



1

Liberty Architecture Overview

2

Version 1.1

3

15 January 2003

4

5 **Document Description:** liberty-architecture-overview-v1.1

6

6 **Notice**

7 Copyright © 2002, 2003 ActivCard; American Express Travel Related Services; America Online,
8 Inc.; Bank of America; Bell Canada; Catavault; Cingular Wireless; Cisco Systems, Inc.; Citigroup;
9 Communicator, Inc.; Consignia; Cyberun Corporation; Deloitte & Touche LLP; Earthlink, Inc.;
10 Electronic Data Systems, Inc.; Entrust, Inc.; Ericsson; Fidelity Investments; France Telecom;
11 Gemplus; General Motors; Hewlett-Packard Company; i2 Technologies, Inc.; Intuit Inc.;
12 MasterCard International; NEC Corporation; Netegrity; NeuStar; Nextel Communications; Nippon
13 Telegraph and Telephone Company; Nokia Corporation; Novell, Inc.; NTT DoCoMo, Inc.;
14 OneName Corporation; Openwave Systems Inc.; PricewaterhouseCoopers LLP; Register.com;
15 RSA Security Inc; Sabre Holdings Corporation; SAP AG; SchlumbergerSema; SK Telecom; Sony
16 Corporation; Sun Microsystems, Inc.; United Airlines; VeriSign, Inc.; Visa International;
17 Vodafone Group Plc; Wave Systems;. All rights reserved.

18 This specification document has been prepared by Sponsors of the Liberty Alliance. Permission is
19 hereby granted to use the document solely for the purpose of implementing the Specification. No
20 rights are granted to prepare derivative works of this Specification. Entities seeking permission to
21 reproduce portions of this document for other uses must contact the Liberty Alliance to determine
22 whether an appropriate license for such use is available.

23 Implementation of certain elements of this Specification may require licenses under third
24 party intellectual property rights, including without limitation, patent rights. The Sponsors
25 of and any other contributors to the Specification are not, and shall not be held responsible
26 in any manner, for identifying or failing to identify any or all such third party intellectual
27 prop-erty rights. **This Specification is provided "AS IS", and no participant in the
28 Liberty Alliance makes any warranty of any kind, express or implied, including any
29 implied warranties of merchantability, non-infringement of third party intellectual
30 property rights, and fitness for a particular purpose.** Implementors of this Specification
31 are advised to review the Liberty Alliance Project's website
32 (<http://www.projectliberty.org/>) for information concerning any Necessary Claims
33 Disclosure Notices that have been received by the Liberty Alliance Management Board.

34 Liberty Alliance Project
35 Licensing Administrator
36 c/o IEEE-ISTO
37 445 Hoes Lane
38 Piscataway, NJ 08855-1331, USA
39 info@projectliberty.org

40

40 **Editor**

41 Jeff Hodges, Sun Microsystems, Inc.
42 Tom Wason, IEEE - ISTO

43 **Contributors**

44
45 The following Liberty Alliance Project Sponsor companies contributed to the development of this
46 specification:
47

ActivCard	Netegrity
American Express Travel Related Services	NeuStar
America Online, Inc.	Nextel Communications
Bank of America	Nippon Telegraph and Telephone Company
Bell Canada	Nokia Corporation
Catavault	Novell, Inc.
Cingular Wireless	NTT DoCoMo, Inc.
Cisco Systems, Inc.	OneName Corporation
Citigroup	Openwave Systems Inc.
Communicator, Inc.	PricewaterhouseCoopers LLP
Consignia	Register.com
Cyberun Corporation	RSA Security Inc
Deloitte & Touche LLP	Sabre Holdings Corporation
Earthlink, Inc.	SAP AG
Electronic Data Systems, Inc.	SchlumbergerSema
Entrust, Inc.	SK Telecom
Ericsson	Sony Corporation
Fidelity Investments	Sun Microsystems, Inc.
France Telecom	United Airlines
Gemplus	VeriSign, Inc.
General Motors	Visa International
Hewlett-Packard Company	Vodafone Group Plc
i2 Technologies, Inc.	Wave Systems
Intuit Inc.	
MasterCard International	
NEC Corporation	

48

49

49 Revision History

50

Version #	Date	Editor	Scope of changes
1.0	14-Mar-02	Jeff Hodges	Initial Draft Based on Liberty V1.0
1.1	05-Nov-02	Jeff Hodges	<p>CR 1107 login via embedded form only "may" reveal users' credentials to SP</p> <p>CR 1103 Argument in line 949 inversed. It says available space in ULR larger than HTML form.</p> <p>CR 1100 Mention "provide non-repudiation"</p> <p>CR 1102 Is Figure 17 supported in Phase1?</p> <p>CR 1101 Added description of this document. Section 1.1.</p> <p>CR 1104 Figure 14 represents double linking instead of simple one.</p> <p>CR 1099 User consent obtained prior to authentication</p> <p>CR 1177 establishing trust relationships in IDP2IDP federation is unspecified</p>
1.1 - 04	15-Nov-02	Tom Wason	<p>CR1217: Added note on authentication state information for principals, Section 5.4.2.</p> <p>CR1218: Added common cookie note to Section 5.5.</p> <p>CR1222: Added note on federation termination with a local session, Section 5.4.1.2</p> <p>CR1226: User handles note #2 change in Section 5.4.1.</p>
1.1 - 05	25-Nov-02	Tom Wason	<p>CR1238: Inserted title "Identity Provider Session State Maintenance" in <u>POLICY/SECURITY NOTE</u> in Section 5.4.2.</p> <p>CR1247: Lowered case of RECOMMENDED, <u>POLICY/SECURITY NOTE</u> in Section 5.4.1.2.</p>
1.1 - 06	20-Dec-02	Tom Wason	<p>CR1179: Re,oved extraneous "[1107]" from Section 5.7.1.3.</p> <p>CR1270: Normalized and formatted reference, Section 6.</p>
1.1 Final	15-Jan-2003	John Kemp	Removed references to

51

52

53

53	Table of Contents	
54	1 Introduction.....	6
55	1.1 About This Document.....	6
56	1.2 What is the Liberty Alliance?.....	6
57	1.2.1 The Liberty Vision	6
58	1.2.2 The Liberty Mission	7
59	1.3 What is Network Identity?	7
60	1.3.1 The Liberty Objectives	7
61	2 Liberty Version 1.0 User Experience Examples.....	9
62	2.1 Example of Identity Federation User Experience	9
63	2.2 Example of Single Sign-on User Experience.....	13
64	3 Liberty Engineering Requirements Summary	15
65	3.1 General Requirements.....	15
66	3.1.1 Client Device/User Agent Interoperability	15
67	3.1.2 Openness Requirements	15
68	3.2 Functional Requirements	15
69	3.2.1 Identity Federation	15
70	3.2.2 Authentication.....	16
71	3.2.3 Pseudonyms	16
72	3.2.4 Global Logout	16
73	4 Liberty Security Framework.....	16
74	5 Liberty Architecture	18
75	5.1 Web Redirection Architectural Component.....	19
76	5.1.1 HTTP-Redirect-Based Redirection	20
77	5.1.2 Form-POST-Based Redirection	21
78	5.1.3 Cookies	21
79	5.1.4 Web Redirection Summary.....	22
80	5.2 Web Services Architectural Component.....	22
81	5.3 Metadata and Schemas Architectural Component	23
82	5.4 Single Sign-On and Identity Federation	23
83	5.4.1 Identity Federation	23
84	5.4.2 Single Sign-on.....	29
85	5.4.3 Profiles of the Single Sign-On and Federation Protocol	31
86	5.5 Identity Provider Introduction.....	34
87	5.6 Single Logout	37
88	5.6.1 Single Logout Profiles.....	38
89	5.7 Example User Experience Scenarios	38
90	5.7.1 Scenario: Not Logged in Anywhere, No Common Domain Cookie.....	39
91	5.7.2 Scenario: Not Logged in Anywhere, Has a Common Domain Cookie.....	43
92	5.7.3 Scenario: Logged in, Has a Common Domain Cookie	43
93	6 References	43
94		
95		

95 1 Introduction

96 The Internet is now a prime vehicle for business, community, and personal interactions. The notion
97 of *identity* is the crucial component of this vehicle. Today, one's identity on the Internet is
98 fragmented across various identity providers — employers, Internal portals, various communities,
99 and business services. This fragmentation yields isolated, high-friction, one-to-one customer-to-
100 business relationships and experiences.

101
102 *Federated network identity* is the key to reducing this friction and realizing new business
103 taxonomies and opportunities, coupled with new economies of scale. In this new world of
104 federated commerce, a user's online identity, personal profile, personalized online configurations,
105 buying habits and history, and shopping preferences will be administered by the user and securely
106 shared with the organizations of the user's choosing. A federated network identity model will
107 ensure that critical private information is used by appropriate parties.

108
109 The path to realizing a rich, fertile federated identity infrastructure can be taken in phases. The
110 natural first phase is the establishment of a standardized, multivendor, Web-based single sign-on
111 with simple federated identities based on today's commonly deployed technologies. This
112 document presents an overview of the *Liberty Version 1.0 architecture*, which offers a viable
113 approach for implementing such a single sign-on with federated identities. This overview first
114 summarizes federated network identity, describes two key Liberty Version 1.0 user experience
115 scenarios, summarizes the Liberty engineering requirements and security framework, and then
116 provides a discussion of the Liberty Version 1.0 architecture.

117 1.1 About This Document

118 This document is *non-normative*. However, it provides implementers and deployers guidance in
119 the form of policy/security and technical notes. Further details of the Liberty architecture are given
120 in several normative technical documents associated with this overview, specifically
121 [LibertyAuthnContext], [LibertyBindProf], [LibertyArchImpl], and [LibertyProtSchema]. Note:
122 The more global term *Principal* is used for *user* in Liberty's technical documents. Definitions for
123 Liberty-specific terms can be found in the [LibertyGloss]. Also, many abbreviations are used in
124 this document without immediate definition because the authors believe these abbreviations are
125 widely known, for example, HTTP and SSL. However, the definitions of these abbreviations can
126 also be found in [LibertyGloss]. Note: Phrases and numbers in brackets [] refer to other
127 documents; details of these references can be found in Section 6 (at the end of this document). As
128 this document is non-normative it does not use terminology "MUST", "MAY", "SHOULD" in a
129 manner consistent with RFC-2119.

130 1.2 What is the Liberty Alliance?

131 The Liberty Alliance Project represents a broad spectrum of industries united to drive a new level
132 of trust, commerce, and communications on the Internet.

133 1.2.1 The Liberty Vision

134 The members of the Liberty Alliance envision a networked world across which individuals and
135 businesses can engage in virtually any transaction without compromising the privacy and security
136 of vital identity information.

137 **1.2.2 The Liberty Mission**

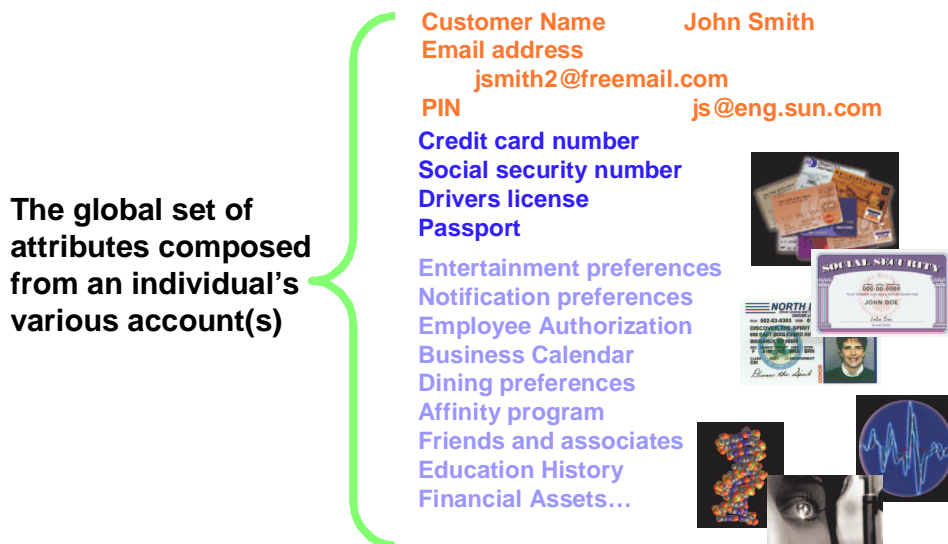
138 To accomplish its vision, the Liberty Alliance will establish open technical specifications that
139 support a broad range of network identity-based interactions and provide businesses with
140

- 141 • A basis for new revenue opportunities that economically leverage their relationships with
142 consumers and business partners and
- 143 • A framework within which the businesses can provide consumers with choice,
144 convenience, and control when using any device connected to the Internet.
145

146 **1.3 What is Network Identity?**

147 When users interact with services on the Internet, they often tailor the services in some way for
148 their personal use. For example, a user may establish an account with a username and password
149 and/or set some preferences for what information the user wants displayed and how the user wants
150 it displayed. The network identity of each user is the overall global set of these attributes
151 constituting the various accounts (see Figure 1).

What is Network Identity?



152 **Figure 1: A network identity is the global set of attributes composed from a user's account(s).**
153

154 Today, users' accounts are scattered across isolated Internet sites. Thus the notion that a user could
155 have a cohesive, tangible network identity is not realized.

156 **1.3.1 The Liberty Objectives**

157 The key objectives of the Liberty Alliance are to

- 158
- 159 • Enable consumers to protect the privacy and security of their network identity information
- 160 • Enable businesses to maintain and manage their customer relationships without third-party
161 participation

- Provide an open single sign-on standard that includes decentralized authentication and authorization from multiple providers
- Create a network identity infrastructure that supports all current and emerging network access devices

These capabilities can be achieved when, first, businesses affiliate together into *circles of trust* based on Liberty-enabled technology and on operational agreements that define *trust relationships* between the businesses and, second, users federate the otherwise isolated accounts they have with these businesses (known as their *local identities*). In other words, a circle of trust is a federation of service providers and identity providers that have business relationships based on Liberty architecture and operational agreements and with whom users can transact business in a secure and apparently seamless environment. See Figure 2. Note: Operational agreement definitions are out of the scope of the Liberty Version 1.0 specifications.

Federated Network Identity

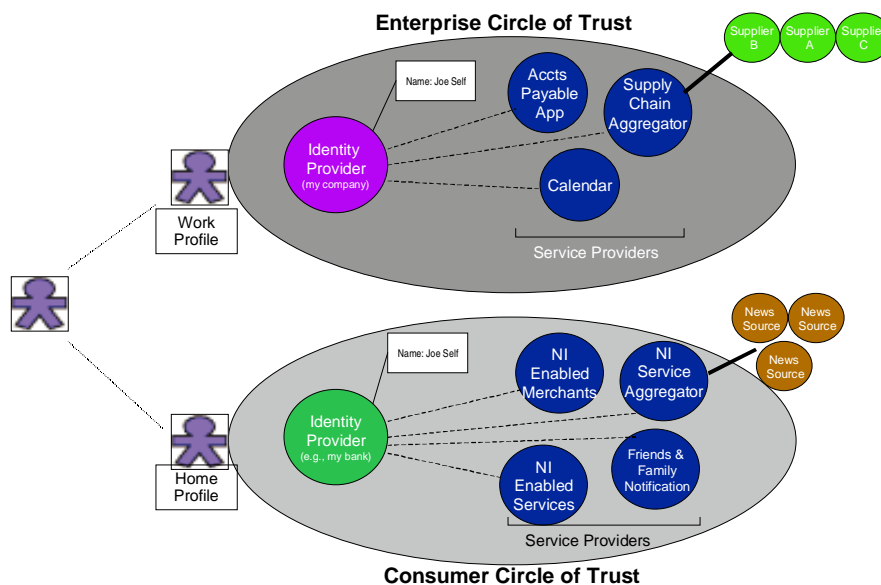


Figure 2: Federated network identity and circles of trust

From a Liberty perspective, the salient actors in Figure 2 are the user, service providers, and identity providers.

