Liberty Architecture Overview

Version 1.1 -05

25-November 2002

Document Description: draft-liberty-architecture-overview-v1.0-05

Notice

Liberty Alliance Project
Licensing Administrator
c/o IEEE-ISTO
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331, USA

Editor

Jeff Hodges, Sun Microsystems, Inc.

Contributors

The following Liberty Alliance Project Sponsor companies contributed to the development of this specification:

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

52 Revision History
53

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	CR 1107 login via embedded form only "may" reveal users' credentials to SP CR 1103 Argument in line 949 inversed. It says available space in ULR larger than HTML form. CR 1100 Mention "provide non-repudiation" CR 1102 Is Figure 17 supported in Phase1? CR 1101 Added description of this document. Section 1.1. CR 1104 Figure 14 represents double linking instead of simple one. CR 1099 User consent obtained prior to authentication CR 1177 establishing trust relationships in IDP2IDP federation is unspecified
1.1 - 04	15-Nov-02	Tom Wason	CR1217: Added note on authentication state information for principals, Section 5.4.2. CR1218: Added common cookie note to Section 5.5. CR1222: Added note on federation termination with a local session, Section 5.4.1.2 CR1226: User handles note #2 change in Section 5.4.1.
1.1 - 05	25-Nov-02	Tom Wason	CR1238: Inserted title "**Identity Provider Session State Maintenance"** in POLICY/SECURITY NOTE in Section 5.4.2. CR1247: Lowered case of RECOMMENDED, POLICY/SECURITY NOTE in Section 5.4.1.2.

54
55
56

Table of Contents

1 Introduction 6
1.1 About This Document 6
1.2 What is the Liberty Alliance? 6
1.2.1 The Liberty Vision 6
1.2.2 The Liberty Mission 7
1.3 What is Network Identity? 7
1.3.1 The Liberty Objectives 7
2 Liberty Version 1.0 User Experience Examples 9
2.1 Example of Identity Federation User Experience 10
2.2 Example of Single Sign-on User Experience 14
3 Liberty Engineering Requirements Summary 16
3.1 General Requirements 16
3.1.1 Client Device/User Agent Interoperability 16
3.1.2 Openness Requirements 16
3.2 Functional Requirements 16
3.2.1 Identity Federation 17
3.2.2 Authentication 17
3.2.3 Pseudonyms 17
3.2.4 Global Logout 17
4 Liberty Security Framework 17
5 Liberty Architecture 19
5.1 Web Redirection Architectural Component 20
5.1.1 HTTP-Redirect-Based Redirection 21
5.1.2 Form-POST-Based Redirection 22
5.1.3 Cookies 22
5.1.4 Web Redirection Summary 23
5.2 Web Services Architectural Component 23
5.3 Metadata and Schemas Architectural Component 24
5.4 Single Sign-On and Identity Federation 24
5.4.1 Identity Federation 24
5.4.2 Single Sign-on 30
5.4.3 Profiles of the Single Sign-On and Federation Protocol 32
5.5 Identity Provider Introduction 36
5.6 Single Logout 38
5.6.1 Single Logout Profiles 39
5.7 Example User Experience Scenarios 39
5.7.1 Scenario: Not Logged in Anywhere, No Common Domain Cookie 40
5.7.2 Scenario: Not Logged in Anywhere, Has a Common Domain Cookie 44
5.7.3 Scenario: Logged in, Has a Common Domain Cookie 44
6 References 44

