricting SAML's rich and flexible syntax for  
289 conveying security information to solve specific business problems (for example, to perform a web  
290 SSO exchange). It has several associated small schemas covering syntax aspects of attribute  
291 profiles.
- 292 • **Metadata** defines how a SAML entity can describe its configuration data (e.g. service endpoint  
293 URLs, key material for verifying signatures) in a standard way for consumption by partner entities. It  
294 has an associated schema.
- 295 • **Authentication Context** defines a syntax for describing authentication context declarations which  
296 describe various authentication mechanisms. It has an associated set of schemas.
- 297 • **Executive Overview** provides a brief executive-level overview of SAML and its primary benefits.  
298 This is a non-normative document.
- 299 • **Technical Overview** is the document you are reading.
- 300 • **Glossary** normatively defines terms used throughout the SAML specifications. Where possible,  
301 terms are aligned with those defined in other security glossaries.
- 302 • **Errata** clarifies interpretation of the SAML V2.0 standard where information in the final published  
303 version was conflicting or unclear. Although the advice offered in this document is non-normative, it  
304 is useful as a guide to the likely interpretations used by implementors of SAML-conforming software,  
305 and is likely to be incorporated in any future revision to the standard. This document is updated on

306 an ongoing basis.

307 • **Security and Privacy Considerations** describes and analyzes the security and privacy properties  
308 of SAML.

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

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

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

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

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

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

## 323 **3 High-Level SAML Use Cases**

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

### 327 **3.1 SAML Participants**

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

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

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

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

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

### 352 **3.2 Web Single Sign-On Use Case**

353 Multi-domain web single sign-on is arguably the most important use case for which SAML is applied. In  
354 this use case, a user has a login session (that is, a *security context*) on a web site ([airline.example.com](http://airline.example.com))  
355 and is accessing resources on that site. At some point, either explicitly or transparently, he is directed over  
356 to a partner's web site ([cars.example.co.uk](http://cars.example.co.uk)). In this case, we assume that a federated identity for the user  
357 has been previously established between [airline.example.com](http://airline.example.com) and [cars.example.co.uk](http://cars.example.co.uk) based on a  
358 business agreement between them. The identity provider site ([airline.example.com](http://airline.example.com)) asserts to the service  
359 provider site ([cars.example.co.uk](http://cars.example.co.uk)) that the user is known (by referring to the user by their federated  
360 identity), has authenticated to it, and has certain identity attributes (e.g. has a “Gold membership”). Since  
361 [cars.example.co.uk](http://cars.example.co.uk) trusts [airline.example.com](http://airline.example.com), it trusts that the user is valid and properly authenticated  
362 and thus creates a local session for the user. This use case is shown in Figure 2, which illustrates the fact  
363 that the user is not required to re-authenticate when directed over to the [cars.example.co.uk](http://cars.example.co.uk) site.

364

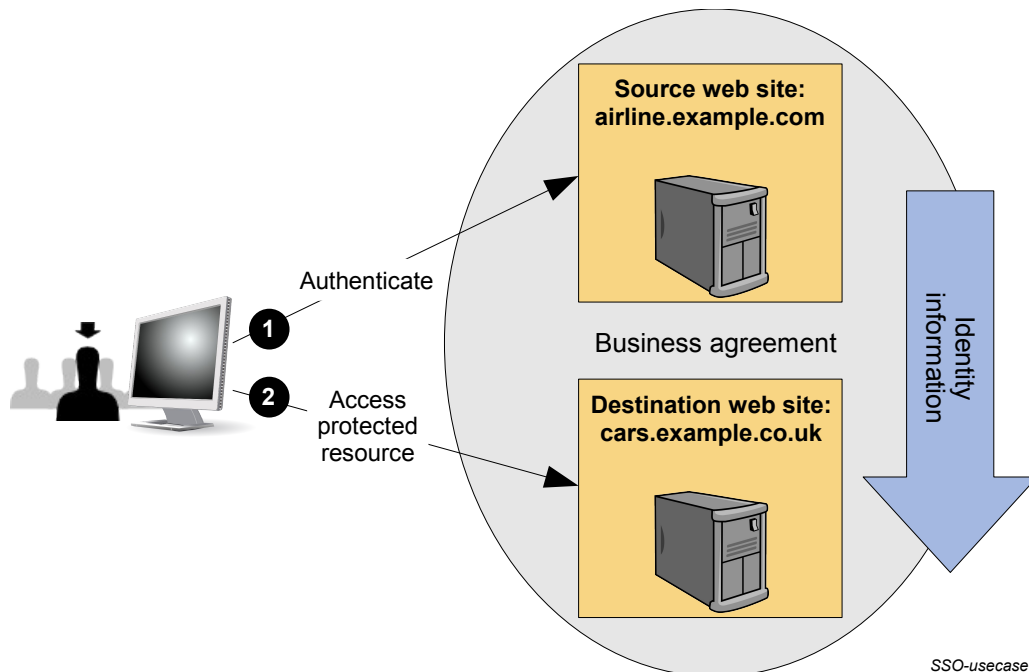


Figure 2: General Single Sign-On Use Case

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

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

### 378 3.3 Identity Federation Use Case

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

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

- 384 • Do the users have existing local identities at the sites that must be linked together through the  
 385 federated identifiers?
- 386 • Will the establishment and termination of federated identifiers for the users be done dynamically or  
 387 will the sites use pre-established federated identifiers?
- 388 • Do users need to explicitly consent to establishment of the federated identity?
- 389 • Do identity attributes about the users need to be exchanged?
- 390 • Should the identity federation rely on transient identifiers that are destroyed at the end of the user  
 391 session?
- 392 • Is the privacy of information to be exchanged of high concern such that the information should be

393 encrypted?

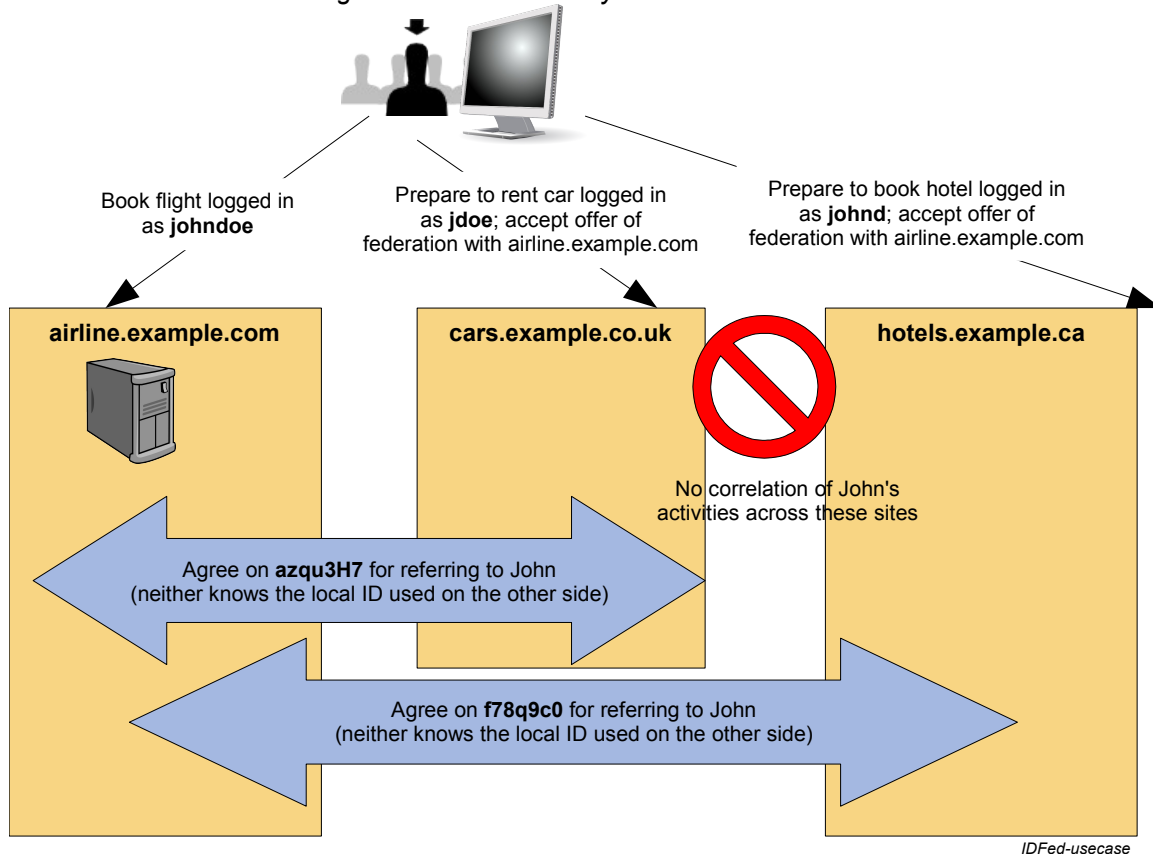
394 Previous versions of the SAML standard relied on out-of-band agreement on the types of identifiers that  
395 would be used to represent a federated identity between partners (e.g. the use of X.509 subject names).  
396 While it supported the use of federated identities, it provided no means to directly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

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

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

417 This use case, shown in Figure 3, demonstrates how, during web SSO, the sites can dynamically  
418 establish the federated name identifiers used in the account linking process. One identity provider,  
419 [airline.example.com](#), and two service providers exist in this example: [cars.example.co.uk](#) for car rentals  
420 and [hotels.example.ca](#) for hotel bookings. The example assumes a user is registered on all three provider  
421 sites (i.e. they have pre-existing local login accounts), but the local accounts all have different account  
422 identifiers. At [airline.example.com](#), user John is registered as **johndoe**, on [cars.example.co.uk](#) his  
423 account is **jdoe**, and on [hotels.example.ca](#) it is **johnd**. The sites have established an agreement to use  
424 **persistent** SAML privacy-preserving pseudonyms for the user's federated name identifiers. John has not  
425 previously federated his identities between these sites.

Figure 3: General Identity Federation Use Case



IDFed-usecase

427 The processing sequence is as follows:

- 428 1. John books a flight at [airline.example.com](http://airline.example.com) using his **johndoe** user account.
- 429 2. John then uses a browser bookmark or clicks on a link to visit [cars.example.co.uk](http://cars.example.co.uk) to reserve a car.  
 430 This site sees that the browser user is not logged in locally but that he has previously visited their IdP  
 431 partner site [airline.example.com](http://airline.example.com) (optionally using the new IdP discovery feature of SAML V2.0). So  
 432 cars.example.co.uk asks John if he would like to consent to federate his local [cars.example.co.uk](http://cars.example.co.uk)  
 433 identity with [airline.example.com](http://airline.example.com).
- 434 3. John consents to the federation and his browser is redirected back to [airline.example.com](http://airline.example.com) where the  
 435 site creates a new pseudonym, **azqu3H7** for John's use when he visits [cars.example.co.uk](http://cars.example.co.uk). The  
 436 pseudonym is linked to his **johndoe** account. Both providers agree to use this identifier to refer to John  
 437 in subsequent transactions.
- 438 4. John is then redirected back to [cars.example.co.uk](http://cars.example.co.uk) with a SAML assertion indicating that the user  
 439 represented by the federated persistent identifier **azqu3H7** is logged in at the IdP. Since this is the first  
 440 time that [cars.example.co.uk](http://cars.example.co.uk) has seen this identifier, it does not know which local user account to  
 441 which it applies.
- 442 5. Thus, John must log in at [cars.example.co.uk](http://cars.example.co.uk) using his **jdoe** account. Then [cars.example.co.uk](http://cars.example.co.uk)  
 443 attaches the identity **azqu3H7** to the local **jdoe** account for future use with the IdP [airline.example.com](http://airline.example.com).  
 444 The user accounts at the IdP and this SP are now *linked* using the federated name identifier **azqu3H7**.
- 445 6. After reserving a car, John selects a browser bookmark or clicks on a link to visit [hotels.example.ca](http://hotels.example.ca) in  
 446 order to book a hotel room.
- 447 7. The federation process is repeated with the IdP [airline.example.com](http://airline.example.com), creating a new pseudonym,  
 448 **f78q9C0**, for IdP user **johndoe** that will be used when visiting [hotels.example.ca](http://hotels.example.ca).

449 8. John is redirected back to the [hotels.example.ca](http://hotels.example.ca) SP with a new SAML assertion. The SP requires John  
450 to log into his local **johnd** user account and adds the pseudonym as the federated name identifier for  
451 future use with the IdP [airline.example.com](http://airline.example.com). The user accounts at the IdP and this SP are now *linked*  
452 using the federated name identifier **f78q9C0**.

453 In the future, whenever John needs to books a flight, car, and hotel, he will only need to log in once to  
454 [airline.example.com](http://airline.example.com) before visiting [cars.example.co.uk](http://cars.example.co.uk) and [hotels.example.ca](http://hotels.example.ca). The [airline.example.com](http://airline.example.com)  
455 IdP will identify John as **azqu3H7** to [cars.example.co.uk](http://cars.example.co.uk) and as **f78q9C0** to [hotels.example.ca](http://hotels.example.ca). Each SP  
456 will locate John's local user account through the linked persistent pseudonyms and allow John to conduct  
457 business after the SSO exchange.



## 4 SAML Architecture

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

### 4.1 Basic Concepts

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

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

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

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

Figure 4 illustrates the relationship between these basic SAML concepts.

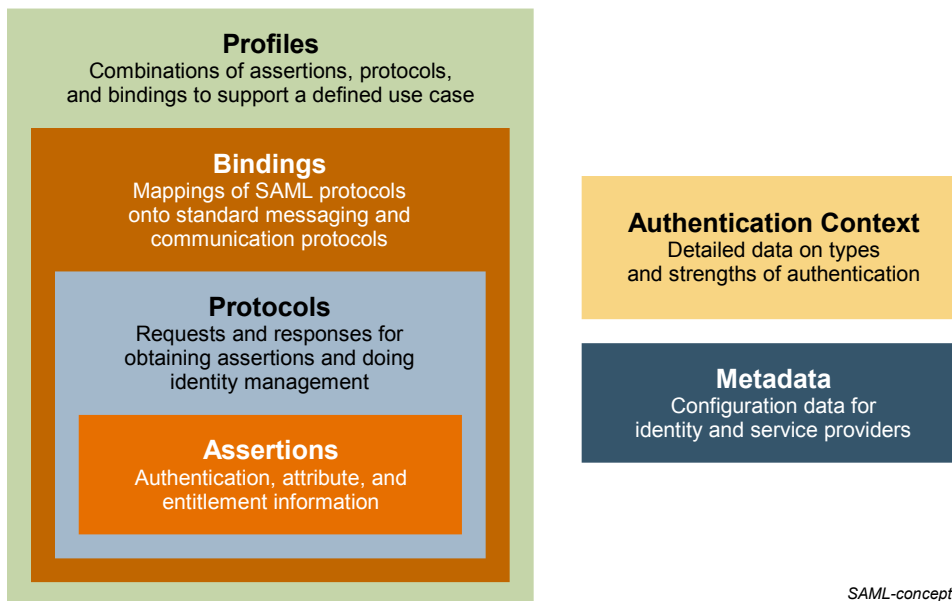


Figure 4: Basic SAML Concepts

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

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

487 expressed using SAML metadata XML documents. SAML Metadata is defined by its own XML  
488 schema.