Service providers are organizations offering Web-based services to users. This broad category includes practically any organization on the Web today, for example, Internet portals, retailers, transportation providers, financial institutions, entertainment companies, not-for-profit organizations, governmental agencies, etc.

Identity providers are service providers offering business incentives so that other service providers affiliate with them. Establishing such relationships creates the circles of trust shown in Figure 2. For example, in the enterprise circle of trust, the identity provider is a company leveraging employee network identities across the enterprise. Another example is the consumer circle of trust, where the user's bank has established business relationships with various other service providers

191 allowing the user to wield his/her bank-based network identity with them. Note: A single
192 organization may be both an identity provider and a service provider, either generally or for a
193 given interaction.

194
195 These scenarios are enabled by service providers and identity providers deploying Liberty-enabled
196 products in their infrastructure, but do not require users to use anything other than today's common
197 Web browser.

198 **2 Liberty Version 1.0 User Experience Examples**

199 This section provides two simple, plausible examples of the Liberty Version 1.0 user experience,
200 from the perspective of the user, to set the overall context for delving into technical details of the
201 Liberty architecture in the Section 5. As such, actual technical details are hidden or simplified.

202
203 Note: the user experience examples presented in this section are non-normative and are presented
204 for illustrative purposes only.

205
206 These user experience examples are based upon the following set of actors:

- 207
- 208 • Joe Self A user of Web-based online services.
- 209 • Airline.inc An airline maintaining an affinity group of partners. Airline.inc is an
210 identity provider.
- 211 • CarRental.inc A car rental company that is a member of the airline's affinity group.
212 CarRental.inc is a service provider.
- 213

214 The Liberty Version 1.0 user experience has two main facets:

- 215
- 216 • Identity federation
- 217 • Single sign-on
- 218

219 Identity federation is based upon linking users' otherwise distinct service provider and identity
220 provider accounts. This account linkage, or *identity federation*, in turn underlies and enables the
221 other facets of the Liberty Version 1.0 user experience.

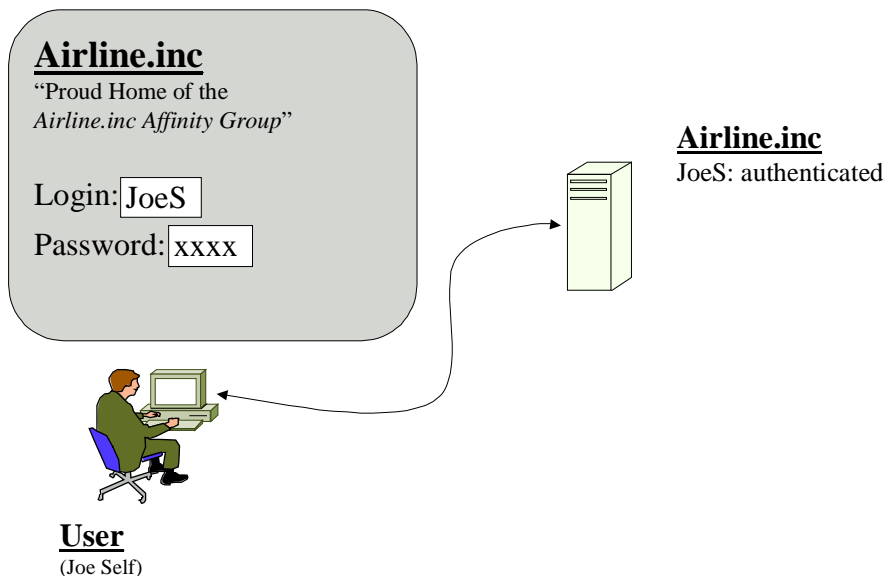
222
223 OVERALL POLICY/SECURITY NOTE: Identity federation must be predicated upon prior agreement
224 between the identity and service providers. It should be additionally predicated upon providing notice to the
225 user, obtaining the user's consent, and recording both the notice and consent in an auditable fashion.
226 Providing an auditable record of notice and consent will enable both users and providers to confirm that
227 notice and consent were provided and to document that the consent is bound to a particular interaction. Such
228 documentation will increase consumer trust in online services. Implementors and deployers of Liberty-
229 enabled technology should ensure that notice and user consent are auditably recorded in Liberty-enabled
230 interactions with users, as appropriate.

231
232 Single sign-on enables users to sign on once with a member of a federated group of identity and
233 service providers (or, from a provider's point of view, with a member of a circle of trust) and
234 subsequently use various Websites among the group without signing on again.

235 **2.1 Example of Identity Federation User Experience**

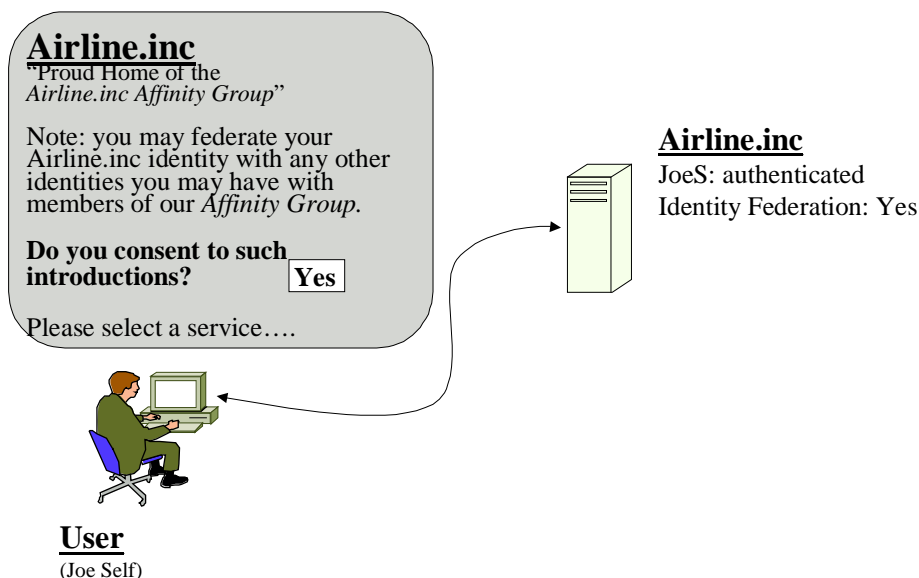
236 The identity federation facet of the Liberty Version 1.0 user experience typically begins when Joe
237 Self logs in to Airline.inc's Website, a Liberty-enabled identity provider, as illustrated in Figure 3.

238
239 Note: Even though Joe Self is unaware of it, behind the scenes the identity provider is using Joe
240 Self’s credentials—his username and password in this case—to *authenticate* his identity. If
241 successful, Joe Self is considered *authenticated*.



242
243 **Figure 3: User logs in at a Liberty-enabled Website.**

244
245 Airline.inc. (as would any other identity provider that has created a circle of trust among its
246 affinity group) will notify its eligible users of the possibility of federating their local identities
247 among the members of the affinity group and will solicit permission to facilitate such
248 introductions. See
249 Figure 4.
250



251
252 **Figure 4: User is notified of eligibility for identity federation and elects to allow introductions.**

253

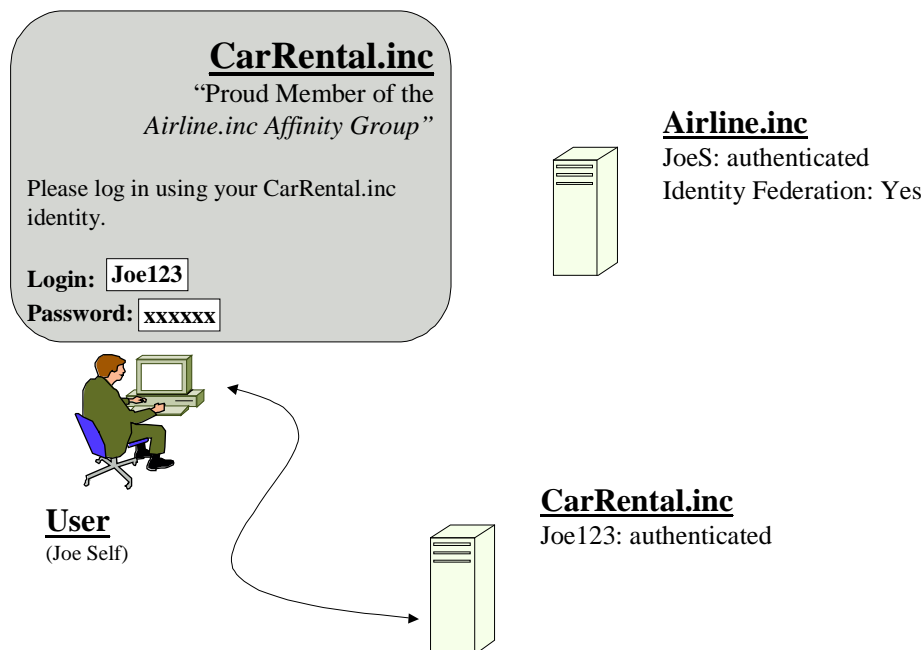
254 POLICY/SECURITY NOTE: Figure 4 illustrates the user’s consenting to introductions. An introduction is
255 the means by which a service provider may discover which identity providers in the circle of trust have
256 authenticated the user. Note: In Figure 4 the user is not consenting to federating his identity with any service
257 providers. Soliciting consent to identity federation is a separate step, as illustrated in Figure 5.
258

259 The act of introduction may be implemented via the Identity Provider Introduction Profile (as detailed in
260 [LibertyBindProf]), or it may be implemented via other unspecified means, such as when the user agent is a
261 Liberty-enabled client or proxy.
262

263 At some later point in time, typically minutes to a few hours, Joe Self may visit the Website of an
264 affinity group member, for example, CarRental, Inc., whose site is CarRental.inc. Indeed, Joe Self
265 may have followed an explicit link from the original Airline.inc Website to the CarRental.inc
266 Website. In either case, CarRental.inc (a Liberty-enabled service provider) is able to discern that
267 Joe Self recently interacted with the Airline.inc Website, because Joe Self elected to allow
268 introductions.
269

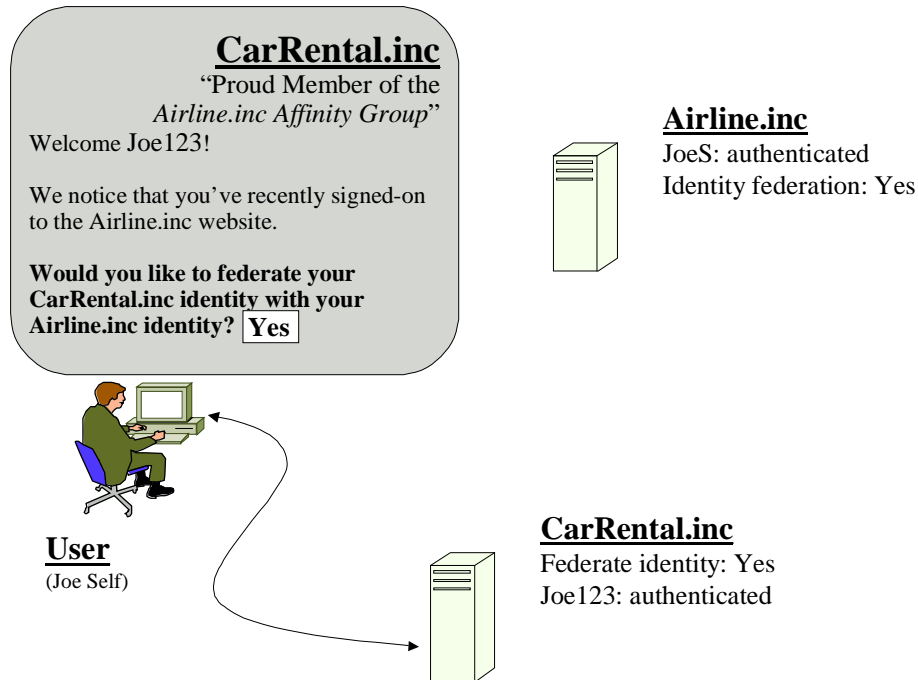
270 TECHNICAL NOTE: The actual means used to perform the introduction is an implementation and
271 deployment decision. One possible means, the Identity Provider Introduction profile, is specified in
272 [LibertyBindProf]. Note that the user may or may not need to log in in order to facilitate introduction – this
273 depends on the specific introduction technique used.
274

275 If the service provider maintains local accounts, as in our example, it will typically, upon Joe
276 Self’s arrival, prompt Joe to log in, which he does using his local CarRental.inc identity.and thus.
277 See Figure 5.
278



279
280 **Figure 5: User signs-on using his local service provider identity.**

281
282 Thereafter, Joe Self is presented with the opportunity to federate his local identities between
283 CarRental.inc and Airline.inc. See Figure 6.
284

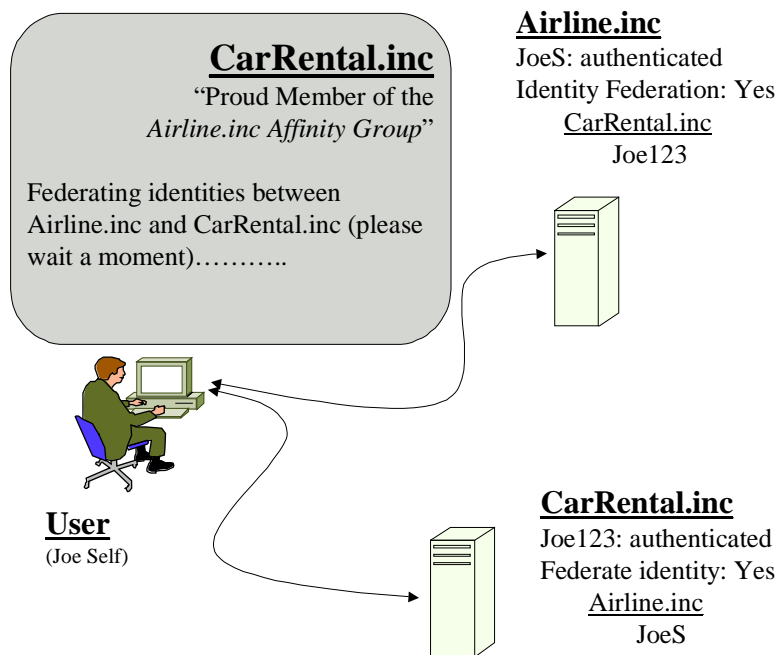


285
 286 **Figure 6: User is prompted to federate his local identities and selects "yes."**

287
 288 POLICY/SECURITY NOTE: Whether the service provider asks for consent to federate the user's local
 289 identity before or after locally authenticating the user is a matter of local deployment policy.

290
 291 As a part of logging in to the CarRental.inc Website, Joe Self's local CarRental.inc identity is
 292 federated with his local Airline.inc identity. See Figure 7.