1 Introduction

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

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

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

1.1 About This Document

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

1.2 What is the Liberty Alliance?

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

1.2.1 The Liberty Vision

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

1.2.2 The Liberty Mission

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

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

1.3 What is Network Identity?

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

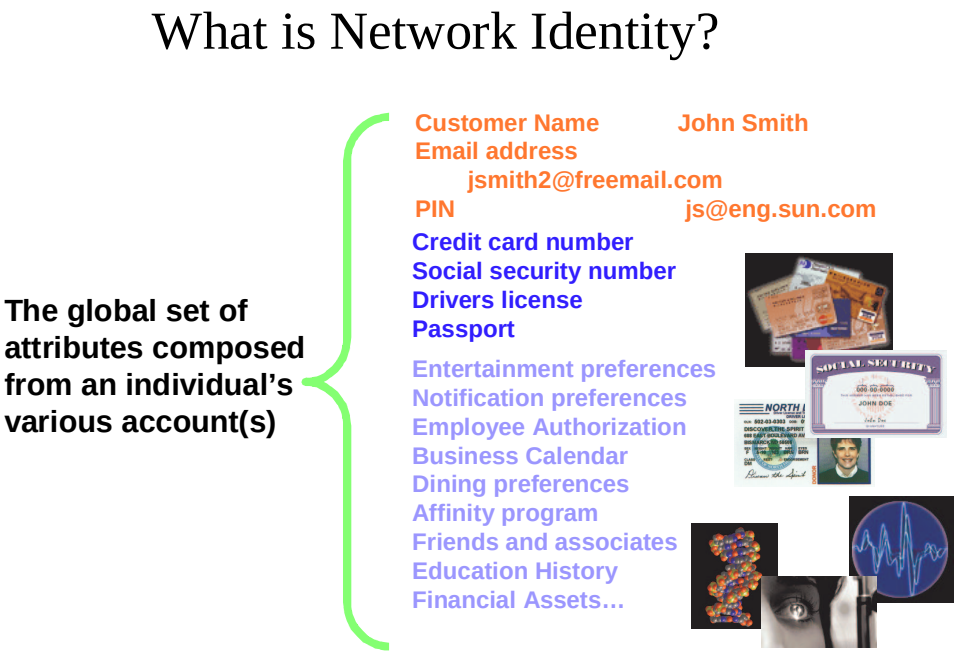


Figure 1: A network identity is the global set of attributes composed from a user's account(s).

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

1.3.1 The Liberty Objectives

The key objectives of the Liberty Alliance are to

- Enable consumers to protect the privacy and security of their network identity information
- Enable businesses to maintain and manage their customer relationships without third-party 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.

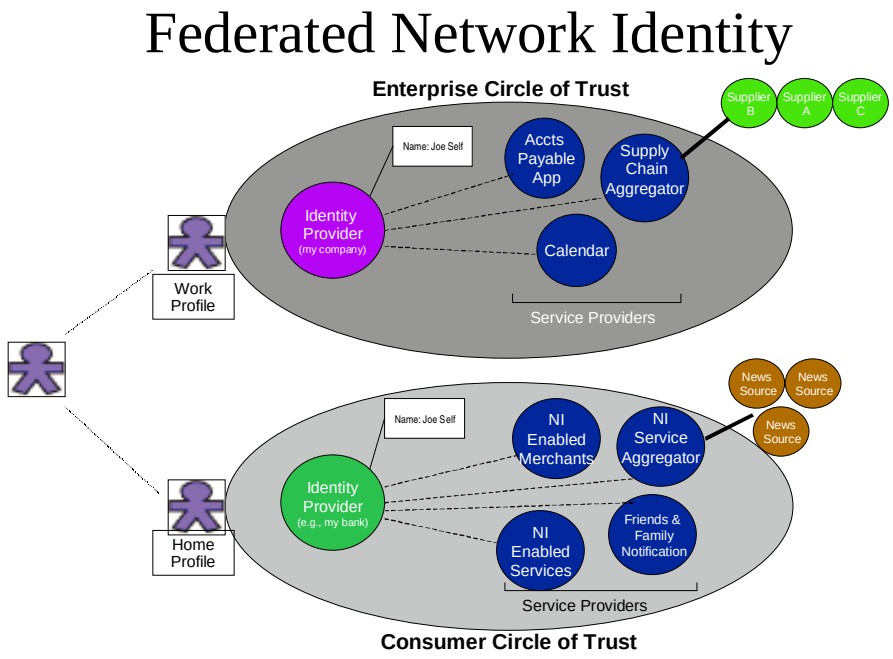


Figure 2: Federated network identity and circles of trust

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

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

2 Liberty Version 1.0 User Experience Examples

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

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

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

- Joe Self — A user of Web-based online services.
- Airline.inc — An airline maintaining an affinity group of partners. Airline.inc is an identity provider.
- CarRental.inc — A car rental company that is a member of the airline's affinity group. CarRental.inc is a service provider.