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

498 This document does not go into further detail about Metadata and Authentication Context; for more  
499 information, see the specifications that focus on them ([SAMLMeta]and[SAMLAuthnCxt], respectively).

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

## 504 4.2 Advanced Concepts

### 505 4.2.1 Subject Confirmation

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

511 The `SubjectConfirmation` element provides the means for a relying party to verify the  
512 correspondence of the subject of the assertion with the party with whom the relying party is  
513 communicating. The `Method` attribute indicates the specific method that the relying party should use to  
514 make this determination.  
515

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

```
518 urn:oasis:names:tc:SAML:2.0:cm:holder-of-key  
519 urn:oasis:names:tc:SAML:2.0:cm:sender-vouches  
520 urn:oasis:names:tc:SAML:2.0:cm:bearer
```

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

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

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

## 530 4.3 SAML Components

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

533 **Assertions:** SAML allows for one party to assert security information in the form of **statements** about a  
534 **subject**. For instance, a SAML assertion could state that the subject is named “John Doe”, has an email  
535 address of john.doe@example.com, and is a member of the “engineering” group.

536 An assertion contains some basic required and optional information that applies to all its statements, and  
537 usually contains a *subject* of the assertion (if not present, the identity determined through other means,  
538 e.g. the certificate used for subject confirmation), *conditions* used to validate the assertion, and assertion  
539 statements.

540 SAML defines three kinds of statements that can be carried within an assertion:

- 541 • **Authentication statements:** These are created by the party that successfully authenticated a  
542 user. At a minimum, they describe the particular means used to authenticate the user and the  
543 specific time at which the authentication took place.
- 544 • **Attribute statements:** These contain specific identifying attributes about the subject (for  
545 example, that user “John Doe” has “Gold” card status).
- 546 • **Authorization decision statements:** These define something that the subject is entitled to do  
547 (for example, whether “John Doe” is permitted to buy a specified item).

548 **Protocols:** SAML defines a number of generalized request/response protocols:  
549

- 550 • **Authentication Request Protocol:** Defines a means by which a principal (or an agent acting on  
551 behalf of the principal) can request assertions containing authentication statements and,  
552 optionally, attribute statements. The Web Browser SSO Profile uses this protocol when  
553 redirecting a user from an SP to an IdP when it needs to obtain an assertion in order to establish  
554 a security context for the user at the SP.
- 555 • **Single Logout Protocol:** Defines a mechanism to allow near-simultaneous logout of active  
556 sessions associated with a principal. The logout can be directly initiated by the user, or initiated  
557 by an IdP or SP because of a session timeout, administrator command, etc.
- 558 • **Assertion Query and Request Protocol:** Defines a set of queries by which SAML assertions  
559 may be obtained. The *Request* form of this protocol can ask an asserting party for an existing  
560 assertion by referring to its assertion ID. The *Query* form of this protocol defines how a relying  
561 party can ask for assertions (new or existing) on the basis of a specific subject and the desired  
562 statement type.
- 563 • **Artifact Resolution Protocol:** Provides a mechanism by which SAML protocol messages may  
564 be passed by reference using a small, fixed-length value called an *artifact*. The artifact receiver  
565 uses the Artifact Resolution Protocol to ask the message creator to dereference the artifact and  
566 return the actual protocol message. The artifact is typically passed to a message recipient using  
567 one SAML binding (e.g. HTTP Redirect) while the resolution request and response take place  
568 over a synchronous binding, such as SOAP.
- 569 • **Name Identifier Management Protocol:** Provides mechanisms to change the value or format  
570 of the name identifier used to refer to a principal. The issuer of the request can be either the  
571 service provider or the identity provider. The protocol also provides a mechanism to terminate an  
572 association of a name identifier between an identity provider and service provider.
- 573 • **Name Identifier Mapping Protocol:** Provides a mechanism to programmatically map one  
574 SAML name identifier into another, subject to appropriate policy controls. It permits, for example,  
575 one SP to request from an IdP an identifier for a user that the SP can use at another SP in an  
576 application integration scenario.

577 **Bindings:** SAML bindings detail exactly how the various SAML protocol messages can be carried over  
578 underlying transport protocols. The bindings defined by SAML V2.0 are:

- 579 • **HTTP Redirect Binding:** Defines how SAML protocol messages can be transported using  
580 HTTP redirect messages (302 status code responses).
- 581 • **HTTP POST Binding:** Defines how SAML protocol messages can be transported within the  
582 base64-encoded content of an HTML form control.

- 583 • **HTTP Artifact Binding:** Defines how an artifact (described above in the Artifact Resolution  
584 Protocol) is transported from a message sender to a message receiver using HTTP. Two  
585 mechanisms are provided: either an HTML form control or a query string in the URL.
- 586 • **SAML SOAP Binding:** Defines how SAML protocol messages are transported within SOAP 1.1  
587 messages, with details about using SOAP over HTTP.
- 588 • **Reverse SOAP (PAOS) Binding:** Defines a multi-stage SOAP/HTTP message exchange that  
589 permits an HTTP client to be a SOAP responder. Used in the Enhanced Client and Proxy Profile  
590 to enable clients and proxies capable of assisting in IDP discovery.
- 591 • **SAML URI Binding:** Defines a means for retrieving an existing SAML assertion by resolving a  
592 URI (uniform resource identifier).

593 **Profiles:** SAML profiles define how the SAML assertions, protocols, and bindings are combined and  
594 constrained to provide greater interoperability in particular usage scenarios. Some of these profiles are  
595 examined in detail later in this document. The profiles defined by SAML V2.0 are:

- 596 • **Web Browser SSO Profile:** Defines how SAML entities use the Authentication Request Protocol  
597 and SAML Response messages and assertions to achieve single sign-on with standard web  
598 browsers. It defines how the messages are used in combination with the HTTP Redirect, HTTP  
599 POST, and HTTP Artifact bindings.
- 600 • **Enhanced Client and Proxy (ECP) Profile:** Defines a specialized SSO profile where  
601 specialized clients or gateway proxies can use the Reverse-SOAP (PAOS) and SOAP bindings.
- 602 • **Identity Provider Discovery Profile:** Defines one possible mechanism for service providers to  
603 learn about the identity providers that a user has previously visited.
- 604 • **Single Logout Profile:** Defines how the SAML Single Logout Protocol can be used with SOAP,  
605 HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
- 606 • **Assertion Query/Request Profile:** Defines how SAML entities can use the SAML Query and  
607 Request Protocol to obtain SAML assertions over a synchronous binding, such as SOAP.
- 608 • **Artifact Resolution Profile:** Defines how SAML entities can use the Artifact Resolution Protocol  
609 over a synchronous binding, such as SOAP, to obtain the protocol message referred to by an  
610 artifact.
- 611 • **Name Identifier Management Profile:** Defines how the Name Identifier Management Protocol  
612 may be used with SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
- 613 • **Name Identifier Mapping Profile:** Defines how the Name Identifier Mapping Protocol uses a  
614 synchronous binding such as SOAP.

## 615 4.4 SAML XML Constructs and Examples

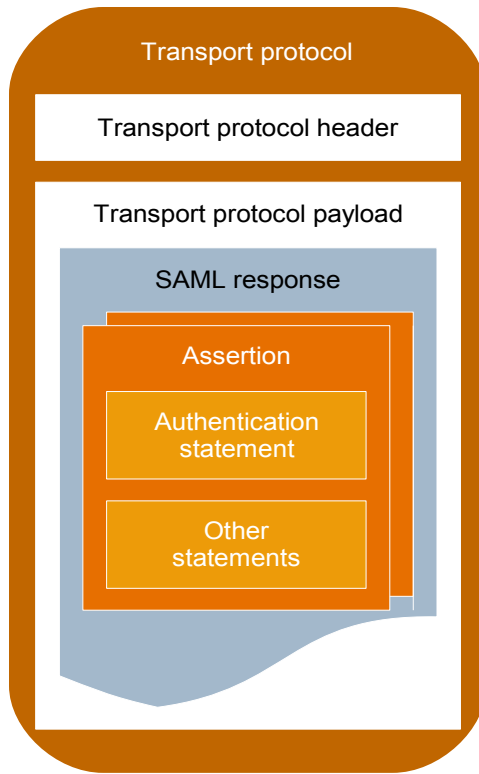
616 This section provides descriptions and examples of some of the key SAML XML constructs.

### 617 4.4.1 Relationship of SAML Components

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

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

624



SAML-component-nesting

Figure 5: Relationship of SAML Components

#### 625 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

626

627 Figure 6 shows an XML fragment containing an example assertion with a single authentication statement.  
628 Note that the XML text in the figure (and elsewhere in this document) has been formatted for presentation  
629 purposes. Specifically, while line breaks and extra spaces are ignored between XML attributes within an  
630 XML element tag, when they appear between XML element start/end tags, they technically become part of  
631 the element value. They are inserted in the example only for readability.

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

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

- 650 • Email address
- 651 • X.509 subject name
- 652 • Windows domain qualified name
- 653 • Kerberos principal name
- 654 • Entity identifier
- 655 • Persistent identifier
- 656 • Transient identifier

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

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

### 671 **4.4.3 Attribute Statement Structure**

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

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

```

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

677

678

679 Note the following:

- 680 • A single statement can contain multiple attributes. In this example, there are three attributes (starting  
681 on lines 2, 10, and 16) within the statement.
- 682 • Attribute names are qualified with a name format (lines 4, 11, and 17) which indicates how the  
683 attribute name is to be interpreted. This example takes advantage of two of the SAML-defined  
684 **attribute profiles** and defines a third custom attribute as well. The first attribute uses the SAML **X.**  
685 **500/LDAP Attribute Profile** to define a value for the LDAP attribute identified by the OID "2.5.4.42".  
686 This attribute in an LDAP directory has a friendly name of "givenName" and the attribute's value is  
687 "John". The second attribute utilizes the SAML **Basic Attribute Profile**, refers to an attribute named  
688 "LastName" which has the value "Doe". The name format of the third attribute indicates the name is  
689 not of a format defined by SAML, but is rather defined by a third party, SmithCo. Note that the use of  
690 private formats and attribute profiles can create significant interoperability issues. See the SAML  
691 Profiles specification for more information and examples.
- 692 • The value of an attribute can be defined by simple data types, as on lines 7 and 14, or can be  
693 structured XML, as on lines 20 through 22.

#### 694 4.4.4 Message Structure and the SOAP Binding

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



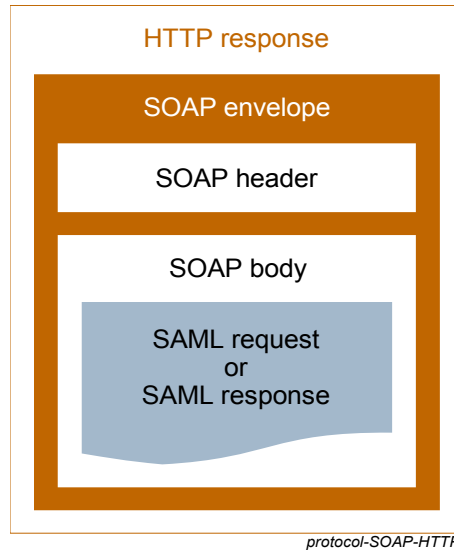


Figure 8: Protocol Messages Carried by SOAP Over HTTP

701 Figure 9 shows an XML document containing an example SAML attribute query message being  
 702 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:AttributeQuery
6.        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
7.        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
8.        ID="aaf23196-1773-2113-474a-fe114412ab72"
9.        Version="2.0"
10.       IssueInstant="2006-07-17T20:31:40Z">
11.       <saml:Issuer>http://example.sp.com</saml:Issuer>
12.       <saml:Subject>
13.         <saml:NameID
14.           Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">
15.           C=US, O=NCSA-TEST, OU=User, CN=trscavo@uiuc.edu
16.         </saml:NameID>
17.       </saml:Subject>
18.       <saml:Attribute
19.         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
20.         Name="urn:oid:2.5.4.42"
21.         FriendlyName="givenName">
22.       </saml:Attribute>
23.     </samlp:AttributeQuery>
24.   </env:Body>
25. </env:Envelope>
  
```

*Figure 9: Attribute Query in SOAP Envelope*

703  
 704  
 705  
 706  
 707  
 708  
 709  
 710

Note the following:

- The SOAP envelope starts at line 2.
- The SAML attribute query starting on line 5 is embedded in a SOAP body element starting on line 4.
- The attribute query contains, from lines 6 through 10, various required and optional XML attributes including declarations of the SAML V2.0 assertion and protocol namespaces, and the message ID, .

- 711 • The request specifies a number of optional elements, from lines 11 through 22, that govern the type  
712 of attributes the requester expects back. This includes, for example, the requested attribute  
713 (givenName) and the subject for which the attribute is sought.

714 An example XML fragment containing a SAML protocol Response message being transported in a SOAP  
715 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="2006-07-17T20:31:41Z"
10:      InResponseTo="aaf23196-1773-2113-474a-fe114412ab72 ">
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

716  
717 Note the following:

- 718 • On line 10, the Response InResponseTo XML attribute references the request to which the  
719 asserting party is responding, and specifies additional information (lines 7 through 14) needed to  
720 process the response, including status information. SAML defines a number of status codes and, in  
721 many cases, dictates the circumstances under which they must be used.
- 722 • Within the response (line 15; detail elided) is a SAML assertion, that would contain the requested  
723 given name attribute in an attribute statement.

