



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

Working Draft 08, 12 September 2005

Document identifier:

sstc-saml-tech-overview-2.0-draft-08

Location:

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

Editors:

John Hughes, Individual
Eve Maler, Sun Microsystems

Contributors:

Hal Lockhart, BEA
Thomas Wisniewski, Entrust
Prateek Mishra, Oracle
Nick Ragouzis, Individual

Abstract:

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.

Status:

This draft is a non-normative document that is intended to be approved as a Committee Draft by the SSTC. This document is not currently on an OASIS Standard track. Readers should refer to the normative specification suite for precise information concerning SAML V2.0.

Committee members should send comments on this specification to the security-services@lists.oasis-open.org list. Others should submit them by filling in the form at http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security.

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

35 Table of Contents

36	1 Introduction.....	5
37	2 SAML Use Cases.....	6
38	2.1 Single Sign-On Use Case.....	6
39	2.2 Identity Federation Use Cases.....	7
40	3 SAML Architecture.....	8
41	3.1 Basic Concepts.....	8
42	3.2 Summary of SAML Components.....	8
43	3.3 SAML Constructs and Examples.....	10
44	3.3.1 The Relationship of Bindings, Protocol Messages, Assertions, and Statements.....	11
45	3.3.2 Assertion, Statement, and Subject Structure.....	11
46	3.3.3 Attribute Statement Structure.....	12
47	3.3.4 Message Structure and the SOAP Binding.....	13
48	3.4 Use of SAML in other Frameworks.....	14
49	3.4.1 Web Services Security (WS-Security).....	15
50	3.4.2 eXtensible Access Control Markup Language (XACML).....	16
51	3.5 Security in SAML.....	18
52	4 Profiles.....	19
53	4.1 Web Browser SSO Profile.....	19
54	4.1.1 SP initiated: POST->POST binding.....	20
55	4.1.2 SP initiated: Redirect->POST binding.....	22
56	4.1.3 SP initiated: Artifact->POST binding.....	23
57	4.1.4 SP initiated: POST->Artifact binding.....	24
58	4.1.5 SP initiated: Redirect->Artifact binding.....	26
59	4.1.6 SP initiated: Artifact->Artifact binding.....	27
60	4.1.7 IdP initiated: POST binding.....	29
61	4.1.8 IdP initiated: Artifact binding.....	30
62	4.2 ECP Profile.....	31
63	4.2.1 Introduction.....	31
64	4.2.2 ECP Profile using PAOS binding.....	32
65	4.3 Identity Federation Protocols.....	32
66	4.3.1 Introduction.....	32
67	4.3.2 Single Sign-On with Out-of-band Account Linking.....	33
68	4.3.3 Attribute Federation.....	34
69	4.3.4 Persistent Federation.....	35
70	4.3.5 Transient Federation.....	38
71	4.3.6 Federation Termination.....	40
72	4.4 Single Logout.....	41
73	5 Documentation roadmap	44
74	6 Comparison Between SAML V2.0 and SAML V1.1.....	45
75	6.1 Differences in the Organization of the Specifications.....	45
76	6.2 Versioning Differences.....	45
77	6.3 Subject and Subject Confirmation Differences.....	45
78	6.4 Encryption-Related Differences.....	46

79	6.5 Attribute-Related Differences.....	46
80	6.6 Differences in the Request-Response Mechanism.....	46
81	6.7 Differences in the Protocols for Retrieving Assertions.....	46
82	6.8 Session-Related Differences.....	47
83	6.9 Federation-Related Differences.....	47
84	6.10 Differences in Bindings and Profiles.....	47
85	6.11 Other Differences.....	47
86	7 References.....	48
87		

88 Table of Figures

89	Figure 1: Single Sign-On Use Case [@@remove attr aspect].....	6
90	Figure 2: Account Linking Use Case [@@replace missing figure].....	7
91	Figure 3: SAML Components.....	10
92	Figure 4: SAML Assertion Containment Overview.....	11
93	Figure 5: Example of SAML Assertion, Subject, and Authentication Statement Constructs.....	12
94	Figure @@nn: Example of Attribute Statement.....	13
95	Figure 6: SAML Protocol Message Conveyed with SOAP over HTTP.....	13
96	Figure 7: Example of SAML Authentication Request Conveyed by SOAP.....	14
97	Figure 8: Example of SAML Response.....	14
98	Figure 10: Typical use of WS-Security and SAML [@@add figure back!].....	16
99	Figure 11: Typical use of XACML and SAML	17
100	Figure 12: Push and Pull Models for Web Browser SSO Profile.....	20
101	Figure 14: SP initiated: POST->POST binding.....	21
102	Figure 15: SP initiated: Redirect->POST binding.....	22
103	Figure 16: SP initiated: Artifact->POST binding.....	23
104	Figure 17: SP initiated: POST->Artifact binding.....	25
105	Figure 18: SP initiated: Redirect->Artifact binding.....	26
106	Figure 19: SP initiated: Artifact->Artifact binding.....	28
107	Figure 20: IdP initiated: POST binding.....	29
108	Figure 21: IdP initiated: Artifact binding.....	31
109	Figure 22: ECP use cases.....	32
110	Figure 23: ECP with PAOS.....	32
111	Figure 24: Single Sign-On with Out-of Band account linking.....	33
112	Figure 25: Attribute Federation.....	35
113	Figure 26: Persistent Federation – SP-initiated.....	36
114	Figure 27: Persistent Federation – IdP-initiated.....	37
115	Figure 28: Transient Federation – SP-initiated.....	39
116	Figure 29: Transient Federation – IdP-initiated.....	40
117	Figure 30: Federation Termination.....	41
118	Figure 31: Single Logout.....	42
119	Figure 32: Multiple Logouts.....	43
120	Figure 33: Multiple Logouts – identity provider initiated.....	43
121		

1 Introduction

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

More precisely, SAML defines a common XML framework for exchanging security assertions between entities. As stated in the Security Services Technical Committee (SSTC) charter, the purpose of the Committee is:

...to define, enhance, and maintain a standard XML-based framework for creating and exchanging authentication and authorization information.

SAML uses the approach of expressing assertions about a subject in a portable fashion that other applications across system domain boundaries can trust.

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

A system entity that makes SAML assertions is known as an **asserting party** or sometimes a **SAML authority**, and a system entity that uses received assertions is known as a **relying party**. This latter entity's willingness to rely on information from an asserting party depends on the existence of a trust relationship between them. The relying party is sometimes called a **SAML requester**, in that it is requesting information from a SAML authority.

Typically there are a number of **service providers** (SPs) that can make use of assertions about a subject in order to control access and provide customized services, and accordingly they become the relying parties of an asserting party called an **identity provider** (IdP). For example, a typical assertion from an identity provider might convey that “This user is John Doe, he has an email address of john.doe@acompany.com, and he was authenticated into this system using a password mechanism.” A service provider could choose to use this information, depending on its access policies, to grant access to local resources.

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

- **SSO interoperability:** How different products implement Cross-Domain Single Sign-On (CDSSO) has traditionally been completely proprietary. Most pre-SAML Single Sign-On products use browser cookies to maintain state so that re-authentication is not required. Browser cookies are not transferred between DNS domains. So, if you obtain a cookie from www.abc.com, then that cookie will not be sent in any HTTP messages to www.xyz.com. This could even apply within a single organization that has separate DNS domains. SAML solves the CDSSO problem by providing a standard vendor-independent protocol for transferring information about a (browser-equipped) user from one web server to another without relying on cookies.
- **Federated identity:** Federated identity deals with the sharing of information about user identities across organizational boundaries while maintaining privacy protection. From an administrative perspective, this type of sharing can help reduce identity management costs as multiple organizations do not need to independently collect and maintain identity-related data (e.g. passwords). From a user-centric viewpoint, as explained under SSO interoperability, this also results in an enhanced user experience with fewer sign-ons. In addition, administrators do not have to maintain the mappings; rather control can reside with the user.
- **Web services:** SAML allows its security assertion format to be used outside of a “native” SAML-based protocol context, and this modularity has proven useful within the web services environment. The WS-Security effort has defined how to use a SAML assertion as a security token. The SAML assertion format (and associated security token definition) provides a means by which security assertions about messages and service requesters can be exchanged between communicating service endpoints. In particular, the advantage offered by the use of a SAML assertion is that it provides a standards-based approach to the exchange of information, including attributes, not easily contained within other WS-Security token formats.

2 SAML Use Cases

- Prior to examining the details of the SAML standard, it's useful to describe some of its high-level use cases. (Later on, more detailed use cases are described based on specific SAML profiles.)

2.1 Single Sign-On Use Case

Single sign-on is the classic use case supported in SAML V1.0 and V1.1. A user has a login session (that is, a *security context*) on a web site (*AirlineInc.com*) and is accessing resources on that site. At some point, either explicitly or transparently, he is directed over to another web site (typically in a different DNS domain). The identity provider site (*AirlineInc.com*) asserts to the service provider site (*CarRentallnc.com*) that the user is known to it and provides the user's name and possibly additional session attributes (e.g. "Gold member"). The user's identity is federated between *AirlineInc.com* and *CarRentallnc.com* by business agreement between the partners. As *CarRentallnc.com* trusts *AirlineInc.com*, it knows that the user is valid and creates a session for the user. This use case illustrates the fact that the user is not required to re-authenticate when directed over to the *CarRentallnc.com* site

Figure 1 illustrates the SSO high-level use case.

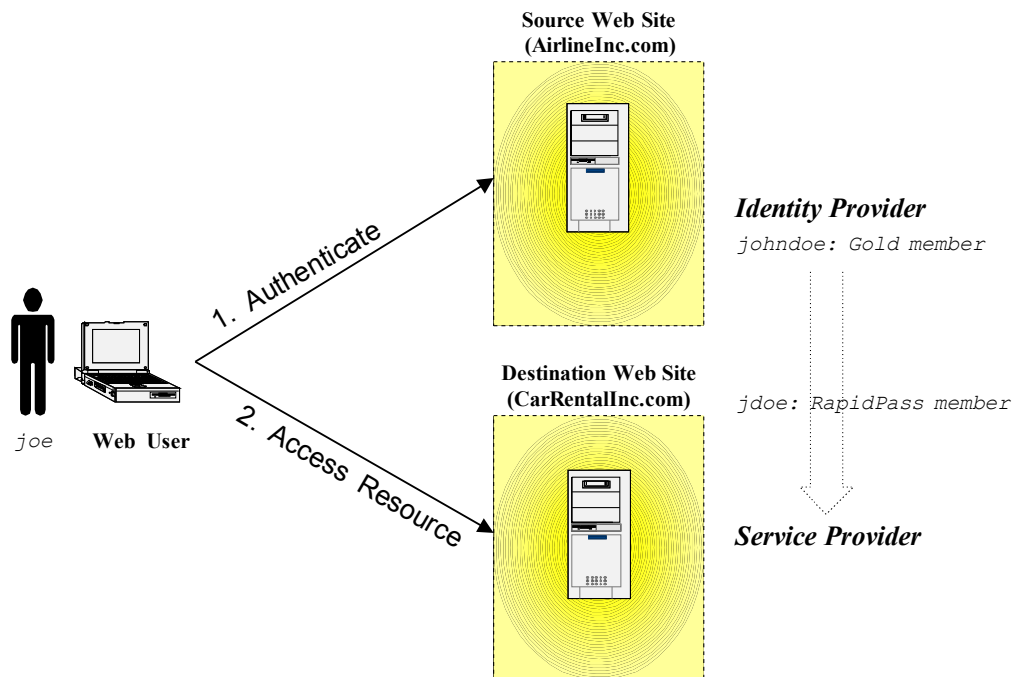


Figure 1: Single Sign-On Use Case [@@remove attr aspect]

This high-level description implies that the user always visits and is authenticated at the IdP as a first step in order to access protected resources at that site, and this is indeed sometimes the case. However, often a user starts out by visiting an SP site, accessing resources that require no special authentication or authorization, and subsequently attempts to access a protected resource at the SP. For these reasons, the SP may need to direct an authentication request to the IdP (which may ultimately require the IdP to interact with the user) in order to validate the user's access rights to the protected resource. SAML provides for both patterns, which it calls *IdP-initiated* and *SP-initiated* flows.

At a more detailed technical level, SAML provides a number of different "delivery mechanisms" (called bindings) for each request and response message to and from a SAML authority. Thus, in practice, there are several flavors of IdP-initiated and SP-initiated flows. These are described in more detail in Section 4.1.

199 2.2 Identity Federation Use Cases

200 Once it is possible to share identity information between providers, further questions arise concerning the
201 properties of the transmitted information. These include such issues as the format and values
202 transmitted, relevant processing rules at identity and service providers, and assumptions about contents
203 of identity stores at the providers. These tasks fall into the general category of identity management and
204 are more specifically described by the term identity federation.

205 A user's identity is said to be federated between a set of providers when there is agreement between the
206 providers on a set of identifiers and/or attributes to use when referring to the user. There are many
207 different techniques that may be used to implement the data flows required for such agreements
208 between providers.

209 In some cases, some of the required exchanges of identity-related information may take place outside of
210 the SAML V2.0 standard using other infrastructure. For example, providers may choose to share
211 information about newly registered or de-registered users via batch "identity feeds" that are driven by
212 identity sources (for example, human resources databases) at the identity provider and propagated to
213 service providers. Subsequently, the user name may be placed in a SAML assertion and propagated
214 between providers to implement single sign-on. Alternatively, identity federation may be achieved purely
215 by a business agreement that states that an identity provider will refer to a user based on certain attribute
216 names and values, with no additional flows required for maintaining and updating user information
217 between providers

218 A typical use case for achieving identity federation using SAML is "account linking." Figure 2 illustrates
219 one scenario. Two service providers exist, one for car rentals, the other for hotel bookings. In addition to
220 [AirlinesInc.com](#), users are registered on both service provider sites, but using different names. At
221 [AirlinesInc.com](#), user **joe** may be registered as **johndoe**, on [CarRentallnc.com](#) as **jdoe**, and on
222 [HotelBookings.com](#) as **johnd**. SAML V2.0 supports a model for federated identity based on
223 pseudonyms. A pseudonym is a privacy-preserving identifier shared between entities. In this use case,
224 [AirlinesInc.com](#) describes the user to [CarRentallnc.com](#) and [HotelBooking.com](#) using (distinct)
225 pseudonyms. Each of [CarRentallnc.com](#) and [HotelBooking.com](#) can link the pseudonym to the existing
226 user account once user consent has been obtained. In subsequent access, the user will only need to log
227 in once to [AirlinesInc.com](#) before beginning to conduct business at [CarRentallnc.com](#) and
228 [HotelBooking.com](#) using account information available at these sites.

229

Figure 2: Account Linking Use Case [@@@replace missing figure]

230 SAML V2.0 supports several features that are desirable when working with federated identity. For
231 example, confidentiality is supported by permitting various SAML constructs to be encrypted. In addition,
232 providers can capture information about user consent and transmit it within SAML messages.

233 3 SAML Architecture

234 This section provides a brief description of the concepts that underlie SAML and the component pieces
235 defined in the standard.

236 3.1 Basic Concepts

237 SAML consists of building-block components that, when put together, allow a number of use cases to be
238 supported. Primarily the components permit transfer of identity, authentication, attribute, and
239 authorization information to be exchanged between autonomous organizations. The “core” SAML
240 specification defines the structure and content of **Assertions** – which carry statements about a Principal
241 as asserted by an Asserting Party. These are defined by an XML schema.

242 Assertions are either requested or just “pushed” out to the service provider. How and which assertions
243 are requested is defined by the **SAML Protocols**, which have their own XML schema. The lower-level
244 communication or messaging protocols (such as HTTP or SOAP) over which the SAML protocol
245 messages can be transported are defined by **Bindings**. SAML Protocols and Bindings, together with the
246 structure of Assertions, can be combined together to create a **Profile** for greater interoperability. In
247 general Profiles can be thought of a satisfying a particular use case, for example the Web Browser SSO
248 profile. There are also Attribute Profiles (for example, LDAP and DCE profiles), which define how to
249 interpret attribute information carried within an Assertion using common attribute/directory technologies.

250 Two other SAML components can be used in building a system:

- 251 • **Metadata:** Metadata defines a way to express and share configuration information between two
252 communicating providers. For instance, an entity's support for given SAML bindings, identifier
253 information, and PKI information can be defined. Metadata is defined by an XML schema. The
254 location of Metadata is defined using DNS records.
- 255 • **Authentication Context:** In a number of situations the service provider may wish to have
256 additional information in determining the authenticity and confidence they have in the information
257 within an assertion. Authentication Context permits the augmentation of Assertions with additional
258 information pertaining to the authentication of the principal at the identity provider. For instance,
259 details of multi-factor authentication can be included.

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

262 3.2 Summary of SAML Components

263 The SAML components and their individual parts are as follows:

- 264 • **Assertions:** SAML allows for one party to assert characteristics and attributes of a subject. For
265 instance, a SAML assertion could state that the subject is “John Doe”, has “Gold” status, has an
266 email address of john.doe@example.com, and is a member of the “engineering” group. SAML
267 assertions are encoded in an XML schema. SAML defines three kinds of statements that can be
268 carried within an assertion:
 - 269 • **Authentication statements:** These are issued by the party that successfully authenticated the
270 user. They describe who issued the assertion, the authenticated subject, and the validity period,
271 plus other authentication-related information.
 - 272 • **Attribute statements:** These contain specific details about the subject (for example, that user
273 “John Doe” has “Gold” status).
 - 274 • **Authorization decision statements:** These define something the subject is entitled to do (for
275 example, whether “John Doe” is permitted to buy a specified item).
- 276 • **Protocols:** SAML defines a number of request/response protocols, which are encoded in an XML
277 schema as a set of request-response pairs. The protocols defined are:
 - 278 • **Assertion Query and Request Protocol:** Defines a set of queries by which existing SAML

279 assertions may be obtained. The query can be on the basis of a SAML message reference, the
280 subject, or the statement type.

281 • **Authentication Request Protocol:** Defines a protocol by which a service provider or principal
282 can request assertions from an identity provider tailored to the requirements of a particular
283 SAML profile, for example the Web Browser SSO Profile.

284 • **Artifact Resolution Protocol:** Provides a mechanism by which protocol messages may be
285 passed by reference using a small, fixed-length value called an *artifact*. The artifact receiver
286 uses the Artifact Protocol to dereference the actual protocol message.

287 • **Name Identifier Management Protocol:** Provides mechanisms to change the value of the
288 principal's name identifier. The issuer of the request can be either the service provider or the
289 identity provider. The protocol also provides a mechanism to terminate an association of a
290 name between an identity provider and service provider.

291 • **Single Logout Protocol:** Defines a request that allows near-simultaneous logout of all
292 sessions associated by a principal. The logout can be directly initiated by the principal, due to a
293 session timeout or because a user access rights have been revoked. Logout can also be
294 initiated by a provider site.

295 • **Name Identifier Mapping Protocol:** Provides a mechanism to programmatically map one
296 SAML name identifier into another, subject to appropriate policy controls.

297 In addition to request and response messages comprising the protocols listed here, SAML provides
298 for a special representation of any type of protocol message, called an *artifact*, which can be
299 dereferenced to obtain the full message. An artifact takes the form of a base-64 encoded string.

300 • **Bindings:** These detail exactly how the SAML protocols map onto underlying transport protocols.
301 For instance, the SAML specification provides a binding of how SAML requests and responses are
302 carried with SOAP exchange messages. The bindings defined are:

303 • **SAML SOAP Binding:** Defines how SAML protocol messages are transported within SOAP 1.1
304 messages, with details about using SOAP over HTTP.

305 • **Reverse SOAP (PAOS) Binding:** Defines a multi-stage SOAP/HTTP message exchange that
306 permits an HTTP client to be a SOAP responder. Used in the Enhanced Client and Proxy
307 Profile and particularly designed to support WAP gateways.

308 • **HTTP Redirect Binding:** Defines how SAML protocol messages can be transported using
309 HTTP redirect messages (302 status code responses).