293



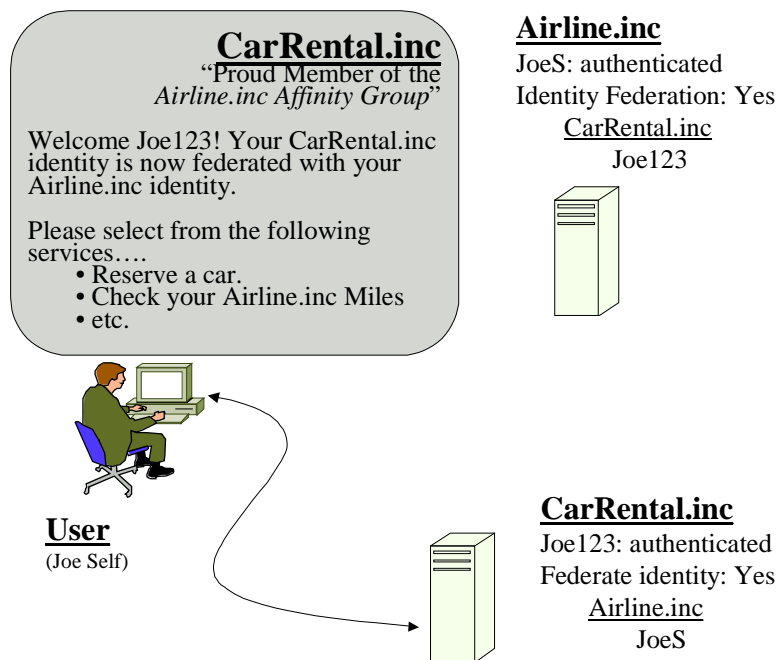
294
 295 **Figure 7: The Websites federate the user's local identities.**

296
297 Upon completion of the login and identity federation activity, Joe User is logged in to the
298 CarRental.inc Website, and CarRental.inc delivers services to him as usual. In addition, the
299 Website may now offer new selections because Joe Self's local service provider (CarRental.inc)
300 identity has been federated with his local identity provider (Airline.inc) identity. See Figure 8.

301
302 TECHNICAL NOTE: Some figures illustrating the user experience, for example, Figure 7, show simplified,
303 user-perspective notions of how identity federation is effected. In actuality, cleartext identifiers, for example,
304 "JoeS" and "Joe123" WILL NOT be exchanged between the identity provider and service provider. Rather,
305 opaque user handles will be exchanged. See 5.4.1 for details.

306
307 Additionally, if errors are encountered in the process of authenticating and/or federating, the service provider
308 will need to present appropriate indications to the user.

309



310

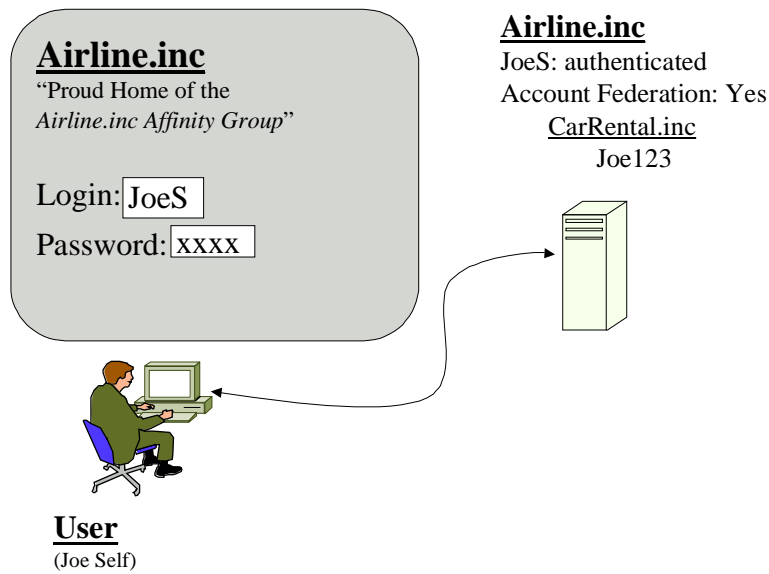
311 **Figure 8: The service provider delivers services to user as usual.**

312
313 POLICY/SECURITY NOTE: Business prerequisites must be met to offer identity federation. Two
314 prerequisites are notifying the user of the capability to federate and soliciting consent to facilitate
315 introductions. Another is creating agreements between the affinity group members to establish their policies
316 for recognizing identities and honoring reciprocal authentication.

317 2.2 Example of Single Sign-on User Experience

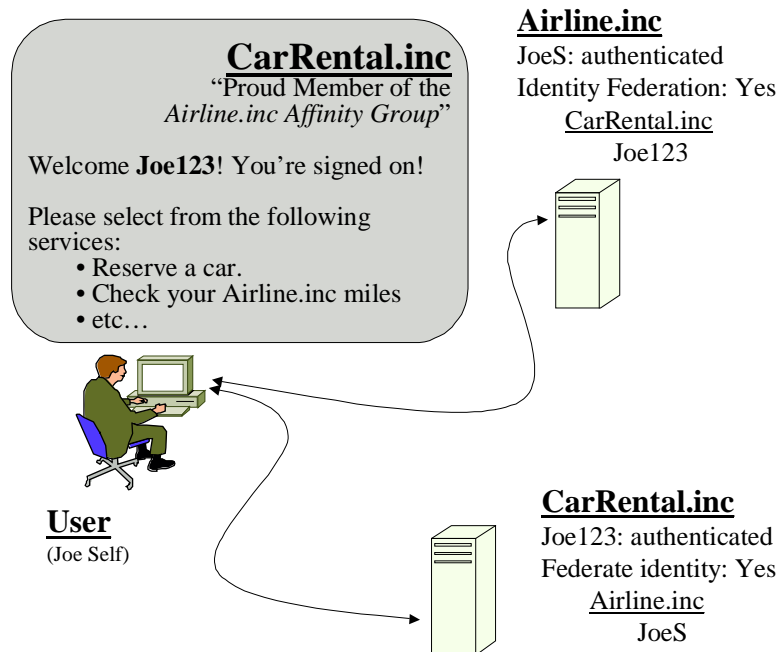
318 Single sign-on builds upon identity federation and has a simple user experience. Joe Self logs in to
319 the Airline.inc Website and later visits the CarRental.inc Website with which he has established
320 identity federation. Joe Self's authentication state with the Airline.inc Website is reciprocally
321 honored by the CarRental.inc Website, and Joe Self is transparently logged in to the latter site. See
322 Figure 9 and Figure 10.

323



324
325
326
327

Figure 9: User logs in to identity provider’s Website using local identity.



328
329
330
331
332
333
334
335
336
337

Figure 10: User proceeds to service provider’s Website, and his authentication state is reciprocally honored by the service provider’s Website.

A perceptive Joe Self will notice that his name in the CarRental.inc session is based upon his local CarRental.inc identity, rather than the local Airline.inc identity with which it has been federated.

TECHNICAL NOTE: Because users’ actual account identifiers are not exchanged during federation, a service provider will not be able to display a user’s identity provider identifier.

338 Also, many types of service provider Websites may not use a personally identifiable identifier in response to
339 the user. For example, advertising-driven sites where users may specify display preferences, for example, a
340 sporting events schedule site. The site may simply transparently refer to the user as “you,” for example, “Set
341 your display preferences here...,” “Here is the list of upcoming events you’re interested in...” etc.

342
343 SECURITY/POLICY NOTE: Even though the user may be validly authenticated via the single sign-on
344 mechanism, the user’s use of the service provider’s Website is still subject to local policy. For example, the
345 site may have time-of-day usage restrictions, the site may be undergoing maintenance, the user’s relationship
346 with the service provider may be in a particular state (for example, highly valued customer – show the user
347 the bonus pages; troublesome customer – remind the user of unpaid bills and restrict some access).

348 **3 Liberty Engineering Requirements Summary**

349 This section summarizes the Liberty general and functional engineering requirements.

350 **3.1 General Requirements**

351 The Liberty-enabled systems should follow the set of general principals outlined in 3.1.1 and 3.1.2.
352 These principles cut across categories of functionality.

353 **3.1.1 Client Device/User Agent Interoperability**

354 Liberty Version 1.0 clients encompass a broad range of presently deployed Web browsers, other
355 presently deployed Web-enabled client access devices, and newly designed Web-enabled browsers
356 or clients with specific Liberty-enabled features.

357
358 The Liberty Version 1.0 architecture and protocol specifications must support a basic level of
359 functionality across the range of Liberty Version 1.0 clients.

360 **3.1.2 Openness Requirements**

361 The Liberty architecture and protocol specifications must provide the widest possible support for

- 362
- 363 • Operating systems
 - 364 • Programming languages
 - 365 • Network infrastructures
- 366

367 and must not impede multivendor interoperability between Liberty clients and services, including
368 interoperability across circle of trust boundaries.

369 **3.2 Functional Requirements**

370 The Liberty architecture and protocols must be specified so that Liberty-enabled implementations
371 are capable of performing the following activities:

- 372
- 373 • Identity federation
 - 374 • Authentication
 - 375 • Use of pseudonyms
 - 376 • Global logout

377 **3.2.1 Identity Federation**

378 Requirements of identity federation stipulate that

379

- 380 • Providers give the user notice upon identity federation and defederation.
- 381 • Service providers and identity providers notify each other about identity defederation.
- 382 • Each identity provider notifies appropriate service providers of user account terminations at
383 the identity provider.
- 384 • Each service provider and/or identity provider gives each of its users a list of the user's
385 federated identities at the identity provider or service provider.

386 **3.2.2 Authentication**

387 Authentication requirements include

388

- 389 • Supporting any method of navigation between identity providers and service providers on
390 the part of the user, that is, how the user navigates from A to B (including click-through,
391 favorites or bookmarks, URL address bar, etc.) must be supported.
- 392 • Giving the identity provider's authenticated identity to the user before the user gives
393 credentials or any other personally identifiable information to the identity provider.
- 394 • Providing for the confidentiality, integrity, and authenticity of information exchanged
395 between identity providers, service providers, and user agents, as well as mutually
396 authenticating the identities of the identity providers and service providers, during the
397 authentication and single sign-on processes.
- 398 • Supporting a range of authentication methods, extensibly identifying authentication
399 methods, providing for coalescing authentication methods into authentication classes, and
400 citing and exchanging authentication classes. Protocols for exchanging this information are
401 out of the scope of the Liberty Version 1.0 specifications, however.
- 402 • Exchanging the following minimum set of authentication information with regard to a user:
403 authentication status, instant, method, and pseudonym.
- 404 • Giving service providers the capability of causing the identity provider to reauthenticate the
405 user using the same or a different authentication class. Programmatic exchange of the set of
406 authentication classes for which a user is registered at an identity provider is out of the
407 scope of the Liberty Version 1.0 specifications, however.

408 **3.2.3 Pseudonyms**

409 Liberty-enabled implementations must be able to support the use of pseudonyms that are unique on
410 a per-identity-federation basis across all identity providers and service providers.

411 **3.2.4 Global Logout**

412 Liberty-enabled implementations must be able to support the notification of service providers
413 when a user logs out at identity provider.

414 **4 Liberty Security Framework**

415 Table 1 generally summarizes the security mechanisms incorporated in the Liberty specifications,
416 and thus in Liberty-enabled implementations, across two axes: channel security and message
417 security. It also generally summarizes the security-oriented processing requirements placed on
418 Liberty implementations. Note: This section is non-normative, please refer to [LibertyProtSchema]
419 and [LibertyBindProf] for detailed normative statements regarding security mechanisms.

420

421

Table 1: Liberty security mechanisms

Security Mechanism	Channel Security	Message Security (for Requests, Assertions)
Confidentiality	Required	Optional
Per-message data integrity	Required	Required
Transaction integrity	—	Required
Peer-entity authentication	Identity provider — Required Service provider — Optional	—
Data origin authentication	—	Required
Nonrepudiation	—	Required

422

423 Channel security addresses how communication between identity providers, service providers, and
 424 user agents is protected. Liberty implementations must use TLS1.0 or SSL3.0 for channel security,
 425 although other communication security protocols may also be employed, for example, IPsec, if
 426 their security characteristics are equivalent to TLS or SSL. Note: TLS, SSL, and equivalent
 427 protocols provide confidentiality and integrity protection to communications between parties as
 428 well as authentication.

429

430 Critical points of channel security include the following:

431

- 432 • In terms of authentication, service providers are required to authenticate identity providers
 433 using identity provider server-side certificates. Identity providers have the option to require
 434 authentication of service providers using service provider client-side certificates.
- 435 • Additionally, each service provider is required to be configured with a list of authorized
 436 identity providers, and each identity provider is required to be configured with a list of
 437 authorized service providers. Thus any service provider-identity provider pair must be
 438 mutually authorized before they will engage in Liberty interactions. Such authorization is
 439 in addition to authentication. (Note: The format of this configuration is a local matter and
 440 could, for example, be represented as lists of names or as sets of X.509 certificates of other
 441 circle of trust members).
- 442 • The authenticated identity of an identity provider must be presented to a user before the
 443 user presents personal authentication data to that identity provider.

444

445 Message security addresses security mechanisms applied to the discrete Liberty protocol messages
 446 passed between identity providers, service providers, and user agents. These messages are
 447 exchanged across the communication channels whose security characteristics were just discussed.
 448

449

450 Critical points of message security include the following:

451

- 452 • Liberty protocol messages and some of their components are generally required to be
 453 digitally signed and verified. Signing and verifying messages provide data integrity,
 454 data origin authentication, and a basis for nonrepudiation. Therefore, identity providers
 455 and service providers are required to use key pairs that are distinct from the key pairs
 456 applied for TLS and SSL channel protection and that are suitable for long-term
 457 signatures.

458

459
460 SECURITY/POLICY NOTE: Specifically, the <AuthnRequest> message of the Single
461 Sign-On and Federation Protocol defined in [LibertyProtSchema] may be signed or not signed
462 as specified by agreement between the identity provider and service provider and indicated by
463 the <AuthnRequestsSigned> element of the provider metadata. Not signing this message
464 may be considered reasonable in some deployment contexts, for example, an enterprise
465 network, where access to the network and its systems is moderated by some means out of the
466 scope of the Liberty architecture.

- 467
468 • In transactions between service providers and identity providers, requests are required
469 to be protected against replay, and received responses are required to be checked for
470 correct correspondence with issued requests. Time-based assurance of freshness may be
471 employed. These techniques provide transaction integrity.

472
473 To become circle of trust members, providers are required to establish bilateral agreements on
474 selecting certificate authorities, obtaining X.509 credentials, establishing and managing trusted
475 public keys, and managing life cycles of corresponding credentials.