## 724 4.5 Privacy in SAML

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

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

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

- 732 • SAML supports the establishment of pseudonyms established between an identity provider and a  
733 service provider. Such pseudonyms do not themselves enable inappropriate correlation between  
734 service providers (as would be possible if the identity provider asserted the same identifier for a  
735 user to every service provider, a so-called *global* identifier).
- 736 • SAML supports *one-time* or transient identifiers – such identifiers ensure that every time a certain  
737 user accesses a given service provider through a single sign-on operation from an identity  
738 provider, that service provider will be unable to recognize them as the same individual as might  
739 have previously visited (based solely on the identifier, correlation may be possible through non-  
740 SAML handles).
- 741 • SAML's Authentication Context mechanisms allow a user to be authenticated at a sufficient (but  
742 not more than necessary) assurance level, appropriate to the resource they may be attempting to  
743 access at some service provider.
- 744 • SAML allows the claimed fact of a user consenting to certain operations (e.g. the act of

745 federation) to be expressed between providers. How, when or where such consent is obtained is  
746 out of scope for SAML.

## 747 **4.6 Security in SAML**

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

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

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

- 759 • Where message integrity and message confidentiality are required, then HTTP over SSL 3.0 or TLS  
760 1.0 is recommended.
- 761 • When a relying party requests an assertion from an asserting party, bi-lateral authentication is  
762 required and the use of SSL 3.0 or TLS 1.0 using mutual authentication or authentication via digital  
763 signatures is recommended.
- 764 • When a response message containing an assertion is delivered to a relying party via a user's web  
765 browser (for example using the HTTP POST binding), then to ensure message integrity, it is  
766 mandated that the response message be digitally signed using XML Signature

## 767 **5 Major Profiles and Federation Use Cases**

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

### 771 **5.1 Web Browser SSO Profile**

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

#### 773 **5.1.1 Introduction**

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

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

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

794 Figure 11 compares the IdP-initiated and SP-initiated models.

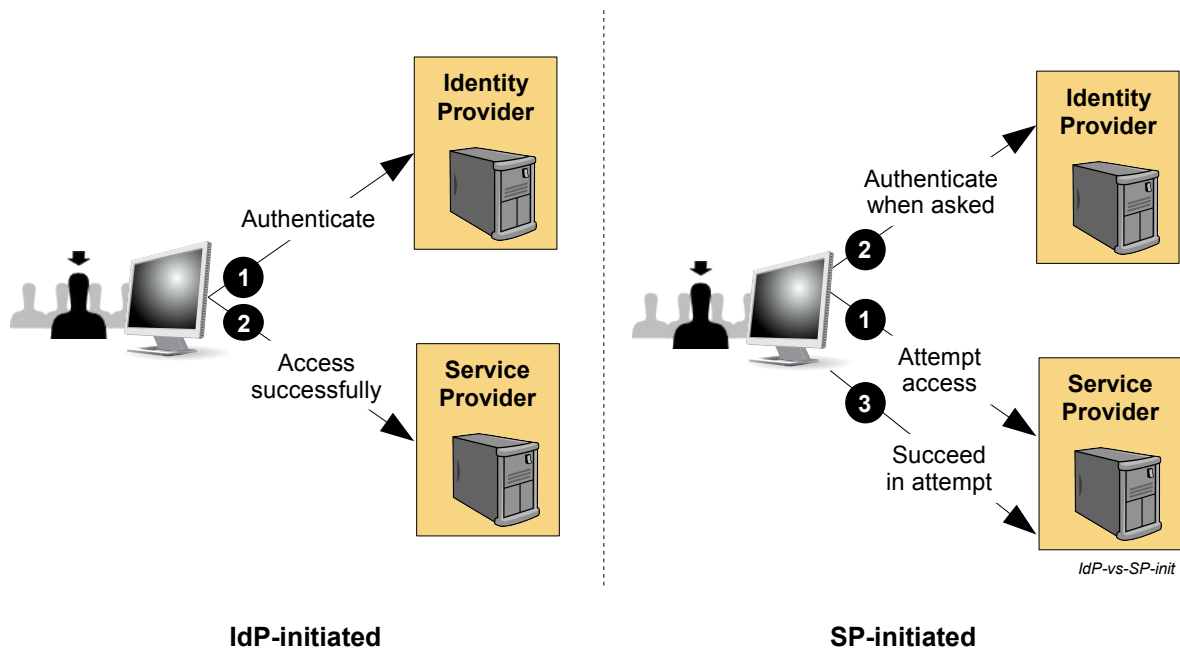


Figure 11: Differences in Initiation of Web Browser SSO

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

803 The SAML Conformance and Profiles specifications identify the SAML bindings that can legally be used  
 804 with these two messages. Specifically, an Authentication Request message can be sent from an SP to an  
 805 IdP using either the HTTP Redirect Binding, HTTP POST Binding, or HTTP Artifact Binding. The  
 806 Response message can be sent from an IdP to an SP using either the HTTP POST Binding or the HTTP  
 807 Artifact Binding. For this pair of messages, SAML permits asymmetry in the choice of bindings used. That  
 808 is, a request can be sent using one binding and the response can be returned using a different binding.  
 809 The decision of which bindings to use is typically driven by configuration settings at the IdP and SP  
 810 systems. Factors such as potential message sizes, whether identity information is allowed to transit  
 811 through the browser (if not the artifact binding may be required) , etc. must be considered in the choice of  
 812 bindings.

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

- 815 • SP-initiated SSO using a Redirect Binding for the SP-to-IdP <AuthnRequest> message and a POST  
 816 Binding for the IdP-to-SP <Response> message
- 817 • SP-initiated SSO using a POST Binding for the <AuthnRequest> message and an Artifact Binding for  
 818 the <Response> message
- 819 • IDP-initiated SSO using a POST Binding for the IdP-to-SP <Response> message; no SP-to-IdP  
 820 <AuthnRequest> message is involved.

### 821 5.1.2 SP-Initiated SSO: Redirect/POST Bindings

822 This first example describes an SP-initiated SSO exchange. In such an exchange, the user attempts to

823 access a resource on the SP, sp.example.com. However they do not have a current logon session on this  
 824 site and their federated identity is managed by their IdP, idp.example.org. They are sent to the IdP to log  
 825 on and the IdP provides a SAML web SSO assertion for the user's federated identity back to the SP.

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

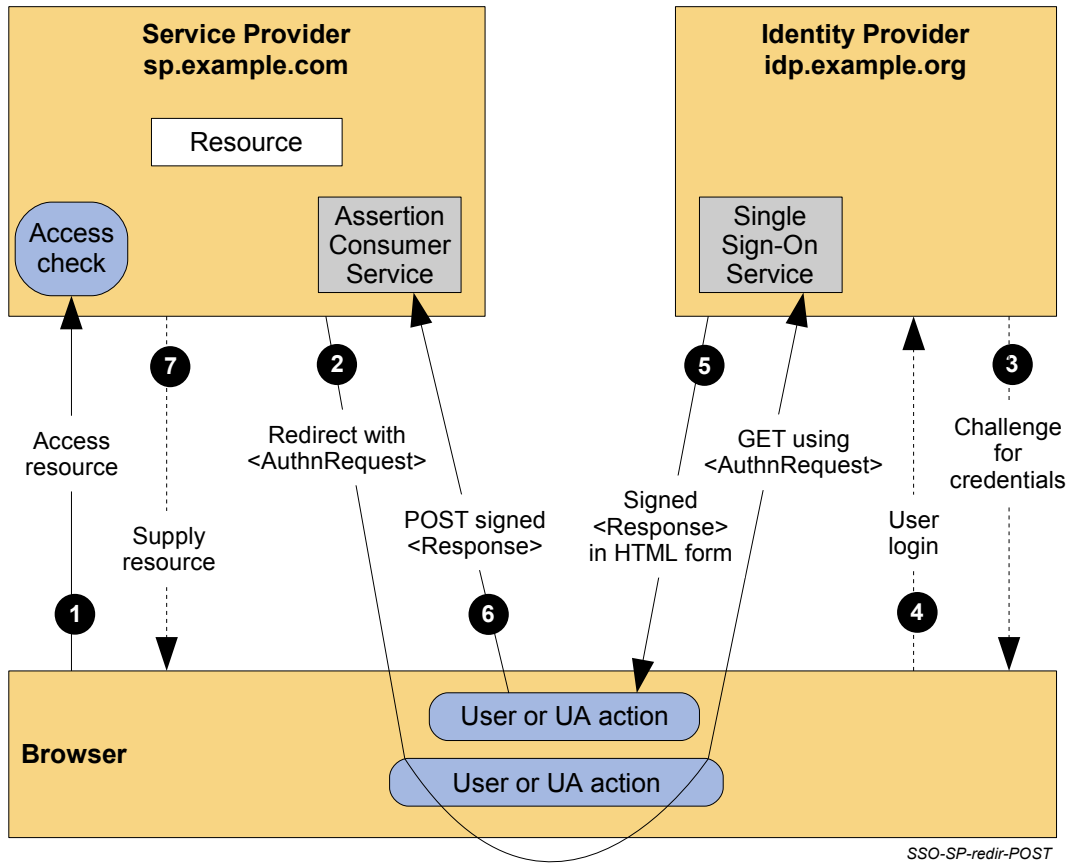


Figure 12: SP-Initiated SSO with Redirect and POST Bindings

830 The processing is as follows:

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

```

837 <samlp:AuthnRequest
838   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
839   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
840   ID="identifier_1"
841   Version="2.0"
842   IssueInstant="2004-12-05T09:21:59Z"
843   AssertionConsumerServiceIndex="1">
844   <saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>
845   <samlp:NameIDPolicy
846     AllowCreate="true"
847     Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>
848 </samlp:AuthnRequest>
  
```

849 The query string is encoded using the DEFLATE encoding. The browser processes the redirect

850 response and issues an HTTP GET request to the IdP's Single Sign-On Service with the  
851 SAMLRequest query parameter. The local state information (or a reference to it) is also included in the  
852 HTTP response encoded in a RelayState query string parameter.

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

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

```
867 <form method="post" action="https://sp.example.com/SAML2/SSO/POST" ...>  
868   <input type="hidden" name="SAMLResponse" value="response" />  
869   <input type="hidden" name="RelayState" value="token" />  
870   ...  
871   <input type="submit" value="Submit" />  
872 </form>
```

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

```
875 <saml:Response  
876   xmlns:saml="urn:oasis:names:tc:SAML:2.0:protocol"  
877   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
878   ID="identifier_2"  
879   InResponseTo="identifier_1"  
880   Version="2.0"  
881   IssueInstant="2004-12-05T09:22:05Z"  
882   Destination="https://sp.example.com/SAML2/SSO/POST">  
883   <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>  
884   <saml:Status>  
885     <saml:StatusCode  
886       Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>  
887   </saml:Status>  
888   <saml:Assertion  
889     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
890     ID="identifier_3"  
891     Version="2.0"  
892     IssueInstant="2004-12-05T09:22:05Z">  
893     <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>  
894     <!-- a POSTed assertion MUST be signed -->  
895     <ds:Signature  
896       xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>  
897     <saml:Subject>  
898       <saml:NameID  
899         Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"  
900         Value="3f7b3dcf-1674-4ecd-92c8-1544f346baf8"/>  
901       </saml:NameID>  
902       <saml:SubjectConfirmation  
903         Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">  
904         <saml:SubjectConfirmationData  
905           InResponseTo="identifier_1"  
906           Recipient="https://sp.example.com/SAML2/SSO/POST"  
907           NotOnOrAfter="2004-12-05T09:27:05Z"/>  
908         </saml:SubjectConfirmation>  
909       </saml:Subject>  
910     <saml:Conditions  
911       NotBefore="2004-12-05T09:17:05Z"  
912       NotOnOrAfter="2004-12-05T09:27:05Z">  
913     <saml:AudienceRestriction>  
914       <saml:Audience>https://sp.example.com/SAML2</saml:Audience>  
915     </saml:AudienceRestriction>
```



```
916 </saml:Conditions>
917 <saml:AuthnStatement
918   AuthnInstant="2004-12-05T09:22:00Z"
919   SessionIndex="identifier_3">
920   <saml:AuthnContext>
921     <saml:AuthnContextClassRef>
922       urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
923     </saml:AuthnContextClassRef>
924   </saml:AuthnContext>
925 </saml:AuthnStatement>
926 </saml:Assertion>
927 </samlp:Response>
```

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

```
930 POST /SAML2/SSO/POST HTTP/1.1
931 Host: sp.example.com
932 Content-Type: application/x-www-form-urlencoded
933 Content-Length: nnn
934
935 SAMLResponse=response&RelayState=token
```

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

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

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

### 947 **5.1.3 SP-Initiated SSO: POST/Artifact Bindings**

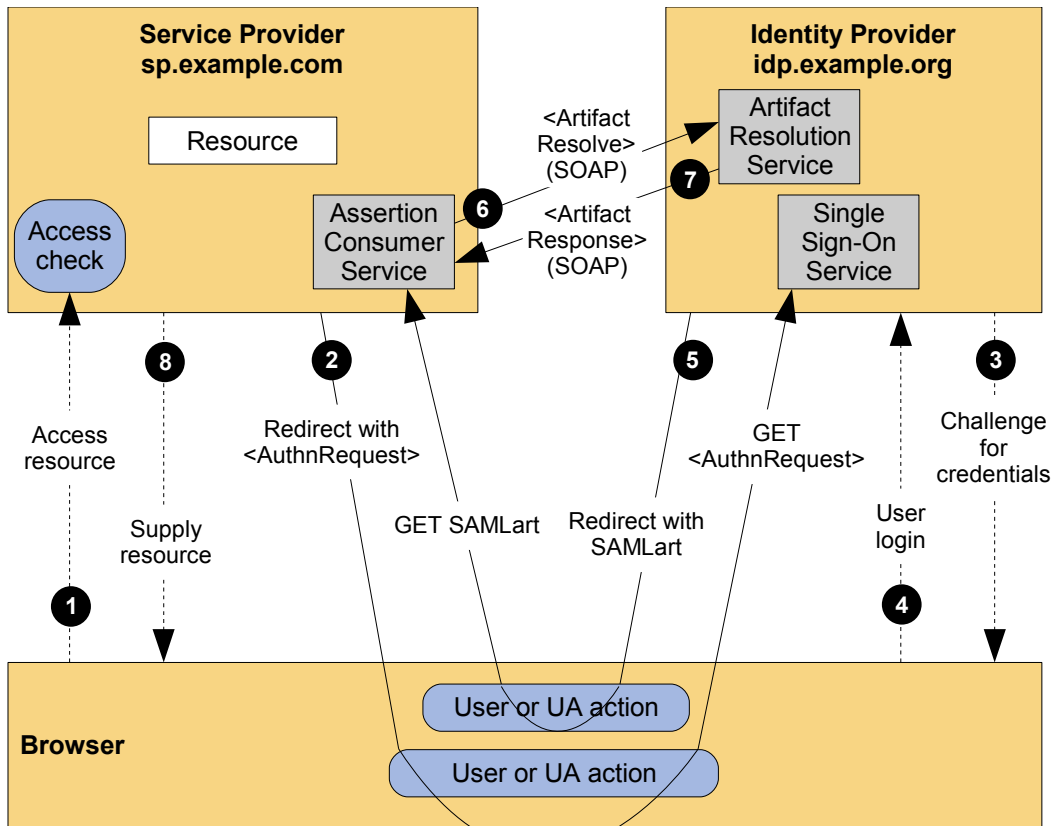
948 This use case again describes an SP-initiated SSO exchange.