310 • **HTTP POST Binding:** Defines how SAML protocol messages can be transported within the
311 base64-encoded content of an HTML form control.

312 • **HTTP Artifact Binding:** Any SAML protocol message can be represented by an artifact, which
313 has a compact base-64 format and allows for the real message to be "pulled" (dereferenced).
314 This binding defines how an artifact is transported by HTTP using one of two mechanisms:
315 either an HTML form control or a query string in the URL.

316 • **SAML URI Binding:** Defines a means for retrieving a SAML assertion by resolving a URI
317 (uniform resource identifier).

318 • **Profiles:** Profiles define how the SAML assertions, protocols, and bindings are combined for
319 interoperability in particular usage scenarios. Some of these are described in detail later on in the
320 document. In summary they are:

321 • **Web Browser SSO Profile:** Defines a mechanism for single sign-on by unmodified web
322 browsers to multiple service providers using the Authentication Request protocol in combination
323 with the HTTP Redirect, HTTP POST, and Artifact bindings.

324 • **Enhanced Client and Proxy (ECP) Profile:** Defines a profile of the Authentication Request
325 protocol in conjunction with the Reverse-SOAP and SOAP bindings suited to clients or gateway

- 326 devices with knowledge of one or more identity providers.
- 327 • **Identity Provider Discovery Profile:** Defines one possible mechanism for a set of cooperating
328 Identity and service providers to obtain the identity providers used by a principal.
 - 329 • **Single Logout Profile:** A profile of the SAML Single Logout protocol is defined. Defines how
330 SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings may be used.
 - 331 • **Name Identifier Management Profile:** Defines how the Name Identifier Management protocol
332 may be used with SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
 - 333 • **Artifact Resolution Profile:** Defines how the Artifact Resolution protocol uses a synchronous
334 binding, for example the SOAP binding.
 - 335 • **Assertion Query/Request Profile:** Defines how the SAML query protocols (used for obtaining
336 SAML assertions) use a synchronous binding such as the SOAP binding.
 - 337 • **Name Identifier Mapping Profile:** Defines how the Name Identifier Mapping protocol uses a
338 synchronous binding such as the SOAP binding.

339 Figure 3 illustrates the relationship between the components:

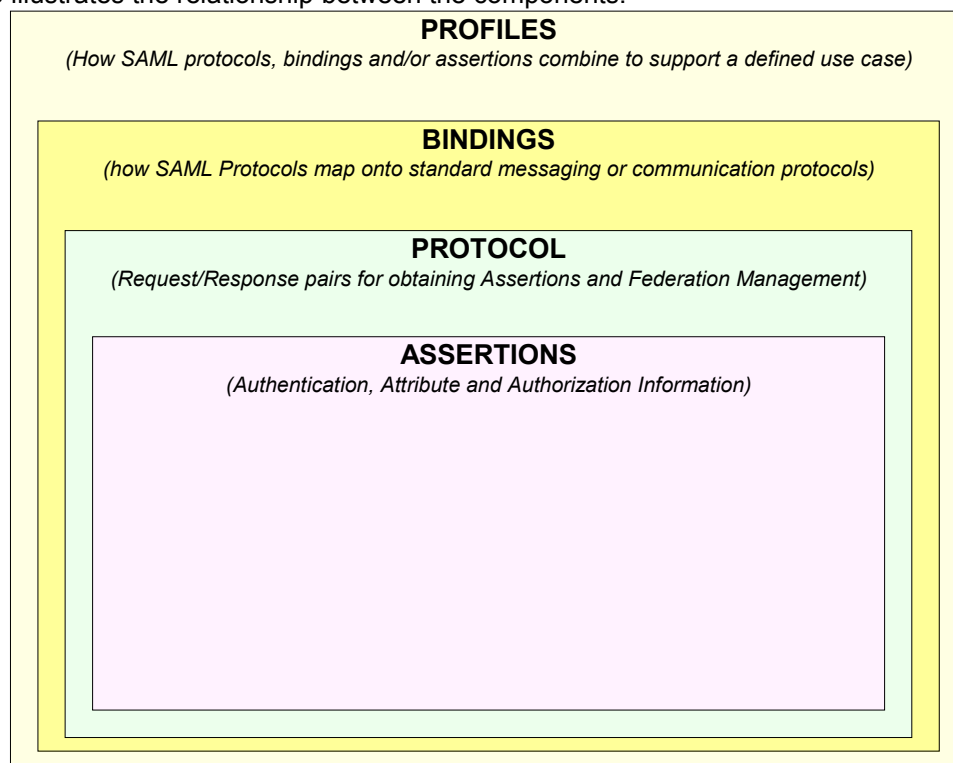


Figure 3: SAML Components

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

344 3.3 SAML Constructs and Examples

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

346 **3.3.1 The Relationship of Bindings, Protocol Messages, Assertions, and**
347 **Statements**

348 An assertion contains one or more statements and some common information that applies to all
349 contained statements or to the assertion as a whole. A SAML assertion is typically conveyed by a SAML
350 protocol response message, which itself must be carried by some sort of transport or messaging
351 protocol.

352 Figure 4 shows a typical example of containment: a SAML Assertion containing a series of statements,
353 the whole being carried within a SAML response, which itself is within a SOAP body.

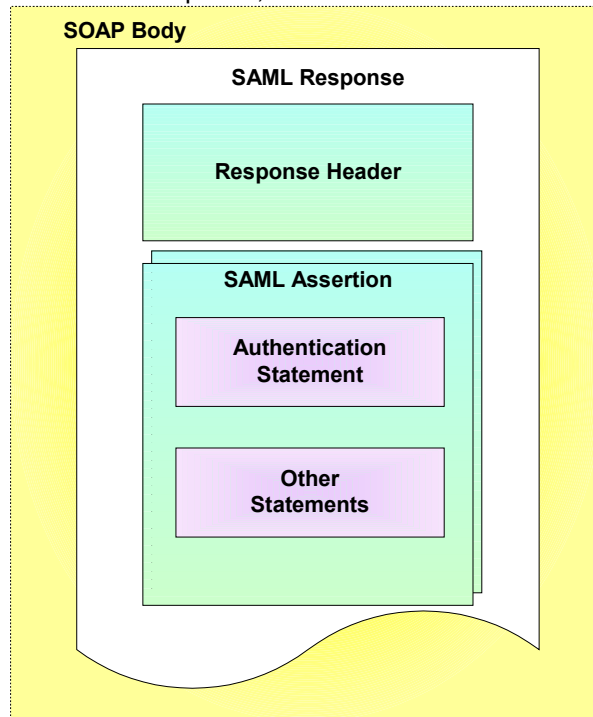


Figure 4: SAML Assertion Containment Overview

354 **3.3.2 Assertion, Statement, and Subject Structure**

355 Figure 5 shows an XML fragment containing an example assertion with a single authentication
356 statement. Note the following:

- 357 • Line 1 contains the declaration of the SAML assertion namespace, which is conventionally
358 represented in the specifications with the `saml:` prefix.
- 359 • Lines 3 through 6 provide information about the nature of the assertion: when it was issued and
360 who issued it.
- 361 • Lines 7 through 12 provide information about the subject of the assertion, to which all of the
362 contained statements (in this case only a single authentication statement) apply. The subject has a
363 name identifier (line 10) whose value is "j.doe@acompany.com", provided in a format whose label
364 is given on line 9. SAML predefines a number of name identifier formats, and you can also define
365 your own.
- 366 • The assertion as a whole has a validity period indicated by lines 14 and 15. Additional conditions
367 on the use of the assertion can be provided inside this element; SAML predefines some and you
368 can define your own. Timestamps in SAML use the XML Schema **dateTime** data type.
- 369 • The authentication statement appearing on lines 17 through 24 shows that this subject was
370 originally authenticated using a password-protected transport mechanism at the time and date

371 shown. SAML predefines some dozens of labels for various authentication mechanisms, and you
372 can also define your own.

```
373 1: <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
374 2:   Version="2.0"  
375 3:   IssueInstant="2005-01-31T12:00:00Z">  
376 4:   <saml:Issuer>  
377 5:     www.acompany.com  
378 6:   </saml:Issuer>  
379 7:   <saml:Subject>  
380 8:     <saml:NameID  
381 9:       Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">  
382 10:      j.doe@acompany.com  
383 11:     </saml:NameID>  
384 12:   </saml:Subject>  
385 13:   <saml:Conditions  
386 14:     NotBefore="2005-01-31T12:00:00Z"  
387 15:     NotOnOrAfter="2005-01-31T12:00:00Z">  
388 16:   </saml:Conditions>  
389 17:   <saml:AuthnStatement  
390 18:     AuthnInstant="2005-01-31T12:00:00Z" SessionIndex="6777527772">  
391 19:     <saml:AuthnContext>  
392 20:       <saml:AuthnContextClassRef>  
393 21:         urn:oasis:names:tc:SAML:2.0:ac:classes>PasswordProtectedTransport  
394 22:       </saml:AuthnContextClassRef>  
395 23:     </saml:AuthnContext>  
396 24:   </saml:AuthnStatement>  
397 25: </saml:Assertion>
```

Figure 5: Example of SAML Assertion, Subject, and Authentication Statement Constructs

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

- 400 • Email address
- 401 • X.509 subject name
- 402 • Windows domain qualified name
- 403 • Kerberos principal name
- 404 • Entity identifier
- 405 • Persistent identifier
- 406 • Transient identifier

407 Of these, persistent and transient identifiers require further discussion. **Transient identifiers** support
408 “anonymity” at the SP since they correspond to a “one-time use” identifier created at the IdP. **Persistent**
409 **identifiers** support pseudonymity at SPs; they are privacy-preserving and their use is restricted to an
410 IdP-SP pair. The concept of **affiliation** permits a group of SPs to consume a single shared persistent
411 identifier used to describe a principal. Affiliations are indicated by the `SPNameQualifier` attribute on
412 the <NameID> and <NameIDPolicy> elements.

413 3.3.3 Attribute Statement Structure

414 Attribute information about a principal is often provided as an adjunct to authentication information in
415 single sign-on, or even in lieu of authentication information in scenarios such as “attribute federation.”
416 SAML's attribute structure does not presume that any particular type of data store is being used for the
417 attributes; it has an agnostic structure.

418 Figure @@nn shows an XML fragment containing an attribute statement. Note the following:

- 419 • A single statement can contain multiple attributes. In this case, there are two attributes (starting on
420 line 2 and line 9) within the statement.
- 421 • Similarly to name identifier formats, attribute names are also qualified with a format label which
422 indicates how the attribute name is to be interpreted. In both of the cases here (lines 3 and 10), the
423 name format is not one of those predefined by SAML, but is rather defined by a third party,
424 SmithCo. This is a fairly artificial example, and interoperability would be increased by either the use of
425 one of SAML's **attribute profiles** or the formal definition of a third-party attribute profile. See the

- 426 SAML Profiles specification [@@add bibref] for more information and examples.
- 427 • The value of an attribute can be plain text, as on line 6, or can be structured XML, as on lines 13
- 428 through 15.

```

429 1: <saml:AttributeStatement xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
430 2:   <saml:Attribute
431 3:     NameFormat="http://smithco.com"
432 4:     Name="PaidStatus">
433 5:     <saml:AttributeValue>
434 6:       PaidUp
435 7:     </saml:AttributeValue>
436 8:   </saml:Attribute>
437 9:   <saml:Attribute
438 10:    NameFormat="http://smithco.com"
439 11:    Name="CreditLimit">
440 12:    <saml:AttributeValue xsi:type="smithco:type">
441 13:      <smithco:amount currency="USD">
442 14:        500.00
443 15:      </my:amount>
444 16:    </saml:AttributeValue>
445 17:  </saml:Attribute>
446 18: </saml:AttributeStatement>

```

Figure @@nn: Example of Attribute Statement

447 3.3.4 Message Structure and the SOAP Binding

448 In environments where the two communicating endpoints are SOAP-enabled, then the SOAP-over-HTTP

449 binding can be used to exchange SAML request/query and response protocol messages. Figure 6

450 provides an overview of the structure. The SAML request or SAML response being carried within the

451 SOAP body, which itself has an HTTP response wrapper.

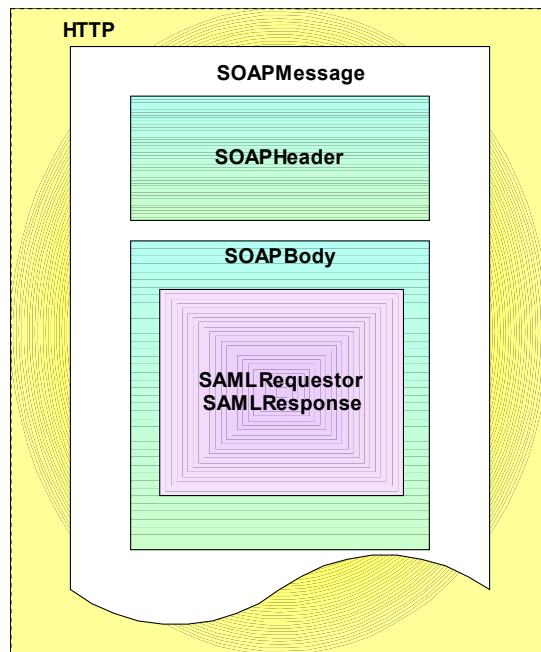


Figure 6: SAML Protocol Message Conveyed with SOAP over HTTP

452 Figure 7 shows a complete XML document containing an example of a SAML authentication request

453 message being transported within a SOAP message. Note the following:

- 454 • The authentication request starting on line 5 is embedded in a SOAP body element starting on line
- 455 4. The request contains, on line 6, a declaration of the SAML protocol namespace, which is
- 456 conventionally represented in the specifications with the `samlp:` prefix. The protocol namespace

- 457 and the assertion namespace are separate (and SAML defines a number of adjunct vocabularies
 458 with their own namespaces as well).
- 459 • The request element provides a number of parameters that govern the type of assertion it expects
 460 back, for example, the requested subject (lines 14 through 20) and whether to force fresh
 461 authentication of the subject (line 7).

```

462 1: <?xml version="1.0" encoding="UTF-8"?>
463 2: <env:Envelope
464 3:   xmlns:env="http://www.w3.org/2003/05/soap/envelope/">
465 4:   <env:Body>
466 5:     <samlp:AuthnRequest
467 6:       xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
468 7:       ForceAuthn="true"
469 8:       AssertionConsumerServiceURL="https://www.sp.example.com/SSO"
470 9:       AttributeConsumingServiceIndex="0" ProviderName="CarRentalInc.com"
471 10:      ID="abe567de6"
472 11:      Version="2.0"
473 12:      IssueInstant="2005-01-31T12:00:00Z"
474 13:      Destination="https://www.idp.example.com/" >
475 14:     <saml:Subject
476 15:       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
477 16:       <saml:NameID
478 17:         Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
479 18:         j.doe@acompany.com
480 19:       </saml:NameID>
481 20:     </saml:Subject>
482 21:   </samlp:AuthnRequest>
483 22: </env:Body>
484 23: </env:Envelope>
  
```

Figure 7: Example of SAML Authentication Request Conveyed by SOAP

485 Figure 8 shows an XML fragment containing a SAML response. Note the following:

- 486 • On line 6, the response references the request to which it is responding, and additional information
 487 is provided in the response header (lines 4 through 13), including status information. SAML
 488 predefines a number of status codes and, in many cases, dictates the circumstances under which
 489 each must be used.
- 490 • Within the response (line 14; detail elided) is a SAML assertion, typically containing one or more
 491 statements as shown above.

```

492 1: <env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
493 2:   <env:Body>
494 3:     <samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
495 4:       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
496 5:       ID="abe567de6"
497 6:       InResponseTo="example-ncname" Version="2.0"
498 7:       IssueInstant="2005-01-31T12:00:00Z"
499 8:       Destination="https://www.idp.example.com/">
500 9:     <samlp:Status>
501 10:       <samlp:StatusCode Value="samlp:Success"/>
502 11:       <samlp:StatusMessage>Success</samlp:StatusMessage>
503 12:       <samlp:StatusDetail/>
504 13:     </samlp:Status>
505 14:     ...SAML assertion...
506 15:   </samlp:Response>
507 16: </env:Body>
508 17: </env:Envelope>
  
```

Figure 8: Example of SAML Response

509

510 3.4 Use of SAML in other Frameworks

511 SAML's components are modular and extensible, and it has been adopted for use with several other
 512 standard frameworks. Following are some examples.

513 3.4.1 Web Services Security (WS-Security)

514 SAML Assertions can be conveyed by means other than the SAML Request/Response protocols or
515 Profiles defined by the SAML specification set. One example of this is the use of SAML by Web Services
516 Security (WS-Security) [@@add bibref], which is a set of specifications that define means for providing
517 security protection of SOAP messages. The primary services provided WS-Security are authentication,
518 data integrity, and confidentiality.

519 WS-Security defines a <Security> element that may be included in the SOAP header. This element
520 contains information that specifies how the message is protected. WS-Security makes use of
521 mechanisms defined in the XML Digital Signature and XML Encryption specifications to sign and encrypt
522 message data in both the header and the body. The information in the <Security> element specifies
523 what operations were performed and in what order, what keys were used for the operations, and what
524 attributes and identity information are associated with that information. WS-Security also contains other
525 features, such as the ability to timestamp the security information and to address it to a specified Role.

526 In WS-Security keys and attributes are specified using tokens. Tokens can either be binary or XML.
527 Binary tokens, such as X.509 Certificates and Kerberos Tickets are carried in an XML wrapper. XML
528 tokens, such as SAML Assertions, are inserted directly as sub-elements of the <Security> element. A
529 Security Token Reference may be used to refer to a token in one of a number of ways.

530 WS-Security consists of a Core Specification which describes the mechanisms independent of the type
531 of token being used and a number of Token Profiles which describe the use of particular types of tokens.
532 Token profiles cover considerations relating to that particular token type and methods of referencing the
533 token using a Security Token Reference. The use of SAML Assertions with WS-Security is described in
534 the SAML Token Profile [@@add bibref].

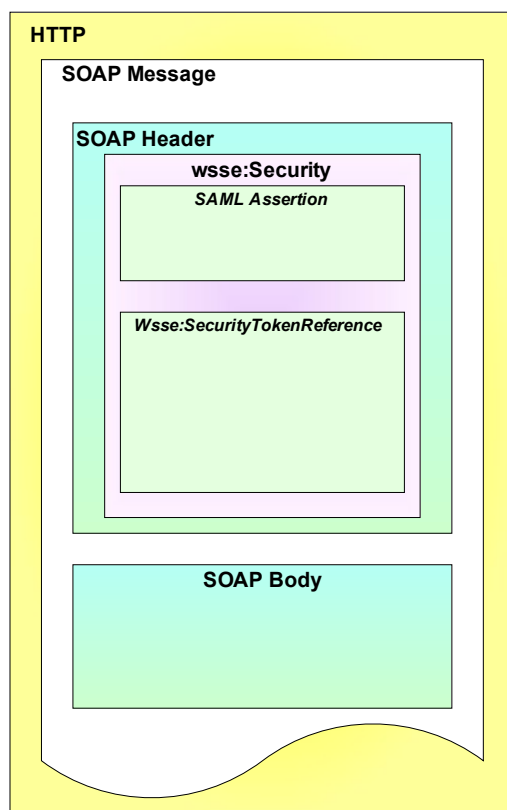
535 Because the SAML protocols have a binding to SOAP, it is easy to get confused between that and the
536 use of SAML Assertions by WS-Security. They can be distinguished by their purpose, the message
537 format, and the parties involved.

538 The characteristics of the SAML Request/Response protocol binding over SOAP are as follows.

- 539 • It is used to obtain SAML Assertions for future use; they play no role in protecting the message.
- 540 • The SAML Assertions are contained within a SAML Response, which is carried in the SOAP body.
- 541 • The SAML Assertions are provided by a trusted authority or repository and may or may not pertain to
542 the party requesting them.