The Liberty Version 1.0 user experience has two main facets:

- Identity federation
- Single sign-on

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

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

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

2.1 Example of Identity Federation User Experience

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

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

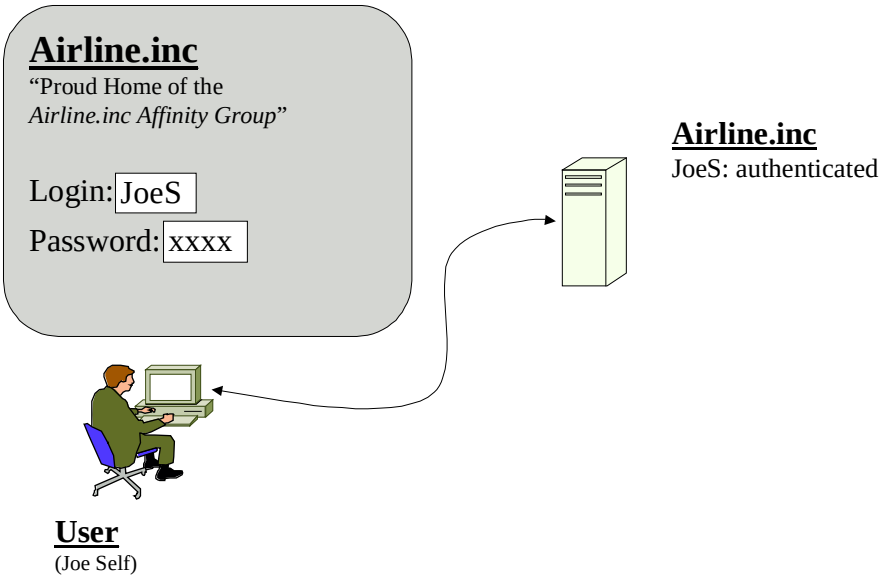


Figure 3: User logs in at a Liberty-enabled Website.

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

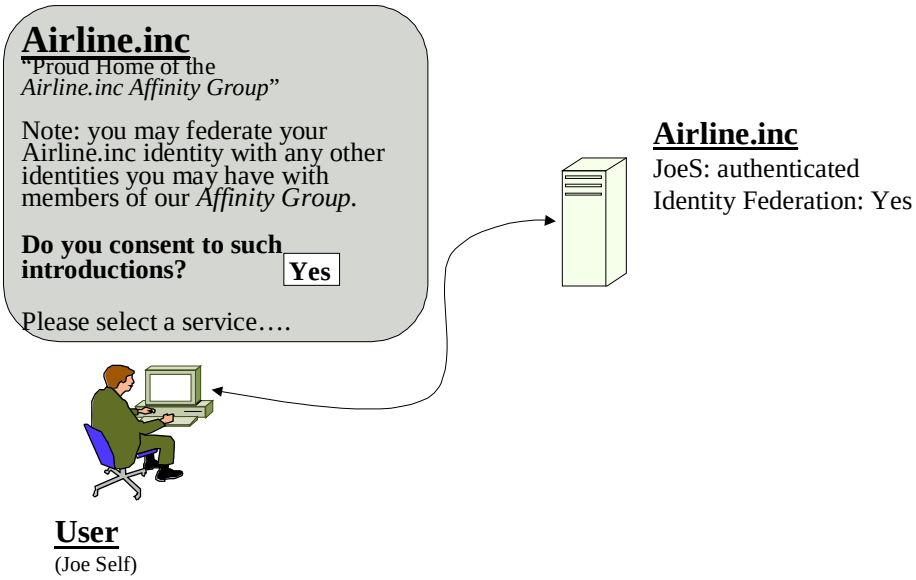


Figure 4: User is notified of eligibility for identity federation and elects to allow introductions.