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

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

959 Once the SP is in possession of the artifact, it contacts the IdP's Artifact Resolution Service using the  
960 synchronous SOAP binding to obtain the SAML message that corresponds to the artifact. Figure 13  
961 illustrates the message flow.

962



SSO-SP-POST-art

Figure 13: SP-Initiated SSO with Binding

964  
965  
966  
967  
968  
969  
970  
971

972 The processing is as follows:

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

```
979 <form method="post" action="https://idp.example.org/SAML2/SSO/POST" ...>
980 <input type="hidden" name="SAMLRequest" value="request" />
981 <input type="hidden" name="RelayState" value="token" />
982 ...
983 <input type="submit" value="Submit" />
984 </form>
```

985 The RelayState token is an opaque reference to state information maintained at the service  
986 provider. (The RelayState mechanism can leak details of the user's activities at the SP to the IdP  
987 and so the SP should take care in its implementation to protect the user's privacy.) The value of  
988 the SAMLRequest parameter is the base64 encoding of the following <samlp:AuthnRequest>  
989 element:

```
990 <samlp:AuthnRequest  
991   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
992   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
993   ID="identifier_1"  
994   Version="2.0"  
995   IssueInstant="2004-12-05T09:21:59Z"  
996   AssertionConsumerServiceIndex="1">  
997   <saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>  
998   <samlp:NameIDPolicy  
999     AllowCreate="true"  
1000     Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>  
1001 </samlp:AuthnRequest>
```

1002 1. For ease-of-use purposes, the HTML FORM typically will be accompanied by script code that will  
1003 automatically post the form to the destination site (which is the IdP in this case). The browser, due  
1004 either to a user action or execution of an "auto-submit" script, issues an HTTP POST request to send  
1005 the form to the identity provider's Single Sign-On Service.

```
1006 POST /SAML2/SSO/POST HTTP/1.1  
1007 Host: idp.example.org  
1008 Content-Type: application/x-www-form-urlencoded  
1009 Content-Length: nnn  
1010  
1011 SAMLRequest=request&RelayState=token
```

1012 3. The Single Sign-On Service determines whether the user has an existing logon security context at the  
1013 identity provider that meets the default or requested authentication policy requirements. If not, the IdP  
1014 interacts with the browser to challenge the user to provide valid credentials.

1015 4. The user provides valid credentials and a local logon security context is created for the user at the IdP.

1016 5. The IdP Single Sign-On Service issues a SAML assertion representing the user's logon security  
1017 context and places the assertion within a SAML <Response> message. Since the HTTP Artifact  
1018 binding will be used to deliver the SAML Response message, it is not mandated that the assertion be  
1019 digitally signed. The IdP creates an artifact containing the source ID for the idp.example.org site and a  
1020 reference to the <Response> message (the MessageHandle). The HTTP Artifact binding allows the  
1021 choice of either HTTP redirection or an HTML form POST as the mechanism to deliver the artifact to  
1022 the partner. The figure shows the use of redirection.

1023 6. The SP's Assertion Consumer Service now sends a SAML <ArtifactResolve> message containing  
1024 the artifact to the IdP's Artifact Resolution Service endpoint. This exchange is performed using a  
1025 synchronous SOAP message exchange.

```
1026 <samlp:ArtifactResolve  
1027   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
1028   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
1029   ID="identifier_2"  
1030   Version="2.0"  
1031   IssueInstant="2004-12-05T09:22:04Z"  
1032   Destination="https://idp.example.org/SAML2/ArtifactResolution">  
1033   <saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>  
1034   <!-- an ArtifactResolve message SHOULD be signed -->  
1035   <ds:Signature  
1036     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>  
1037   <samlp:Artifact>artifact</samlp:Artifact>  
1038 </samlp:ArtifactResolve>
```

1039 7. The IdP's Artifact Resolution Service extracts the MessageHandle from the artifact and locates the  
1040 original SAML <Response> message associated with it. This message is then placed inside a SAML  
1041 <ArtifactResponse> message, which is returned to the SP over the SOAP channel.

```
1042  
1043 <samlp:ArtifactResponse  
1044   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
```

```

1045 ID="identifier_3"
1046 InResponseTo="identifier_2"
1047 Version="2.0"
1048 IssueInstant="2004-12-05T09:22:05Z">
1049 <!-- an ArtifactResponse message SHOULD be signed -->
1050 <ds:Signature
1051   xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
1052 <samlp:Status>
1053   <samlp:StatusCode
1054     Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
1055 </samlp:Status>
1056 <samlp:Response
1057   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
1058   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1059   ID="identifier_4"
1060   InResponseTo="identifier_1"
1061   Version="2.0"
1062   IssueInstant="2004-12-05T09:22:05Z"
1063   Destination="https://sp.example.com/SAML2/SSO/Artifact">
1064 <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
1065 <ds:Signature
1066   xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
1067 <samlp:Status>
1068   <samlp:StatusCode
1069     Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
1070 </samlp:Status>
1071 <saml:Assertion
1072   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1073   ID="identifier_5"
1074   Version="2.0"
1075   IssueInstant="2004-12-05T09:22:05Z">
1076 <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
1077 <!-- a Subject element is required -->
1078 <saml:Subject>
1079   <saml:NameID
1080     Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
1081     user@mail.example.org
1082   </saml:NameID>
1083   <saml:SubjectConfirmation
1084     Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
1085     <saml:SubjectConfirmationData
1086       InResponseTo="identifier_1"
1087       Recipient="https://sp.example.com/SAML2/SSO/Artifact"
1088       NotOnOrAfter="2004-12-05T09:27:05Z"/>
1089   </saml:SubjectConfirmation>
1090 </saml:Subject>
1091 <saml:Conditions
1092   NotBefore="2004-12-05T09:17:05Z"
1093   NotOnOrAfter="2004-12-05T09:27:05Z">
1094   <saml:AudienceRestriction>
1095     <saml:Audience>https://sp.example.com/SAML2</saml:Audience>
1096   </saml:AudienceRestriction>
1097 </saml:Conditions>
1098 <saml:AuthnStatement
1099   AuthnInstant="2004-12-05T09:22:00Z"
1100   SessionIndex="identifier_5">
1101   <saml:AuthnContext>
1102     <saml:AuthnContextClassRef>
1103       urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
1104     </saml:AuthnContextClassRef>
1105   </saml:AuthnContext>
1106 </saml:AuthnStatement>
1107 </saml:Assertion>
1108 </samlp:Response>
1109 </samlp:ArtifactResponse>

```

1110 The SP extracts and processes the `<Response>` message and then processes the embedded  
1111 assertion in order to create a local logon security context for the user at the SP. Once this is  
1112 completed, the SP retrieves the local state information indicated by the `RelayState` data to recall the  
1113 originally-requested resource URL. It then sends an HTTP redirect response to the browser directing it  
1114 to access the originally requested resource (not shown).

1115 8. An access check is made to establish whether the user has the correct authorization to access the  
1116 resource. If the access check passes, the resource is then returned to the browser.

1117 **5.1.4 IdP-Initiated SSO: POST Binding**

1118 In addition to supporting the new SP-Initiated web SSO use cases, SAML v2 continues to support the IdP-  
 1119 initiated web SSO use cases originally supported by SAML v1. In an IdP-initiated use case, the identity  
 1120 provider is configured with specialized links that refer to the desired service providers. These links actually  
 1121 refer to the local IdP's Single Sign-On Service and pass parameters to the service identifying the remote  
 1122 SP. So instead of visiting the SP directly, the user accesses the IdP site and clicks on one of the links to  
 1123 gain access to the remote SP. This triggers the creation of a SAML assertion that, in this example, will be  
 1124 transported to the service provider using the HTTP POST binding.

1125 Figure 14 shows the process flow for an IdP-initiated web SSO exchange.

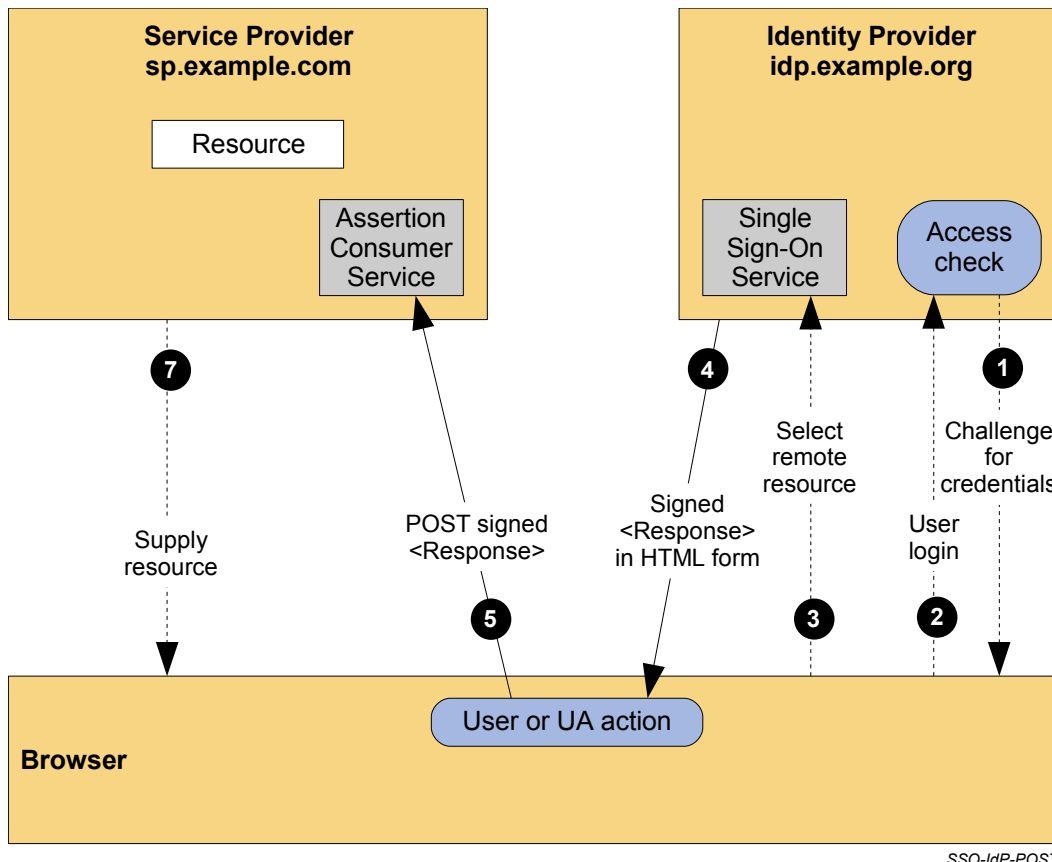


Figure 14: IdP-Initiated SSO with POST Binding

1127 The processing is as follows:

- 1128 1. If the user does not have a valid local security context at the IdP, at some point the user will be  
 1129 challenged to supply their credentials to the IdP site, idp.example.org.
- 1130 2. The user provides valid credentials and a local logon security context is created for the user at the IdP.
- 1131 3. The user selects a menu option or link on the IdP to request access to an SP web site,  
 1132 sp.example.com. This causes the IdP's Single Sign-On Service to be called.
- 1133 4. The Single Sign-On Service builds a SAML assertion representing the user's logon security context.  
 1134 Since a POST binding is going to be used, the assertion is digitally signed before it is placed within a  
 1135 SAML <Response> message. The <Response> message is then placed within an HTML FORM as  
 1136 a hidden form control named SAMLResponse. (If the convention for identifying a specific application  
 1137 resource at the SP is supported at the IdP and SP, the resource URL at the SP is also encoded into  
 1138 the form using a hidden form control named RelayState.) The Single Sign-On Service sends the  
 1139 HTML form back to the browser in the HTTP response. For ease-of-use purposes, the HTML FORM

- 1140 typically will contain script code that will automatically post the form to the destination site.
- 1141 5. The browser, due either to a user action or execution of an “auto-submit” script, issues an HTTP POST  
1142 request to send the form to the SP’s Assertion Consumer Service. The service provider’s Assertion  
1143 Consumer Service obtains the <Response> message from the HTML FORM for processing. The  
1144 digital signature on the SAML assertion must first be validated and then the assertion contents are  
1145 processed in order to create a local logon security context for the user at the SP. Once this completes,  
1146 the SP retrieves the RelayState data (if any) to determine the desired application resource URL and  
1147 sends an HTTP redirect response to the browser directing it to access the requested resource (not  
1148 shown).
- 1149 6. An access check is made to establish whether the user has the correct authorization to access the  
1150 resource. If the access check passes, the resource is then returned to the browser.

## 1151 5.2 ECP Profile

1152 The browser SSO profile discussed above works with commercial browsers that have no special  
1153 capabilities. This section describes a SAML V2.0 profile that takes into account enhanced client devices  
1154 and proxy servers.