543 The characteristics of the use of SAML Assertions as defined by WS-Security are as follows.

- 544 • The SAML Assertions usually play a role in the protection of the message they are carried in, typically
545 they contain a key used for digital signatures.
- 546 • The SAML Assertions are carried in a <Security> element within the SOAP header.
- 547 • The SAML Assertions will have been obtained previously and typically pertain to the identity of the
548 sender. Figure 9: *WS-Security* and SAML Relationship



549 Note that in principle, SAML Assertions could be used in both ways in a single SOAP message. In this
 550 case the Assertions in the header would refer to the identity of the Responder (and Requester) of the
 551 message.

552 The following sequence of steps typifies the use of SAML Assertions with WS-Security.

- 553 1. Sender obtains SAML Assertion by means of SAML Request/Response or other SAML Profile.
 554 Assertion contains attribute statement and Subject Confirmation Method of Holder of Key.
- 555 2. Sender constructs SOAP message, including Security header. SAML Assertion is included in
 556 Security header. Key referred to by SAML Assertion is used to construct digital signature over
 557 data in message body. Signature information is also included in Security header.
- 558 3. Receiver verifies digital signature.
- 559 4. The information in the SAML Assertion is used for purposes such as Access Control and Audit
 560 logging.

561 Figure 10 illustrates this usage scenario.

Figure 10: Typical use of WS-Security and SAML [@@@add figure back!]

562 3.4.2 eXtensible Access Control Markup Language (XACML)

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

569 The eXtensible Access Control Markup Language (XACML) is an OASIS Standard that defines the
 570 syntax and semantics of a language for expressing and evaluating access control policies. The work to
 571 define XACML was started slightly after SAML began. From the beginning they were viewed as related

572 efforts and consideration was given to specifying both within the same Technical Committee. Ultimately,
573 it was decided to allow them to proceed independently but to align them. Compatibility with SAML was
574 written in to the charter of the XACML TC.

575 As a result, SAML and XACML can each be used independently of the other, or both can be used
576 together. Using SAML and XACML in combination would typically involve the following steps.

- 577 1. An XACML Policy Enforcement Point (PEP) receives a request to access some resource.
- 578 2. The PEP obtains SAML Assertions containing information about the parties to the request,
579 such as the requester, the receiver (if different) or intermediaries. These Assertions might
580 accompany the request or be obtained directly from a SAML Authority, depending on the
581 SAML profile used.
- 582 3. The PEP obtains other information relevant to the request, such as time, date, location, and
583 properties of the resource.
- 584 4. The PEP presents all the information to a Policy Decision Point (PDP) to decide if the access
585 should be allowed.
- 586 5. The PDP obtains all the policies relevant to the request and evaluates them, combining
587 conflicting results if necessary.
- 588 6. The PDP informs the PEP of the decision result.
- 589 7. The PEP enforces the decision, by either allowing the requested access or indicating that
590 access is not allowed.

591 Figure 11 illustrates the typical use of SAML with XACML.

592

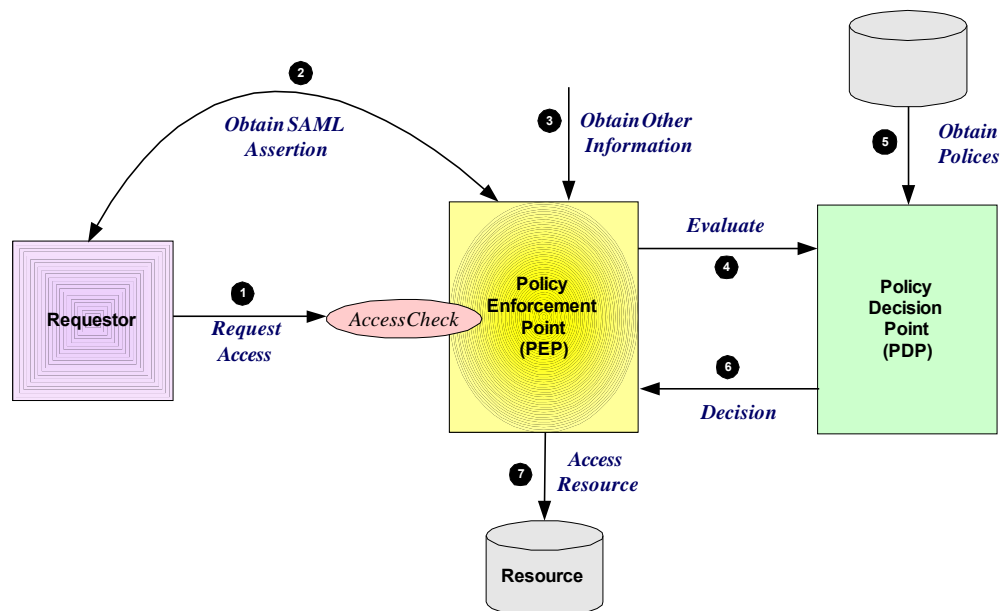


Figure 11: Typical use of XACML and SAML

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

596 The XACML Attribute Profile, which can be found in the SAML Profiles specification, defines how SAML
597 attributes may be mapped to XACML Attributes. A schema is provided by SAML to facilitate this.

598 One of the XACML specifications, SAML V2.0 profile of XACML, provides additional information on
599 mapping SAML Attributes to XACML Attributes.

600 The SAML V2.0 profile of XACML also defines a new type of Authorization decision query specifically

601 designed for use in an XACML environment. It extends the SAML protocol schema and provides a
602 request and response that contains exactly the inputs and outputs defined by XACML.

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

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

611 **3.5 Security in SAML**

612 [@@@add SecConsider bibref]Just providing assertions from an asserting party to a relying party may not
613 be adequate for a secure system. How does the relying party trust what is being asserted to it? In
614 addition, what prevents a “man-in-the-middle” attack that grabs assertions to be illicitly “replayed” at a
615 later date? SAML defines a number of security mechanisms that prevent or detect such attacks. The
616 primary mechanism is for the relying party and asserting party to have a pre-existing trust relationship,
617 typically involving a Public Key Infrastructure (PKI). While use of a PKI is not mandated, it is
618 recommended. Use of particular mechanisms is described for each profile; however, an overview of
619 what is recommended is provided below:

- 620 • Where message integrity and message confidentiality are required, then HTTP over SSL 3.0 or
621 TLS 1.0 is recommended.
- 622 • When a relying party requests an assertion from an asserting party then bi-lateral authentication is
623 required and the use of SSL 3.0 or TLS 1.0 using server and client authentication is recommended.
- 624 • When an assertion or request “pushed” to a relying party (for example using the HTTP POST
625 binding), then it is mandated that the response message be digitally signed using the XML digital
626 signature standard.

627

628

629 **4 Profiles**

630 SAML supports a number of use cases and profiles. The purpose of this section is to describe a number
631 of the more important ones. The following are described:

- 632 • Web Browser SSO Profile
- 633 • Enhanced Client and Proxy (ECP) Profile
- 634 • Federation Use Cases
- 635 • Single Logout

636 **4.1 Web Browser SSO Profile**

637 The Web Browser SSO profile supports a variety of options, having to do with two dimensions of choice:
638 whether the message flows are IdP-initiated or SP-initiated (as discussed in Section 2.1) and whether the
639 IdP pushes SAML assertions to the SP or the SP pulls them from the IdP. The push approach involves
640 using either HTTP redirects or HTTP POST messages to deliver a SAML message. The pull approach
641 involves sending an artifact to the receiver, which then uses the artifact to dereference and obtain the
642 related SAML message. A combination of message flow and binding techniques gives rise to eight
643 different combinations, all of which are described in this section:

- 644 • Message flow initiated by the SP:
 - 645 • POST binding used for both the request and the response
 - 646 • Redirect binding used for the request and POST binding used for the response
 - 647 • Artifact binding used for the request and POST binding used for the response
 - 648 • POST binding used for the request and artifact binding used for the response
 - 649 • Redirect binding used for the request and artifact binding used for the response
 - 650 • Artifact binding used for both the request and the response
- 651 • Message flow initiated by the IdP:
 - 652 • POST binding used for the (soliary) response
 - 653 • Artifact binding used for the response

654 Figure 12 compares the push and pull approaches.

655

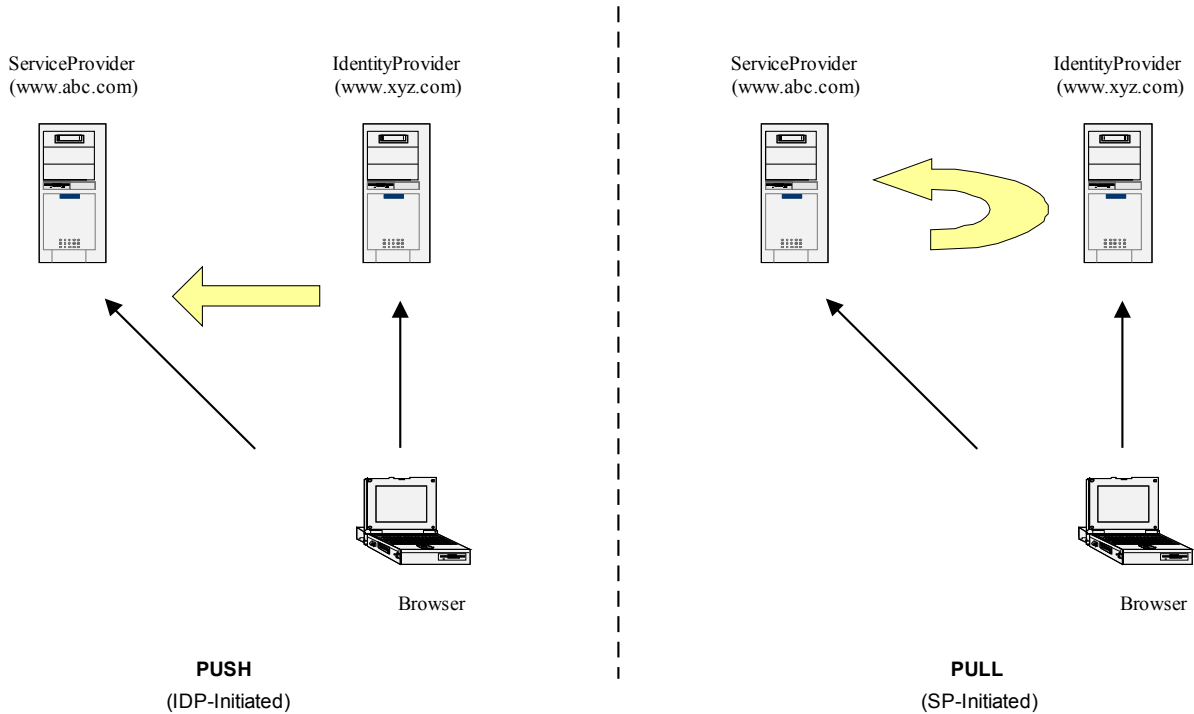


Figure 12: Push and Pull Models for Web Browser SSO Profile

656

657 4.1.1 SP initiated: POST->POST binding

658 In this use case the user attempts to access a resource on www.abc.com. However they do not have
 659 current an existing session context on this site (e.g. logged on) and their identity is managed by
 660 www.xyz.com. A SAML <AuthnRequest> is sent to their identity provider so that the identity provider
 661 can provide back a SAML assertion concerning the user. HTTP POST messages are used to deliver the
 662 SAML <AuthnRequest> to the identity provider as well as receive back the SAML response.

663 Figure 14 illustrates the message flow:

664

665

666

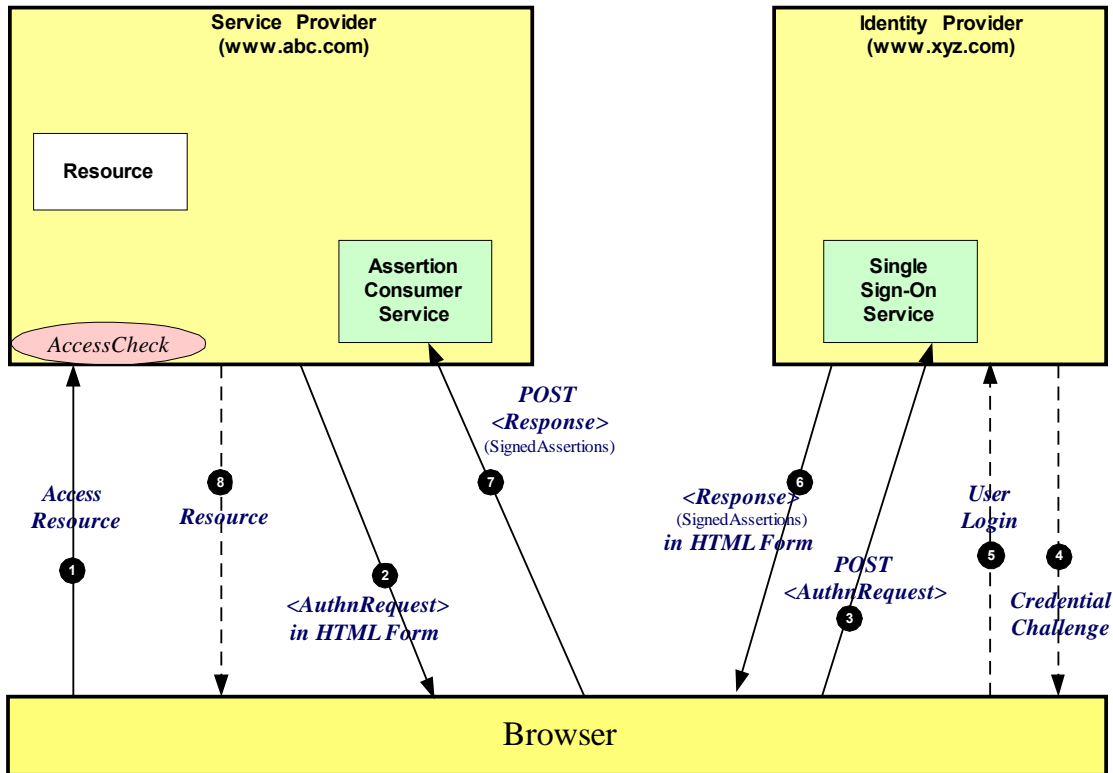


Figure 14: SP initiated: POST->POST binding

667 The processing is as follows:

- 668 1. The user attempts to access a resource on www.abc.com. The user does not have any current logon
669 session (i.e. security context) on this site, and is unknown to it.
- 670 2. The SP sends a HTML form back to the browser. The HTML FORM contains a SAML
671 <AuthnRequest> defining the user for which authentication and authorization information is
672 required. Typically the HTML FORM will contain an input or submit action that will result in an HTTP
673 POST.
- 674 3. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing
675 the SAML <AuthnRequest> to the identity provider's Single Sign-On service.
- 676 4. If the user does not have an existing security context on the identity provider, or the policy defines
677 that authentication is required, they user will be challenged to provide valid credentials.
- 678 5. The user provides valid credentials and a security context is created for the user.
- 679 6. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a
680 SAML response, within which is a SAML assertion. The SAML specification mandates that the
681 response must be digitally signed. Typically the HTML FORM will contain an input or submit action
682 that will result in an HTTP POST.
- 683 7. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing
684 the SAML response to be sent to the service provider's Assertion Consumer service.
- 685 8. The service provider's Assertion Consumer service validates the digital signature on the SAML
686 Response. If this, and the Assertion validate correctly, it sends an HTTP redirect to the browser
687 causing it to access the TARGET resource, with a cookie that identifies the local session (use of a
688 cookie is implementation specific, other techniques to maintain the security context at the SP can be
689 used). An access check is then made to establish whether the user has the correct authorization to
690 access the www.abc.com web site and the TARGET resource. If the access check passes, the
691 TARGET resource is then returned to the browser.

692 **4.1.2 SP initiated: Redirect->POST binding**

693 In this use case the user attempts to access a resource on www.abc.com. However they do not have
 694 current logon session on this site and their identity is managed by www.xyz.com. A SAML
 695 <AuthnRequest> is sent to their identity provider so that the identity provider can provide back a SAML
 696 assertion concerning the user. An HTTP redirect message is used to deliver the SAML
 697 <AuthnRequest> to the identity provider and an HTTP POST is used to return the SAML response.

698 Figure 15 illustrates the message flow:

699

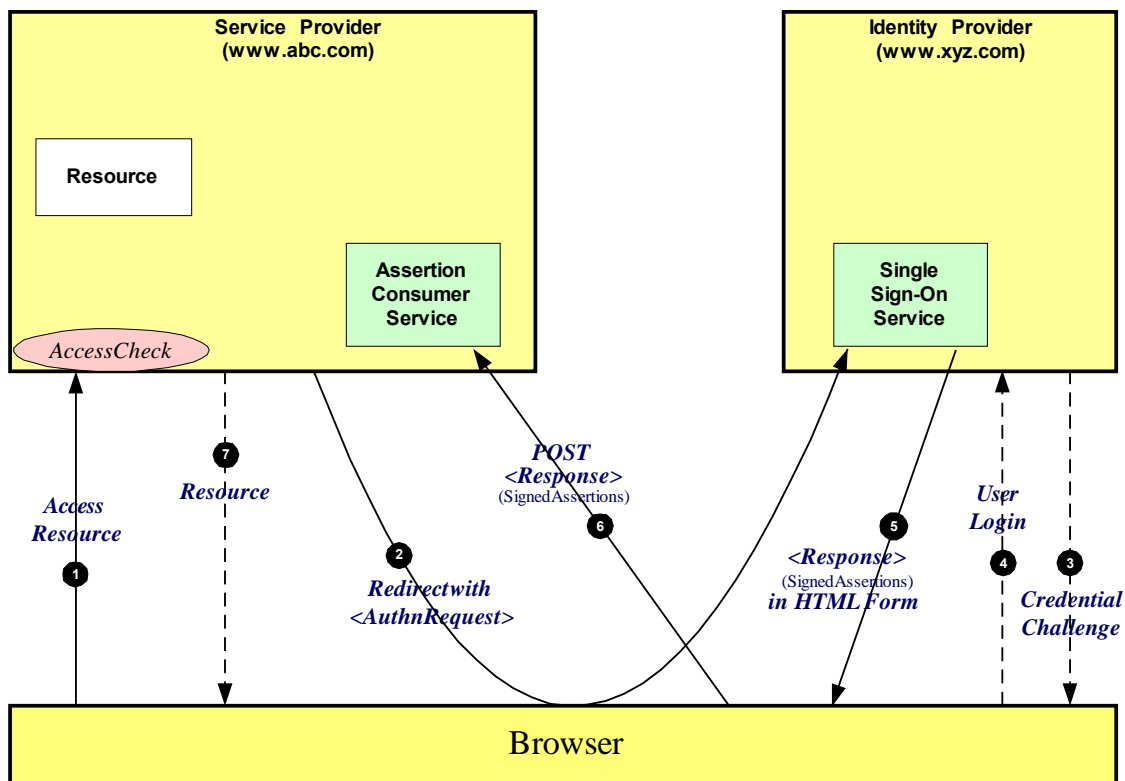


Figure 15: SP initiated: Redirect->POST binding

700 The processing is as follows:

- 701 1. The user attempt to access a resource on www.abc.com. The user does not have any current logon
 702 session (i.e. security context) on this site, and is unknown to it.
- 703 2. The SP sends a redirect message to the browser with HTTP status code of either 302 or 303. The
 704 Location HTTP header contains the destination URI of the Sign-On Service of the identity provider
 705 together with the <AuthnRequest> as a query variable named SAMLRequest. The query string is
 706 encoded using the DEFLATE encoding. The browser processes the redirect message and issues a
 707 GET to the Sign-on Service with the SAMLRequest query parameter.
- 708 3. The Sign-on Service determines whether the user has an existing security context on the identity
 709 provider, or that the policy defines that authentication is required. If the user requires to be
 710 authenticated he will be challenged to provide valid credentials.
- 711 4. The user provides valid credentials and a security context is created for the user.
- 712 5. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a
 713 SAML response, within which is a SAML assertion. The SAML specification mandates that the
 714 response must be digitally signed. Typically the HTML FORM will contain an input or submit action
 715 that will result in an HTTP POST.
- 716 6. The browser, either due to a user action or via an “auto-submit”, issues an HTTP POST containing
 717 the SAML response to be sent to the service provider’s Assertion Consumer service.