476 SECURITY/POLICY NOTE: Many of the security mechanisms mentioned above, for example, SSL and
477 TLS, have dependencies upon, or interact with, other network services and/or facilities such as the DNS, time
478 services, firewalls, etc. These latter services and/or facilities have their own security considerations upon
479 which Liberty-enabled systems are thus dependent.
480

481 **5 Liberty Architecture**

482 The overall Liberty architecture is composed of three orthogonal architectural components (see
483 Figure 11):

- 484
485 • Web redirection
486 • Web services
487 • Metadata and schemas

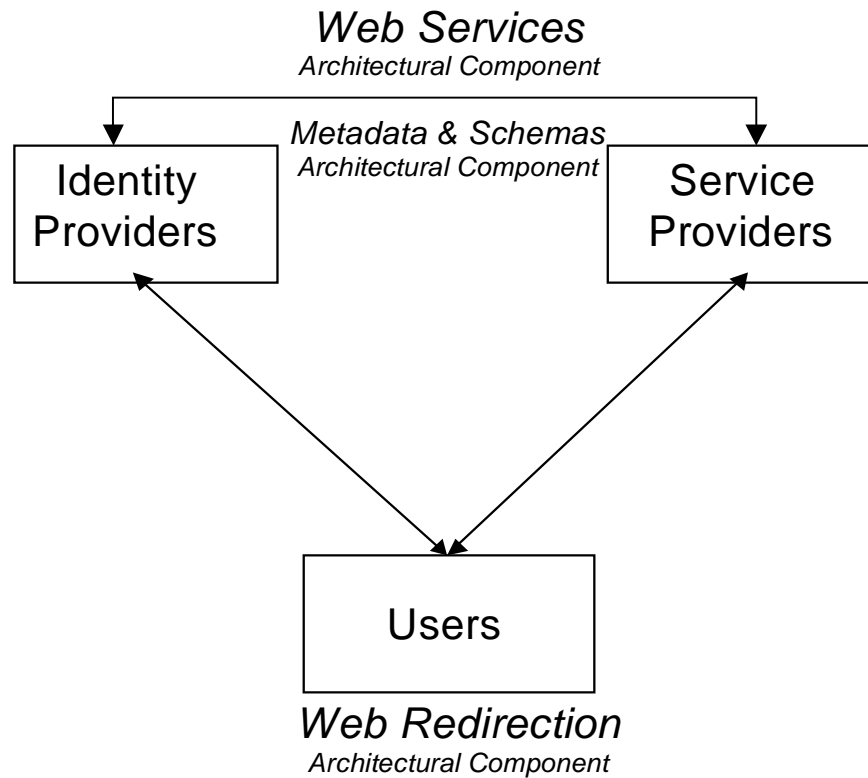


Figure 11: Overall Liberty architecture

The role of each architectural component is summarized in Table 2:

Table 2: Components of Liberty architecture

Web redirection	Action that enables Liberty-enabled entities to provide services via today's user-agent-installed base.
Web services	Protocol profiles that enable Liberty-enabled entities to directly communicate.
Metadata and schemas	A common set of metadata and formats used by Liberty-enabled sites to communicate various provider-specific and other information.

Sections 5.1 through 5.3 describe each architectural component. Sections 5.4 through 5.6 then relate the architectural components to the concrete protocols and profiles detailed in [LibertyProtSchema] and [LibertyBindProf], and 5.7 provides illustrations of user experience.

5.1 Web Redirection Architectural Component

The Web redirection architectural component is composed of two generic variants: HTTP-redirect-based redirection and form-POST-based redirection. Both variants create a communication channel between identity providers and service providers that is rooted in the user agent. See Figure 12.

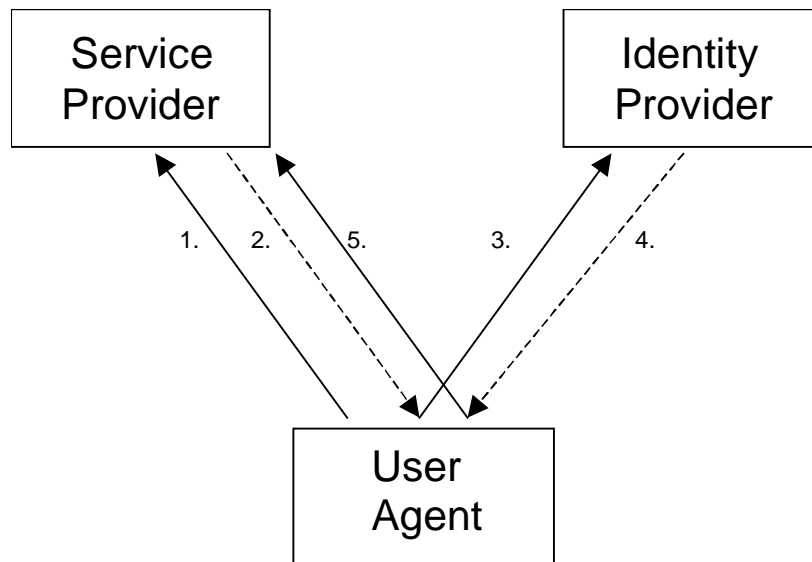


Figure 12: Web redirection between a service provider and an identity provider via the user agent

5.1.1 HTTP-Redirect-Based Redirection

HTTP-redirect-based redirection uses the HTTP redirection class of response (that is, *redirects*) of the HTTP protocol (see [RFC2616]) and the syntax of URIs (see [RFC1738] and [RFC2396]) to provide a communication channel between identity providers and service providers. Thus the steps shown in Figure 12 create a communication channel between the service provider and identity provider as follows:

1. The user agent sends an HTTP request to the service provider (typically a GET). In this step the user has typically clicked on a link in the Webpage presently displayed in the user agent.
2. The service provider responds with an HTTP response with a status code of 302 (that is, a redirect) and an alternate URI in the Location header field. In this example, the Location URI will point to the identity provider and will also contain a second, embedded URI pointing back to the service provider.
3. The user agent sends an HTTP request to the identity provider (typically a GET), specifying the complete URI taken from the Location field of the response returned in Step 2 as the argument of the GET. Note: This URI contains the second, embedded URI pointing back to the service provider.
4. The identity provider can then respond in kind with a redirect whose Location header field contains the URI pointing to the service provider (extracted from the GET argument URI supplied in Step 3) and optionally contains an embedded, second URI pointing back to itself.
5. The user agent sends an HTTP request to the service provider (typically a GET), specifying the complete URI taken from the Location field of the response returned in Step 4 as the argument of the GET. Note: This URI might contain any second, embedded URI pointing back to the identity provider.

Note: Both URIs are passed as arguments of HTTP GET requests, and the Location response-header field of redirect responses can contain either or both embedded URIs and other arbitrary

536 data. Thus the identity provider and service provider can relatively freely exchange arbitrary
537 information between themselves across this channel. See Table 3.

538
539 **Table 3: Embedding a parameter within an HTTP redirect**

Location: http://www.foobar.com/auth	Redirects to foobar.com
Location: http://www.foobar.com/auth?XYZ=1234	Redirects to foobar.com and also passes a parameter “XYZ” with the value “1234”

540 **5.1.2 Form-POST-Based Redirection**

541 In form-POST-based redirection, the following steps in Figure 12 are modified as follows:

542
543 2. The service provider responds by returning an HTML form to the user agent containing
544 an action parameter pointing to the identity provider and a method parameter with the value of
545 POST. Arbitrary data may be included in other form fields. The form may also include a
546 JavaScript or ECMAScript fragment that causes the next step to be performed without user
547 interaction.

548 3. Either the user clicks on the Submit button, or the JavaScript or ECMAScript executes.
549 In either case, the form and its arbitrary data contents are sent to the identity provider via the
550 HTTP POST method.

551
552 The above process can be reversed in Steps 4 and 5 to effect form-POST-based communication in
553 the opposite direction.

554 **5.1.3 Cookies**

555 POLICY/SECURITY NOTE: Use of cookies by implementors and deployers should be carefully considered,
556 especially if a cookie contains either or both personally identifying information and authentication
557 information. Cookies can be either ephemeral (that is, this session only) or persistent. Persistent cookies are
558 of special concern because they are typically written to disk and persist across user agent invocations. Thus if
559 a session authentication token is cached in a persistent cookie, the user exits the browser, and another person
560 uses the system and relaunches the browser, then the second person could impersonate the user (unless any
561 authentication time limits imposed by the authentication mechanism have expired).

562
563 Additionally, persistent cookies should be used *only* with the consent of the user. This consent step allows,
564 for example, a user at a public machine to prohibit a persistent cookie that would otherwise remain in the user
565 agent’s cookie cache after the user is finished.

566 **5.1.3.1 Why Not Use Cookies in General?**

567 Cookies are the HTTP state management mechanism specified in [RFC2965] and are a means for
568 Web servers to store information, that is, *maintain state*, in the user agent. However, the default
569 security setting in the predominant user agents allow cookies to be read only by the Website that
570 wrote them. This discrimination is based on the DNS domains of the reading and writing sites.

571
572 To permit multiple identity providers and service providers in different DNS domains to
573 communicate using cookies, users must lower the default security settings of their user agents.
574 This option is often an unacceptable requirement.

575
576 Additionally, it is not uncommon for users and/or their organizations to operate their user agents
577 with cookies turned off.

578 **5.1.3.2 Where Cookies are Used**

579 In the Liberty context, cookies might be used for maintaining local session state, and cookies are
580 used in addressing the introduction problem (see 5.5).

581
582 The fact that identity providers cannot arbitrarily send data to service providers via cookies does
583 not preclude identity providers and service providers from writing cookies to store local session
584 state and other, perhaps persistent, information.

585 **5.1.4 Web Redirection Summary**

586 Web redirection is not an ideal distributed systems architecture.

587
588 POLICY/SECURITY NOTE: Communications across Web redirection channels as described in 5.1.1
589 through 5.1.3 have many well-documented security vulnerabilities, which should be given careful
590 consideration when designing protocols utilizing Web redirection. Such consideration was incorporated into
591 the design of the profiles specified in [LibertyBindProf], and specific considerations are called out as
592 appropriate in that document (for example, regarding cleartext transmissions and caching vulnerabilities).
593 Examples of security vulnerabilities include

- 594
595
 - **Interception:** Such communications go across the wire in cleartext unless all the steps in 5.1.1 through
596 5.1.3 are carried out over an SSL or TLS session or across another secured communication transport, for
597 example, an IPsec-based VPN.
 - **User agent leakage:** Because the channel is redirected through the user agent, many opportunities arise
598 for the information to be cached in the user agent and revealed later. This caching is possible even if a
599 secure transport is used because the conveyed information is kept in the clear in the browser. Thus any
600 sensitive information conveyed in this fashion needs to be encrypted on its own before being sent across
601 the channel.

602
603
604 TECHNICAL NOTE: A key limitation of Web redirection is the overall size of URIs passed as arguments of
605 GET requests and as values of the Location field in redirects. These elements have size limitations that vary
606 from browser to browser and are particularly small in some mobile handsets. These limitations were
607 incorporated into the design of the protocols specified in [LibertyProtSchema] and [LibertyBindProf].

608
609 In spite of the vulnerabilities and limitations of Web redirection, use of this mechanism enables
610 distributed, cross-domain interactions, such as single sign-on, with today's deployed HTTP
611 infrastructure on the Internet.

612
613 Both generic variants of Web redirection underlie several of the profiles specified in
614 [LibertyBindProf]: Single Sign-On and Federation, Identity Federation Termination Notification,
615 Identity Provider Introduction, and Single Logout.

616 **5.2 Web Services Architectural Component**

617 Various Liberty protocol interaction steps are profiled to occur directly between system entities in
618 addition to other steps occurring via Web redirection and are based on RPC-like protocol messages
619 conveyed via SOAP (see [SOAP1.1]). SOAP is a widely implemented specification for RPC-like
620 interactions and message communications using XML and HTTP and hence is a natural fit for this
621 architectural component.

622 5.3 Metadata and Schemas Architectural Component

623 *Metadata and schemas* is an umbrella term generically referring to various subclasses of
624 information and their formats exchanged between service providers and identity providers,
625 whether via protocol or out of band. The subclasses of exchanged information are
626

- 627 • **Account/Identity:** In Liberty Version 1.0, account/identity is simply the opaque user
628 handle that serves as the name that the service provider and the identity provider use in
629 referring to the user when communicating. In future Liberty phases, it will encompass
630 various attributes.
- 631
- 632 • **Authentication Context:** Liberty explicitly accommodates identity provider use of
633 arbitrary authentication mechanisms and technologies. Different identity providers will
634 choose different technologies, follow different processes, and be bound by different legal
635 obligations with respect to how they authenticate users. The choices that an identity
636 provider makes here will be driven in large part by the requirements of the service
637 providers with which the identity provider has federated. Those requirements, in turn, will
638 be determined by the nature of the service (that is, the sensitivity of any information
639 exchanged, the associated financial value, the service providers risk tolerance, etc) that the
640 service provider will be providing to the user. Consequently, for anything other than trivial
641 services, if the service provider is to place sufficient confidence in the authentication
642 assertions it receives from an identity provider, the service provider must know which
643 technologies, protocols, and processes were used or followed for the original authentication
644 mechanism on which the authentication assertion is based. The authentication context
645 schema provides a means for service providers and identity providers to communicate such
646 information (see [LibertyAuthnContext]).
- 647
- 648 • **Provider Metadata:** For identity providers and service providers to communicate with
649 each other, they must a priori have obtained metadata regarding each other. These provider
650 metadata include items such as X.509 certificates and service endpoints.
651 [LibertyProtSchema] defines metadata schemas for identity providers and service providers
652 that may be used for provider metadata exchange. However, provider metadata exchange
653 protocols are outside the scope of the Liberty Version 1.0 specifications.

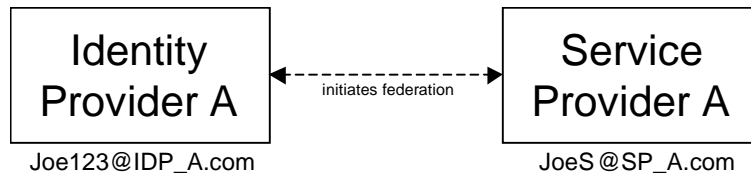
654 5.4 Single Sign-On and Identity Federation

655 The single sign-on and identity federation aspects of Liberty are facilitated by the Single Sign-On
656 and Federation Protocol, which is specified in [LibertyProtSchema]. It facilitates both identity
657 federation (see 5.4.1) and single sign-on (see 5.4.2) in a single overall protocol flow. The various
658 profiles of the overall protocol flow that are defined in [LibertyBindProf] are discussed in 5.4.3.

659 5.4.1 Identity Federation

660 The first time that users use an identity provider to log in to a service provider they must be given
661 the option of federating an existing local identity on the service provider with the identity provider
662 login to preserve existing information under the single sign-on. See Figure 13. It is critical that, in
663 a system with multiple identity providers and service providers, a mechanism exists by which
664 users can be (at their discretion) uniquely identified across the providers. However, it is technically
665 challenging to create a globally unique ID that is not tied to a particular identity provider and a
666 business challenge to ensure the portability of globally unique IDs.

667



668

669

Figure 13: User initiates federation of two identities

670

671 An explicit trust relationship, or chain, is created with the opt-in identity federation that occurs the
 672 first time a user logs in to a service provider using an identity provider. While multiple identities
 673 can be federated to each other, an explicit link exists between each identity. Providers cannot skip
 674 over each other in the trust chain to request information on or services for a user because user
 675 identity information must be checked at each step. Therefore, the only requirement is that, when
 676 two elements of a trust chain communicate, they can differentiate users.

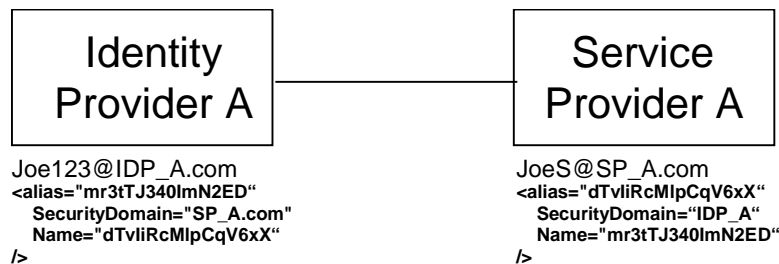
677

678 Members of the circle of trust are not required to provide the actual account identifier for a user
 679 and can instead provide a handle for a particular user. Members can also choose to create multiple
 680 handles for a particular user. However, identity providers must create a single handle for each
 681 service provider that has multiple Websites so that the handle can be resolved across the Websites.

682

683 Because both the identity provider and service provider in such a federation need to remember the
 684 other's handle for the user, they create entries in their user directories for each other and note each
 685 other's handle for the user. See Figure 14 and Figure 15.