### 1155 5.2.1 Introduction

1156 The Enhanced Client and Proxy (ECP) Profile supports several SSO use cases, in particular:

- 1157 ● Clients with capabilities beyond those of a browser, allowing them to more actively participate in  
1158 IDP discovery and message flow.
- 1159 ● Using a proxy server, for example a WAP gateway in front of a mobile device which has limited  
1160 functionality.
- 1161 ● When other bindings are precluded (e.g. where the client does not support redirects, or when auto  
1162 form post is not possible without Javascript, or when the artifact binding is ruled out because the  
1163 identity provider and service provider cannot directly communicate).

1164 Figure 15 illustrates two such use cases for using the ECP Profile.

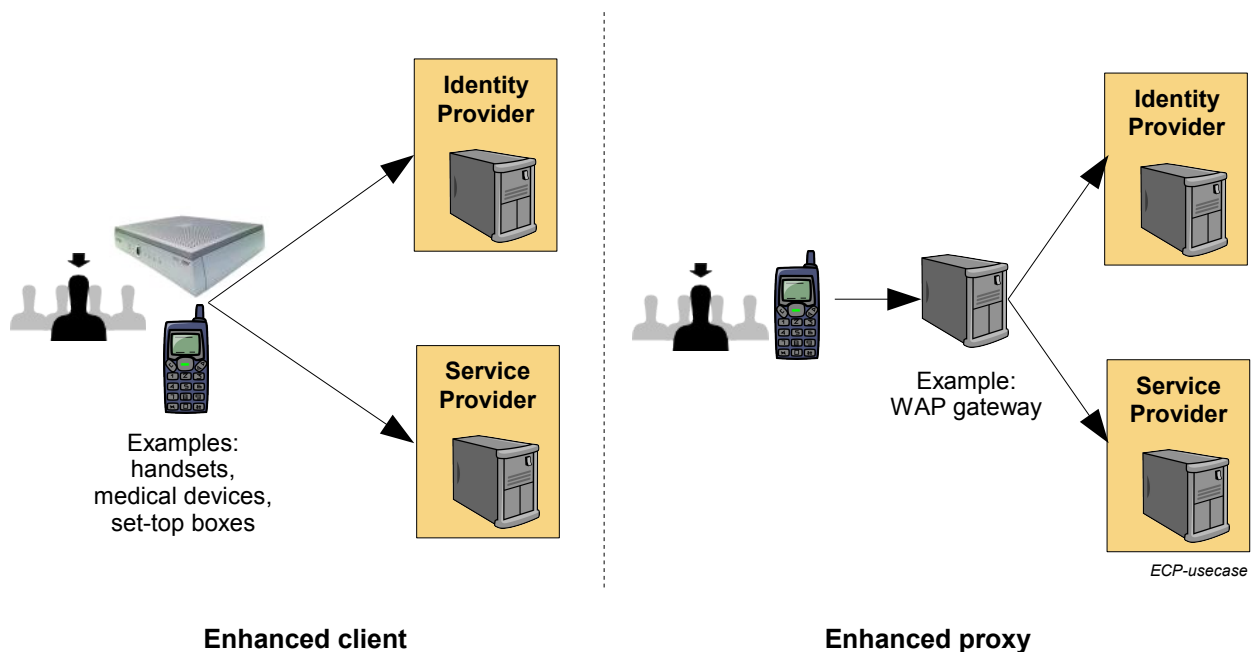
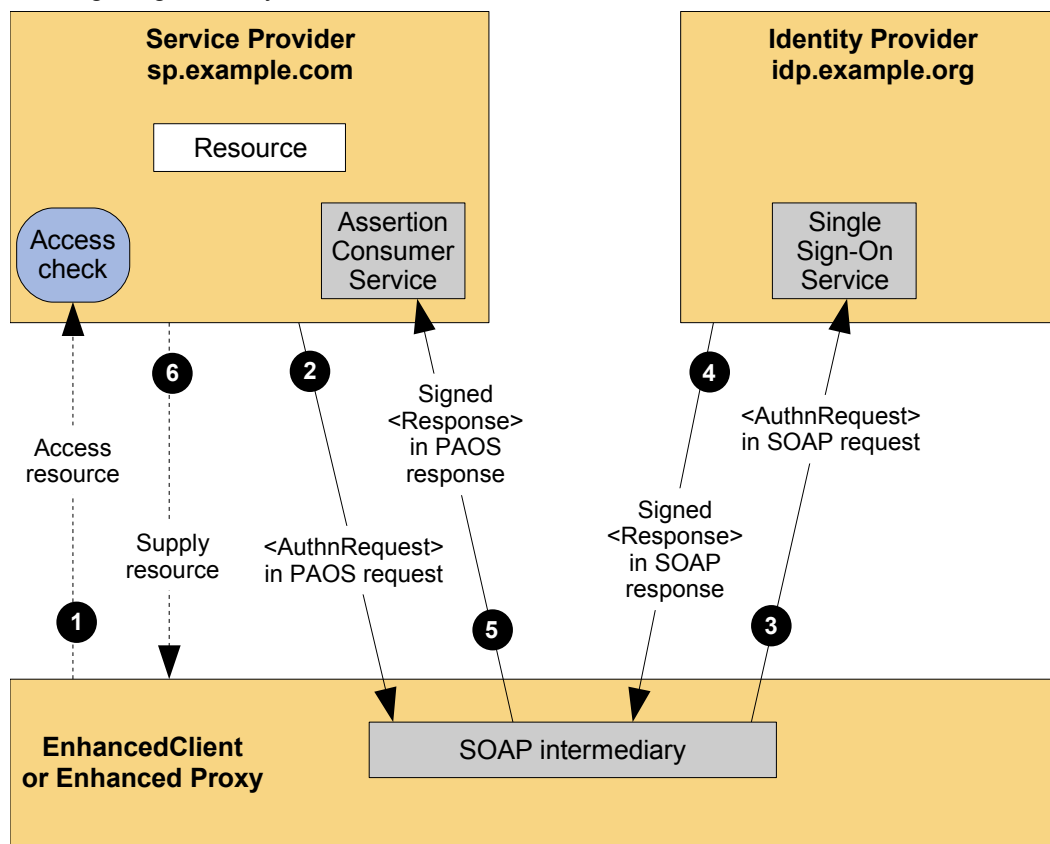


Figure 15: Enhanced Client/Proxy Use Cases

1166 The ECP profile defines a single binding – PAOS (Reverse SOAP). The profile uses SOAP headers and  
 1167 SOAP bodies to transport SAML <AuthnRequest> and SAML <Response> messages between the  
 1168 service provider and the identity provider.

## 1169 5.2.2 ECP Profile Using PAOS Binding

1170 Figure 16 shows the message flows between the ECP, service provider and identity provider. The ECP is  
 1171 shown as a single logical entity.



SSO-ECP-PAOS

Figure 16: SSO Using ECP with the PAOS Binding

1173 The processing is as follows:

- 1174 1. The ECP wishes to gain access to a resource on the service provider, sp.example.com. The ECP will  
 1175 issue an HTTP request for the resource. The HTTP request contains a PAOS HTTP header defining  
 1176 that the ECP service is to be used.
- 1177 2. Accessing the resource requires that the principal has a valid security context, and hence a SAML  
 1178 assertion needs to be supplied to the service provider. In the HTTP response to the ECP an  
 1179 <AuthnRequest> is carried within a SOAP body. Additional information, using the PAOS binding, is  
 1180 provided back to the ECP
- 1181 3. After some processing in the ECP the <AuthnRequest> is sent to the appropriate identity provider  
 1182 using the SAML SOAP binding.
- 1183 4. The identity provider validates the <AuthnRequest> and sends back to the ECP a SAML  
 1184 <Response>, again using the SAML SOAP binding.
- 1185 5. The ECP extracts the <Response> and forwards it to the service provider as a PAOS response.
- 1186 6. The service provider sends to the ECP an HTTP response containing the resource originally



1187 requested.

## 1188 **5.3 Single Logout Profile**

1189 Once single sign-on has been achieved, several individual sessions with service providers share a single  
1190 authentication context. This section discusses SAML's profile for single logout, which allows for reversing  
1191 the sign-on process with all of these providers at once.

1192 One representative flow option is discussed in detail: single logout that is initiated at one SP and results in  
1193 logout from multiple SPs.

### 1194 **5.3.1 Introduction**

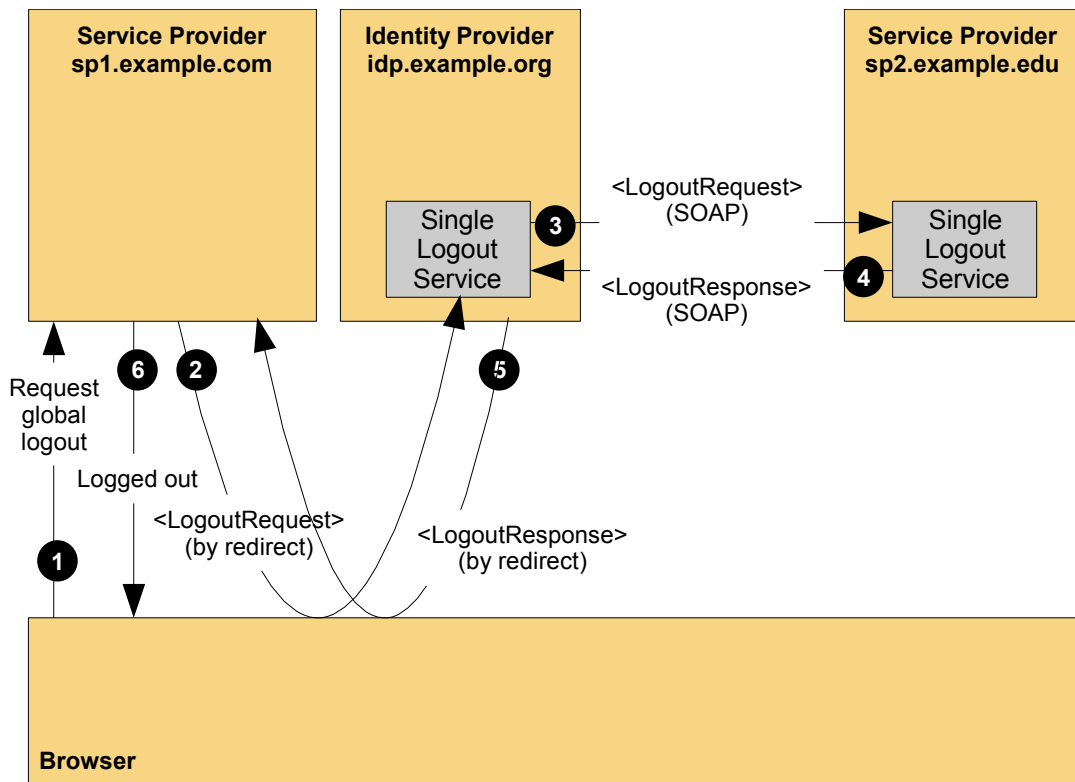
1195 Single logout permits near real-time session logout of a user from all participants in a session. A request  
1196 can be issued by any session participant to request that the session is to be ended. As specified in the  
1197 SAML Conformance specification, the SAML logout messages can be exchanged over either the  
1198 synchronous SOAP over HTTP binding or using the asynchronous HTTP Redirect, HTTP POST, or HTTP  
1199 Artifact bindings. Note that a browser logout operation often requires access to local authentication  
1200 cookies stored in the user's browser. Thus, asynchronous front-channel bindings are typically preferred for  
1201 these exchanges in order to force the browser to visit each session participant to permit access to the  
1202 browser cookies. However, user interaction with the browser might interrupt the process of visiting each  
1203 participant and thus, the result of the logout process cannot be guaranteed.

### 1204 **5.3.2 SP-Initiated Single Logout with Multiple SPs**

1205 In the example shown in Figure 16, a user visiting the sp1.example.com service provider web site decides  
1206 that they wish to log out of their web SSO session. The identity provider idp.example.org determines that  
1207 other service providers are also participants in the web SSO session, and thus sends <LogoutRequest>  
1208 messages to each of the other SPs. In this example, different bindings are used for the exchanges  
1209 between the various pairs of session participants. The SP initiating the single logout uses the HTTP  
1210 Redirect binding with the IdP, while the IdP uses a back-channel SOAP over HTTP binding to  
1211 communicate with the other SP sp2.example.edu.

1212

1213



SLO-SP-init-mult

Figure 17: SP-initiated Single Logout with Multiple SPs

1214 The processing is as follows:

- 1215 1. A user was previously authenticated by the idp.example.org identity provider and is interacting with the  
 1216 sp1.example.com service provider through a web SSO session. The user decides to terminate their  
 1217 session and selects a link on the SP that requests a global logout.
- 1218 2. The SP sp1.example.com destroys the local authentication session state for the user and then sends  
 1219 the idp.example.org identity provider a SAML <LogoutRequest> message requesting that the user's  
 1220 session be logged out. The request identifies the principal to be logged out using a <NameID>  
 1221 element, as well as providing a <SessionIndex> element to uniquely identify the session being  
 1222 closed. The <LogoutRequest> message is digitally signed and then transmitted using the HTTP  
 1223 Redirect binding. The identity provider verifies that the <LogoutRequest> originated from a known  
 1224 and trusted service provider. The identity provider processes the request and destroys any local  
 1225 session information for the user.
- 1226 3. Having determined that other service providers are also participants in the web SSO session, the  
 1227 identity provider similar sends a <LogoutRequest> message to those providers. In this example,  
 1228 there is one other service provider, sp2.example.edu. The <LogoutRequest> message is sent using  
 1229 the SOAP over HTTP Binding.
- 1230 4. The service provider sp2.example.edu returns a <LogoutResponse> message containing a suitable  
 1231 status code response to the identity provider. The response is digitally signed and returned (in this  
 1232 case) using the SOAP over HTTP binding.
- 1233 5. The identity provider returns a <LogoutResponse> message containing a suitable status code  
 1234 response to the original requesting service provider, sp1.example.com. The response is digitally  
 1235 signed and returned (in this case) using the HTTP Redirect binding.
- 1236 6. Finally, the service provider sp1.example.com informs the user that they are logged out of all the  
 1237 providers.

## 1238 **5.4 Establishing and Managing Federated Identities**

1239 Thus far, the use case examples that have been presented have focused on the SAML message  
1240 exchanges required to facilitate the implementation of web single sign-on solutions. This section  
1241 examines issues surrounding how these message exchanges are tied to individual local and federated  
1242 user identities shared between participants in the solution.