718 7. The service provider's Assertion Consumer service validates the digital signature on the SAML
 719 Response. If this, and the Assertion validate correctly, it sends an HTTP redirect to the browser
 720 causing it to access the TARGET resource, with a cookie that identifies the local session (use of a
 721 cookie is implementation specific, other techniques to maintain the security context at the SP can be
 722 used). An access check is then made to establish whether the user has the correct authorization to
 723 access the www.abc.com web site and the TARGET resource. If the access check passes, the
 724 TARGET resource is then returned to the browser.

725 4.1.3 SP initiated: Artifact->POST binding

726 In this use case the user attempts to access a resource on www.abc.com. However they do not have a
 727 current logon session on this site and their identity is managed by www.xyz.com. A SAML artifact is sent
 728 to the identity provider (using an HTTP redirect), which it uses to obtain a SAML `<AuthnRequest>` from
 729 the service provider's SAML Responder. When the identity provider obtains the SAML
 730 `<AuthnRequest>` it provides back to the service provider the SAML response using the HTTP POST
 731 binding mechanism.

732 Figure 16 illustrates the message flow:

733

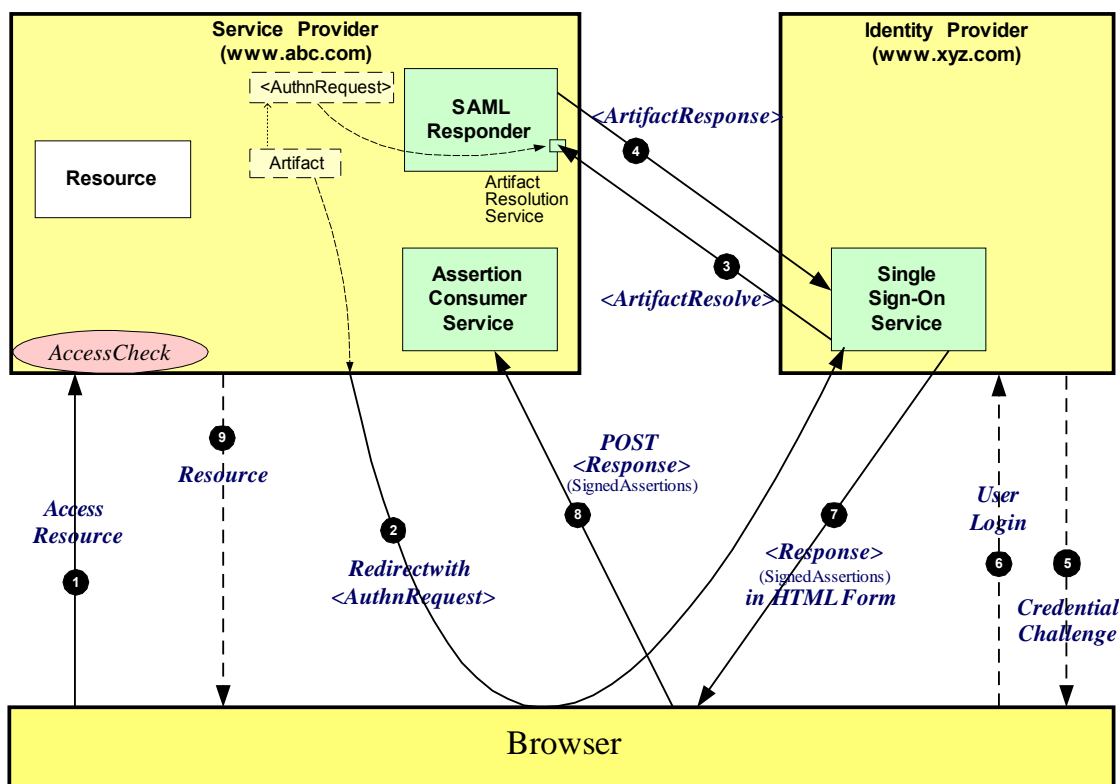


Figure 16: SP initiated: Artifact->POST binding

734 The processing is as follows:

- 735 1. The user attempt to access a resource on www.abc.com. The user does not have any current logon
 736 session (i.e. security context) on this site, and is unknown to it.
- 737 2. The SP generates the `<AuthnRequest>` while also creating an artifact. The artifact contains the
 738 source ID of the www.abc.com SAML responder together with a reference to the assertion (the
 739 MessageHandle). The HTTP Artifact binding allows the choice of either HTTP redirection or a HTML
 740 form as the delivery mechanism to the service provider. The figure shows the use of the HTML form
 741 mechanism. The service provider sends a HTML form back to the browser. The HTML FORM

742 contains the SAML artifact, the control name being `SAMLart`. Typically the HTML FORM will
743 contain an input or submit action that will result in an HTTP POST.

- 744 3. On receiving the HTTP message, the Single Sign-On Service, extracts the SourceID from the SAML
745 artifact. A mapping between source IDs and remote Responders will already have been established
746 administratively. The Assertion Consumer service will therefore know that it has to contact the
747 www.abc.com SAML responder at the prescribed URL. It sends the SAML `<ArtifactResolve>`
748 message to the service provider's SAML responder containing the artifact supplied by its service
749 provider.
- 750 4. The SAML responder supplies back a SAML `<ArtifactResponse>` message containing the
751 `<Authn Request>` previously generated.
- 752 5. The Sign-on Service determines whether the user, for which the `<AuthnRequest>` pertains, has an
753 existing security context on the identity provider, or that the policy defines that authentication is
754 required. If the user requires to be authenticated he will be challenged to provide valid credentials.
- 755 6. The user provides valid credentials and a security context is created for the user.
- 756 7. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a
757 SAML response, within which is a SAML assertion. The SAML specification mandates that the
758 response must be digitally signed. Typically the HTML FORM will contain an input or submit action
759 that will result in an HTTP POST.
- 760 8. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing
761 the SAML response to be sent to the service provider's Assertion Consumer service.
- 762 9. The service provider's Assertion Consumer service validates the digital signature on the SAML
763 Response. If this, and the Assertion validate correctly, it sends an HTTP redirect to the browser
764 causing it to access the TARGET resource, with a cookie that identifies the local session (use of a
765 cookie is implementation specific, other techniques to maintain the security context at the SP can be
766 used). An access check is then made to establish whether the user has the correct authorization to
767 access the www.abc.com web site and the TARGET resource. If the access check passes, the
768 TARGET resource is then returned to the browser.

769 **4.1.4 SP initiated: POST->Artifact binding**

770 In this use case the user attempts to access a resource on www.abc.com. However they do not have
771 current logon session on this site and their identity is managed by www.xyz.com. A SAML
772 `<AuthnRequest>` is sent to their identity provider so that the identity provider can provide back a SAML
773 assertion concerning the user. A HTTP POST message is used to deliver the SAML `<AuthRequest>` to
774 the identity provider. The response is in the form of a SAML Artifact. In this example the SAML Artifact
775 is provided back within an HTTP POST message. The service provider uses the SAML artifact to obtain
776 the SAML response (containing the SAML assertion) from the identity provider's SAML Responder.

777 Figure 17 illustrates the message flow:

778

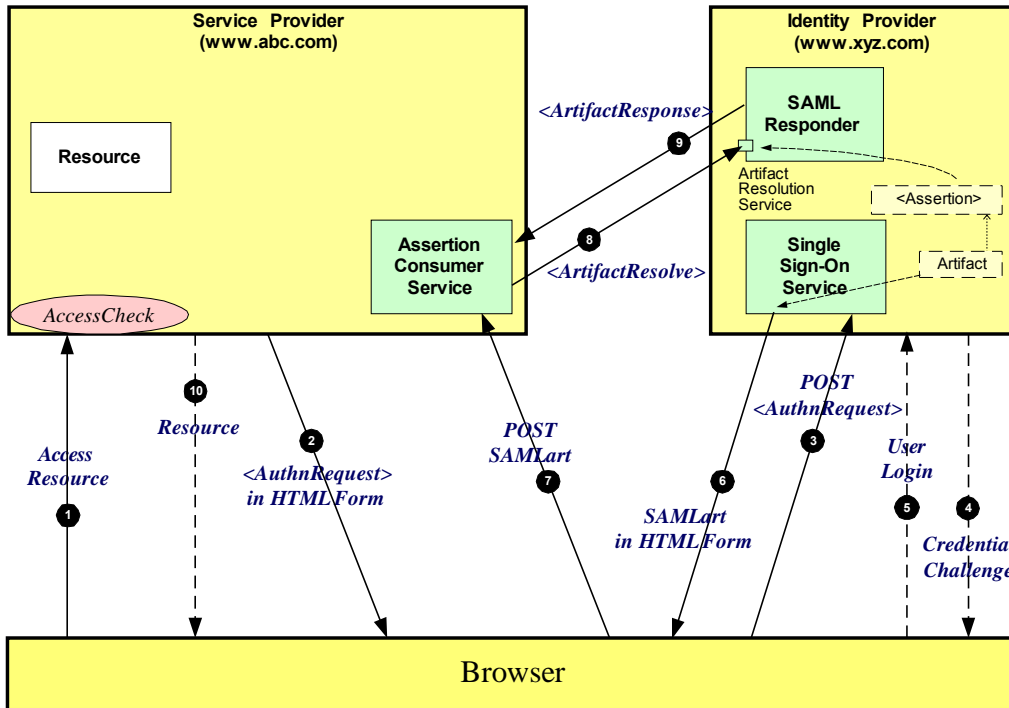


Figure 17: SP initiated: POST->Artifact binding

779 The processing is as follows:

- 780 1. The user attempt to access a resource on www.abc.com. The user does not have any current logon
781 session (i.e. security context) on this site, and is unknown to it.
- 782 2. The SP sends a HTML form back to the browser. The HTML FORM contains a SAML
783 <AuthnRequest> defining the user for which authentication and authorization information is
784 required. Typically the HTML FORM will contain an input or submit action that will result in an HTTP
785 POST.
- 786 3. The browser, either due to a user action or via an “auto-submit”, issues an HTTP POST containing
787 the SAML <AuthnRequest> to the identity provider's Single Sign-On service.
- 788 4. If the user does not have an existing security context on the identity provider, or the policy defines
789 that authentication is required, they user will be challenged to provide valid credentials.
- 790 5. The user provides valid credentials and a security context is created for the user.
- 791 6. The Single Sign-On Service generates an assertion for the user while also creating an artifact. The
792 artifact contains the source ID of the www.xyz.com SAML responder together with a reference to the
793 assertion (the MessageHandle). The HTTP Artifact binding allows the choice of either HTTP
794 redirection or a HTML form as the delivery mechanism to the service provider. The figure shows the
795 use of the HTML form mechanism. The Single Sign-On Service sends a HTML form back to the
796 browser. The HTML FORM contains the SAML artifact, the control name being SAMLart. Typically
797 the HTML FORM will contain an input or submit action that will result in an HTTP POST.
- 798 7. On receiving the HTTP message, the Assertion Consumer service, extracts the SourceID from the
799 SAML artifact. A mapping between source IDs and remote Responders will already have been
800 established administratively. The Assertion Consumer service will therefore know that it has to
801 contact the www.xyz.com SAML responder at the prescribed URL.
- 802 8. The www.abc.com Assertion Consumer service will send a SAML <ArtifactResolve> message to
803 the identity provider's SAML responder containing the artifact supplied by the identity provider.
- 804 9. The SAML responder supplies back a SAML <ArtifactResponse> message containing the
805 assertion previously generated. In most implementations, if a valid assertion is received back, then a
806 session on www.abc.com is established for the user (the relying party) at this point.

807 10. Typically the Assertion Consumer service then sends a redirection message containing a cookie back to
 808 the browser (use of a cookie is implementation specific, other techniques to maintain the security
 809 context at the SP can be used). The cookie identifies the session. The browser then processes the
 810 redirect message and issues an HTTP GET to the TARGET resource on www.abc.com. The GET
 811 message contains the cookie supplied back by the Assertion Consumer service. An access check is
 812 then back to established whether the user has the correct authorization to access the www.abc.com
 813 web site and the index.asp resource.

814 4.1.5 SP initiated: Redirect->Artifact binding

815 In this use case the user attempts to access a resource on www.abc.com. However they do not have
 816 current logon session on this site and their identity is managed by www.xyz.com. A SAML
 817 `<AuthnRequest>` is sent to their identity provider so that the identity provider can provide back a SAML
 818 assertion concerning the user. A HTTP redirect message is used to deliver the SAML `<AuthnRequest>`
 819 to the identity provider. The response is in the form of a SAML Artifact. In this example the SAML
 820 Artifact is provided back within an HTTP POST message. The service provider uses the SAML artifact
 821 to obtain the SAML response (containing the SAML assertion) from the identity provider's SAML
 822 Responder.

823 Figure 18 illustrates the message flow:

824

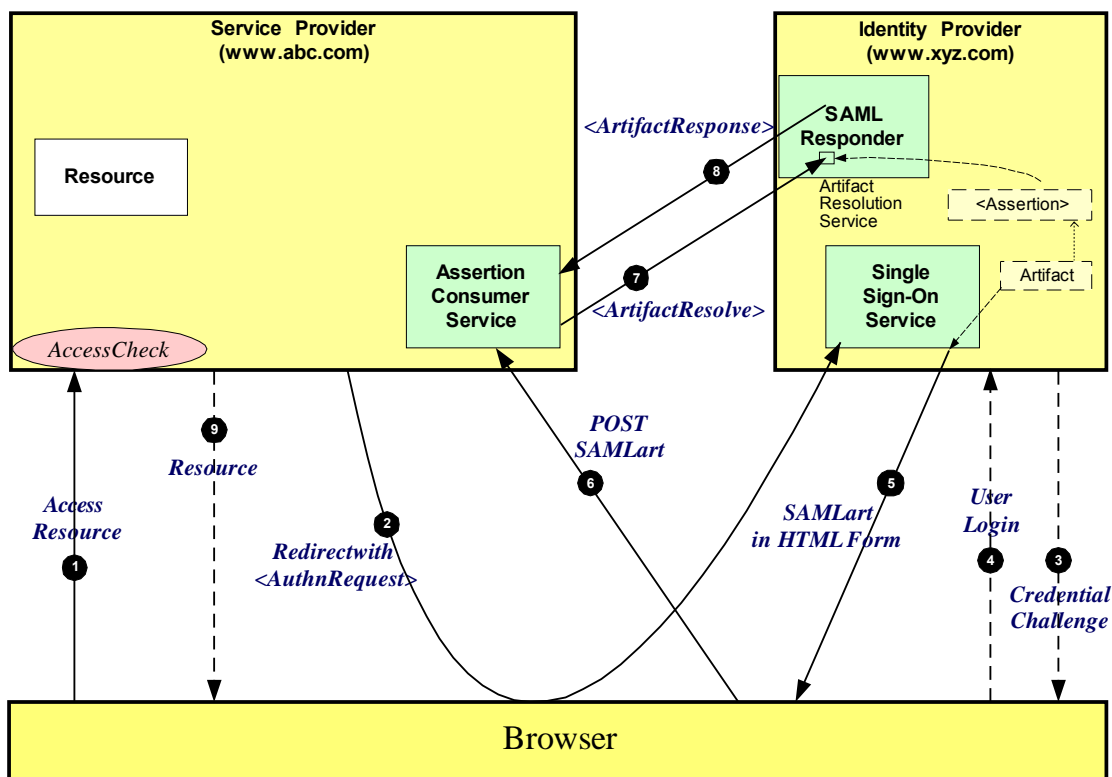


Figure 18: SP initiated: Redirect->Artifact binding

826 The processing is as follows:

- 827 1. The user attempt to access a resource on www.abc.com. The user does not have any current logon
 828 session (i.e. security context) on this site, and is unknown to it.
- 829 2. The SP sends a redirect message to the browser with HTTP status code of either 302 or 303. The
 830 Location HTTP header contains the destination URI of the Sign-On Service of the identity provider
 831 together with the `<AuthnRequest>` as a query variable named `SAMLRequest`. The query string is
 832 encoded using the DEFLATE encoding. The browser processes the redirect message and issues a

- 833 GET to the Sign-on Service with the `SAMLRequest` query parameter.
- 834 3. The Sign-on Service determines whether the user has an existing security context on the identity
835 provider, or that the policy defines that authentication is required. If the user requires to be
836 authenticated he will be challenged to provide valid credentials.
 - 837 4. The user provides valid credentials and a security context is created for the user.
 - 838 5. The Single Sign-On Service generates an assertion for the user while also creating an artifact. The
839 artifact contains the source ID of the www.xyz.com SAML responder together with a reference to the
840 assertion (the MessageHandle). The HTTP Artifact binding allows the choice of either HTTP
841 redirection or a HTML form as the delivery mechanism to the service provider. The figure shows the
842 use of the HTML form mechanism. The Single Sign-On Service sends a HTML form back to the
843 browser. The HTML FORM contains the SAML artifact, the control name being `SAMLart`. Typically
844 the HTML FORM will contain an input or submit action that will result in an HTTP POST.
 - 845 6. On receiving the HTTP message, the Assertion Consumer service, extracts the SourceID from the
846 SAML artifact. A mapping between source IDs and remote Responders will already have been
847 established administratively. The Assertion Consumer service will therefore know that it has to
848 contact the www.xyz.com SAML responder at the prescribed URL.
 - 849 7. The www.abc.com Assertion Consumer service will send a SAML `<ArtifactResolve>` message to
850 the identity provider's SAML responder containing the artifact supplied by the identity provider.
 - 851 8. The SAML responder supplies back a SAML `<ArtifactResponse>` message containing the
852 assertion previously generated. In most implementations, if a valid assertion is received back, then a
853 session on www.abc.com is established for the user (the relying party) at this point.
 - 854 9. Typically the Assertion Consumer service then sends a redirection message containing a cookie back
855 to the browser. The cookie identifies the session (use of a cookie is implementation specific, other
856 techniques to maintain the security context at the SP can be used). The browser then processes the
857 redirect message and issues an HTTP GET to the TARGET resource on www.abc.com. The GET
858 message contains the cookie supplied back by the Assertion Consumer service. An access check is
859 then back to established whether the user has the correct authorization to access the www.abc.com
860 web site and the index.asp resource.

861 **4.1.6 SP initiated: Artifact->Artifact binding**

862 In this use case the user attempts to access a resource on www.abc.com. However they do not have a
863 current logon session on this site and their identity is managed by www.xyz.com. A SAML artifact is sent
864 to the identity provider (using an HTTP redirect), which it uses to obtain a SAML `<AuthnRequest>` from
865 the service provider's SAML Responder. When the identity provider obtains the SAML
866 `<AuthnRequest>` it provides back to the service provider another SAML Artifact. In this example the
867 SAML Artifact is provided back within an HTTP POST message. The service provider uses the SAML
868 artifact to obtain the SAML response (containing the SAML assertion) from the identity provider's SAML
869 Responder.