686



687

Figure 14: User directories of the identity provider and service provider upon identity federation

689

690 TECHNICAL NOTE: Figure 14, along with the three following figures, illustrate bilateral identity federation;
 691 this is where both the service provider and identity provider exchange handles for the user. However, bilateral
 692 handle exchange is an *optional* feature of the Liberty Single Sign-On and Federation protocol. In some
 693 scenarios, only the identity provider's handle will be conveyed to the service provider(s). This will typically
 694 be the case where the service provider doesn't otherwise maintain its own user repository.

695

696 The lines connecting the identity and service providers in the aforementioned figures signify federation
 697 relationships rather than communication exchanges.

698

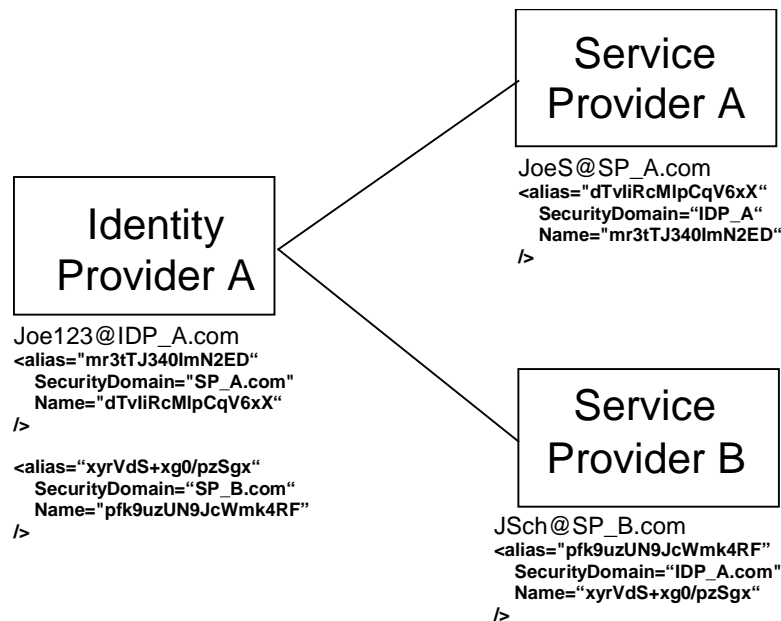


Figure 15: User directories of the identity provider and multiple service providers upon identity federation

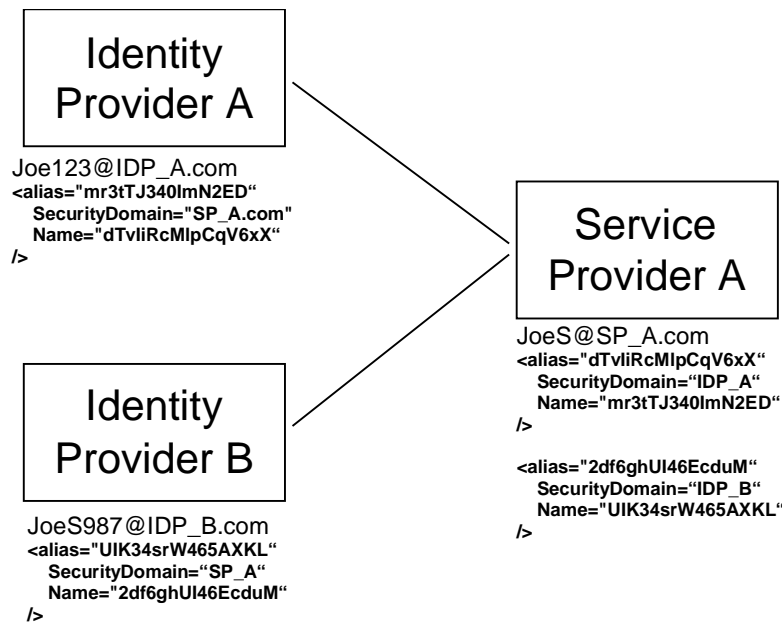
POLICY/SECURITY NOTE:

1. Observe in Figure 15 that SP_A and SP_B cannot communicate directly about Joe Self. They can only communicate with the identity provider individually. This feature is desirable from policy and security perspectives. If Joe Self wishes the service providers to be able to exchange information about him, then he must explicitly federate the two service provider identities, effectively opting in.

Another aspect of this feature is that if the user's local identity is compromised on, for example, SP_A, the local identities at IDP_A or SP_B are not necessarily also compromised.

2. Properties of the user handles, for example, `mr3tTJ340ImN2ED`, (also known as *name identifiers*) need to be carefully considered. It may not be enough for them to be opaque. Considerations of the construction of name identifiers are discussed in [LibProtSchema]. Additionally, user handles should be refreshed periodically. Service providers may refresh the user handles they optionally supply to identity providers via the register name identifier profile defined in [LibertyBindProf]. Identity providers may also use the same profile to optionally refresh the user handles they supply to service providers.

While it is obvious that a user can sign in at multiple service providers with an identity provider, a user can also link multiple identity providers to a particular service provider. See Figure 16. This ability proves useful when a user switches from a work computer to a home computer or from a computer to a mobile device, each of which may be associated with a different identity provider and circle of trust.



727

728

Figure 16: A user with two identity providers federated to a service provider

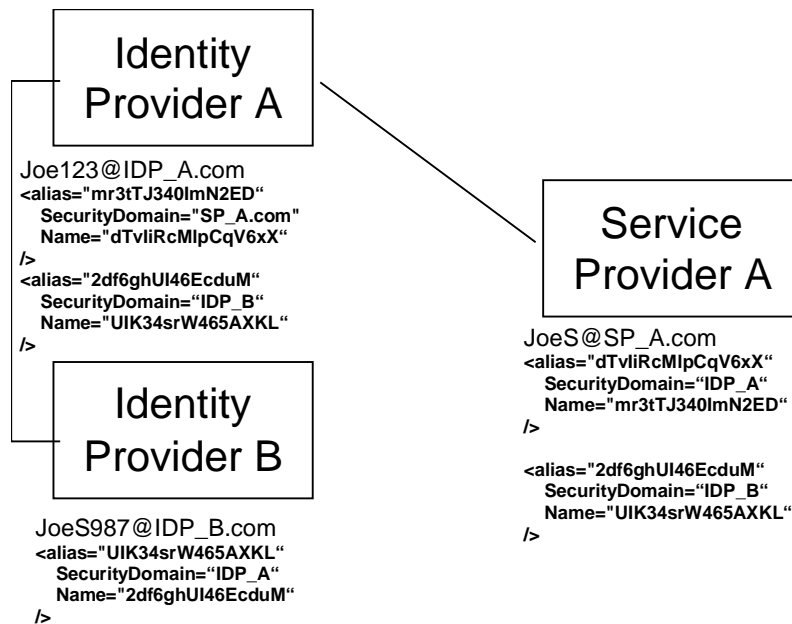
729

730 POLICY/SECURITY NOTE: Subtle considerations arise here in terms of how easy it is for a user to switch
731 between identities and how this capability is materialized. IDP_A may belong to the same circles of trust as
732 more than one of the user's devices. Therefore, certain questions arise, for example, How do users know to
733 which (or both) identity provider they are presently logged in? Features satisfying such questions are a way
734 for identity providers and circles of trust to differentiate themselves.

735

736 While federating two identity providers to a service provider, as illustrated in Figure 16, enables
737 the user to log in to the service provider using either identity provider, the user must remember to
738 federate new service providers to both identity providers, which can be a cumbersome process. An
739 alternative is for the user to federate identity providers together and set policies enabling identity
740 providers to access each other's information. See Figure 17 and the following POLICY/SECURITY
741 NOTE.. The user can then use a preferred identity provider to log in to service providers, but always
742 has the choice of adding additional identity providers to a service provider.

743



744
745 **Figure 17: A user with two identity providers federated**

746
747 **TECHNICAL NOTE:** In Figure 17, Identity Provider A is acting as both a service provider and an identity
748 provider. T

749
750 **POLICY/SECURITY NOTE:**

- 751
- 752 1. The semantics of such a federated relationship (Figure 17) between identity providers are not
753 dictated by the underlying Liberty protocols, nor are they precluded. These semantics need to be
754 addressed by the agreements between the identity providers and supported by the capabilities of the
755 deployed Liberty-enabled implementations.
756
 - 757 2. Additionally, how trust relationships between identity providers are established, and how those
758 relationships are represented to service providers, are unspecified. Identity providers enabling
759 relationships such as that illustrated in Figure 17 must mutually define governing policies and means
760 of representing such trust relationships to relying service providers (for example Service Provider A
761 in Figure 17).
762
 - 763 3. Circle of trust agreements should address how federation failures are materialized to users.
764
 - 765 4. Appropriate portions of the assertions passed between the identity provider and the service provider
766 to effect federation should be logged.
767
 - 768 5. By creating many local identities with many service providers and/or identity providers and then
769 federating them, users possess many sets of local credentials that may be used as a basis to
770 authenticate with many service providers via single sign-on. This situation constitutes a risk. For
771 example, every identity provider that possesses reusable user credentials, for example, a username
772 and password, can impersonate the user at every service provider federated with that account.
773

774 In the normal course of events, some local credentials may go unused for periods of time because the
775 user is making use of the local account via single sign-on from another identity provider. Thus a
776 means of controlling the growth of a user's set of local credentials might be to offer the user the
777 option of invalidating local credentials at identity federation time and also perhaps after a certain
778 number of times of visiting the Website without using them.

779 **5.4.1.1 No Need for Global Account/Identity Namespace**

780 Given the above architecture where users opt to federate identities at different identity providers
781 and service providers, a global namespace across all of the players should not be needed. Circle of
782 trust members can communicate with each other, about or on a user's behalf, only when a user has
783 created a specific federation between the local identities and has set policies for that federation.
784 Although long chains of identity providers and service providers can be created, the user's identity
785 is federated in each link in the chain and, therefore, a globally unique ID need not exist for that
786 user across all of the elements of the chain. See Figure 17.

787 **5.4.1.2 Federation Management: Defederation**

788 Users will have the ability to terminate federations, or *defederate identities*. [LibertyProtSchema]
789 and [LibertyBindProf] specify a Federation Termination Notification Protocol and related profiles.
790 Using this protocol, a service provider may initiate defederation with an identity provider or vice
791 versa. The nominal user experience is for the user to select a Defederate link on a service
792 provider's or identity provider's Webpage. This link initiates defederation with respect to some
793 other, specific, identity provider or service provider.

794
795 When defederation is initiated at an identity provider, the identity provider is stating to the service
796 provider that it will no longer provide user identity information to the service provider and that the
797 identity provider will no longer respond to any requests by the service provider on behalf of the
798 user.

799
800 When defederation is initiated at a service provider, the service provider is stating to the identity
801 provider that the user has requested that the identity provider no longer provide the user identity
802 information to the service provider and that service provider will no longer ask the identity
803 provider to do anything on the behalf of the user.

804
805 POLICY/SECURITY NOTE: Regarding defederation, several issues must be considered:

- 806 • The user should be authenticated by the provider at which identity defederation is being initiated.
- 807 • Providers should ask the user for confirmation before performing defederation and appropriately log
- 808 the event and appropriate portions of the user's authentication information.
- 809 • It is recommended that the service provider, after initiating or receiving a federation termination
- 810 notification for a Principal, check whether that Principal is presently logged in to the service
- 811 provider on the basis of an assertion from the identity provider with which the federation termination
- 812 notification was exchanged. If so, then the local session information that was based on the identity
- 813 provider's assertion should be invalidated.
- 814
- 815
- 816

817
818 If the service provider has local session state information for the Principal that is not based on
819 assertions made by the identity provider with which the federation termination notification was
820 exchanged, then the service provider may continue to maintain that information.

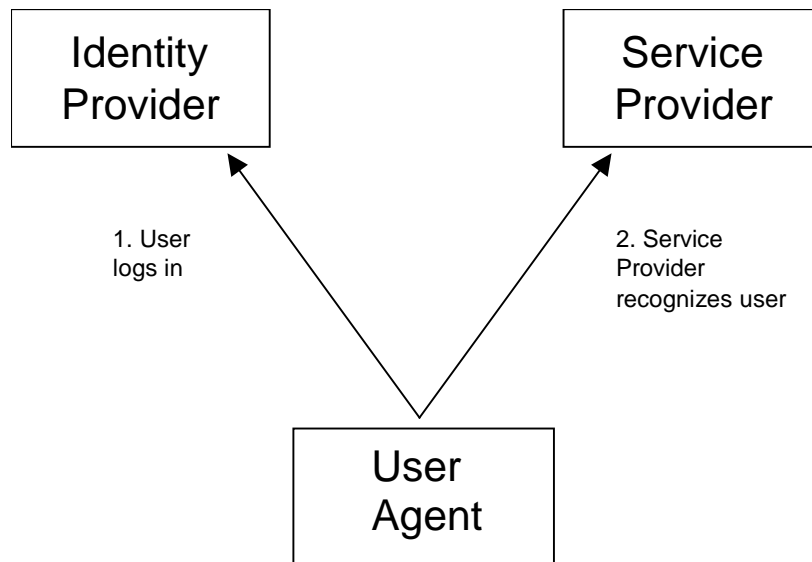
821
822 If the Principal subsequently initiates a single sign-on session with the same identity provider, the
823 service provider will need to request federation as well as authentication from the identity provider.

- 824 • Other means of federation termination are possible, such as federation expiration and termination of
- 825 business agreements between service providers and identity providers.
- 826

827 **5.4.2 Single Sign-on**

828 Single sign-on is enabled once a user's identity provider and service provider identities are
829 federated. From a user's perspective, single sign-on is realized when the user logs in to an identity
830 provider and uses multiple affiliated service providers without having to sign on again (see Figure
831 18). This convenience is accomplished by having federated the user's local identities between the
832 applicable identity providers and the service providers. The basic user single sign-on experience is
833 illustrated in the 5.4.1.

834



835

836 **Figure 18: User logs in at identity provider and is recognized by service provider**

837

838 [LibertyBindProf] specifies single sign-on by profiling both the "Browser/Artifact Profile" and the
839 "Browser/Post Profile" of SAML (see [SAMLBind]).

840

841 POLICY/SECURITY NOTE: Regarding authentication, single sign-on, credentials, etc., several issues must
842 be considered:

843

844 **Authentication Mechanisms are Orthogonal to Single Sign-On**

845

846 Single sign-on is a means by which a service provider or identity provider may convey to another service
847 provider or identity provider that the user is in fact authenticated. The means by which the user was originally
848 authenticated is called the authentication mechanism. Examples of authentication mechanisms are username
849 with password (*not* HTTP Basic Auth), certificate-based (for example, via SSL or TLS), Kerberos, etc.

850

851 **Identity Provider Session State Maintenance**

852

853 Identity providers need to maintain authentication state information for principals. This is also known as
854 "local session state maintenance", where "local" implies "local to the identity provider". There are several
855 mechanisms for maintaining local session state information in the context of HTTP-based [RFC2616] user
856 agents (commonly known as "web browsers"). Cookies are one such mechanism and are specified in
857 [RFC2965]. Identity providers use local session state information, mapped to the participating user agent (see
858 Figure 18), as the basis for issuing authentication assertions to service providers who are performing the
859 "Single Sign-On and Federation" protocol [LibertyBindProf] with the identity provider. Thus, when the
860 Principal uses his user agent to interact with yet another service provider, that service provider will send an
861 <AuthnRequest> to the identity provider. The identity provider will check its local session state information
862 for that user agent, and return to the service provider an <AuthnResponse> containing an authentication
863 assertion if its local session state information indicates the user agent's session with the identity provider is
864 presently active.