### 1243 **5.4.1 Introduction**

1244 The following sections describe mechanisms supported by SAML for establishing and managing federated  
1245 identities. The following use cases are described:

- 1246 • **Federation via Out-of-Band Account Linking:** The establishment of federated identities for users  
1247 and the association of those identities to local user identities can be performed without the use of  
1248 SAML protocols and assertions. This was the only style of federation supported by SAML V1 and is  
1249 still supported in SAML v2.0.
- 1250 • **Federation via Persistent Pseudonym Identifiers:** An identity provider federates the user's local  
1251 identity principal with the principal's identity at the service provider using a persistent SAML name  
1252 identifier.
- 1253 • **Federation via Transient Pseudonym Identifiers:** A temporary identifier is used to federate  
1254 between the IdP and the SP for the life of the user's web SSO session.
- 1255 • **Federation via Identity Attributes:** Attributes of the principal, as defined by the identity provider,  
1256 are used to link to the account used at the service provider.
- 1257 • **Federation Termination:** termination of an existing federation.

1258 To simplify the examples, not all possible SAML bindings are illustrated.

1259 All the examples are based on the use case scenarios originally defined in Section 3.2, with  
1260 [airline.example.com](http://airline.example.com) being the identity provider.

### 1261 **5.4.2 Federation Using Out-of-Band Account Linking**

1262 In this example, shown in Figure 18, the user John has accounts on both [airline.example.com](http://airline.example.com) and  
1263 [cars.example.co.uk](http://cars.example.co.uk) each using the same local user ID (**john**). The identity data stores at both sites are  
1264 synchronized by some out-of-band means, for example using database synchronization or off-line batch  
1265 updates.

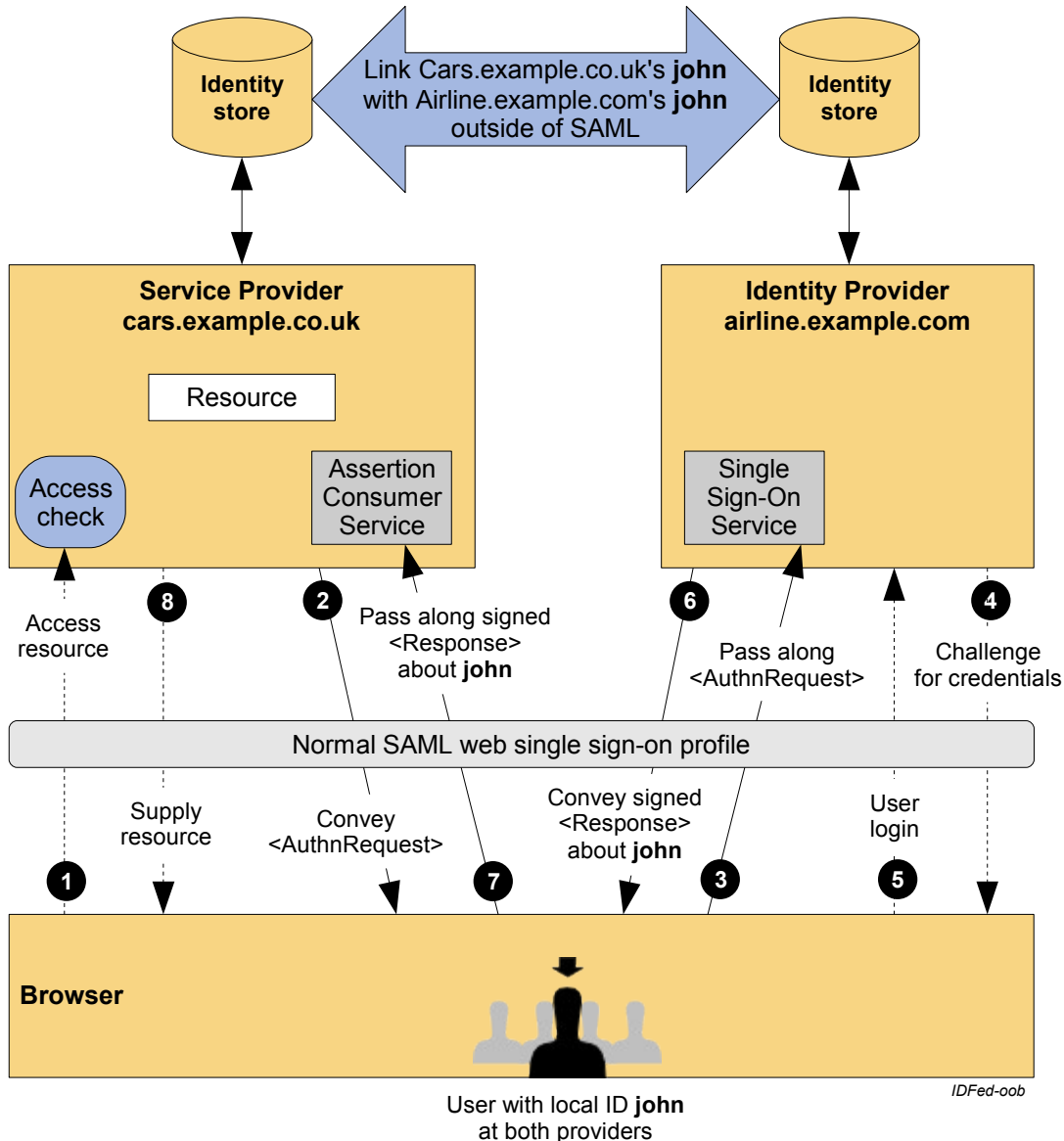


Figure 18: Identity Federation with Out-of-Band Account Linking

1267 The processing is as follows:

- 1268 1. The user is challenged to supply their credentials to the site [airline.example.com](http://airline.example.com).
- 1269 2. The user successfully provides their credentials and has a security context with the
- 1270 [airline.example.com](http://airline.example.com) identity provider.
- 1271 3. The user selects a menu option (or function) on the [airline.example.com](http://airline.example.com) application that means the
- 1272 user wants to access a resource or application on [cars.example.co.uk](http://cars.example.co.uk). The [airline.example.com](http://airline.example.com)
- 1273 identity provider sends a HTML form back to the browser. The HTML FORM contains a SAML
- 1274 response, within which is a SAML assertion about user john.
- 1275 4. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing the
- 1276 SAML response to be sent to the [cars.example.co.uk](http://cars.example.co.uk) Service provider.

1277 The [cars.example.co.uk](http://cars.example.co.uk) service provider's Assertion Consumer Service validates the digital signature on

1278 the SAML Response. If this, and the assertion validate correctly it creates a local session for user john,

1279 based on the local john account. It then sends an HTTP redirect to the browser causing it to access the  
 1280 TARGET resource, with a cookie that identifies the local session. An access check is then made to  
 1281 establish whether the user john has the correct authorization to access the cars.example.co.uk web site  
 1282 and the TARGET resource. The TARGET resource is then returned to the browser.

### 1283 5.4.3 Federation Using Persistent Pseudonym Identifiers

1284 In this use case scenario, the partner sites take advantage of SAML V2.0's ability to dynamically establish  
 1285 a federated identity for a user as part of the web SSO message exchange. SAML V2.0 provides the  
 1286 NameIDPolicy element on the AuthnRequest to allow the SP to constrain such dynamic behaviour. The  
 1287 user **jd**oe on *cars.example.co.uk* wishes to federate this account with his **john** account on the IdP,  
 1288 *airline.example.com*. Figure 19 illustrates dynamic identity federation using persistent pseudonym  
 1289 identifiers in an SP-initiated web SSO exchange.

1290

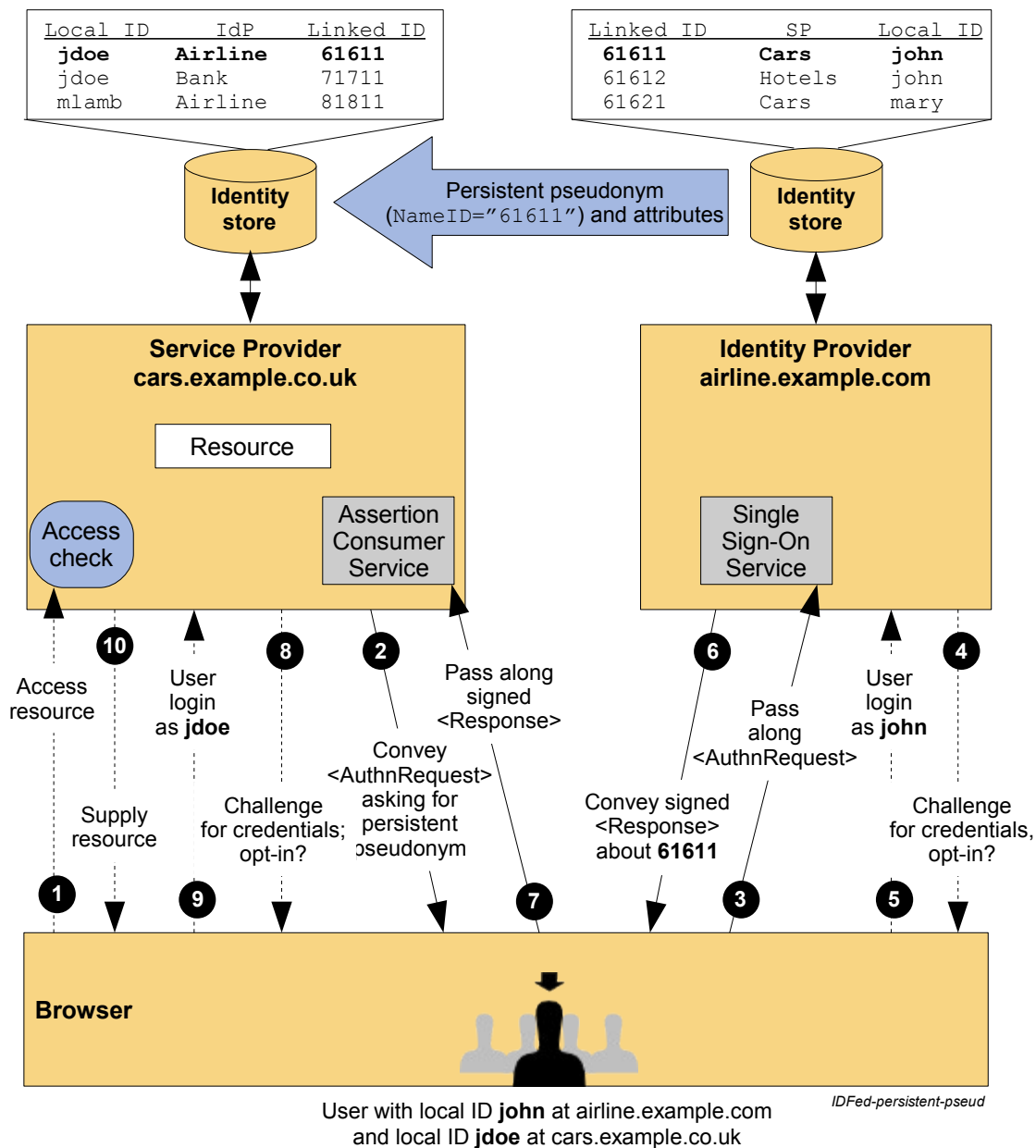


Figure 19: SP-Initiated Identity Federation with Persistent Pseudonym

1292 The processing is as follows:

- 1293 1. The user attempts to access a resource on [cars.example.co.uk](http://cars.example.co.uk). The user does not have any current  
1294 logon session (i.e. security context) on this site, and is unknown to it. The resource that the user  
1295 attempted to access is saved as `RelayState` information.
- 1296 2. The service provider uses the HTTP Redirect Binding to send the user to the Single Sign-On Service at  
1297 the identity provider ([airline.example.com](http://airline.example.com)). The HTTP redirect includes a SAML `<AuthnRequest>`  
1298 message requesting that the identity provider provide an assertion using a persistent name identifier  
1299 for the user. As the service provider desires the IdP have the flexibility to generate a new identifier for  
1300 the user should one not already exist, the SP sets the `AllowCreate` attribute on the `NameIDPolicy`  
1301 element to 'true'.
- 1302 3. The user will be challenged to provide valid credentials.
- 1303 4. The user provides valid credentials identifying himself as **john** and a local security context is created  
1304 for the user at the IdP.
- 1305 5. The Single Sign-On Service looks up user **john** in its identity store and, seeing that the `AllowCreate`  
1306 attribute allows it to, creates a persistent name identifier (`61611`) to be used for the session at the  
1307 service provider. It then builds a signed SAML web SSO assertion where the subject uses a transient  
1308 name identifier format. The name **john** is not contained anywhere in the assertion. Note that  
1309 depending on the partner agreements, the assertion might also contain an attribute statement  
1310 describing identity attributes about the user (e.g. their membership level).
- 1311 6. The browser, due either to a user action or execution of an "auto-submit" script, issues an HTTP POST  
1312 request to send the form to the service provider's Assertion Consumer Service.
- 1313 7. The [cars.example.co.uk](http://cars.example.co.uk) service provider's Assertion Consumer service validates the digital signature  
1314 on the SAML Response and validates the SAML assertion. The supplied name identifier is then used  
1315 to determine whether a previous federation has been established. If a previous federation has been  
1316 established (because the name identifier maps to a local account) then go to step 9. If no federation  
1317 exists for the persistent identifier in the assertion, then the SP needs to determine the local identity to  
1318 which it should be assigned. The user will be challenged to provide local credentials at the SP.  
1319 Optionally the user might first be asked whether he would like to federate the two accounts.
- 1320 8. The user provides valid credentials and identifies his account at the SP as **jdoe**. The persistent name  
1321 identifier is then stored and registered with the **jdoe** account along with the name of the identity  
1322 provider that created the name identifier.
- 1323 9. A local logon session is created for user **jdoe** and an access check is then made to establish whether  
1324 the user **jdoe** has the correct authorization to access the desired resource at the [cars.example.co.uk](http://cars.example.co.uk)  
1325 web site (the resource URL was retrieved from state information identified by the `RelayState`  
1326 information).
- 1327 10. If the access check passes, the desired resource is returned to the browser.