870

871 Figure 19 illustrates the message flow:

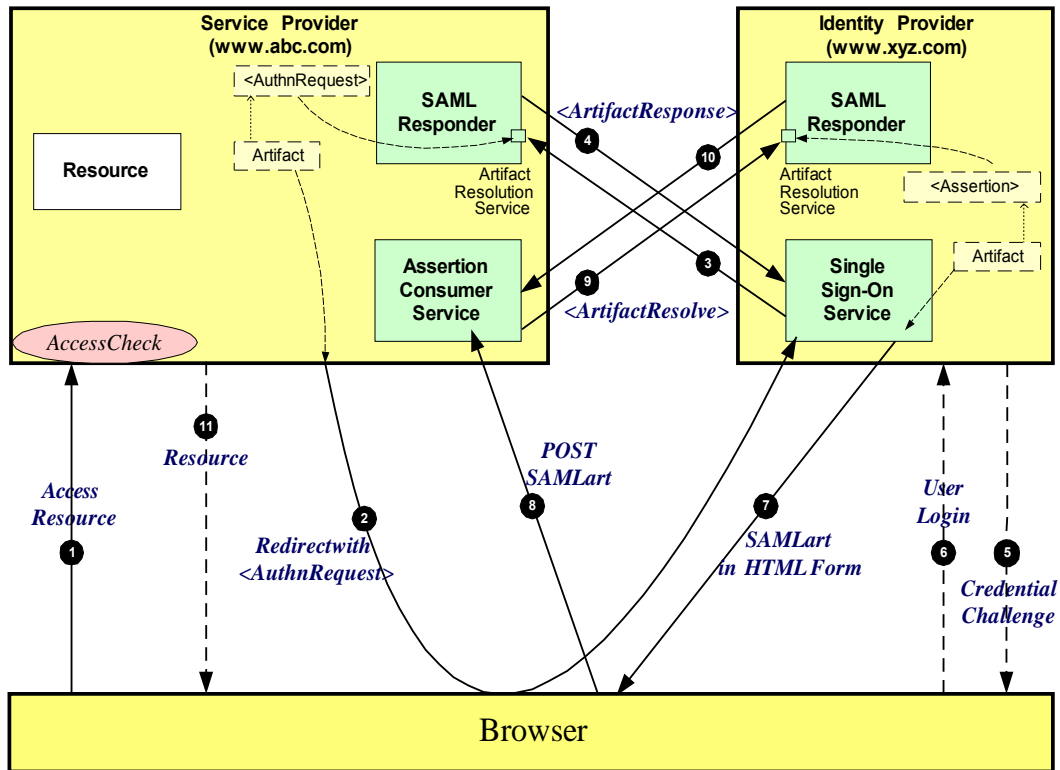


Figure 19: SP initiated: Artifact->Artifact binding

872 The processing is as follows:

- 873 1. The user attempt to access a resource on www.abc.com. The user does not have any current logon
874 session (i.e. security context) on this site, and is unknown to it.
- 875 2. The SP generates the `<AuthnRequest>` while also creating an artifact. The artifact contains the
876 source ID of the www.abc.com SAML responder together with a reference to the assertion (the
877 MessageHandle). The HTTP Artifact binding allows the choice of either HTTP redirection or a HTML
878 form as the delivery mechanism to the service provider. The figure shows the use of the HTML form
879 mechanism. The service provider sends a HTML form back to the browser. The HTML FORM
880 contains the SAML artifact, the control name being `SAMLart`. Typically the HTML FORM will
881 contain an input or submit action that will result in an HTTP POST.
- 882 3. On receiving the HTTP message, the Single Sign-On Service, extracts the SourceID from the SAML
883 artifact. A mapping between source IDs and remote Responders will already have been established
884 administratively. The Assertion Consumer service will therefore know that it has to contact the
885 www.abc.com SAML responder at the prescribed URL. It sends the SAML `<ArtifactResolve>`
886 message to the service provider's SAML responder containing the artifact supplied by its service
887 provider.
- 888 4. The SAML responder supplies back a SAML `<ArtifactResponse>` message containing the
889 `<Authn Request>` previously generated.
- 890 5. The Sign-on Service determines whether the user, for which the `<AuthnRequest>` pertains, has an
891 existing security context on the identity provider, or that the policy defines that authentication is
892 required. If the user requires to be authenticated he will be challenged to provide valid credentials.
- 893 6. The user provides valid credentials and a security context is created for the user.
- 894 7. The Single Sign-On Service generates an assertion for the user while also creating an artifact. The
895 artifact contains the source ID of the www.xyz.com SAML responder together with a reference to the
896 assertion (the MessageHandle). The HTTP Artifact binding allows the choice of either HTTP
897 redirection or a HTML form as the delivery mechanism to the service provider. The figure shows the
898 use of the HTML form mechanism. The Single Sign-On Service sends a HTML form back to the

899 browser. The HTML FORM contains the SAML artifact, the control name being `SAMLart`. Typically
 900 the HTML FORM will contain an input or submit action that will result in an HTTP POST.

901 8. On receiving the HTTP message, the Assertion Consumer service, extracts the SourceID from the
 902 SAML artifact. A mapping between source IDs and remote Responders will already have been
 903 established administratively. The Assertion Consumer service will therefore know that it has to
 904 contact the `www.xyz.com` SAML responder at the prescribed URL.

905 9. The `www.abc.com` Assertion Consumer service will send a SAML `<ArtifactResolve>` message to
 906 the identity provider's SAML responder containing the artifact supplied by the identity provider.

907 10. The SAML responder supplies back a SAML `<ArtifactResponse>` message containing the
 908 assertion previously generated. In most implementations, if a valid assertion is received back, then a
 909 session on `www.abc.com` is established for the user (the relying party) at this point.

910 11. Typically the Assertion Consumer service then sends a redirection message containing a cookie back
 911 to the browser. The cookie identifies the session (use of a cookie is implementation specific, other
 912 techniques to maintain the security context at the SP can be used). The browser then processes the
 913 redirect message and issues an HTTP GET to the TARGET resource on `www.abc.com`. The GET
 914 message contains the cookie supplied back by the Assertion Consumer service. An access check is
 915 then back to established whether the user has the correct authorization to access the `www.abc.com`
 916 web site and the `index.asp` resource.

917 4.1.7 IdP initiated: POST binding

918 In this use case the user has a security context on the identity provider (`www.xyz.com`) and wishes to
 919 access a resource at a service provider (`www.abc.com`). The SAML assertion is transported to the
 920 service provider using the HTTP POST binding.

921 Figure 20 shows the process flow:

922

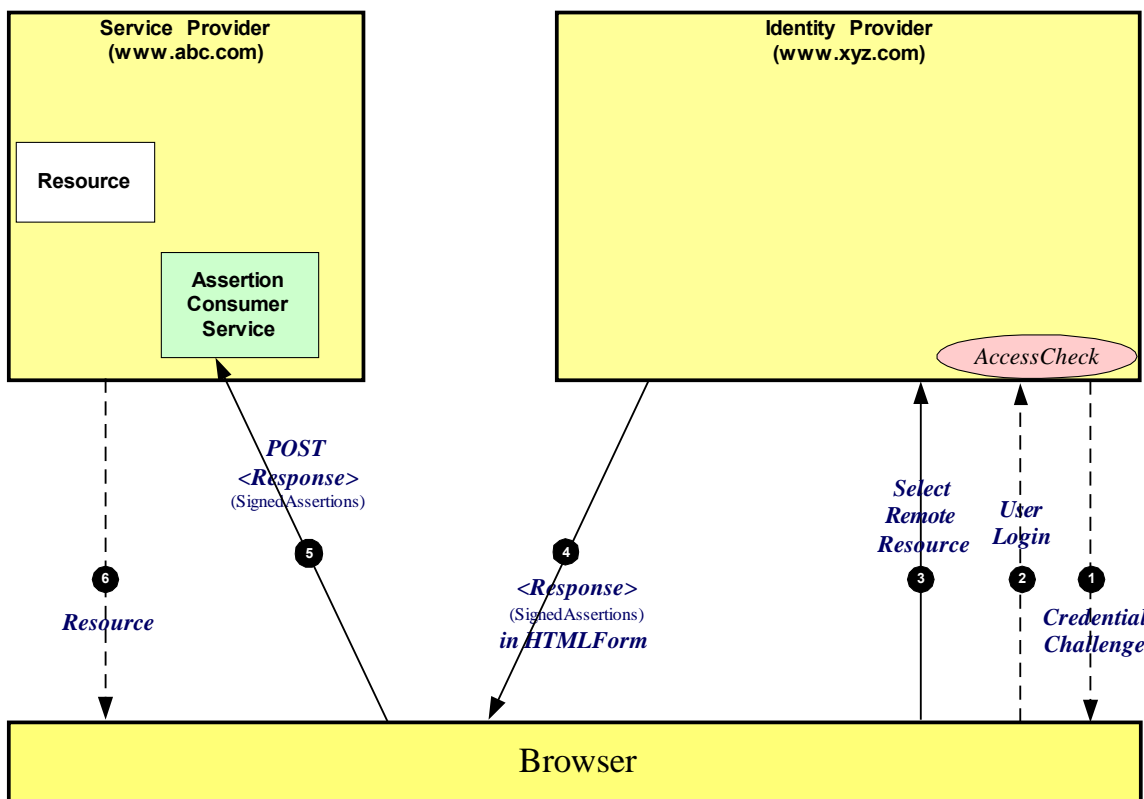


Figure 20: IdP initiated: POST binding

924 The processing is as follows:

- 925 1. At some point the user will have been challenged to supply their credentials to the site www.xyz.com.
- 926 2. The user successfully provides their credentials and has a security context with the identity provider.
- 927 3. The user selects a menu option (or function) on the displayed screen that means the user wants to
928 access a resource or application on another web site www.xyz.com.
- 929 4. The SP sends a HTML form back to the browser. The HTML FORM contains a SAML response,
930 within which is a SAML assertion. The SAML specification mandates that the response must be
931 digitally signed. Typically the HTML FORM will contain an input or submit action that will result in an
932 HTTP POST.
- 933 5. The browser, either due to a user action or via an “auto-submit”, issues an HTTP POST containing
934 the SAML response to be sent to the Service provider's Assertion Consumer service.
- 935 6. The service provider's Assertion Consumer service validates the digital signature on the SAML
936 Response. If this, and the Assertion validate correctly, it sends an HTTP redirect to the browser
937 causing it to access the TARGET resource, withing with a cookie that identifies the local session (use
938 of a cookie is implementation specific, other techniques to maintain the security context at the SP can
939 be used). An access check is then made to establish whether the user has the correct authorization to
940 access the www.abc.com web site and the TARGET resource. If the access check passes, the
941 TARGET resource is then returned to the browser.

942 4.1.8 IdP initiated: Artifact binding

943 In this use case the user has a security context on the identity provider (www.xyz.com) and wishes to
944 access a resource at a service provider (www.abc.com). An artifact is provided to the service provider,
945 which it can use (e.g. “de-reference”) to obtain the associated SAML response from the identity provider.

946 Figure 21 shows the process flow:

947

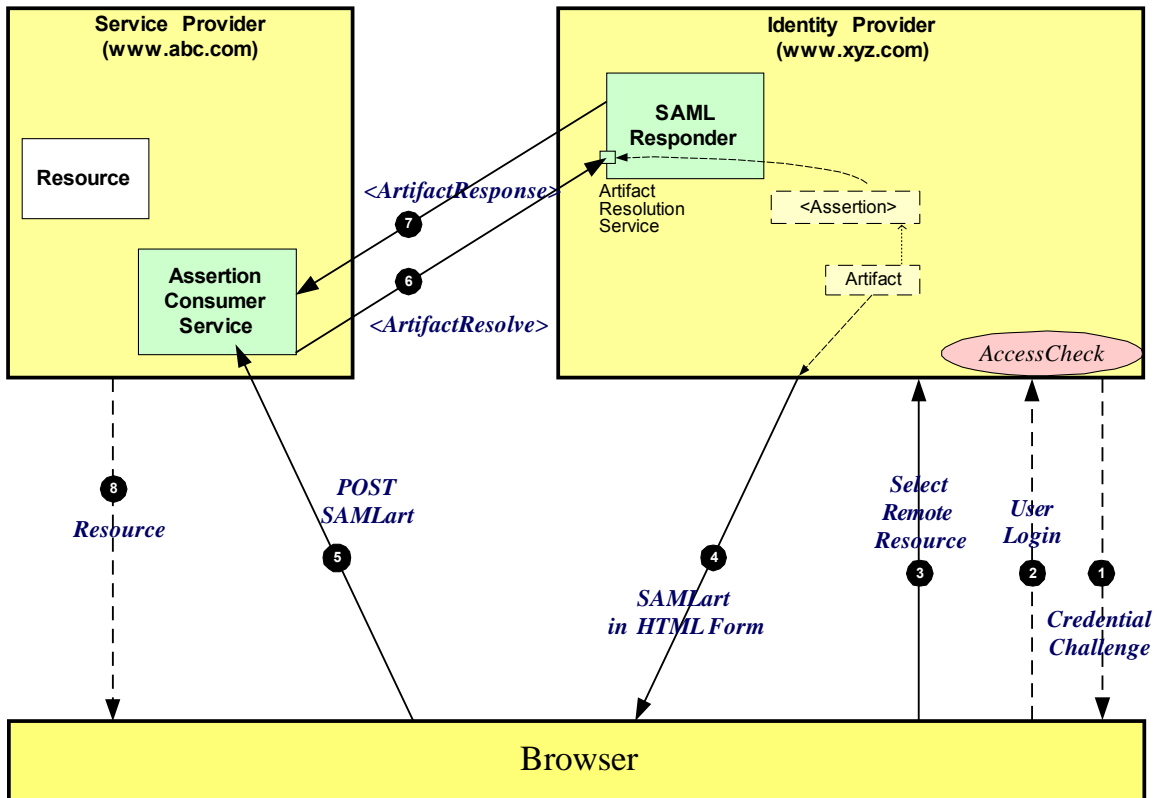


Figure 21: IdP initiated: Artifact binding

949 The processing is as follows:

- 950 1. At some point the user will have been challenged to supply their credentials to the site www.xyz.com.
- 951 2. The user successfully provides their credentials and has a security context with the identity provider.
- 952 3. The user selects a menu option (or function) on the displayed screen that means the user wants to
953 access a resource or application on a destination web site www.abc.com.
- 954 4. The SP generates an assertion for the user while also creating an artifact. The artifact contains the
955 source ID of the www.xyz.com SAML responder together with a reference to the assertion (the
956 MessageHandle). The HTTP Artifact binding allows the choice of either HTTP redirection or a HTML
957 form as the delivery mechanism to the service provider. The figure shows the use of the HTML form
958 mechanism. The service provider sends a HTML form back to the browser. The HTML FORM
959 contains the SAML artifact, the control name being `SAMLart`. Typically the HTML FORM will
960 contain an input or submit action that will result in an HTTP POST.
- 961 5. On receiving the HTTP message, the Assertion Consumer service, extracts the SourceID from the
962 SAML artifact. A mapping between source IDs and remote Responders will already have been
963 established administratively. The Assertion Consumer service will therefore know that it has to
964 contact the www.xyz.com SAML responder at the prescribed URL.
- 965 6. The www.abc.com Assertion Consumer service will send a SAML `<ArtifactResolve>` message to
966 the identity provider's SAML responder containing the artifact supplied by its service provider.
- 967 7. The SAML responder supplies back a SAML `<ArtifactResponse>` message containing the
968 assertion previously generated. In most implementations, if a valid assertion is received back, then a
969 session on www.abc.com is established for the user (the relying party) at this point.
- 970 8. Typically the Assertion Consumer service then sends a redirection message containing a cookie back
971 to the browser. The cookie identifies the session (use of a cookie is implementation specific, other
972 techniques to maintain the security context at the SP can be used). The browser then processes the
973 redirect message and issues an HTTP GET to the TARGET resource on www.abc.com. The GET
974 message contains the cookie supplied back by the Assertion Consumer service. An access check is
975 then back to established whether the user has the correct authorization to access the www.abc.com
976 web site and the `index.asp` resource.

977 4.2 ECP Profile

978 4.2.1 Introduction

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

- 980 • Use of a proxy server, for example a WAP gateway in front of a mobile device which has limited
981 functionality
- 982 • Clients where it is impossible to use redirects
- 983 • It is impossible for the identity provider and service provider to directly communicate (and hence
984 the HTTP Artifact binding cannot be used)
- 985 • Figure 22 illustrates two use cases for using the ECP Profile.
- 986 •
- 987 •

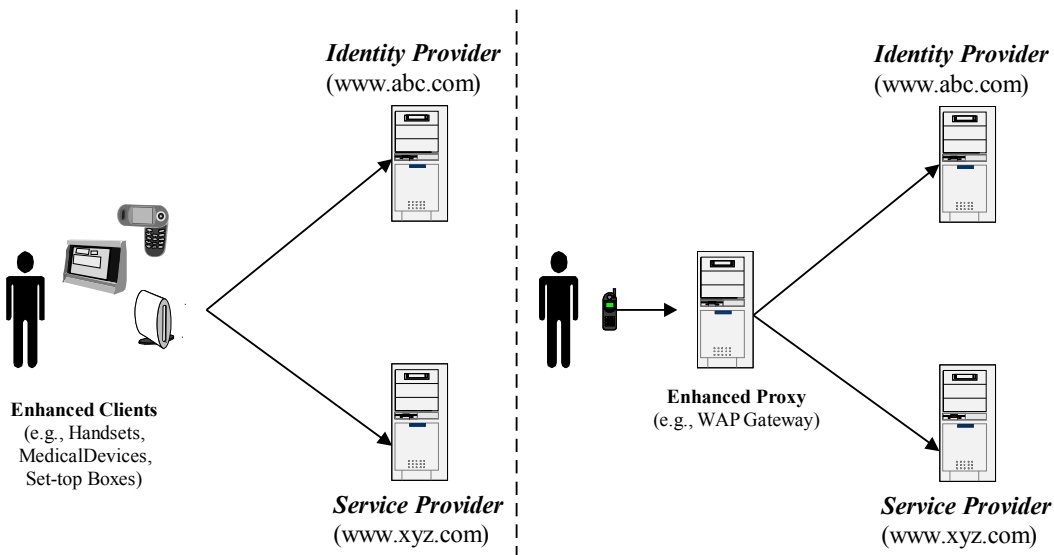


Figure 22: ECP use cases

988 The ECP profile defines a single binding – PAOS (Reserve SOAP). The Profile uses SOAP headers and
 989 SOAP bodies to transport SAML <AuthnRequest> and SAML <Response> messages between the
 990 service provider and the identity provider.

991 4.2.2 ECP Profile using PAOS binding

992 Figure 23 shows the message flows between the ECP, service provider and identity provider. The ECP is
 993 shown as a single logical entity.

Figure 23: ECP with PAOS

994 The processing is as follows:

- 995 1. The ECP wishes to gain access to a resource on the service provider (www.abc.com). The ECP will
 996 issue an HTTP request for the resource. The HTTP request contains a PAOS HTTP header defining
 997 that the ECP service is to be used.
- 998 2. Accessing the resource requires that the principal has a valid security context, and hence a SAML
 999 assertion needs to be supplied to the service provider. In the HTTP response to the ECP an
 1000 <AuthnRequest> is carried within a SOAP body. Additional information, using the PAOS binding, is
 1001 provided back to the ECP
- 1002 3. After some processing in the ECP the <AuthnRequest> is sent to the appropriate identity provider
 1003 using the SAML SOAP binding.
- 1004 4. The identity provider validates the <AuthnRequest> and sends back to the ECP a SAML
 1005 <Response>, again using the SAML SOAP binding.
- 1006 5. The ECP extracts the <Response> and forwards it to the service provider as a PAOS response.
- 1007 6. The service provider sends to the ECP an HTTP response containing the resource originally
 1008 requested.

1009 4.3 Identity Federation Protocols

1010 4.3.1 Introduction

1011 This section provides details of a number of use cases when identities are federated. The following use
 1012 cases are described:

- 1013 • **Single Sign-on with Out-of-Band Account Linking:** Not a true example of federation but a
1014 worth while example of what is required to be established if only the Single Sign-On features of
1015 SAML are used.
- 1016 • **Attribute Federation:** Attributes of the principal, as defined by the identity provider, are used to
1017 link to the account used at the service provider.
- 1018 • **Persistent Federation:** an identity provider federates the identity provider's principal with the
1019 principal's identity at the service provider using a persistent ID.
- 1020 • **Transient Federation:** a transient ID is used to federate between the IdP and the SP.
- 1021 • **Federation Termination:** termination of a Federation

1022 To simplify the examples not each permutation of the bindings are illustrated.

1023 All the examples are based on the use case scenarios originally defined in section 2, with [AirlinesInc.com](#)
1024 being the identity provider.