864

865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921

Credentials

Credentials are relied upon in a number of ways in a single sign-on system and are often the basis for establishing trust with the credential bearer. Credentials may represent security-related attributes of the bearer, including the owner's identity. Sensitive credentials that require special protection, such as private cryptographic keys, must be protected from unauthorized exposure. Some credentials are intended to be shared, such as public-key certificates.

Credentials are a general notion of the data necessary to prove an assertion. For example, in a password-based authentication system, the user name and password would be considered credentials. However, the use of credentials is not limited to authentication. Credentials may also be relied upon in the course of making an authorization decision.

As mentioned above, certain credentials must be kept confidential. However, some credentials not only need to remain confidential, but also must be integrity-protected to prevent them from being tampered with or even fabricated. Other credentials, such as the artifacts described in 5.4.3.1, must have the properties of a nonce. A nonce is a random or nonrepeating value that is included in data exchanged by a protocol, usually for guaranteeing liveness and thus detecting and protecting against replay attacks.

Authentication Type, Multitiered Authentication

All authentication assertions should include an authentication type that indicates the quality of the credentials and the mechanism used to vet them. Credentials used to authenticate a user or supplied to authorize a transaction and/or the authentication mechanism used to vet the credentials may not be of sufficient quality to complete the transaction.

For example, a user initially authenticates to the identity provider using username and password. The user then attempts to conduct a transaction, for instance, a bank withdrawal, which requires a stronger form of authentication. In this case the user must present a stronger assertion of identity, such as a public-key certificate or something ancillary such as birthdate, mother's maiden name, etc. This act is *reauthentication* and the overall functionality is *multitiered authentication*. Wielding multitiered authentication can be a policy decision at the service provider and can be at the discretion of the service provider. Or it might be established as part of the contractual arrangements of the circle of trust. In this case, the circle of trust members can agree among themselves upon the trust they put in different authentication types and of each other's authentication assertions. Such an agreement's form may be similar to today's certificate practice statements (CPS) (for example, see <http://www.verisign.com/repository/cps20/cps20.pdf>). The information cited in such a document may include

- User identification methods during credentials enrollment
- Credentials renewal frequency
- Methods for storing and protecting credentials (for example, smartcard, phone, encrypted file on hard drive, etc.)

Note: While the current Liberty specifications allow service providers, identity providers, and user agents to support authentication using a range of methods, the methods and their associated protocol exchanges are not specified within Liberty documents. Further, the scope of the current Liberty specifications does not include a means for a communicating identity provider and user agent to identify a set of methods that they are both equipped to support. As a result, support for the Liberty specifications is not in itself sufficient to ensure effective interoperability between arbitrary identity providers and user agents using arbitrary methods and must, instead, be complemented with data obtained from other sources.

Also, the scope of the current Liberty specifications does not include a means for a service provider to interrogate an identity provider and determine the set of authentication profiles for which a user is registered at that identity provider. As a result, effective service provider selection of specific profiles to authenticate a particular user will require access to out-of-band information describing users' capabilities.

922 For example, members of a given circle of trust may agree that they will label an authentication assertion
923 based on PKI technology and face-to-face user identity verification with substantiating documentation at
924 enrollment time to be of type “Strong.” Then, when an identity provider implementing these policies and
925 procedures asserts that a user has logged in using the specified PKI-based authentication mechanism, service
926 providers rely upon said assertion to a certain degree. This degree of reliance is likely different from the
927 degree put into an assertion by an identity provider who uses the same PKI-based authentication mechanism,
928 but who does not claim to subject the user to the same amount of scrutiny at enrollment time.

929
930 This issue has another dimension: Who performs the reauthentication? An identity provider or the service
931 provider itself? This question is both an implementation and deployment issue and an operational policy
932 issue. Implementations and deployments need to support having either the identity provider or the service
933 provider perform reauthentication when the business considerations dictate it (that is, the operational policy).
934 For example, a circle of trust may decide that the risk factors are too large for having the identity provider
935 perform reauthentication in certain high-value interactions and that the service provider taking on the risk of
936 the interaction must be able to perform the reauthentication.

937 938 **Mutual Authentication**

939
940 Another dimension of the authentication type and quality space is mutual authentication. For a user
941 authenticating himself to an identity provider, mutual authentication implies that the identity provider server
942 authenticates itself with the user as well as vice versa. Mutual authentication is a function of the particular
943 authentication mechanism employed. For example, any user authentication performed over SSL or TLS is
944 mutual authentication because the server is authenticated to the client by default with SSL or TLS. This
945 feature can be the basis of some greater assurance, but does have its set of vulnerabilities. The server may be
946 wielding a bogus certificate, and the user may not adequately inspect it or understand the significance.

947 948 **Validating Liveness**

949
950 *Liveness* refers to whether the user who authenticated at time t_0 is the same user who is about to perform a
951 given operation at time t_1 . For example, a user may log in and perform various operations and then attempt to
952 perform a given operation that the service provider considers high-value. The service provider may initiate
953 reauthentication to attempt to validate that the user operating the system is still the same user that
954 authenticated originally. Even though such an approach has many vulnerabilities, that is, it fails completely in
955 the case of a rogue user, it does at least augment the service provider’s audit trail. Therefore, at least some
956 service providers will want to do it.

957
958 Authentication assertions from identity providers contain a `<ReauthenticationOnOrAfter>` element. If
959 this attribute was specified and the time of the user request is past the specified reauthentication time, the
960 service provider should redirect the user back to the identity provider for reauthentication.

961 962 **Communication Security**

963
964 A service provider can reject communications with an identity provider for various reasons. For example, it
965 may be the policy of a service provider to require that all protocol exchanges between it and the bearer of a
966 credential commence over a communication protocol that has certain qualities such as bilateral
967 authentication, integrity protection, and message confidentiality.

968 **5.4.3 Profiles of the Single Sign-On and Federation Protocol**

969 The Single Sign-On and Federation Protocol, as specified in [LibertyProtSchema], defines
970 messages exchanged between service providers and identity providers. The concrete mapping of
971 these messages to particular transfer (for example, HTTP) and/or messaging (for example, SOAP)
972 protocols and precise protocol flows are specified in [LibertyBindProf]. These mappings are called
973 *profiles*. The Single Sign-On and Federation Protocol specifies four profiles. The following
974 sections summarize each profile. For a detailed discussion of the common interactions and
975 processing rules of these profiles and for details about each profile, see [LibertyBindProf].

976

977 TECHNICAL NOTE: The Single Sign-On and Federation Protocol and related profiles specify means by
978 which service providers indicate to identity providers the particular profile they wish to employ. The primary
979 means is the <lib:ProtocolProfile> element of the <lib:AuthnRequest> message, which is
980 employed by all profiles of the Single Sign-On and Federation Protocol. Note: The Liberty-enabled client and
981 proxy profile employs additional means.

982 5.4.3.1 Liberty Browser Artifact Profile

983 The Liberty browser artifact profile specifies embedding an artifact in a URI exchanged between
984 the identity provider and service provider via Web redirection and also requires direct
985 communication between the service provider and the identity provider. The artifact itself is an
986 opaque user handle with which the service provider can query the identity provider to receive a full
987 SAML assertion. The motivation for this approach is that the artifact can be small enough in its
988 URI-encoded form to fit in a URI without concern for size limitations. The artifact has the
989 property of being an opaque, pseudo-random nonce that can be used only once. These properties
990 are countermeasures against replay attacks. The randomness property protects the artifact from
991 being guessed by an adversary.

992 5.4.3.2 Liberty Browser POST Profile

993 Modern browsers that support JavaScript or ECMAScript can perform the redirect by sending an
994 HTML page with form elements that contain data with a JavaScript or ECMAScript that
995 automatically posts the form. Legacy browsers, or browsers with scripting disabled, must embed
996 the data within the URI.
997

The Liberty browser POST profile embeds an assertion within an HTTP form per the form-POST-based redirection (see 5.1.2). As a result, this profile does not require any direct communication between the service provider and the identity provider to obtain an assertion. An entire authentication assertion can be included in the posted HTML form because the size allowances for HTML forms are great enough to accommodate one.. See Figure 19.

```
998  
999 <HTML>  
1000 <BODY ONLOAD=" javascript:document.forms[0].submit()">  
1001 <FORM METHOD="POST" ACTION="www.foobar.com/auth">  
1002 <INPUT TYPE="HIDDEN" NAME="FOO" VALUE="1234" />  
1003 </FORM>  
1004 </BODY>  
1005 </HTML>
```

1006 **Figure 19: Example of JavaScript-based HTML form autosubmission with hidden fields**

1007
1008 TECHNICAL NOTE: It must be stressed that Liberty browser POST profile should be supported only in
1009 addition to Liberty browser artifact profile due to its dependence on JavaScript (or ECMAScript).
1010

1011 POLICY/SECURITY NOTE: Implementors and deployers should provide for logging appropriate portions
1012 of the authentication assertion.

1013 5.4.3.3 Liberty WML POST Profile

1014 The Liberty WML POST profile relies on the use of WML events to instruct a WML browser to
1015 submit a HTTP form. WML browsers are typical on mobile handsets. The browsers on such
1016 handsets communicate via a dedicated proxy, a WAP gateway. This proxy converts the Wireless
1017 Session Protocol of the handset into HTTP. Note: The service provider and identity provider will
1018 be contacted using only HTTP.

1019
1020 TECHNICAL NOTE: The primary difference between this profile and the Liberty browser POST profile is
1021 that certain responses from the service provider and identity provider to the user agent contain WML rather
1022 than HTML.
1023
1024 The difference between this profile and the Liberty-enabled client and proxy profile is that this profile is
1025 designed to accommodate standard, unmodified WML browsers, while the Liberty-enabled client and proxy
1026 profile assumes a browser and/or proxy with built-in Liberty protocol capabilities.

1027 **5.4.3.4 Liberty-Enabled Client and Proxy Profile**

1028 The Liberty-enabled client and proxy profile specifies interactions between Liberty-enabled clients
1029 and/or proxies, service providers, and identity providers. A Liberty-enabled client is a client that
1030 has, or knows how to obtain, knowledge about the identity provider that the user wishes to use
1031 with the service provider. In addition a Liberty-enabled client receives and sends Liberty messages
1032 in the body of HTTP requests and responses using POST, rather than relying upon HTTP redirects
1033 and encoding protocol parameters into URLs. Therefore, Liberty-enabled clients have no
1034 restrictions on the size of the Liberty protocol messages.

1035
1036 A Liberty-enabled proxy is a HTTP proxy (typically a WAP gateway) that emulates a Liberty-
1037 enabled client.

1038
1039 TECHNICAL NOTE: The differences between this profile and the other Liberty POST-based profiles are that
1040

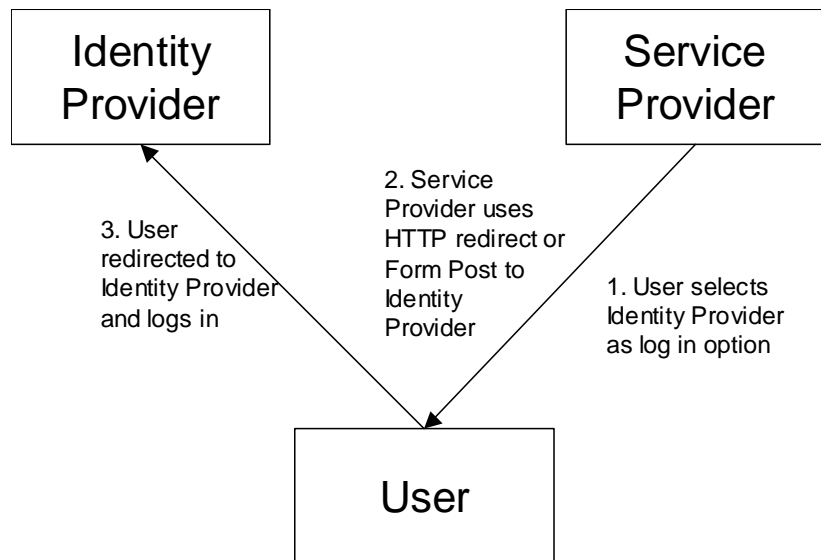
- 1041 • It does not rely upon HTTP redirects.
- 1042 • The interactions between the user agent and the identity provider are SOAP-based.
- 1043 • The Liberty-enabled client and proxy profile includes Liberty-specified HTTP headers in the
1044 protocol messages it sends, signifying to identity providers and service providers that it is Liberty-
1045 enabled and thus can support capabilities beyond those supported by common non-Liberty-enabled
 user agents.

1046 **5.4.3.5 Single Sign-On Protocol Flow Example: Liberty Browser Artifact Profile**

1047 The first step in the single sign-on process in a Liberty browser artifact profile is that the user goes
1048 to a service provider and chooses to log in via the user's preferred identity provider. This login is
1049 accomplished by selecting the preferred identity provider from a list presented on the service
1050 provider's login page.

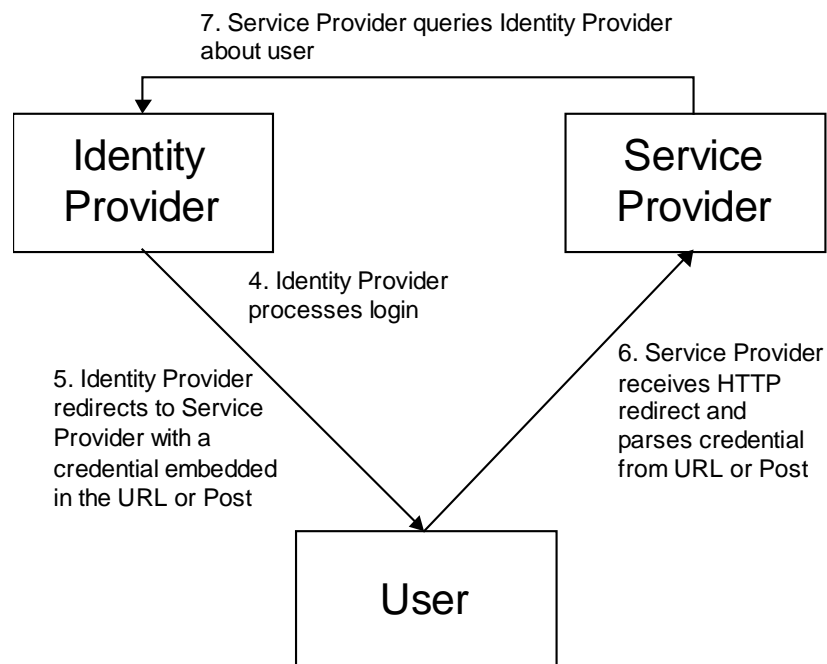
1051
1052 TECHNICAL NOTE: The service provider may discover the preferred identity provider via the identity
1053 provider introduction mechanism discussed 5.5 or, in the case of a Liberty-enabled client or proxy, by some
1054 other implementation-specific and unspecified means.

1055
1056 Once the user selects the identity provider, the user's browser is redirected to the identity provider
1057 with an embedded parameter indicating the originating service provider. The user can then log in
1058 to the identity provider as the user normally would. See Figure 20.
1059



1060
1061 **Figure 20: Single sign-on using HTTP redirect / form POST (1 of 2)**

1062
1063 The identity provider then processes the login as normal and, upon successful login, redirects the
1064 user's browser back the originating service provider with a transient, encrypted credential, called
1065 an *artifact*, embedded within the URI. The service provider then parses the artifact from the URI
1066 and directly uses it to query the identity provider about the user. In its response, the identity
1067 provider vouches for the user, and the service provider may then establish a local notion of session
1068 state. See Figure 21.