#### 1328 **5.4.4 Federation Using Transient Pseudonym Identifiers**

1329 The previous use case showed the use of persistent identifiers. So what if you do not want to establish a  
1330 permanent federated identity between the partner sites? This is where the use of transient identifiers are  
1331 useful. Transient identifiers allow you to:

- 1332 • Completely avoid having to manage user ID's and passwords at the service provider.
- 1333 • Have a scheme whereby the service provider does not have to manage specific user accounts, for  
1334 instance it could be a site with a "group-like" access policy.
- 1335 • Support a truly anonymous service

1336 As with the Persistent Federation use cases, one can have SP and IdP-initiated variations. Figure 20  
1337 shows the SP-initiated use case using transient pseudonym name identifiers.

1338

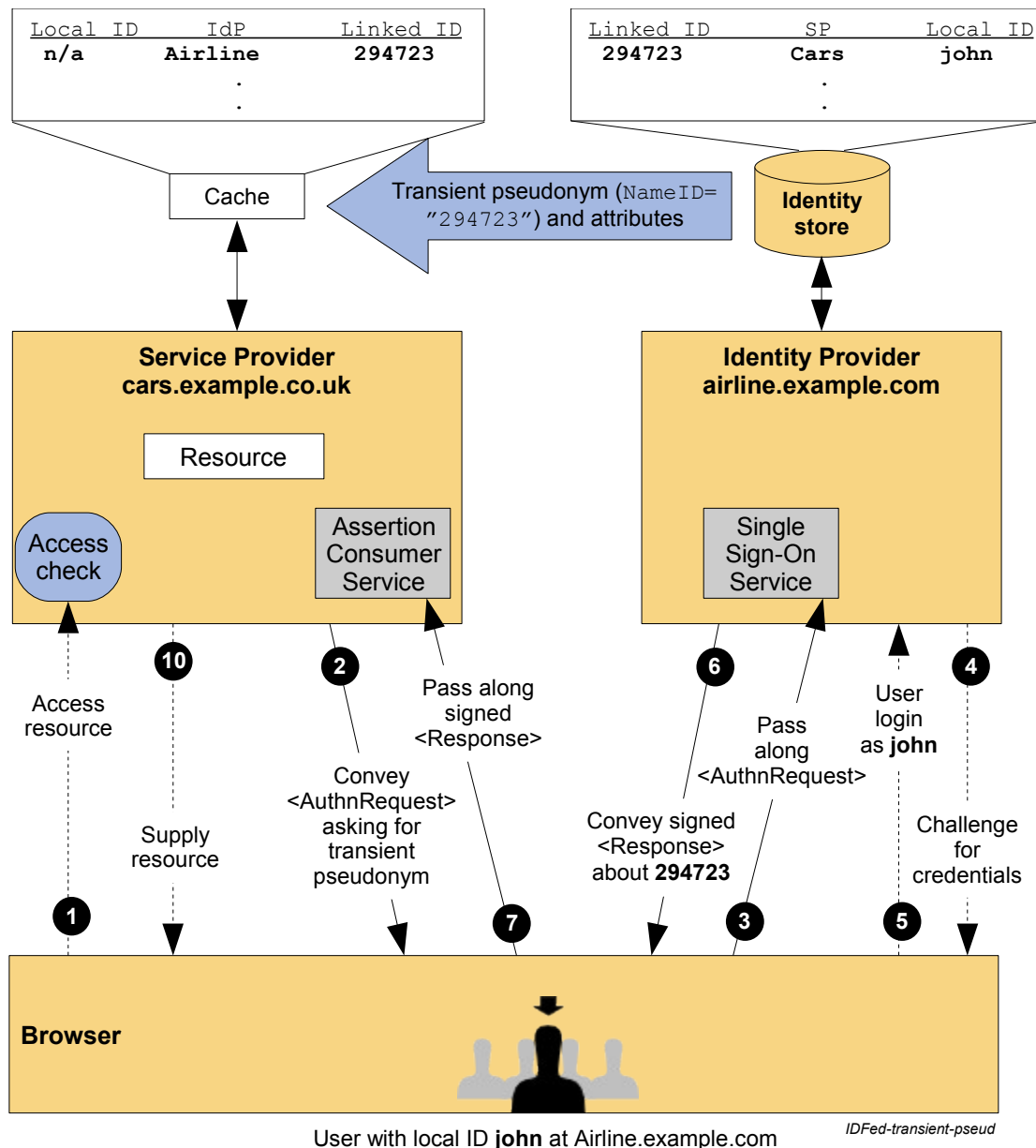


Figure 20: SP-Initiated Identity Federation with Transient Pseudonym

- 1339 The processing is as follows:
- 1340 1. The user attempts to access a resource on [cars.example.co.uk](http://cars.example.co.uk). The user does not have any current
- 1341 logon session (i.e. security context) on this site, and is unknown to it. The resource that the user
- 1342 attempted to access is saved as `RelayState` information.
- 1343 2. The service provider uses the HTTP Redirect Binding to send the user to the Single Sign-On Service at the
- 1344 identity provider ([airline.example.com](http://airline.example.com)). The HTTP redirect includes a SAML `<AuthnRequest>`
- 1345 message requesting that the identity provider provide an assertion using a transient name identifier for
- 1346 the user.
- 1347 3. The user will be challenged to provide valid credentials at the identity provider.
- 1348 4. The user provides valid credentials identifying himself as **john** and a local security context is created
- 1349 for the user at the IdP.
- 1350 5. The Single Sign-On Service looks up user **john** in its identity store and creates a transient name

1351 identifier (294723) to be used for the session at the service provider. It then builds a signed SAML web  
 1352 SSO assertion where the subject uses a transient name identifier format. The name **john** is not  
 1353 contained anywhere in the assertion. The assertion also contains an attribute statement with a  
 1354 membership level attribute ("Gold" level). The assertion is placed in a SAML response message and  
 1355 the IdP uses the HTTP POST Binding to send the Response message to the service provider.

1356 6. The browser, due either to a user action or execution of an "auto-submit" script, issues an HTTP POST  
 1357 request to send the form to the service provider's Assertion Consumer Service.

1358 7. The [cars.example.co.uk](http://cars.example.co.uk) service provider's Assertion Consumer service validates the SAML Response  
 1359 and SAML assertion. The supplied transient name identifier is then used to dynamically create a  
 1360 session for the user at the SP. The membership level attribute might be used to perform an access  
 1361 check on the requested resource and customize the content provided to the user.

1362 8. If the access check passes, the requested resource is then returned to the browser.

1363 While not shown in the diagram, the transient identifier remains active for the life of the user authentication  
 1364 session. If needed, the SP could use the identifier to make SAML attribute queries back to an attribute  
 1365 authority at [airline.example.com](http://airline.example.com) to obtain other identity attributes about the user in order to customize their  
 1366 service provider content, etc.

### 1367 5.4.5 Federation Termination

1368 This example builds upon the previous federation example using persistent pseudonym identifiers and  
 1369 shows how a federation can be terminated. In this case the **jd** account on [cars.example.co.uk](http://cars.example.co.uk) service  
 1370 provider has been deleted, hence it wishes to terminate the federation with [airline.example.com](http://airline.example.com) for this  
 1371 user.

1372 The Terminate request is sent to the identity provider using the Name Identifier Management Protocol,  
 1373 specifically using the <ManageNameIDRequest>. The example shown in Figure 21 uses the SOAP over  
 1374 HTTP binding which demonstrates a use of the back channel. Bindings are also defined that permit the  
 1375 request (and response) to be sent via the browser using asynchronous "front-channel" bindings, such as  
 1376 the HTTP Redirect, HTTP POST, or Artifact bindings.

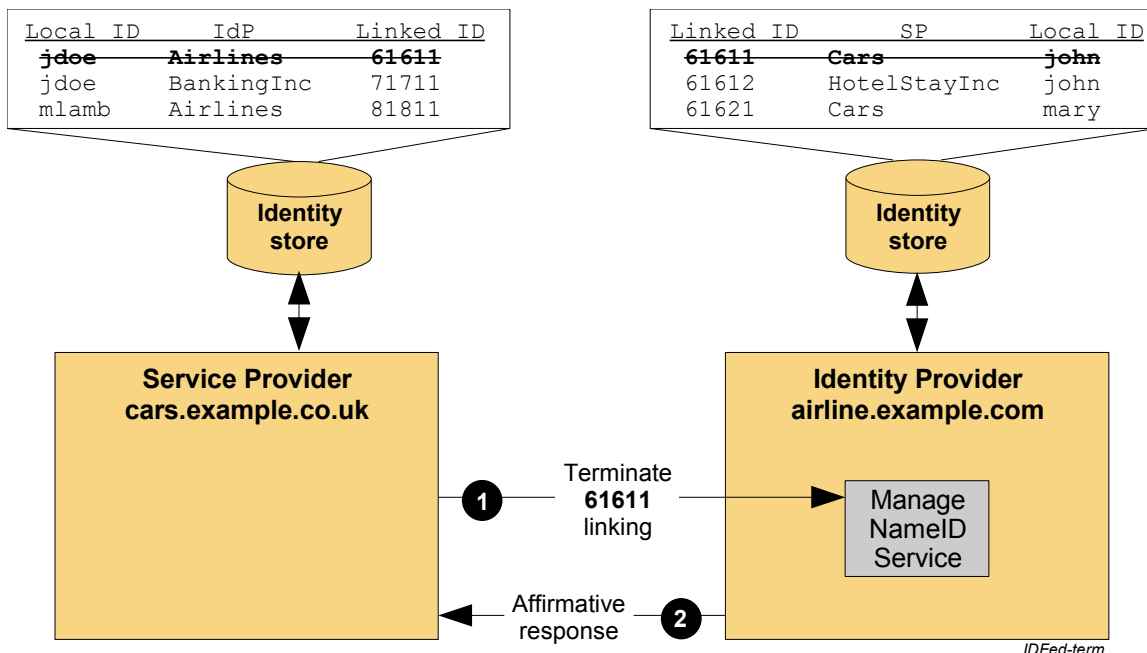


Figure 21: Identity Federation Termination

1378 In this example the processing is as follows:

1379 1. The service provider, [cars.example.co.uk](http://cars.example.co.uk), determines that the local account, **jd**, should no longer be



1380 federated. An example of this could be that the account has been deleted. The service provider sends  
1381 to the [airline.example.com](http://airline.example.com) identity provider a <ManageIDNameRequest> defining that the persistent  
1382 identifier (previously established) must no longer be used. The request is carried in a SOAP message  
1383 which is transported using HTTP, as defined by the SAML SOAP binding. The request is also digitally  
1384 signed by the service provider.

1385 2. The identity provider verifies the digital signature ensuring that the <ManageIDNameRequest>  
1386 originated from a known and trusted service provider. The identity Provider processes the request  
1387 and returns a <ManageIDNameResponse> containing a suitable status code response. The response  
1388 is carried within a SOAP over HTTP message and is digitally signed.

## 1389 **5.5 Use of Attributes**

1390 As explained in Section 3.2, in describing the web single sign-on use case, the SAML assertion  
1391 transferred from an identity provider to a service provider may include attributes describing the user. The  
1392 ability to transfer attributes within an assertion is a powerful SAML feature and it may also be combined  
1393 with the forms of identity federation described above.

1394 The following are some typical use patterns:

- 1395 • Transfer of profile information

1396 Attributes may be used to convey user profile information from the identity provider to the service  
1397 provider. This information may be used to provide personalized services at the service provider, or to  
1398 augment or even create a new account for the user at the service provider. The user should be  
1399 informed about the transfer of information, and, if required, user consent explicitly obtained.

- 1400 • Authorization based on attributes

1401 In this model, the attributes provided in the SAML assertion by the identity provider are used to  
1402 authorize specific services at the service provider. The service provider and identity provider need  
1403 prior agreement (out of band) on the attribute names and values included in the SAML assertion. An  
1404 interesting use of this pattern which preserves user anonymity but allows for differential classes of  
1405 service is found in Shibboleth : federation using transient pseudonyms combined with authorization  
1406 based on attributes.

## 1407 **6 Extending and Profiling SAML for Use in Other** 1408 **Frameworks**

1409 SAML's components are modular and extensible. The SAML Assertions and Protocols specification has a  
1410 section describing the basic extension features provided. The SAML Profiles specification provides  
1411 guidelines on how to define new profiles and attribute profiles. The SAML Bindings specification likewise  
1412 offers guidelines for defining new bindings.

1413 As a result of this flexibility, SAML has been adopted for use with several other standard frameworks.  
1414 Following are some examples.

### 1415 **6.1 Web Services Security (WS-Security)**

1416 SAML assertions can be conveyed by means other than the SAML Request/Response protocols or  
1417 profiles defined by the SAML specification set. One example of this is their use with Web Services  
1418 Security (WS-Security), which is a set of specifications that define means for providing security protection  
1419 of SOAP messages. The services provided by WS-Security are authentication, data integrity, and  
1420 confidentiality.

1421 WS-Security defines a `<Security>` element that may be included in a SOAP message header. This  
1422 element specifies how the message is protected. WS-Security makes use of mechanisms defined in the  
1423 W3C XML Signature and XML Encryption specifications to sign and encrypt message data in both the  
1424 SOAP header and body. The information in the `<Security>` element specifies what operations were  
1425 performed and in what order, what keys were used for these operations, and what attributes and identity  
1426 information are associated with that information. WS-Security also contains other features, such as the  
1427 ability to timestamp the security information and to address it to a specified Role.

1428 In WS-Security, security data is specified using security *tokens*. Tokens can either be binary or structured  
1429 XML. Binary tokens, such as X.509 certificates and Kerberos tickets, are carried in an XML wrapper. XML  
1430 tokens, such as SAML assertions, are inserted directly as sub-elements of the `<Security>` element. A  
1431 Security Token Reference may also be used to refer to a token in one of a number of ways.

1432 WS-Security consists of a core specification, which describes the mechanisms independent of the type of  
1433 token being used, and a number of token profiles which describe the use of particular types of tokens.  
1434 Token profiles cover considerations relating to that particular token type and methods of referencing the  
1435 token using a Security Token Reference. The use of SAML assertions with WS-Security is described in  
1436 the SAML Token Profile.

1437 Because the SAML protocols have a binding to SOAP, it is easy to get confused between that SAML-  
1438 defined binding and the use of SAML assertions by WS-Security. They can be distinguished by their  
1439 purpose, the message format, and the parties involved in processing the messages.