1025 4.3.2 Single Sign-On with Out-of-band Account Linking

1026 In this example the same user, joe, has accounts on both [AirlinesInc.com](#) and [CarRentalInc.com](#) each with
1027 the same user name (**joe**). The identity stores at both sites are synchronized by some means, for
1028 example either via database synchronization or off-line batch updates. This example purely illustrates
1029 the support for Single Sign-On by SAML. This form of account linking uses persistent identifiers.

1030

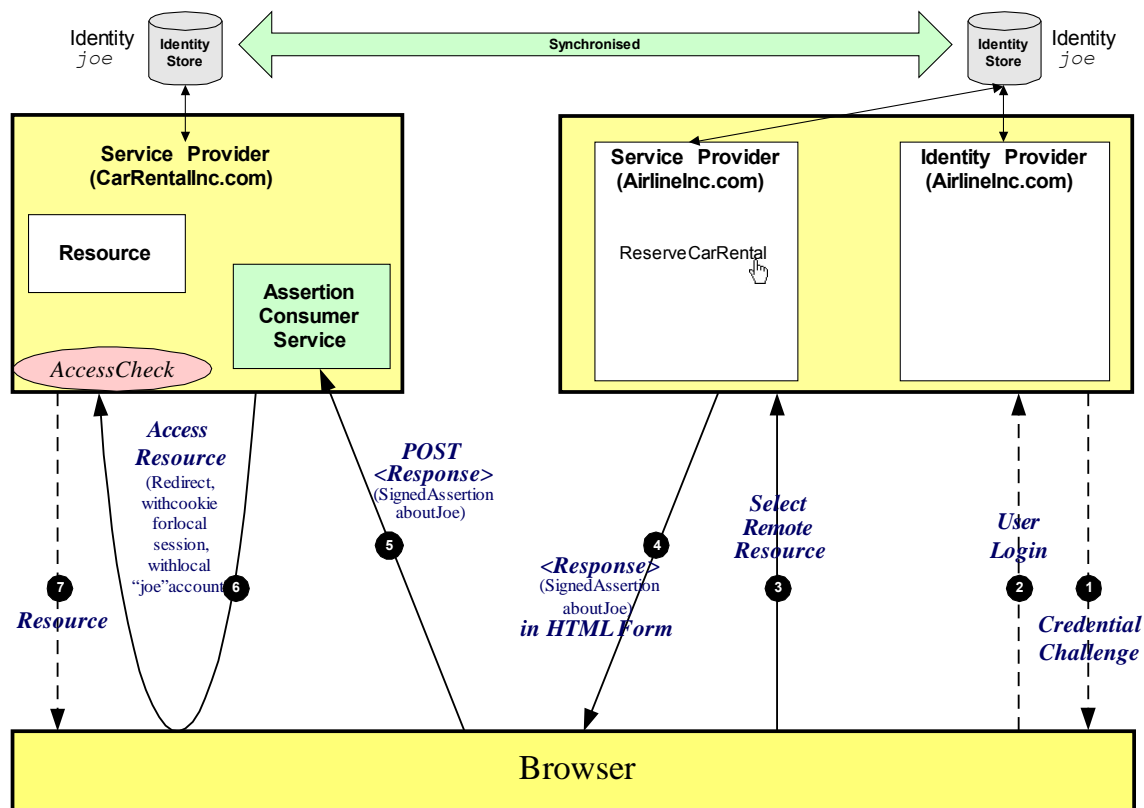


Figure 24: Single Sign-On with Out-of Band account linking

1032 The processing is as follows:

- 1033 1. The user is challenged to supply their credentials to the site [AirlinesInc.com](#).
- 1034 2. The user successfully provides their credentials and has a security context with the [AirlinesInc.com](#)
1035 identity provider.

- 1036 3. The user selects a menu option (or function) on the AirlineInc.com application that means the user
 1037 wants to access a resource or application on CarRentalInc.com.
- 1038 4. The AirlineInc.com service provider sends a HTML form back to the browser. The HTML FORM
 1039 contains a SAML response, within which is a SAML assertion about user **joe**.
- 1040 5. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing
 1041 the SAML response to be sent to the CarRentalInc.com Service provider.
- 1042 6. The CarRentalInc.com service provider's Assertion Consumer service validates the digital signature
 1043 on the SAML Response. If this, and the Assertion validate correctly it creates a local session for user
 1044 **joe**, based on the local joe account. It then sends an HTTP redirect to the browser causing it to
 1045 access the TARGET resource, with a cookie that identifies the local session. An access check is then
 1046 made to establish whether the user **joe** has the correct authorization to access the CarRentalInc.com
 1047 web site and the TARGET resource. The TARGET resource is then returned to the browser.

1048 **4.3.3 Attribute Federation**

1049 Attribute Federation is when the identity provider sends an assertion to the service provider where the
 1050 supplied NameID is not used to map or create a session on the SP, rather an attribute (or possibly
 1051 several attributes) are used to define the account to be used.

1052 [@@Add high-level use case attribute federation figure and explanation here, based on original Figure 1,
 1053 but with attribute aspect emphasized and with details changed to match figure 25]

1054
 1055
 1056
 1057

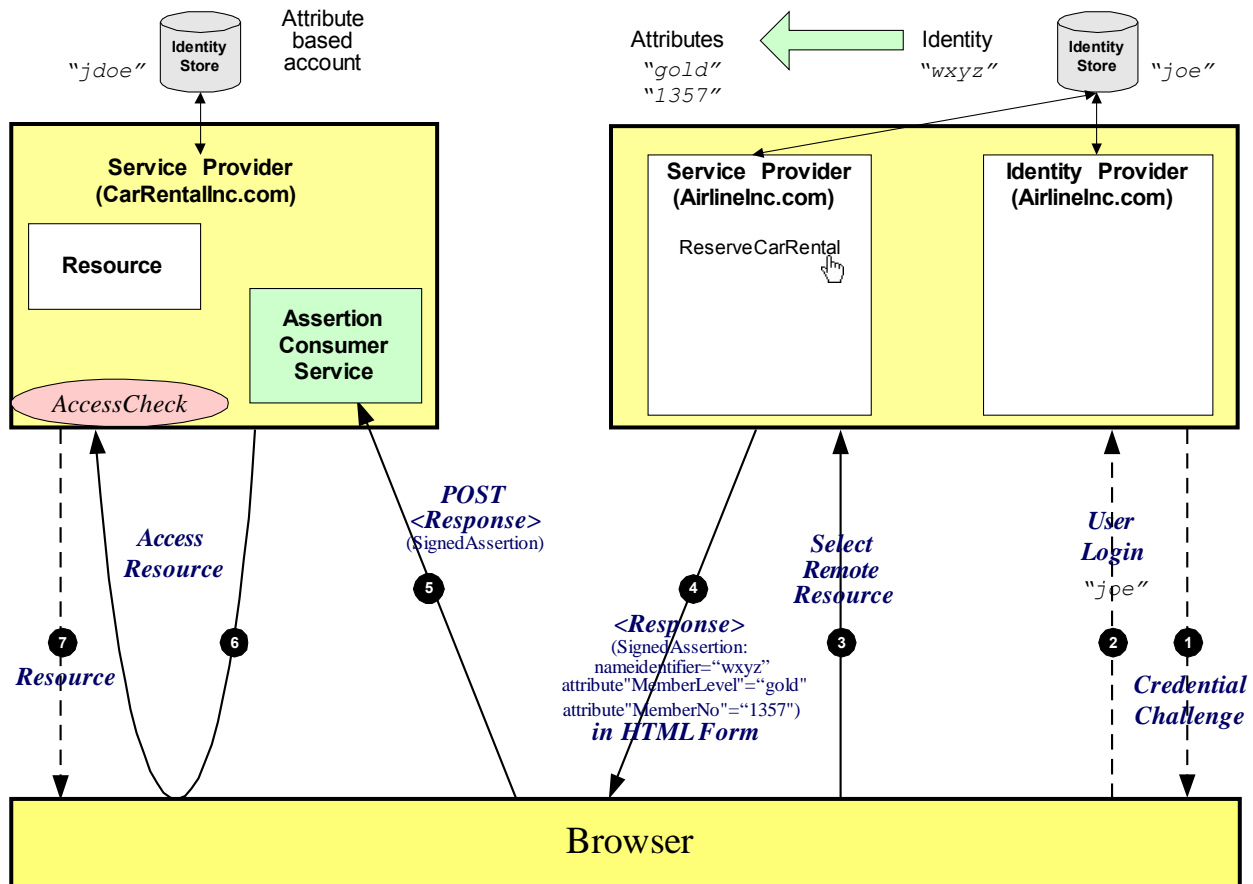


Figure 25: Attribute Federation

1059 In this example the processing is as follows:

- 1060 1. The user is challenged to supply their credentials to the site [AirlineInc.com](#).
- 1061 2. The user successfully provides their credentials and has a security context with the [AirlineInc.com](#)
1062 identity provider, the user named supplied is **joe**.
- 1063 3. The user selects a menu option (or function) on the [AirlineInc.com](#) application that means the user
1064 wants to access a resource or application on [CarRentallnc.com](#).
- 1065 4. The [AirlineInc.com](#) service provider sends a HTML form back to the browser. The HTML FORM
1066 contains a SAML response, within which is a SAML assertion about user **joe**. The name identifier
1067 used in the assertion is an arbitrary value (“wxyz”). The attributes “gold member” and a membership
1068 number attribute (“1357”) are provided. The name **joe** is not contained anywhere in the assertion.
- 1069 5. The browser, either due to a user action or via an “auto-submit”, issues an HTTP POST containing
1070 the SAML response to be sent to the [CarRentallnc.com](#) Service provider.
- 1071 6. The [CarRentallnc.com](#) service provider's Assertion Consumer service validates the digital signature
1072 on the SAML Response. If this, and the Assertion validate correctly it creates a local session. The
1073 session created is for user **jdjoe**. It determines this from a combination of the gold member and
1074 membership number attributes. It then sends an HTTP redirect to the browser causing it to access
1075 the TARGET resource, with a cookie that identifies the local session. An access check is then made
1076 to establish whether the user **jdjoe** has the correct authorization to access the [CarRentallnc.com](#) web
1077 site and the TARGET resource. If the access check passes, the TARGET resource is then returned
1078 to the browser.

1079 **4.3.4 Persistent Federation**

1080 This Federation example is similar to the previous one, except in this case the identity provider provides
1081 to the service provider an assertion with a persistent name identifier using a <Response>. For the
1082 following set of examples will shall illustrate the information maintained at both the IdP and the SP. In all
1083 cases the user **joe** on [AirlineInc.com](#) wishes to federate this account with his **jdjoe** account on
1084 [CarRentallnc.com](#) . There are two use cases that could occur, firstly the user accesses a resource on
1085 [CarRentallnc.com](#) and is then redirected to [AirlineInc.com](#), secondly the user is accessing a resource on
1086 [AirlineInc.com](#) and is directed to [CarRentallnc.com](#). The former we refer to as being SP-initiated the
1087 later IdP-initiated.

1088 Figure 26 illustrates Persistent Federation when it is SP-initiated.

1089

1090

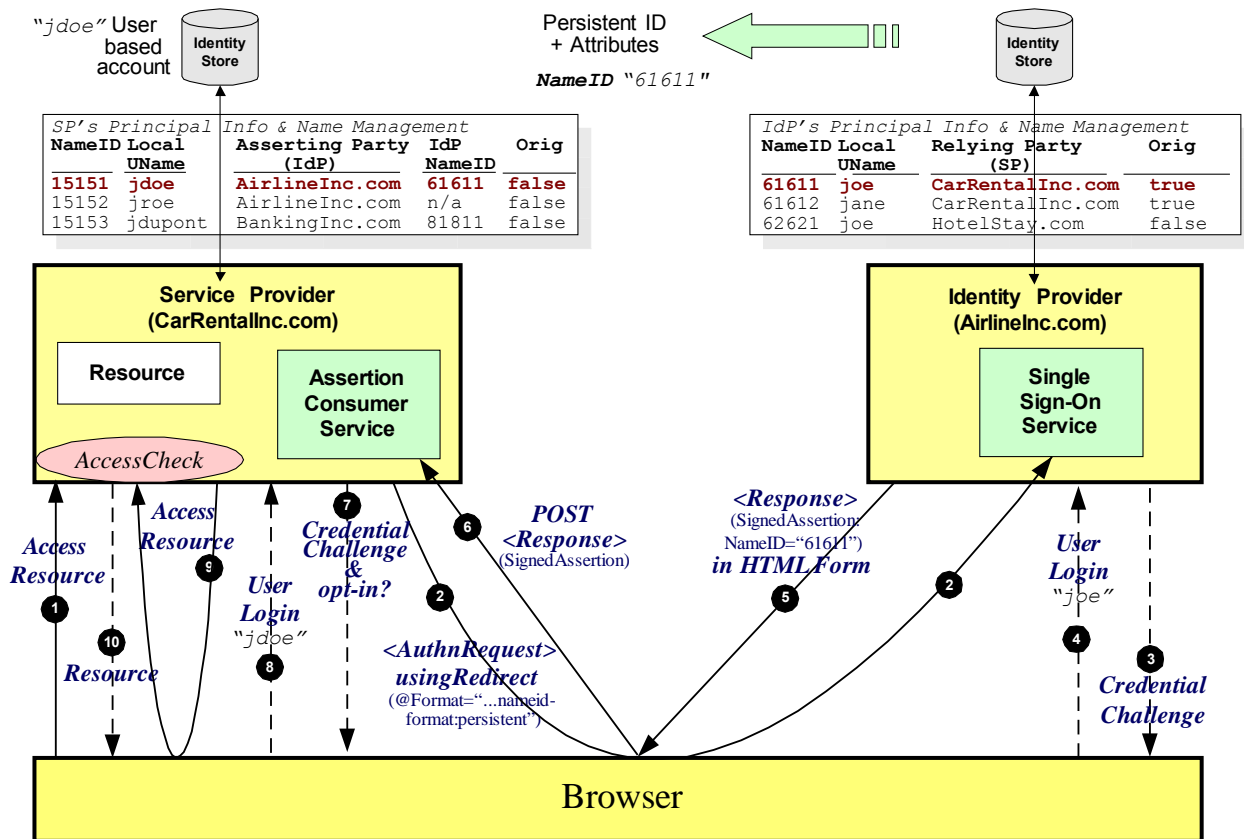


Figure 26: Persistent Federation – SP-initiated

1091 The processing is as follows:

- 1092 1. The user attempt to access a resource on [CarRentallnc.com](#). The user does not have any current
- 1093 logon session (i.e. security context) on this site, and is unknown to it.
- 1094 2. The service provider sends a HTTP redirect to the identity provider ([AirlinesInc.com](#)). The HTTP
- 1095 redirect contains a SAML `<AuthnRequest>` requesting that the identity provider provide an
- 1096 assertion about the requesting user. The request asks that the identity provider sends back a
- 1097 persistent identifier.
- 1098 3. The user will be challenged to provide valid credentials.
- 1099 4. The user provides valid credentials and a security context is created for the user. The user identifies
- 1100 themselves as **joe**. The identity provider looks up user **joe** in its identity store and determines the
- 1101 persistent name identifier to be used for this federation (61611).
- 1102 5. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a
- 1103 SAML response, within which is a SAML assertion. The name identifier used in the assertion is a
- 1104 persistent identifier. The attribute "gold member" and a membership number attribute are provided.
- 1105 The name **joe** is not contained anywhere in the assertion.
- 1106 6. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing
- 1107 the SAML response to be sent to the service provider's Assertion Consumer service.
- 1108 7. The [CarRentallnc.com](#) service provider's Assertion Consumer service validates the digital signature
- 1109 on the SAML Response and validates the SAML assertion. The supplied name identifier is then used
- 1110 to look up to establish whether a previous federation has been established. If a previous federation
- 1111 has been established (because the name identifier maps to a local account) then go to step 9. If no
- 1112 federation is in existence then the user will be challenged to provide valid credentials. Optionally the
- 1113 user could be asked whether he would like to federate the two accounts.
- 1114 8. The user provides valid credentials and identifies themselves as **jdoe**. The persistent name identifier
- 1115 is then stored and registered with the **jdoe** account along with who the identity provider is.

- 1116 9. A session created is for user **jd**oe and an access check is then made to establish whether the user
 1117 **jd**oe has the correct authorization to access the [CarRentalInc.com](#) web site and the TARGET
 1118 10.If the access check passes, the TARGET resource is then returned to the browser.
 1119
 1120 The second use case is when a user accesses a resource on the identity provider ([AirlineInc.com](#)) that
 1121 points to a resource on [CarRentalInc.com](#). Figure 27 illustrates this use case.
 1122

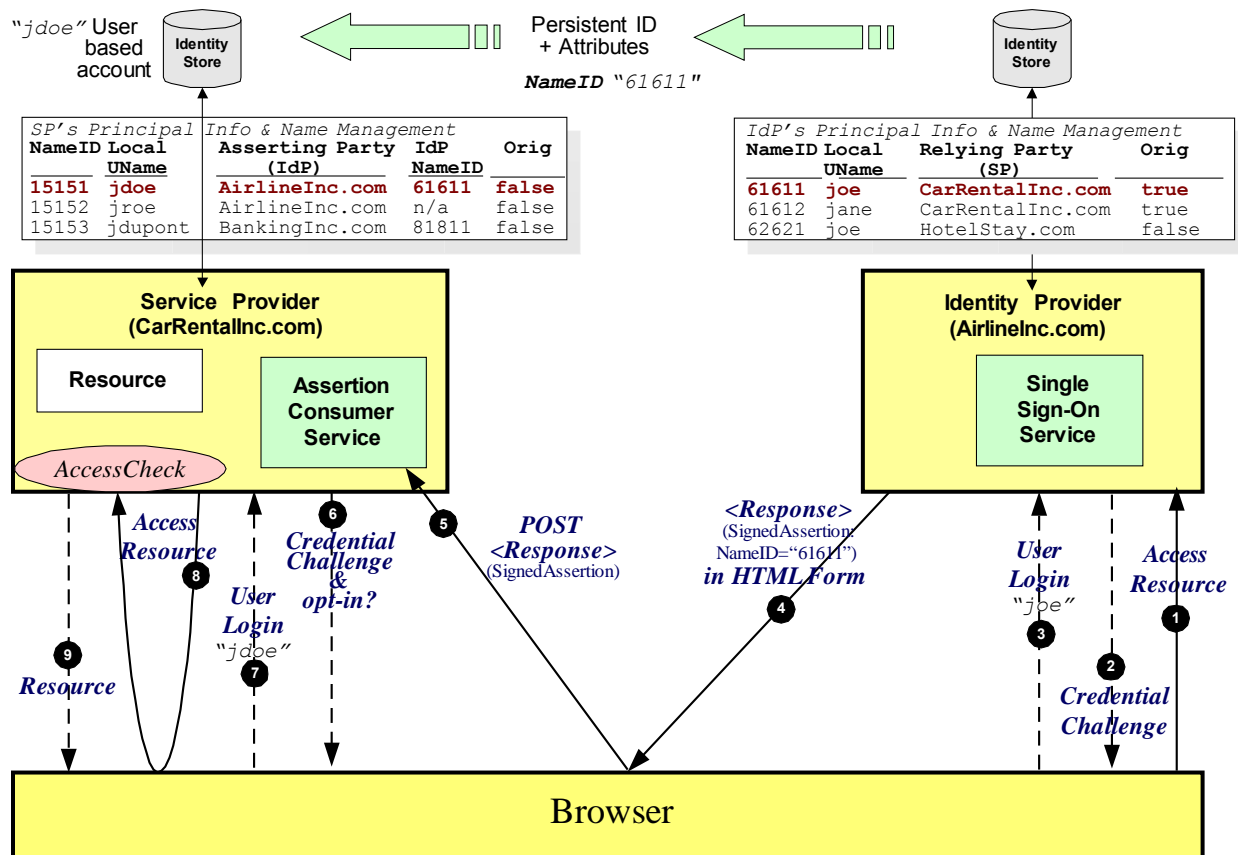


Figure 27: Persistent Federation – IdP-initiated

- 1124 The processing is as follows:
- 1125 1. The user attempt to access a resource on [AirlineInc.com](#) which eventually will pass them over to a
 1126 resource on [CarRentalInc.com](#) .
 - 1127 2. If the user does not have a current security context they will be challenged to provide valid
 1128 credentials.
 - 1129 3. The user provides valid credentials and a security context is created for the user. The user identifies
 1130 themselves as **jo**e. The identity provider looks up user **jo**e in its identity store and determines the
 1131 persistent name identifier to be used for this federation (61611).
 - 1132 4. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a
 1133 SAML response, within which is a SAML assertion. The name identifier used in the assertion is a
 1134 persistent identifier. The attribute "gold member" and a membership number attribute are provided.
 1135 The name **jo**e is not contained anywhere in the assertion.
 - 1136 5. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing
 1137 the SAML response to be sent to the service provider's Assertion Consumer service.
 - 1138 6. The [CarRentalInc.com](#) service provider's Assertion Consumer service validates the digital signature