1070
1071 **Figure 21: Single sign-on using HTTP redirect / form POST (2 of 2)**

1072 **5.5 Identity Provider Introduction**

1073 In circle of trusts having more than one identity provider, service providers need a means to
1074 discover which identity providers a user is using. Ideally, an identity provider could write a cookie

1075 that a service provider could read. However, due to the cookie constraint outlined in 5.1.3, an
1076 identity provider in one DNS domain has no standardized way to write a cookie that a service
1077 provider in another DNS domain can read.

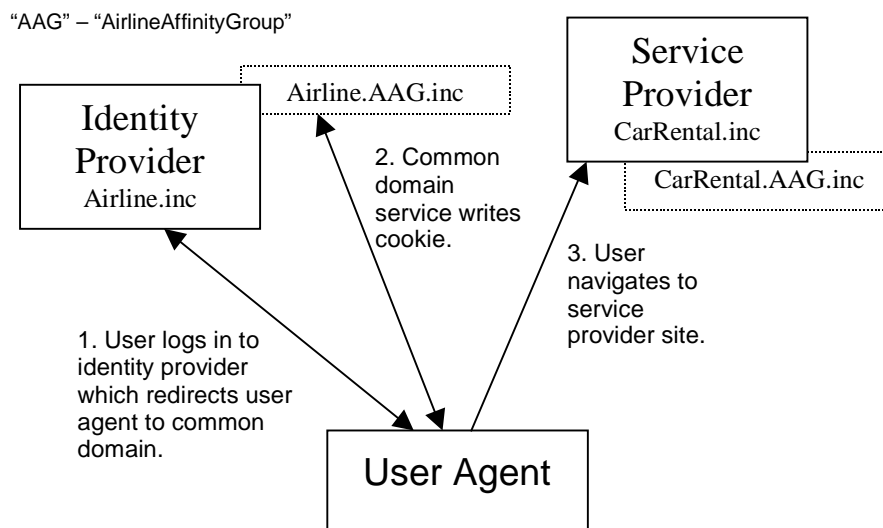
1078

1079 A solution to this introduction problem is to use a domain common to the circle of trust in question
1080 and thus accessible to all parties, for example, AirlineAffinityGroup.inc or AAG.inc. Entries
1081 within this DNS domain will point to IP addresses specified by each affinity group member. For
1082 example, service provider CarRental.inc might receive a third-level domain “CarRental.AAG.inc”
1083 pointing to an IP address specified by CarRental.inc. The machines hosting this *common domain*
1084 *service* would be stateless. They would simply read and write cookies based on parameters passed
1085 within redirect URLs. This is one of several methods suggested for setting a common cookie in
1086 Section 3.6.2 of [LibertyBindProf].

1087

1088 When a user authenticates with an identity provider, the identity provider would redirect the user’s
1089 browser to the identity provider’s instance of a common domain service with a parameter
1090 indicating that the user is using that identity provider. The common domain service writes a cookie
1091 with that preference and redirects the user’s browser back to the identity provider. Then, the user
1092 can navigate to a service provider within the circle of trust. See Figure 22.

1093



1094

1095 **Figure 22: Using a common domain to facilitate introductions (1 of 2)**

1096

1097 When the user navigates to a service provider within the circle of trust, the service provider can
1098 redirect the user’s browser to its instance of the common domain service, which reads the cookie
1099 and redirects the user’s browser back to the service provider with the user’s identity provider
1100 embedded in the URL and thus available to service provider systems operating within the service
1101 provider’s typical DNS domain. See Figure 23.

1102

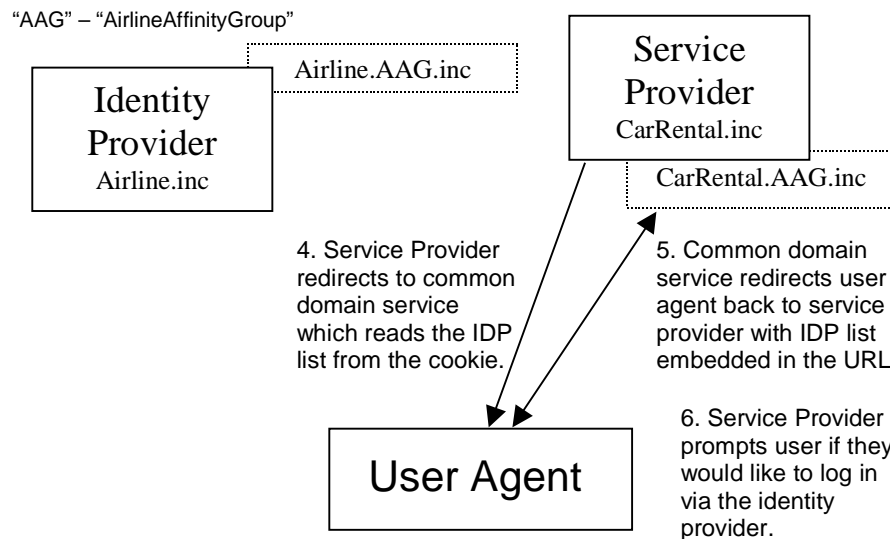


Figure 23: Using a common domain to facilitate introductions (2 of 2)

The service provider now knows with which identity provider the user has authenticated within its circle of trust and can engage in further Liberty protocol operations with that identity provider, for example, single sign-on, on the user's behalf.

POLICY/SECURITY NOTE:

Common Domain Cookie Implications

The identity provider can create either a session common domain cookie (for example, *this session only*; in practice having ephemeral behavior, see [RFC2965]) or a persistent common domain cookie. The implications with a session cookie are that it will disappear from the user agent cookie cache when the user logs out (although this action would have to be explicitly implemented) or when the user agent is exited. This feature may inconvenience some users. However, whether to use a session or a persistent cookie could be materialized to the user at identity provider login time in the form of a Remember Me checkbox. If not checked, a session cookie is used; if checked, a persistent one is used.

A user security implication of the persistent cookie is that if another person uses the machine, even if the user agent had been exited, the persistent common domain cookie is still present—indeed all persistent cookies are present. See the policy/security note in 5.1.3.

However, if the only information contained in a common domain cookie is a list of identity providers—that is, it does not contain any personally identifiable information or authentication information, then the resultant security risk to the user from inadvertent disclosure is low.

Common Domain Cookie Processing

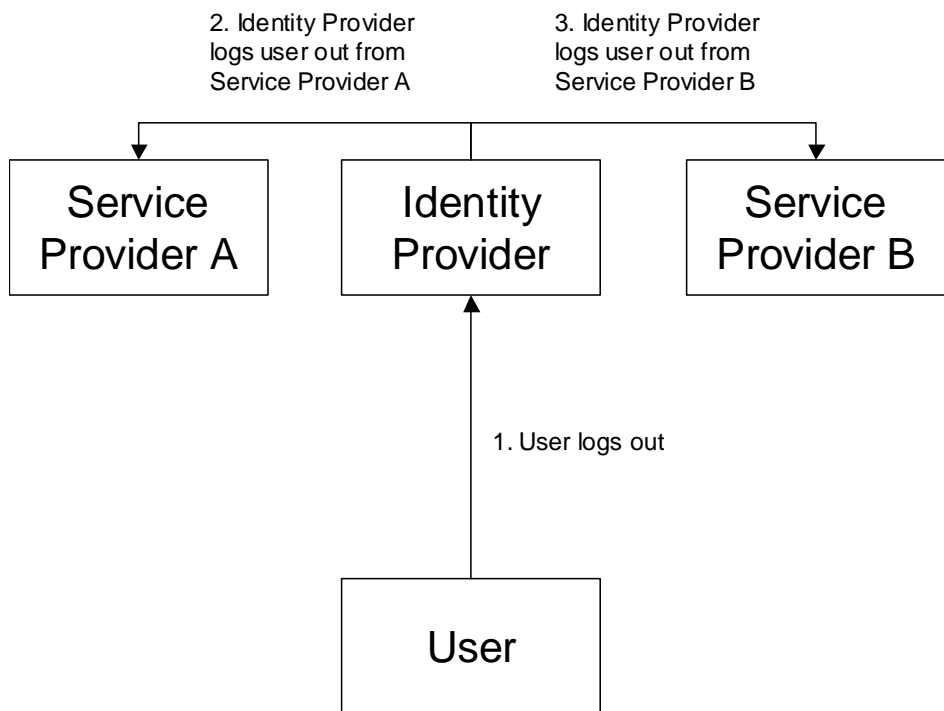
The manner in which the common domain cookie writing service manipulates the common domain cookie is specified in 3.6.2 of [LibertyBindProf]. The identity provider with which the user most recently authenticated should be the last one in the list of identity providers in the cookie. However, the manner in which service providers interpret the common domain cookie and display choices to the user is unspecified. This lack of specificity implies that service providers may approach it in various ways. One way is to display identity providers in a list ordered in reverse to the order in the common domain cookie. This approach will nominally be in order of most-recently used if the common domain cookie writing service is adhering to the above guideline. Or, the service provider may display only the last identity provider in the list. Or the service provider may display the identity providers in some other order, if needed for some reason(s).

1141 **5.6 Single Logout**

1142 The Single Logout Protocol and related profiles synchronize session logout functionality across all
1143 sessions that were authenticated by a particular identity provider. The single logout can be initiated
1144 at either the identity provider (see Figure 24) or the service provider (see Figure 25). In either case,
1145 the identity provider will then communicate a logout request to each service provider with which it
1146 has established a session for the user.

1147
1148 POLICY/SECURITY NOTE: When using a single sign-on system, it is critical that, when users log out at a
1149 service provider, their expectations are set about whether they are logging out from the identity provider or
1150 only that particular service provider. It may be necessary to provide both Single Logout and Site Logout
1151 buttons or links in Websites so that users' expectations are set. However, site logout may be regarded to come
1152 into play only where users have to take a positive action to use their current authentication assertion at a site
1153 that they have previously associated with their single sign-on.

1154



1155

1156

Figure 24: Single logout from an identity provider

1157

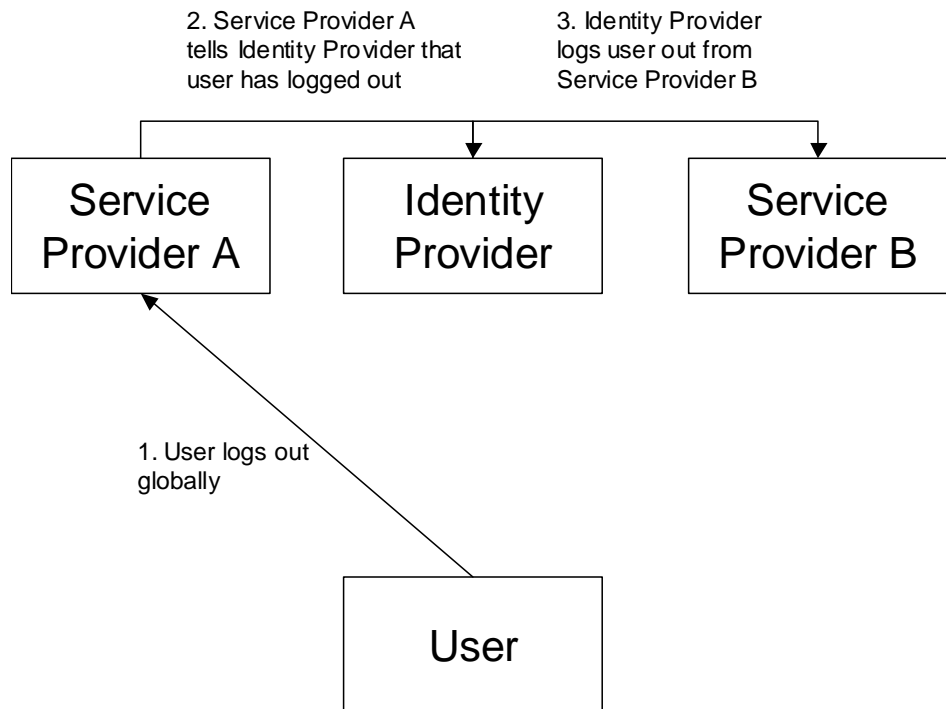


Figure 25: Single logout from a service provider

1158

1159

1160

1161 5.6.1 Single Logout Profiles

1162 [LibertyBindProf] specifies three overall profiles for communicating the logout request among
1163 service providers and an identity provider:

1164

- 1165 • **HTTP-Redirect-Based:** Relies on using HTTP 302 redirects
- 1166 • **HTTP-GET-Based:** Relies on using HTTP GET requests of IMG tags
- 1167 • **SOAP/HTTP-Based:** Relies on SOAP over HTTP messaging

1168

1169 All three profiles may be initiated at an identity provider. Only the first and the last may be
1170 initiated at a service provider. See [LibertyBindProf] for details.

1171

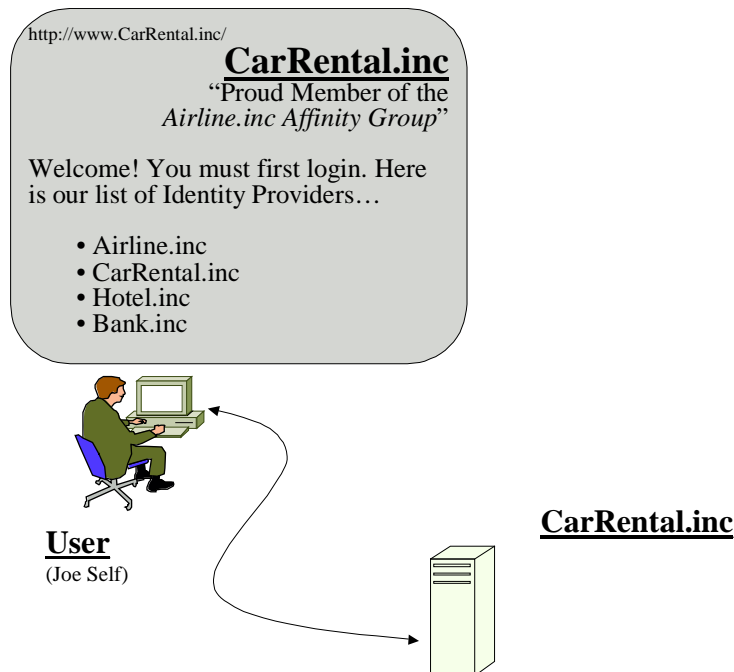
1172 TECHNICAL NOTE: The user-perceivable salient difference between the single logout profiles is that with
1173 the HTTP-redirect-based and SOAP/HTTP-based profiles, the Webpage from which the user initiates the
1174 logout process will remain in place as the logout process occurs (that is, each service provider is contacted in
1175 turn), while with the HTTP-GET-based profile, the identity provider has the opportunity to reload images
1176 (one per service provider, for example, completion check marks) on the viewed Webpage as the logout
1177 process proceeds.

1178 5.7 Example User Experience Scenarios

1179 This section presents several example user experience scenarios based upon the federation,
1180 introduction, and single sign-on facets of the Liberty Version 1.0 architecture. The intent is to
1181 illustrate the more subtle aspects of the user experience at login time and to illustrate
1182 commonWeb-specific user interface techniques that may be employed in prompting for, and
1183 collecting, the user's credentials. Specific policy and security considerations are called out.