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

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

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

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

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

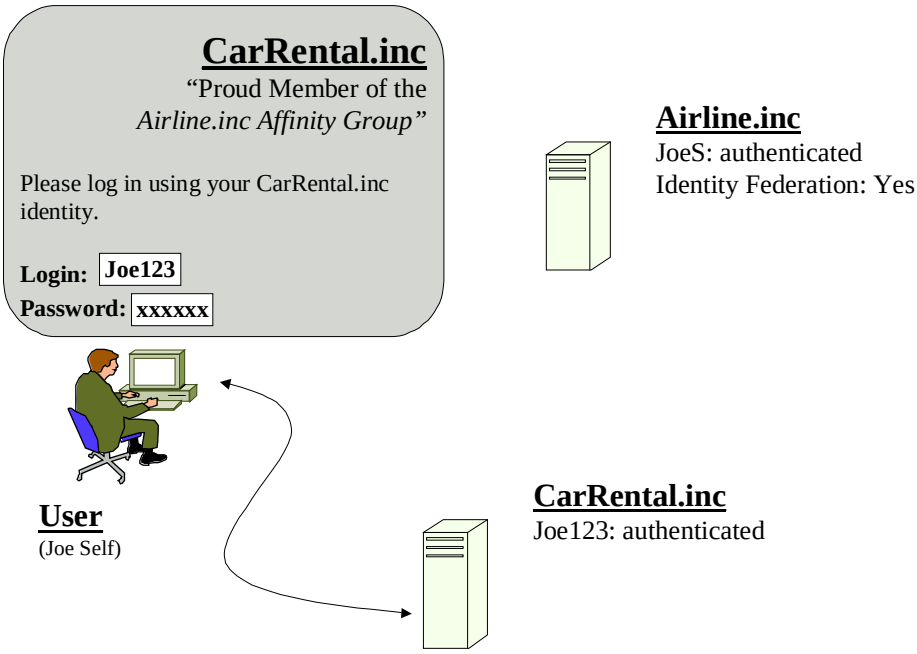


Figure 5: User signs-on using his local service provider identity.

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

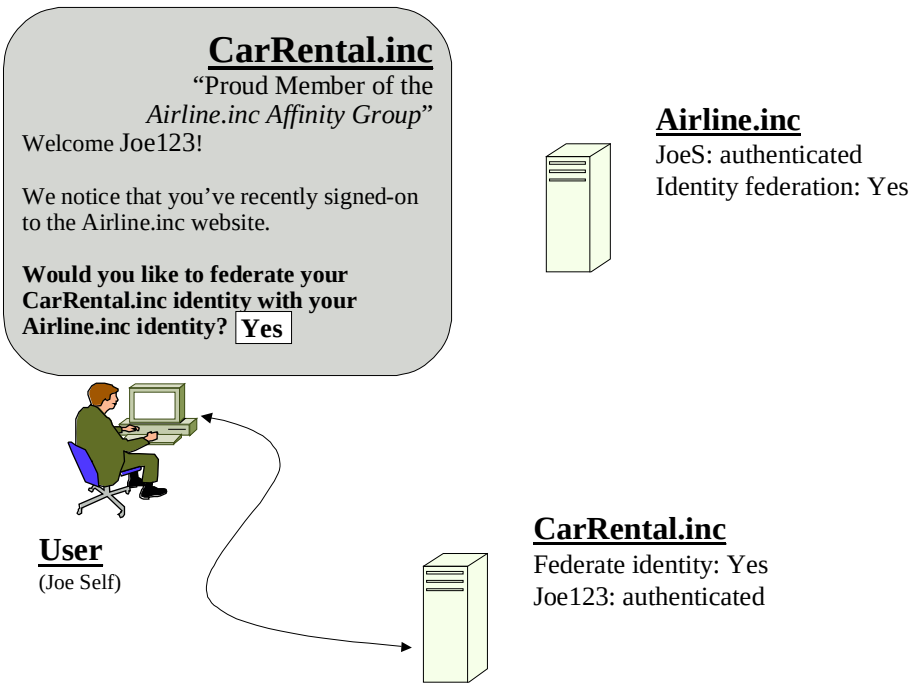
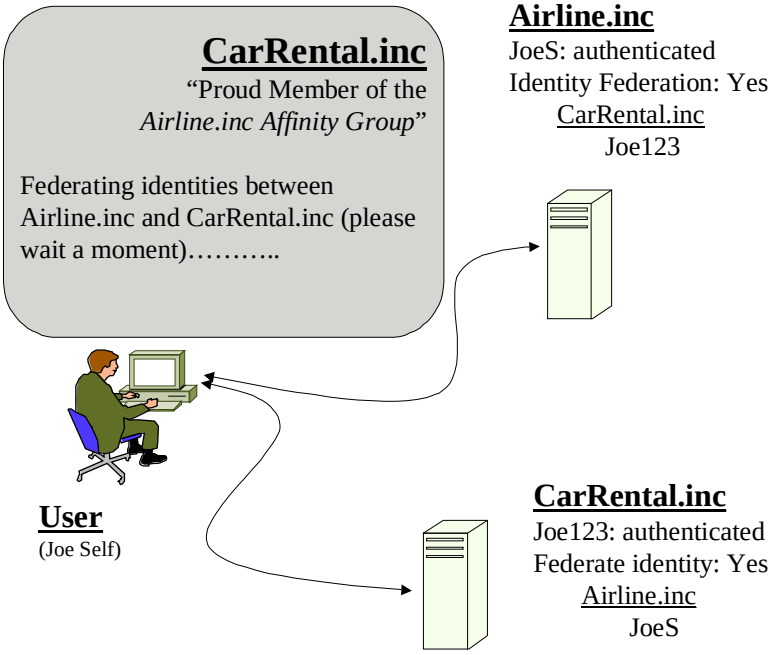