1440 The characteristics of the SAML Request/Response protocol binding over SOAP are as follows:

- 1441 • It is used to obtain SAML assertions for use external to the SOAP message exchange; they play no  
1442 role in protecting the SOAP message.
- 1443 • The SAML assertions are contained within a SAML Response, which is carried in the body of the  
1444 SOAP envelope.
- 1445 • The SAML assertions are provided by a trusted authority and may or may not pertain to the party  
1446 requesting them.

1447 The characteristics of the use of SAML assertions as defined by WS-Security are as follows:

- 1448 • The SAML assertions are carried in a `<Security>` element within the header of the SOAP  
1449 envelope as shown in Figure 22.
- 1450 • The SAML assertions usually play a role in the protection of the message they are carried in;  
1451 typically they contain a key used for digitally signing data within the body of the SOAP message.
- 1452 • The SAML assertions will have been obtained previously and typically pertain to the identity of the  
1453 sender of the SOAP message.

1454 Note that in principle, SAML assertions could be used in both ways in a single SOAP message. In this  
1455 case the assertions in the header would refer to the identity of the Responder (and Requester) of the  
1456 message. However, at this time, SAML has not profiled the use of WS-Security to secure the SOAP  
1457 message exchanges that are made within a SAML deployment.

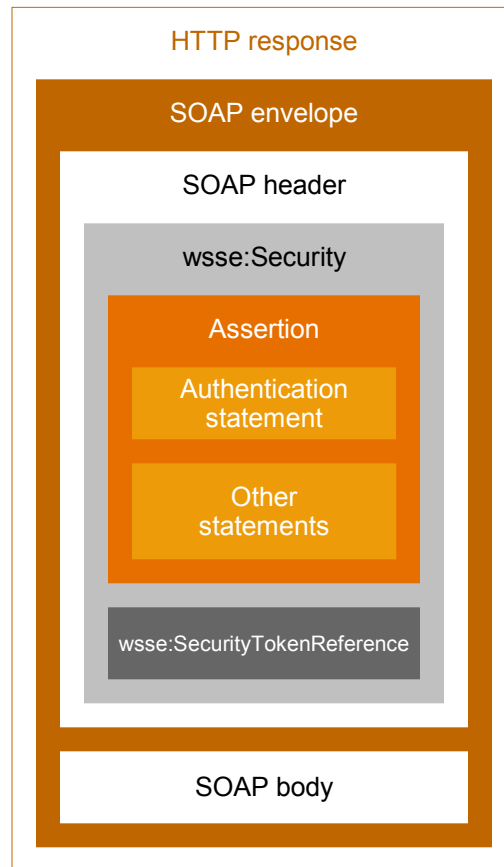


Figure 22: WS-Security with a SAML Token

1459 The following sequence of steps typifies the use of SAML assertions with WS-Security.

1460 A SOAP message sender obtains a SAML assertion by means of the SAML Request/Response protocol  
1461 or other means. In this example, the assertion contains an attribute statement and a subject with a  
1462 confirmation method called *Holder of Key*.

1463 To protect the SOAP message:

- 1464 1. The sender constructs the SOAP message, including a SOAP header with a WS-Security header.  
1465 A SAML assertion is placed within a WS-Security token and included in the security header. The  
1466 key referred to by the SAML assertion is used to construct a digital signature over data in the  
1467 SOAP message body. Signature information is also included in the security header.
- 1468 2. The message receiver verifies the digital signature.
- 1469 3. The information in the SAML assertion is used for purposes such as Access Control and Audit  
1470 logging.

1471 Figure 23 illustrates this usage scenario.

1472

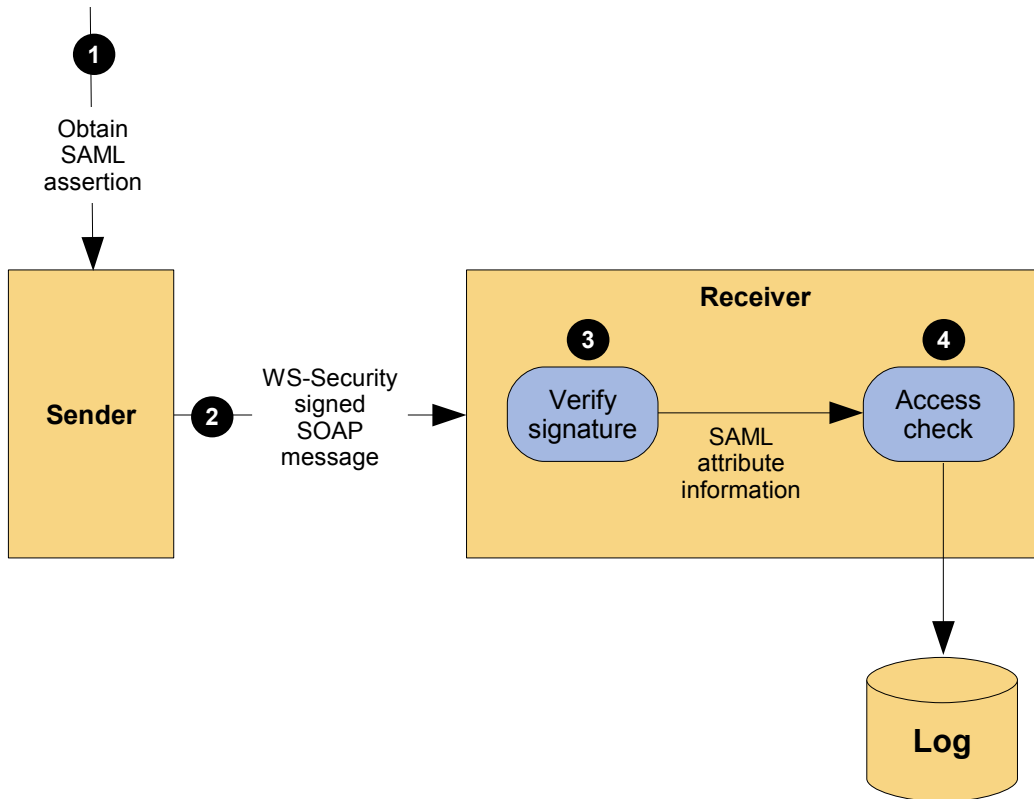


Figure 23: Typical Use of WS-Security with SAML Token

## 1473 6.2 eXtensible Access Control Markup Language (XACML)

1474 SAML assertions provide a means to distribute security-related information that may be used for a number  
 1475 of purposes. One of the most important of these purposes is as input to Access Control decisions. For  
 1476 example, it is common to consider when and how a user authenticated or what their attributes are in  
 1477 deciding if a request should be allowed. SAML does not specify how this information should be used or  
 1478 how access control policies should be addressed. This makes SAML suitable for use in a variety of  
 1479 environments, including ones that existed prior to SAML.

1480 The eXtensible Access Control Markup Language (XACML) is an OASIS Standard that defines the syntax  
 1481 and semantics of a language for expressing and evaluating access control policies. Compatibility with  
 1482 SAML has been a key goal of the XACML TC.

1483 As a result, SAML and XACML can each be used independently of the other, or both can be used  
 1484 together. Figure 24 illustrates the typical use of SAML with XACML.

1485

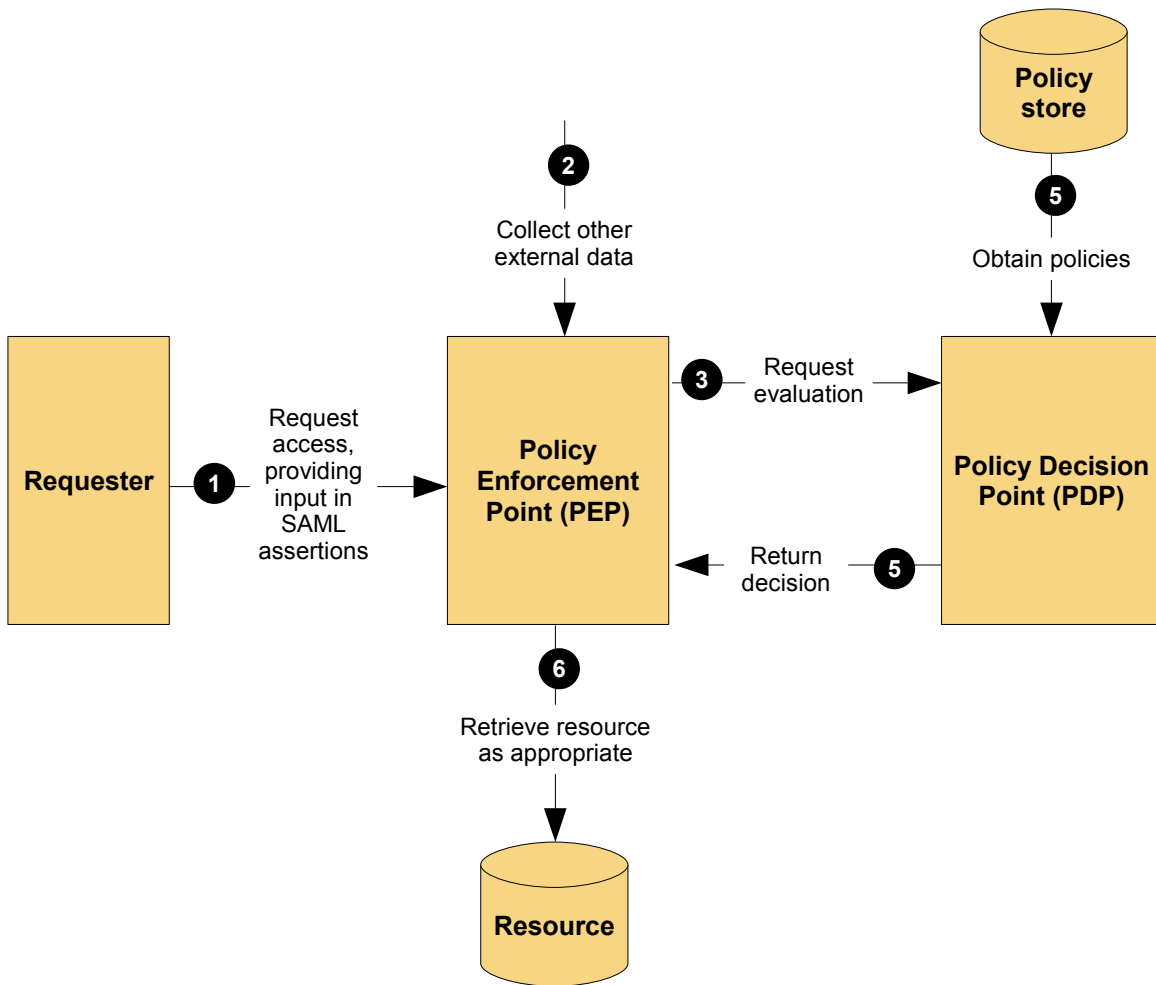


Figure 24: SAML and XACML Integration

1486 Using SAML and XACML in combination would typically involve the following steps.

- 1487 1. An XACML Policy Enforcement Point (PEP) receives a request to access some resource.
- 1488 2. The PEP obtains SAML assertions containing information about the parties to the request,
- 1489 such as the requester, the receiver (if different) or intermediaries. These assertions might
- 1490 accompany the request or be obtained directly from a SAML Authority, depending on the SAML
- 1491 profile used.
- 1492 3. The PEP obtains other information relevant to the request, such as time, date, location, and
- 1493 properties of the resource.
- 1494 4. The PEP presents all the information to a Policy Decision Point (PDP) to decide if the access
- 1495 should be allowed.
- 1496 5. The PDP obtains all the policies relevant to the request and evaluates them, combining
- 1497 conflicting results if necessary.
- 1498 6. The PDP informs the PEP of the decision result.
- 1499 7. The PEP enforces the decision, by either allowing the requested access or indicating that
- 1500 access is not allowed.

1501 The SAML and XACML specification sets contain some features specifically designed to facilitate their  
1502 combined use.

1503 The XACML Attribute Profile in the SAML Profiles specification defines how attributes can be described  
1504 using SAML syntax so that they may be automatically mapped to XACML Attributes. A schema is provided

1505 by SAML to facilitate this.

1506 A document that was produced by the XACML Technical Committee, SAML V2.0 profile of XACML v2.0,  
1507 provides additional information on mapping SAML Attributes to XACML Attributes. This profile also defines  
1508 a new type of Authorization decision query specifically designed for use in an XACML environment. It  
1509 extends the SAML protocol schema and provides a request and response that contains exactly the inputs  
1510 and outputs defined by XACML.

1511 That same document also contains two additional features that extend the SAML schemas. While they  
1512 are not, strictly speaking, intended primarily to facilitate combining SAML and XACML, they are worth  
1513 noting. The first is the XACML Policy Query. This extension to the SAML protocol schema allows the  
1514 SAML protocol to be used to retrieve XACML policy which may be applicable to a given access decision.

1515 The second feature extends the SAML schema by allowing the SAML assertion envelope to be used to  
1516 wrap an XACML policy. This makes available to XACML features such as Issuer, Validity interval and  
1517 signature, without requiring the definition of a redundant or inconsistent scheme. This promotes code and  
1518 knowledge reuse between SAML and XACML.

## 1519 **A. Acknowledgments**

1520 The editors would like to acknowledge the contributions of the OASIS Security Services Technical  
1521 Committee, whose voting members at the time of publication were:  
1522

- 1523 ● Brian Campbell Ping Identity Corporation
- 1524 ● George Fletcher AOL
- 1525 ● Hal Lockhart BEA Systems, Inc.
- 1526 ● Rob Philpott EMC Corporation
- 1527 ● Scott Cantor Internet2
- 1528 ● Bob Morgan Internet2
- 1529 ● Eric Tiffany Liberty Alliance Project
- 1530 ● Tom Scavo National Center for Supercomputing Applications
- 1531 ● Jeff Hodges NeuStar, Inc.
- 1532 ● Frederick Hirsch Nokia Corporation
- 1533 ● Paul Madsen NTT Corporation
- 1534 ● Ari Kermaier Oracle Corporation
- 1535 ● Anil Saldhana Red Hat
- 1536 ● Emily Xu Sun Microsystems
- 1537 ● David Staggs Veterans Health Administration
- 1538 ● Eve Maler Sun Microsystems

1539 Of particular note are the contributions from: Hal Lockhart BEA, Thomas Wisniewski Entrust, Scott Cantor  
1540 Internet2, Prateek Mishra Oracle, and Jim Lien EMC Corporation.