- 1139 on the SAML Response and validates the SAML assertion. The supplied name identifier is then used
 1140 to look up to establish whether a previous federation has been established. If a previous federation
 1141 has been established (because the name identifier maps to a local account) then go to step 9. If no
 1142 federation is in existence then the user will be challenged to provide valid credentials. Optionally the
 1143 user could be asked whether he would like to federate the two accounts.
- 1144 7. The user provides valid credentials and identifies themselves as **jdoe**. The persistent name identifier
 1145 is then stored and registered with the **jdoe** account along with who the identity provider is.
- 1146 8. A session created is for user **jdoe** and an access check is then made to establish whether the user
 1147 **jdoe** has the correct authorization to access the **CarRentalInc.com** web site and the TARGET
- 1148 9. If the access check passes, the TARGET resource is then returned to the browser.
 1149

1150 4.3.5 Transient Federation

1151 The previous use cases showed the use of persistent identifiers, what if you do want to establish a
 1152 permanent federation. This is where the use of transient identifiers are useful. Transient identifiers
 1153 allow you to:

- 1154 • avoid having to manage userids and passwords at the service provider. Therefore all authentication
 1155 is performed at the identity provider.
- 1156 • have a scheme whereby the service provider does not have to manage specific user accounts, for
 1157 instance it could be a site with a “group-like” access policy.
- 1158 • support a truly anonymous service

1159 As with the Persistent Federation use case one can have SP and IdP-initiated variations. Figure 28
 1160 shows the SP-initiated use case

1161

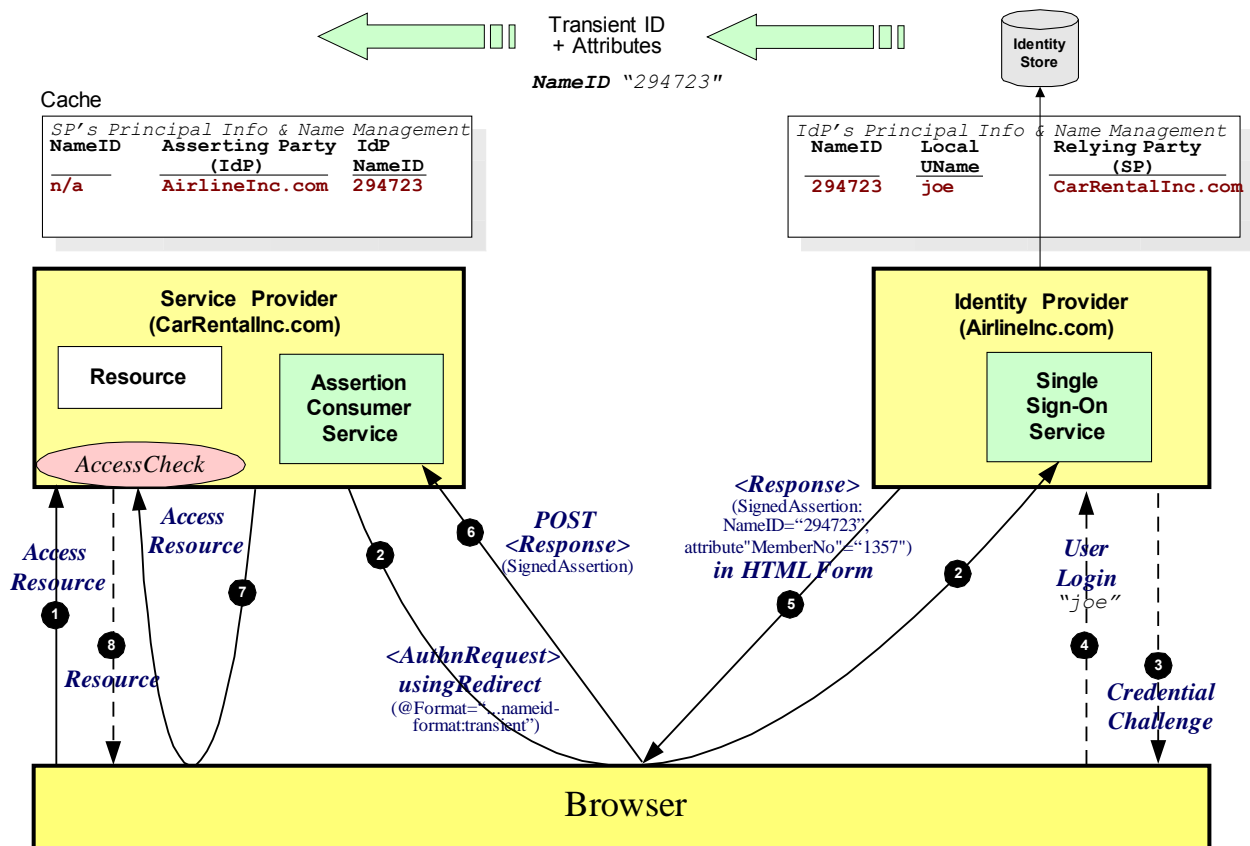


Figure 28: Transient Federation – SP-initiated

- 1162 The processing is as follows:
- 1163 1. The user attempt to access a resource on [CarRentalInc.com](#). The user does not have any current
1164 logon session (i.e. security context) on this site, and is unknown to it.
 - 1165 2. The service provider sends a HTTP redirect to the identity provider ([AirlineInc.com](#)). The HTTP
1166 redirect contains a SAML <AuthnRequest> requesting that the identity provider provide an
1167 assertion about the requesting user. The request asks that the identity provider sends back a
1168 transient identifier.
 - 1169 3. The user will be challenged to provide valid credentials.
 - 1170 4. The user provides valid credentials and a security context is created for the user. The user identifies
1171 themselves as **joe**. The identity provider looks up user **joe** in its identity store and creates a transient
1172 name identifier to be used for this federation (294723).
 - 1173 5. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a
1174 SAML response, within which is a SAML assertion. The name identifier used in the assertion is a
1175 transients identifier. The attribute “gold member” and a membership number attribute (1357) are
1176 provided. The name **joe** is not contained anywhere in the assertion.
 - 1177 6. The browser, either due to a user action or via an “auto-submit”, issues an HTTP POST containing
1178 the SAML response to be sent to the service provider's Assertion Consumer service.
 - 1179 7. The [CarRentalInc.com](#) service provider's Assertion Consumer service validates the digital signature
1180 on the SAML Response and validates the SAML assertion. The supplied name identifier is then used
1181 to dynamically create a session based in the received assertion. In this example it could be the
1182 membership number attribute which maps to the **jdoe** account. A session created is for user **jdoe** and
1183 an access check is then made to establish whether the user **jdoe** has the correct authorization to
1184 access the [CarRentalInc.com](#) web site and the TARGET.
 - 1185 8. If the access check passes, the TARGET resource is then returned to the browser.
- 1186 The IdP-initiated use case is shown in figure 29.

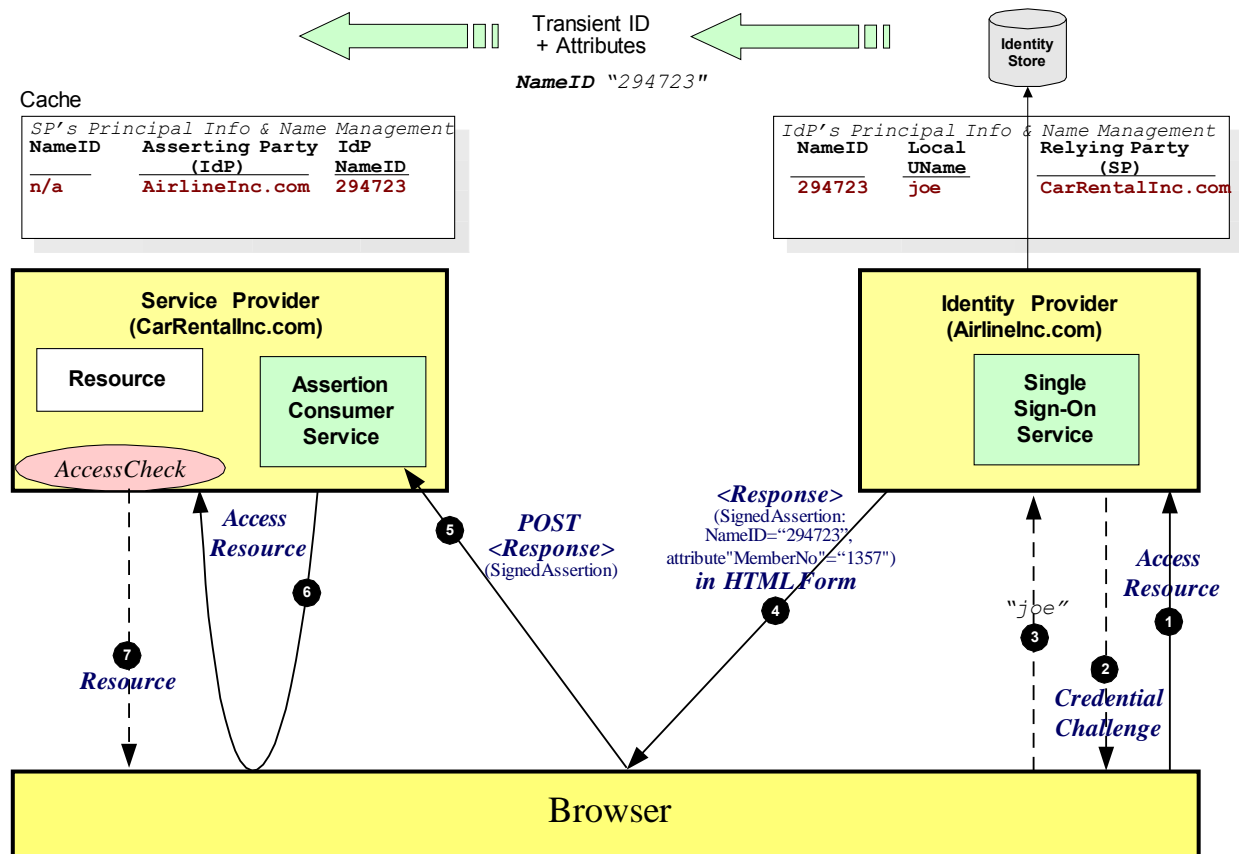


Figure 29: Transient Federation – IdP-initiated

1187 The processing is as follows:

- 1188 1. The user attempt to access a resource on [AirlinesInc.com](#) which eventually will pass them over to a
- 1189 resource on [CarRentalInc.com](#) .
- 1190 2. If the user does not have a current security context they will be challenged to provide valid
- 1191 credentials.
- 1192 3. The user provides valid credentials and a security context is created for the user. The user identifies
- 1193 themselves as **joe**. The identity provider looks up user **joe** in its identity store and creates a transient
- 1194 name identifier to be used for this federation (294723).
- 1195 4. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a
- 1196 SAML response, within which is a SAML assertion. The name identifier used in the assertion is a
- 1197 transient identifier. The attribute "gold member" and a membership number attribute (1357) are
- 1198 provided. The name **joe** is not contained anywhere in the assertion.
- 1199 5. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing
- 1200 the SAML response to be sent to the service provider's Assertion Consumer service.
- 1201 6. The [CarRentalInc.com](#) service provider's Assertion Consumer service validates the digital signature
- 1202 on the SAML Response and validates the SAML assertion. The supplied name identifier is then used
- 1203 to dynamically create a session based in the received assertion. In this example it could be the
- 1204 membership number attribute which maps to the **jdjoe** account. A session created is for user **jdjoe** and
- 1205 an access check is then made to establish whether the user **jdjoe** has the correct authorization to
- 1206 access the [CarRentalInc.com](#) web site and the TARGET.
- 1207 7. If the access check passes, the TARGET resource is then returned to the browser.

1208 4.3.6 Federation Termination

1209 This example builds upon the previous example and shows how a federation can be terminated. In this

1210 case the **jd** account on [CarRentalInc.com](#) service provider has been deleted, hence it wishes to
 1211 terminate the federation with [AirlineInc.com](#) for this user.

1212 The Terminate request is sent to the identity provider using the Name Identifier Management Protocol,
 1213 specifically using the `<ManageNameIDRequest>`. The example shown uses the SOAP over HTTP
 1214 binding which demonstrates a use of the back-channel. Bindings are also defined that permit the
 1215 request (and response) to be sent via the browser using asynchronous "front-channel" bindings, such as
 1216 the HTTP Redirect, HTTP POST, or Artifact bindings.

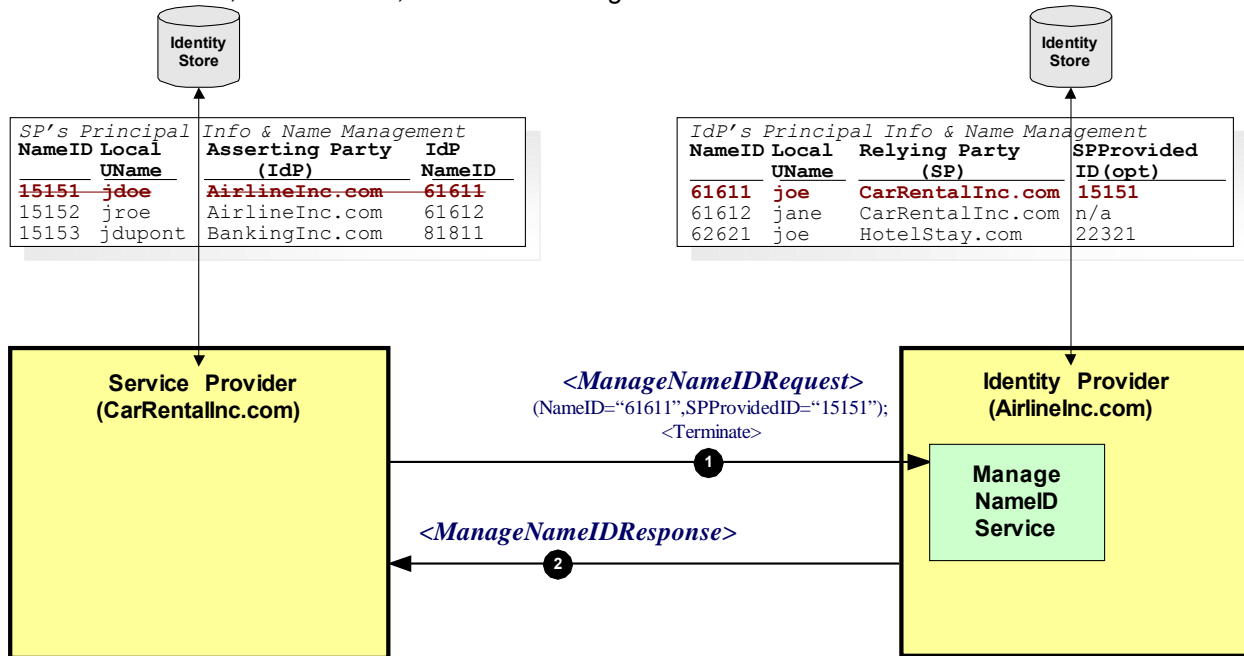


Figure 30: Federation Termination

1218

1219 In this example the processing is as follows:

- 1220 1. The service provider, [CarRentalInc.com](#), determines that the local account, **jd**, should no longer be
 1221 federated. An example of this could be that the account has been deleted. The service provider
 1222 sends to the [AirlineInc.com](#) identity provider a `<ManageIDNameRequest>` defining that the
 1223 persistent identifier (previously established) must no longer be used. The request is carried in a
 1224 SOAP message which is transported using HTTP, as defined by the SAML SOAP binding. The
 1225 request is also digitally signed by the service provider.
- 1226 2. The identity provider verifies the digital signature ensuring that the `<ManageIDNameRequest>`
 1227 originated from a known and trusted service provider. The identity Provider processes the request
 1228 and returns a `<ManageIDNameResponse>` containing a suitable status code response. The
 1229 response is carried within a SOAP over HTTP message and is digitally signed.

1230

1231 4.4 Single Logout

1232 Single Logout permits near real-time session logout of all participants in a session. A request can be
 1233 issued by any session participant to request that the session is to be finished. In this example a user on
 1234 the [CarRentalInc.com](#) service provider decides that they wish to logout out of the session.

1235 The example shows the use of the SOAP over HTTP binding, however asynchronous front-channel
 1236 bindings can also be used.

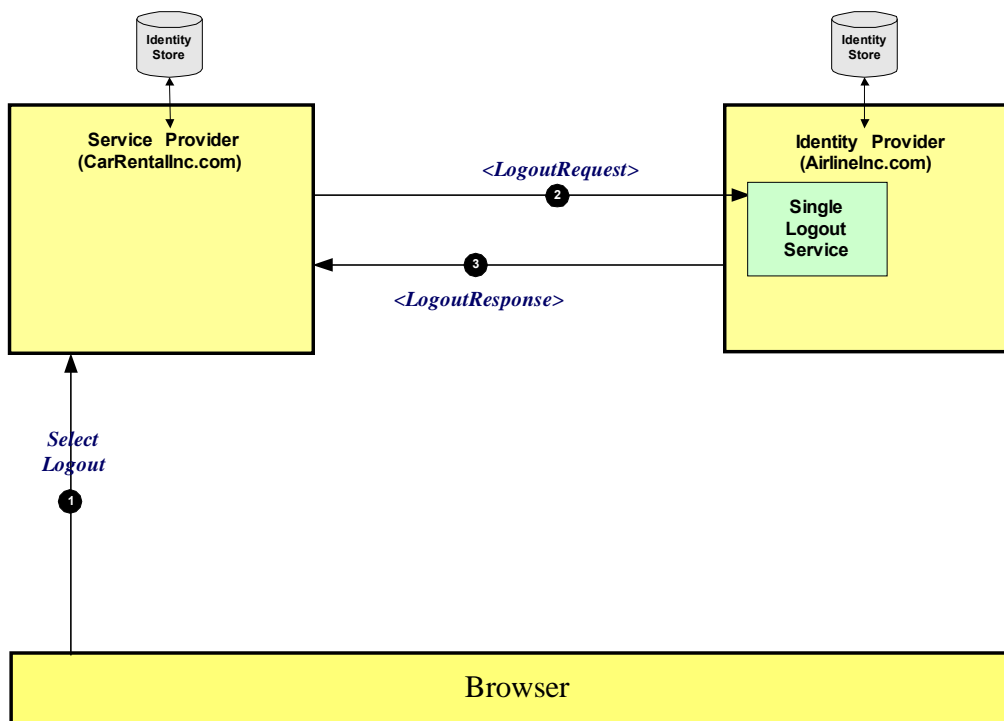


Figure 31: Single Logout

1237

1238 The processing is as follows:

- 1239 1. A user was previously authenticated by the [AirlinesInc.com](#) identity provider and is interacting with the
 1240 [CarRentallnc.com](#) service provider. The user decides to terminate their session and logout.
- 1241 2. The service provider, sends to the [AirlinesInc.com](#) identity provider a `<LogoutRequest>` defining
 1242 that the session is to be logged out. The request identifies the principal to be logged out, by using the
 1243 `<NameID>` element, as well as providing a `<SessionIndex>` element to uniquely identify the
 1244 session being closed. The request is carried in a SOAP message which is transported using HTTP,
 1245 as defined by the SAML SOAP binding. The request is also digitally signed by the service provider.
- 1246 3. The identity provider verifies the digital signature ensuring that the `<LogoutRequest>` originated
 1247 from a known and trusted service provider. The identity Provider processes the request and returns
 1248 a `<LogoutResponse>` containing a suitable status code response. The response is carried within a
 1249 SOAP over HTTP message and is digitally signed.