Figure 6: User is prompted to federate his local identities and selects "yes."

295 POLICY/SECURITY NOTE: Whether the service provider asks for consent to federate the user's local
296 identity before or after locally authenticating the user is a matter of local deployment policy.
297
298 As a part of logging in to the CarRental.inc Website, Joe Self's local CarRental.inc identity is
299 federated with his local Airline.inc identity. See Figure 7.
300



301
302 **Figure 7: The Websites federate the user's local identities.**
303
304 Upon completion of the login and identity federation activity, Joe User is logged in to the
305 CarRental.inc Website, and CarRental.inc delivers services to him as usual. In addition, the
306 Website may now offer new selections because Joe Self's local service provider
307 (CarRental.inc) identity has been federated with his local identity provider (Airline.inc)
308 identity. See Figure 8.
309
310 TECHNICAL NOTE: Some figures illustrating the user experience, for example, Figure 7, show
311 simplified, user-perspective notions of how identity federation is effected. In actuality, cleartext
312 identifiers, for example, "JoeS" and "Joe123" WILL NOT be exchanged between the identity provider
313 and service provider. Rather, opaque user handles will be exchanged. See 5.4.1 for details.
314
315 Additionally, if errors are encountered in the process of authenticating and/or federating, the service
316 provider will need to present appropriate indications to the user.
317

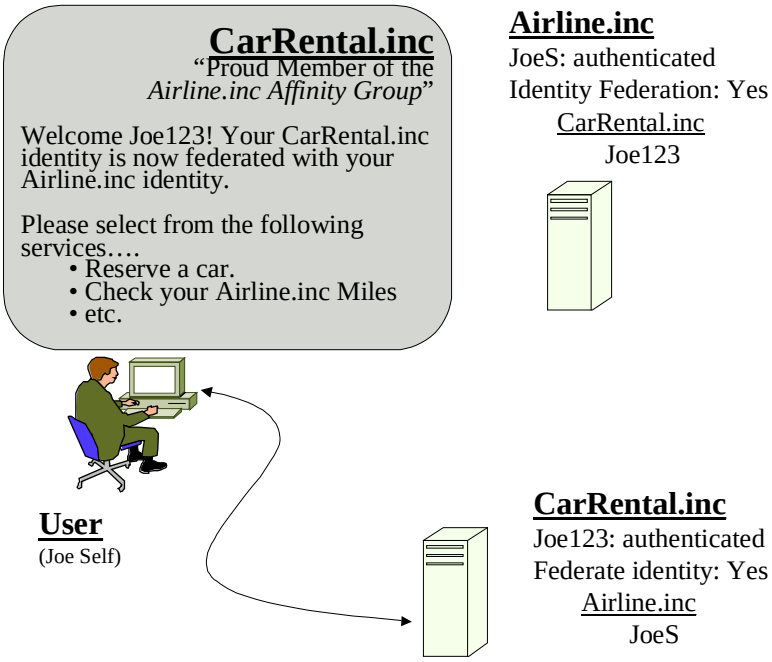


Figure 8: The service provider delivers services to user as usual.