1184 **5.7.1 Scenario: Not Logged in Anywhere, No Common Domain Cookie**

1185 In this scenario, Joe Self is not logged in at any Website, does not have a common domain cookie
1186 (for example, he restarted his user agent and/or flushed the cookie cache), and surfs to
1187 CarRental.inc. without first visiting his identity provider, Airline.inc.
1188



1189 **Figure 26: User arrives at service provider's Website without any authentication evidence or**
1190 **common domain cookie**
1191

1192 CarRental.inc presents Joe Self with a welcome page listing identity providers from which he can
1193 select (see Figure 26). Joe Self selects Airline.inc from the list.
1194

1195 Sections 5.7.1.1 through 5.7.1.3 illustrate three different, plausible, Web-specific user interface
1196 techniques CarRental.inc, working in concert with Airline.inc, may use to facilitate Joe Self's
1197 login:
1198

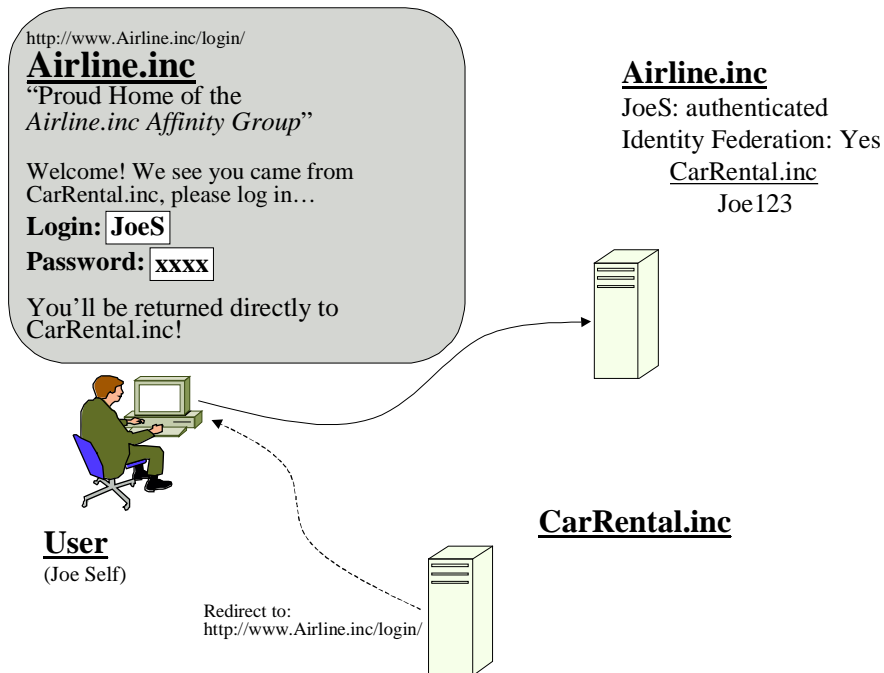
- 1199
- 1200 • Redirect to identity provider Website
- 1201 • Identity provider dialog box
- 1202 • Embedded form

1203 **TECHNICAL NOTE:** These user interface techniques are commonly employed in Web-based systems. They
1204 are not particular to, or specified by, Liberty. They are presented for illustrative purposes only.
1205

1206 **5.7.1.1 Login via Redirect to Identity Provider Website**

1207 With login via redirect to the identity provider's Website, service providers provide direct links,
1208 likely effected via redirects, to the identity provider's appropriate login page. Joe Self's browser
1209 will display an identity provider's Webpage (see Figure 27); and upon successful login, his
1210 browser will be redirected back to the service provider's Website where Joe Self will be provided
1211 access (see Figure 30).

1212



1213

1214

Figure 27: Service provider redirects to identity provider's login page.

1215

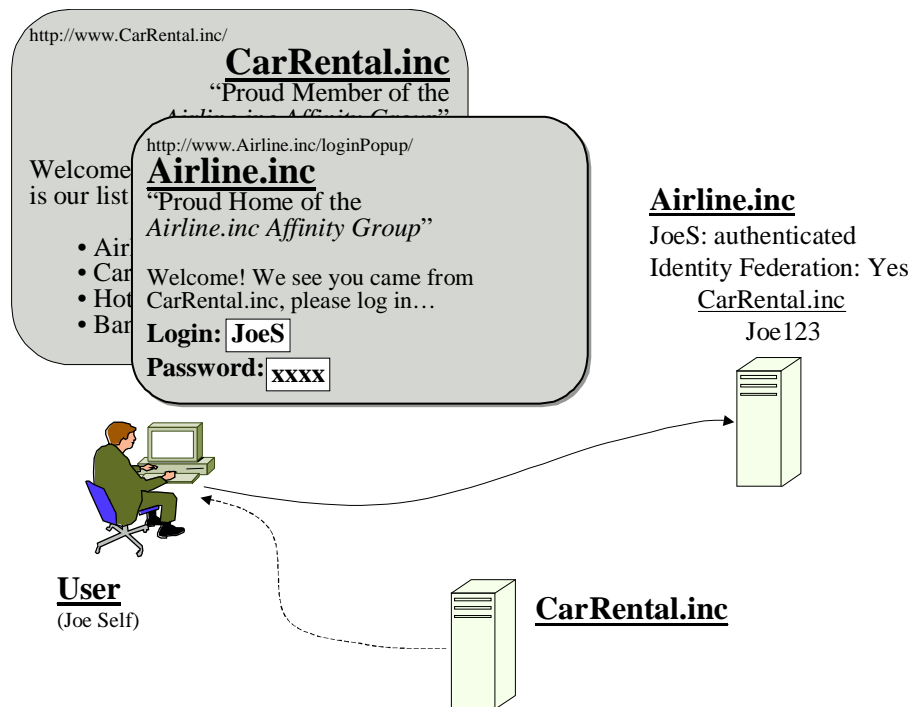
POLICY/SECURITY NOTE: Login via redirect to the identity provider's Website is relatively secure in that the user reveals his credentials directly to the identity provider. Of course, the usual security considerations surrounding login and authentication events apply.

1219

5.7.1.2 Login via Identity Provider Dialog Box

1220 With login via a dialog box from the identity provider, the links on the service provider's Webpage
1221 invoke a dialog or popup box. Joe Self's browser will display an identity provider popup (see
1222 Figure 28); and upon successful login, the popup box will close, and Joe Self will be provided
1223 access at the service provider's Website (see Figure 30).

1224



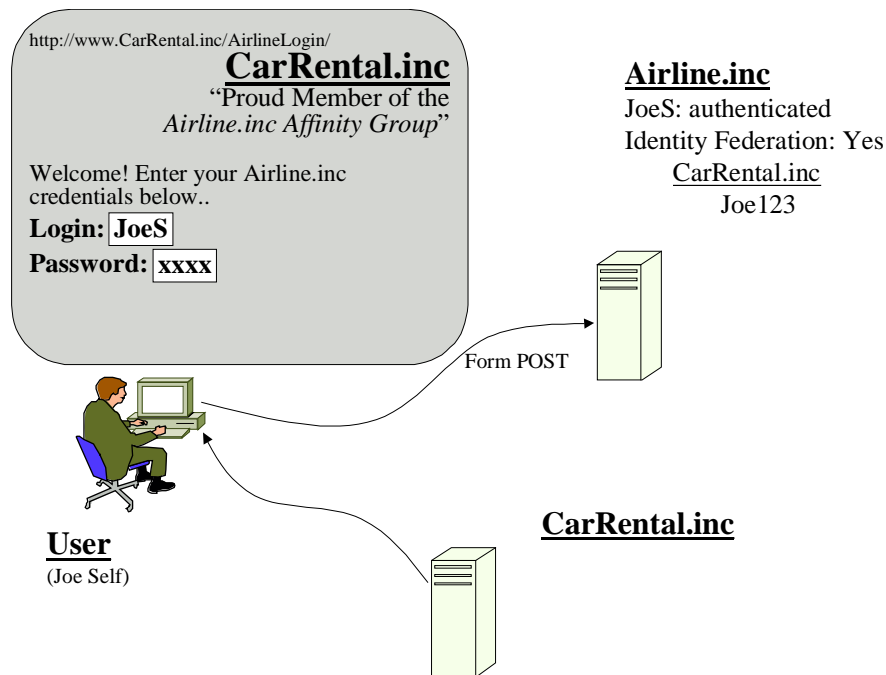
1225
1226 **Figure 28: Service provider invokes dialog or popup box from identity provider.**

1227
1228 POLICY/SECURITY NOTE: Login via a dialog box from the identity provider is relatively secure in that the
1229 user reveals his credentials directly to the identity provider. Of course, the usual security considerations
1230 surrounding login and authentication events apply.

1231 **5.7.1.3 Login via Embedded Form**

1232 With login via embedded form, the links on the service provider's Webpage cause the service
1233 provider to display embedded login forms. In other words, the displayed page comes from the
1234 service provider, but when Joe Self presses the Submit button, the information is conveyed to the
1235 identity provider, typically via POST (see Figure 29). To Joe Self, it appears as if he has not left
1236 the service provider's Webpages. Upon successful login, Joe Self will be provided access at the
1237 service provider's Website (see Figure 30).

1238



1239

1240

Figure 29: Login via embedded form

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

POLICY/SECURITY NOTE: Although users may like the seamlessness of this embedded form mechanism and deployers will like that the user does not leave their Website, it has serious policy and security considerations. In this mechanism, the user may be revealing his identity provider credentials to the service provider in cleartext. This is because the service provider controls the actual code implementing both the page and the embedded form and thus can conceivably capture users' credentials. In this way, privacy surrounding the user's identity provider account may be compromised by such a rogue service provider, who could then wield those credentials and impersonate the user. Because of this, when using authentication via embedded form, deployers may want to consider appropriate contract terms between identity providers and service providers to address this risk.

1252

5.7.1.4 The User is Logged in at CarRental.inc

1253

CarRental.inc and Airline.inc then work in conjunction to effect login, and the CarRental.inc

1254

Website establishes a session based upon Joe Self's identity federation with Airline.inc (see Figure 30).

1255

1256

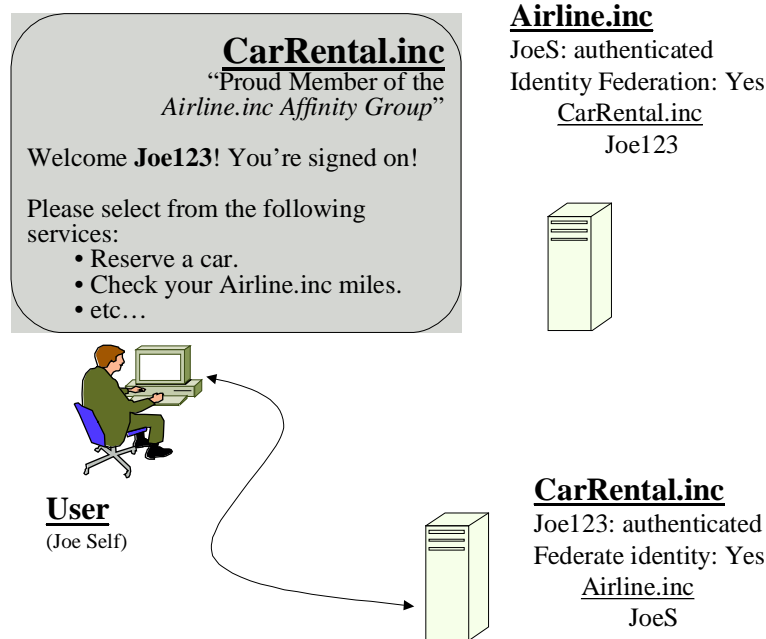


Figure 30: Service provider's Website delivers services on basis of federated identity.

1257
1258
1259

5.7.2 Scenario: Not Logged in Anywhere, Has a Common Domain Cookie

This scenario is similar the prior one. The only difference is that Joe Self's browser already has a common domain cookie cached. Therefore, when he arrives at a CarRental.inc Webpage, CarRental.inc will immediately know with which identity provider Joe Self is affiliated (Airline.inc in this case). It can immediately perform login via one of the three mechanisms outlined in the prior example or may prompt the user first.

POLICY/SECURITY NOTE: Implementors and deployers should make allowance for the user to decide whether to immediately authenticate with the identity provider or be offered the chance to decline and authenticate either locally with the service provider or select from the service provider's list of affiliated identity providers.

5.7.3 Scenario: Logged in, Has a Common Domain Cookie

This scenario is the one illustrated in 2.2.

6 References

- [LibertyArchImpl] Kannappan, L., Lachance, M., & Kemp, J., eds. (January 2003). "Liberty Architecture Implementation Guidelines," Version 1.1. Liberty Alliance Project, <<http://www.projectliberty.org/specs/>>.
- [LibertyAuthnContext] Madsen, P., & Kemp, J., eds. (January 2003). "Liberty Authentication Context Specification," Version 1.1. Liberty Alliance Project, <<http://www.projectliberty.org/specs/>>.
- [LibertyBindProf] Rouault, J., & Wason, T., eds. (January 2003). "Liberty Bindings and Profiles Specification," Version 1.1. Liberty Alliance Project, <<http://www.projectliberty.org/specs/>>.

1283	[LibertyGloss]	Mauldin, H., & Wason, T., eds. (January 2003). "Liberty Architecture Glossary," Version 1.1. Liberty Alliance Project, < http://www.projectliberty.org/specs/ >.
1284		
1285	[LibertyProtSchema]	Beatty, J., & Kemp, J., eds. (January 2003). "Liberty Protocols and Schema Specification," Version 1.1. Liberty Alliance Project, < http://www.projectliberty.org/specs/ >.
1286		
1287	[RFC1738]	Berners-Lee, T., Masinter, L., & McCahill, M. (December 1994). "Uniform Resource Locators (URL)," RFC 1738. The Internet Engineering Task Force, < http://www.rfc-editor.org/rfc/rfc1738.txt > [18 December 2002].
1288		
1289		
1290	[RFC2119]	Bradner, S. (March 1997). "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119. The Internet Engineering Task Force, < http://www.rfc-editor.org/rfc/rfc2119.txt > [18 December 2002].
1291		
1292		
1293	[RFC2246]	Dierks, T.,& Allen, C. (January 1999). "The TLS Protocol Version 1.0," RFC 2246. The Internet Engineering Task Force, < http://www.rfc-editor.org/rfc/rfc2246.txt > [18 December 2002]
1294		
1295		
1296	[RFC2396]	Berners-Lee, T., Fielding, R., & Masinter, L. (August 1998). "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396. The Internet Engineering Task Force, < http://www.rfc-editor.org/rfc/rfc2396.txt > [18 December 2002].
1297		
1298		
1299	[RFC2616]	Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (June 1999). "Hypertext Transfer Protocol -- HTTP/1.1," RFC 2616. The Internet Engineering Task Force, < http://www.rfc-editor.org/rfc/rfc2616.txt > [18 December 2002].
1300		
1301		
1302	[RFC2617]	Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., & Stewart, L. (June 1999). "HTTP Authentication: Basic and Digest Access Authentication," RFC 2617. The Internet Engineering Task Force, < http://www.rfc-editor.org/rfc/rfc2617.txt > [18 December 2002]
1303		
1304		
1305		
1306	[RFC2965]	Kristol, D., & Montulli, L. (October 2000). "HTTP State Management Mechanism," RFC 2965. The Internet Engineering Task Force, < http://www.rfc-editor.org/rfc/rfc2965.txt > [18 December 2002].
1307		
1308		
1309	[SAMLBind]	Mishra, P., ed. (05 Nov. 2002). "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)," Version 1.0, OASIS Standard. Organization for the Advancement of Structured Information Standards, < http://www.oasis-open.org/committees/security/#documents > [18 December 2002].
1310		
1311		
1312		
1313	[SOAP1.1]	D. Box et al. (May 2000). "Simple Object Access Protocol (SOAP) 1.1," Note. World Wide Web Consortium, < http://www.w3.org/TR/SOAP > [18 December 2002].
1314		
1315	[SSLv3]	Freier, A. O., Karlton, P., & Kocher, P. (November 1996). "The SSL Protocol," Version 3.0, Internet Draft 02. Internet Engineering Task Force, < http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt > [18 December 2002].
1316		
1317		
1318		