1250

1251 If in step 3 the identity provider determines that other service providers are participants in the session,
 1252 then the identity provider will send `<LogoutRequest>` messages to them. Figure 32 illustrates this
 1253 processing. Notice that different bindings are used between the two different exchanges with the service
 1254 providers. One using the redirect binding the other using a back channel, illustrating the point that
 1255 different combinations of bindings can be used.

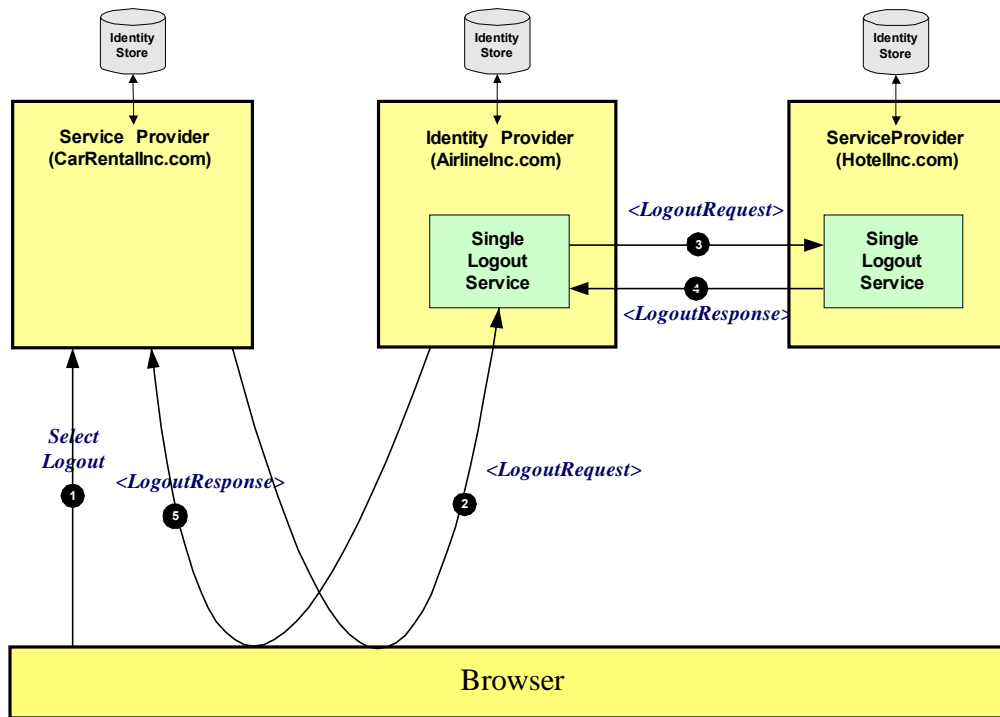


Figure 32: Multiple Logouts

1256 The two previous examples showed the user instigating the logout. Of course the service provider itself
 1257 could initiate the logout, and in that case step 1 would not occur. There is one other use case possible,
 1258 and that is when the identity provider initiates the logout. Figure 33 illustrates this example.
 1259

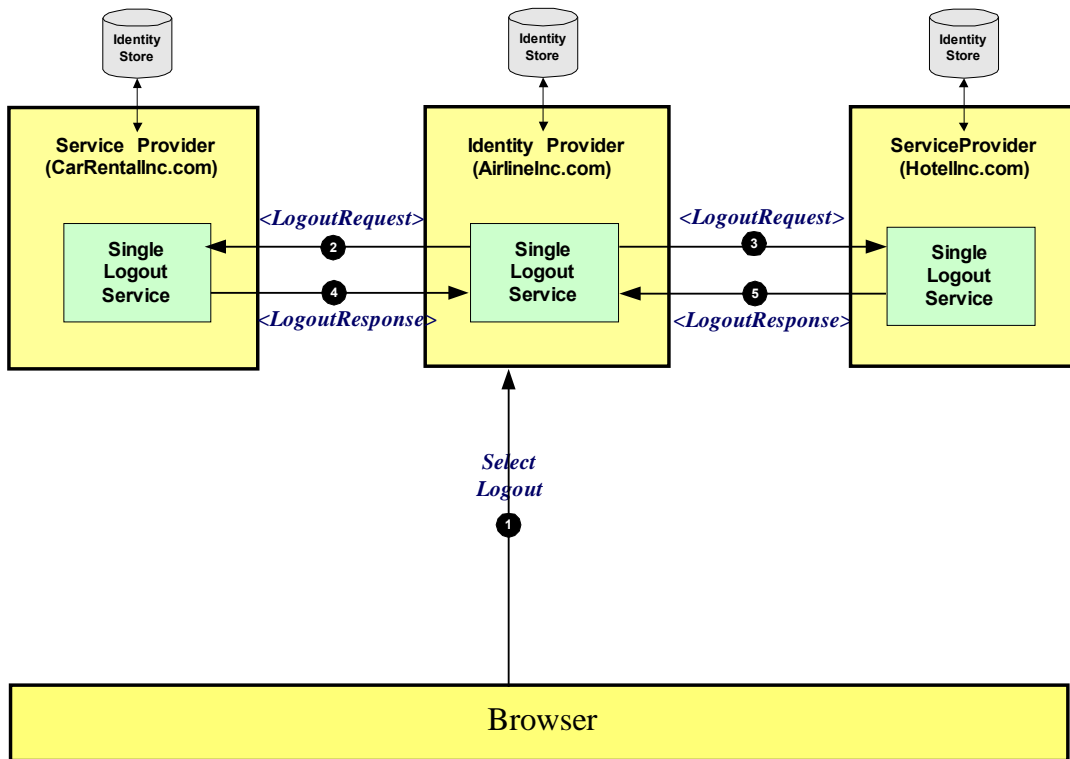


Figure 33: Multiple Logouts – identity provider initiated

5 Documentation roadmap

1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288

- **Security Assertion Markup Language (SAML) V2.0 Executive Overview.** (sstc-saml-exec-overview-2.0) Provides a brief overview of SAML and describes its primary benefits.
- **Security Assertion Markup Language (SAML) V2.0 Technical Overview.** (sstc-saml-tech-overview-2.0). This document.
- **Assertions and Protocol for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-core-2.0). Defines the syntax and semantics for XML-encoded assertions about authentication, attributes and authorization, and for the protocol that conveys this information.
- **Security and Privacy Considerations for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-sec-consider-2.0). Describes and analyzes the security and privacy properties of SAML
- **Bindings for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-bindings-2.0). Defines protocol bindings for the use of SAML assertions and request-response messages in communications protocols and frameworks.
- **Profiles for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-profiles-2.0). Defines how the assertions, protocols and bindings combine to define specific profiles.
- **Conformance Program Specification for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-conform-2.0). Describes the program and technical requirements for SAML conformance.
- **Metadata for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-metadata-2.0). Describes metadata format to enable configuration data to be shared in a standardized format.
- **Glossary for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-glossary-2.0). Defines terms used throughout the OASIS Security Assertion Markup Language (SAML) specifications.
- **Authentication Context for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-authn-context-2.0). Defines a syntax for the definition of authentication context declarations.

1289 **6 Comparison Between SAML V2.0 and SAML V1.1**

1290 **Note that this appendix contains information that is known to be out of date; it only covers differences**
1291 **through about core-10 in most cases. To be updated soon with other differences.**

1292 SAML V2.0 constitutes a large-scale realization of features derived from the Liberty Alliance Identity
1293 Federation Framework (ID-FF) V1.2 specifications that were contributed to the SSTC in 2003, along with
1294 other requested features, improvements, and streamlining.

1295 The on-the-wire representations of SAML V2.0 assertions and messages are incompatible with SAML
1296 V1.x processors. As is explained in the SAML assertions and protocols specification [SAMLCore], only
1297 new major versions of SAML (of which this is one) typically cause this sort of incompatibility. However,
1298 most such incompatibility is syntactic in nature; the expressiveness of SAML has increased rather than
1299 markedly changed.

1300 The differences are described in the sections below. Note that these descriptions may not be complete;
1301 for a full accounting of precise differences to SAML V1.1 specification text, see [some change-bar
1302 version of specs that doesn't exist yet].

1303 **6.1 Differences in the Organization of the Specifications**

- 1304 • The assertion and protocol (“core”) specification is now referred to as **Assertions and Protocols**,
1305 because it now defines a set of protocols.
- 1306 • Processing rules are now clearly called out in each protocol.
- 1307 • Bibliographic references have been divided into normative and non-normative categories.
- 1308 • The single bindings and profiles specification has been split into two documents, one for bindings
1309 and one for profiles, and the latter now includes “attribute profiles”.
- 1310 • There is a new authentication context specification and several accompanying schemas.
- 1311 • There is a new metadata specification and an accompanying schema.
- 1312 • There is a new non-normative executive overview.
- 1313 • The conformance specification now serves explicitly as the entry point for the SAML V2.0 OASIS
1314 Standard specifications.

1315 **6.2 Versioning Differences**

- 1316 • The SAML assertions namespace (known by its convention prefix `saml:`) and protocols
1317 namespace (known by its conventional prefix `samlp:`) now contain the string “2.0” in recognition of
1318 this new major version of SAML.
- 1319 • The `MajorVersion` and `MinorVersion` attributes that appeared on various elements have been
1320 changed to a single `Version` attribute that must have the value “2.0”.
- 1321 • A series of changes planned during SAML the V1.x design cycles have been made:
 - 1322 • The deprecated `<AuthorityBinding>` element has been removed.
 - 1323 • The deprecated `<RespondWith>` element has been removed.
 - 1324 • The deprecated name identifier and artifact URI-based identifiers have been removed.
 - 1325 • URI references are now required to be absolute.
 - 1326 • The description of appearance of the `<Status>` element in SOAP messages has been
1327 improved.

1328 **6.3 Subject and Subject Confirmation Differences**

- 1329 • The `<SubjectStatement>` element and its type have been removed.
- 1330 • The `<Subject>` element has been moved up to appear on the `<Assertion>` element, where the

- 1331 subject so specified applies to all enclosed statements. The `<Subject>` element is now optional
1332 for extensibility reasons, but is required for all SAML-specified statement types.
- 1333 • The new **BaseID** complex type is an extension point that permits non-string identification of
1334 subjects.
 - 1335 • The `<SubjectConfirmation>` element is now repeatable, with the formerly repeatable
1336 `<ConfirmationMethod>` element now an attribute within that element.
 - 1337 • The `<ds:KeyInfo>` element is now allowed only inside `<SubjectConfirmationData>`.
1338 Further, the usage of `<ds:KeyInfo>` within `<SubjectConfirmationData>` has been clarified
1339 to more clearly allow for impersonation.
 - 1340 • A set of generic attributes in `<SubjectConfirmationData>` have been defined for use in
1341 constraining the bearer method or other confirmation methods. Overall assertion validity is more
1342 flexible within profiles that use bearer as a result.

1343 6.4 Encryption-Related Differences

- 1344 • The name identifier structure, the attribute structure, and the assertion structure have all been
1345 refactored to allow encryption.

1346 6.5 Attribute-Related Differences

- 1347 • The `AttributeNameSpace` field has been removed in favor of `NameFormat`, and two new URI-
1348 based identifiers of attribute name format types have been defined for use in this field. This field
1349 can be left blank, as a default has been defined.
- 1350 • The name of the `AttributeName` field has been changed to just `Name`.
- 1351 • Arbitrary XML attributes can now appear on the `<Attribute>` element without a supporting
1352 extension schema.
- 1353 • Clearer instructions have been provided for how to represent null and multi-valued attributes.
- 1354 • A series of attribute profiles has now been defined. They provide for proper interpretation of
1355 attributes specified using common attribute/directory technologies.

1356 6.6 Differences in the Request-Response Mechanism

- 1357 • The request datatype hierarchy has been reorganized; all queries are now kinds of requests, not
1358 inside requests, and the plain `<Query>` has been removed.
- 1359 • `Consent` and `<Extensions>` constructs have been added to all requests and responses.
- 1360 • The `Issuer` field is now an element and is based on the same datatype that underlies name
1361 identifiers, for more unified treatment. Also, in addition to appearing on assertions, it now appears
1362 on requests and responses as well.
- 1363 • The response type hierarchy has been reorganized; most response elements in the various
1364 protocols are simply of **StatusResponseType**.
- 1365 • New status codes have been added to reflect possible statuses when using the new protocols.
1366 Status codes are now URIs instead of QNames.

1367 6.7 Differences in the Protocols for Retrieving Assertions

- 1368 • Instead of the `<AssertionIDReference>` in `<Request>`, the `<AssertionIDRequest>`
1369 element is now used to get an assertion by means of its ID.
- 1370 • Instead of the `<AssertionArtifact>` element to retrieve assertions in a response message,
1371 now a special `<ArtifactResolve>` protocol is used to get SAML protocol messages by means of
1372 an artifact. All types of protocol messages can theoretically be retrieved in this fashion, but in
1373 practice only some kinds will appear in profiles.

1374 6.8 Session-Related Differences

- 1375 • The `<AuthnStatement>` element can now contain a `SessionIndex` attribute in support of single
1376 logout and other session management requirements.
- 1377 • There is a new single logout protocol for near-simultaneous logout from multiple related sessions.

1378 6.9 Federation-Related Differences

- 1379 • There is a new protocol for requesting that authentication be performed and a new assertion with
1380 an authentication statement returned. As part of this, the policy for the desired form of name
1381 identifier can be specified.
- 1382 • In such an assertion, it is now possible to specify many more details about the authentication that
1383 was performed using the new authentication context schemas; the old `AuthenticationMethod`
1384 field has been removed.
- 1385 • There is a new federated name management (registration and deregistration) protocol.
- 1386 • There is a new name identifier mapping protocol.

1387 6.10 Differences in Bindings and Profiles

- 1388 • A lot of profile detail has been refactored out to become new, more generic bindings; the profiles
1389 are much thinner. For example, there's now an HTTP redirect/HTTP POST binding.
- 1390 • There is a new HTTP-based binding added for retrieval of assertions by means of URIs.
- 1391 • A PAOS (reverse SOAP) binding has been added.
- 1392 • An enhanced client profile has been added.
- 1393 • The two original browser profiles (browser/artifact and browser/POST) have become a single web
1394 SSO profile.
- 1395 • A set of mechanisms for relaying state have been added to most of the bindings.
- 1396 • As noted above, a series of attribute profiles has now been defined.

1397 6.11 Other Differences

- 1398 • A number of elements, attributes, and types have been renamed for brevity and consistency. **List**
1399 **them**
- 1400 • The SAML schema extensibility mechanisms have been rationalized and, in some cases,
1401 enhanced. XSD element substitution has been blocked in favor of type extension. The
1402 `<xs:anyAttribute>` wildcard has been added selectively to structures where it has been
1403 deemed valuable to add arbitrary “foreign” attributes without having to create a schema extension;
1404 these structures include subject confirmation data and attributes.
- 1405 • The notion of special “SAML namespaces” (attribute namespaces and action namespaces) has
1406 been deemphasized, with attribute namespaces being removed entirely in favor of URIs as
1407 attribute format identifiers.
- 1408 • The `<ds:Signature>` that allows for the digital signing of assertions and messages has been
1409 positioned earlier in the respective content models.
- 1410 • The authorization decision feature (statement and query) has been frozen; if more functionality is
1411 desired, it is suggested that XACML [XACML] be used.
- 1412 • Two new conditions, `<ProxyRestriction>` element and `<OneTimeUse>`, have been added. The
1413 relationship of the latter to the `NotBefore` and `NotOnOrAfter` conditions has been delineated.
- 1414 • The terminology used to describe various SAML system entities has been rationalized and
1415 enhanced to incorporate the Liberty Alliance notion of “identity providers” as opposed to
1416 “authentication authorities” and similar.

1417 **TBS: validity period semantics and syntax extended, removal of QNames in content, etc.**

7 References

1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454

- [SAMLAuthnCxt]** J. Kemp et al., *Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, January 2005. Document ID sstc-saml-authn-context-2.0-os. See <http://www.oasis-open.org/committees/security/>.
- [SAMLBind]** S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC. Document ID sstc-saml-bindings-2.0-os. See <http://www.oasis-open.org/committees/security/>.
- [SAMLConform]** P. Mishra et al., *Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC. Document ID sstc-saml-conformance-2.0-os. <http://www.oasis-open.org/committees/security/>.
- [SAMLCore]** S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC. Document ID sstc-saml-core-2.0-os. 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. Document ID sstc-saml-glossary-2.0-os. 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. Document ID sstc-saml-metadata-2.0-os. 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. Document ID sstc-saml-sec-consider-2.0-os. See <http://www.oasis-open.org/committees/security/>.
- [SAMLWeb]** OASIS Security Services Technical Committee web site, <http://www.oasis-open.org/committees/security/>.
- [WSS]** A. Nadalin, ed., *OASIS Web Services Security: SOAP Message Security 1.0*. Available on the OASIS WS-Security web page at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- [WSSSAML]** R. Monzillo, ed. OASIS Web Services Security: SAML 2.0 Token Profile 1.0. wss-saml-2.0-token-profile-1.0. Available on the OASIS XACML TC web page at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- [XACML]** T. Moses, ed., *OASIS eXtensible Access Control Markup Language (XACML) Version 2.0*. Available on the OASIS XACML TC web page at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

1455 **A. Acknowledgments**

1456 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
1457 Committee, whose voting members at the time of publication were:

- 1458 • TBD

B. Revision History

Rev	Date	By Whom	What
00	Nov 6, 2003	John Hughes	Storyboard version
01	Jul 22, 2004	John Hughes	First draft
02	27 Sept 2004	John Hughes	Second Draft.General updates, limited distribution
03	Feb 20, 2005	John Hughes	DCE/Kerberos use section removed. Use of SAML in other frameworks added. SAML V2.0 XML examples included. Updated Web SSO examples to remove use of ITS
04	10 Apr 2005	Eve Maler	Edits based on comments made by myself and Scott Cantor. Fleshed out the list of 1.1->2.0 differences, but it's not complete yet. More work to come.
05	May 10, 2005	Prateek Mishra	Updated Section 2 and 3.4, Section 4.3 remains incomplete
06	Jun 3, 2005	John Hughes	Added Section 4.3 plus a few minor corrections
07	Jul 13, 2005	John Hughes	Addressed comments from SSTC, primarily re-vamping section 4.3
08	12 Sep 2005	Eve Maler	Incorporated many, though not all, of the comments that arose from the special Tech Overview review meeting (see notes sent to the SSTC list on 24 August 2005)

1461 C. Notices

1462 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
1463 might be claimed to pertain to the implementation or use of the technology described in this document or
1464 the extent to which any license under such rights might or might not be available; neither does it
1465 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with
1466 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights
1467 made available for publication and any assurances of licenses to be made available, or the result of an
1468 attempt made to obtain a general license or permission for the use of such proprietary rights by
1469 implementors or users of this specification, can be obtained from the OASIS Executive Director.

1470 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,
1471 or other proprietary rights which may cover technology that may be required to implement this
1472 specification. Please address the information to the OASIS Executive Director.

1473 **Copyright © OASIS Open 2005. All Rights Reserved.**

1474 This document and translations of it may be copied and furnished to others, and derivative works that
1475 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published
1476 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright
1477 notice and this paragraph are included on all such copies and derivative works. However, this document
1478 itself does not be modified in any way, such as by removing the copyright notice or references to OASIS,
1479 except as needed for the purpose of developing OASIS specifications, in which case the procedures for
1480 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required
1481 to translate it into languages other than English.

1482 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
1483 or assigns.

1484 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1485 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1486 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS
1487 OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR
1488 PURPOSE.