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

2.2 Example of Single Sign-on User Experience

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

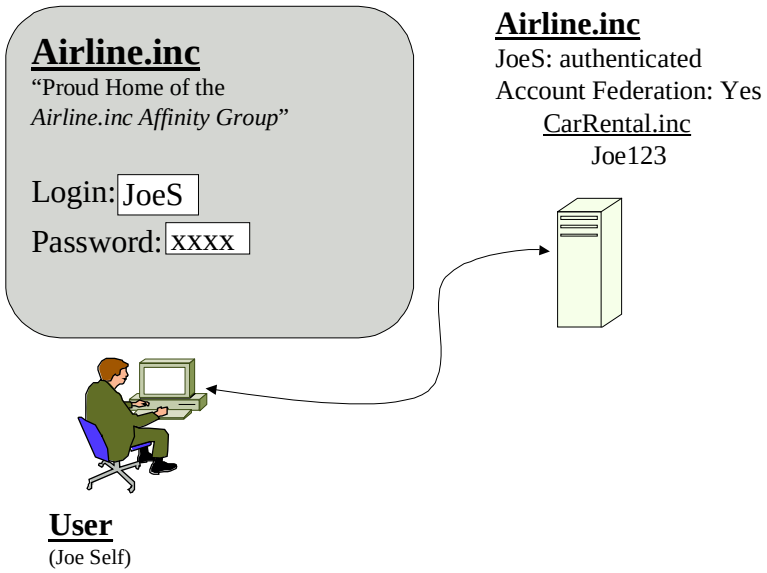


Figure 9: User logs in to identity provider's Website using local identity.

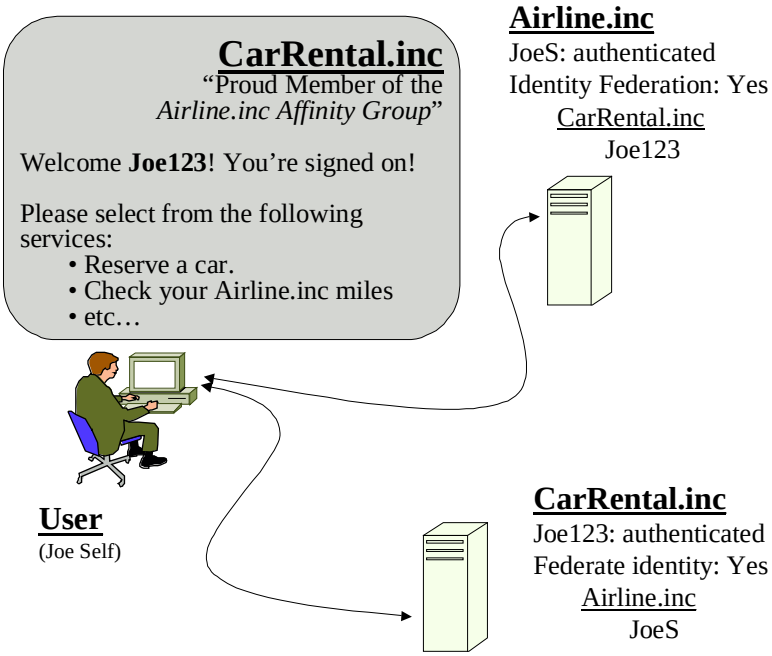


Figure 10: User proceeds to service provider's Website, and his authentication state is reciprocally honored by the service provider's Website.

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.

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

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

3 Liberty Engineering Requirements Summary

This section summarizes the Liberty general and functional engineering requirements.

3.1 General Requirements

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

3.1.1 Client Device/User Agent Interoperability

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

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

3.1.2 Openness Requirements

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

- Operating systems
- Programming languages
- Network infrastructures

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

3.2 Functional Requirements

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

- Identity federation
- Authentication
- Use of pseudonyms
- Global logout

3.2.1 Identity Federation

Requirements of identity federation stipulate that

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

3.2.2 Authentication

Authentication requirements include

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

3.2.3 Pseudonyms

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

3.2.4 Global Logout

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

4 Liberty Security Framework

Table 1 generally summarizes the security mechanisms incorporated in the Liberty specifications, and thus in Liberty-enabled implementations, across two axes: channel security and message security. It also generally summarizes the security-oriented processing requirements placed on Liberty implementations. Note: This section is non-normative, please

refer to [LibertyProtSchema] and [LibertyBindProf] for detailed normative statements regarding security mechanisms.

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

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

Critical points of channel security include the following:

- In terms of authentication, service providers are required to authenticate identity providers using identity provider server-side certificates. Identity providers have the option to require authentication of service providers using service provider client-side certificates.

- Additionally, each service provider is required to be configured with a list of authorized identity providers, and each identity provider is required to be configured with a list of authorized service providers. Thus any service provider-identity provider pair must be mutually authorized before they will engage in Liberty interactions. Such authorization is in addition to authentication. (Note: The format of this configuration is a local matter and could, for example, be represented as lists of names or as sets of X.509 certificates of other circle of trust members).

- The authenticated identity of an identity provider must be presented to a user before the user presents personal authentication data to that identity provider.

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

Critical points of message security include the following:

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

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

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

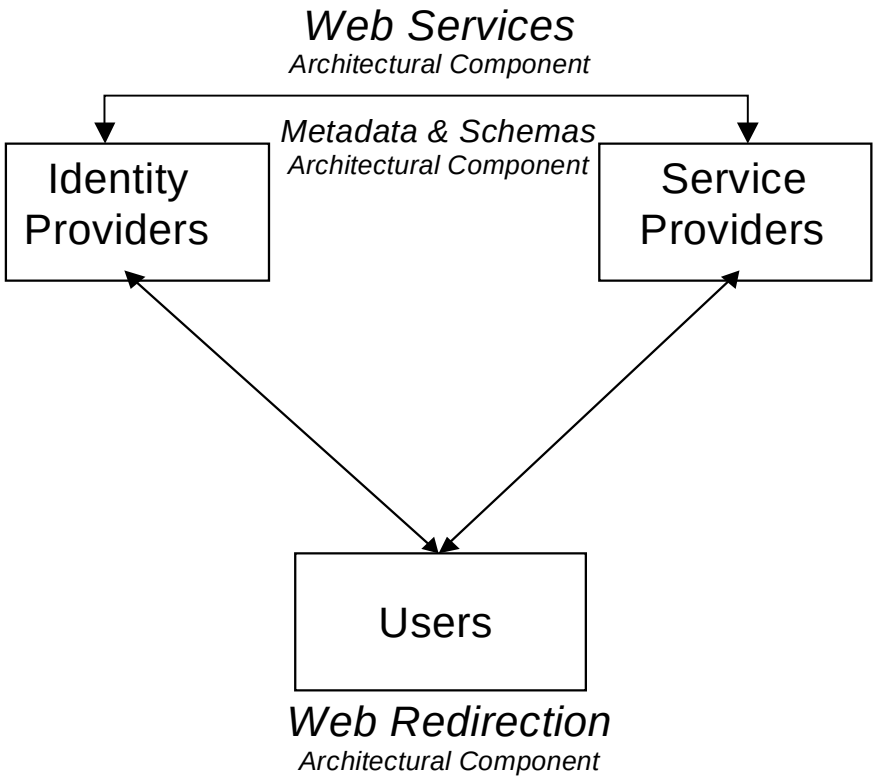
To become circle of trust members, providers are required to establish bilateral agreements on selecting certificate authorities, obtaining X.509 credentials, establishing and managing trusted public keys, and managing life cycles of corresponding credentials.

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

5 Liberty Architecture

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

- Web redirection
- Web services
- Metadata and schemas



503
504 **Figure 11: Overall Liberty architecture**

505
506 The role of each architectural component is summarized in Table 2:
507

508 **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.

509
510 Sections 5.1 through 5.3 describe each architectural component. Sections 5.4 through 5.6 then
511 relate the architectural components to the concrete protocols and profiles detailed in
512 [LibertyProtSchema] and [LibertyBindProf], and 5.7 provides illustrations of user experience.

513 5.1 Web Redirection Architectural Component

514 The Web redirection architectural component is composed of two generic variants: HTTP-
515 redirect-based redirection and form-POST-based redirection. Both variants create a
516 communication channel between identity providers and service providers that is rooted in the
517 user agent. See Figure 12.
518

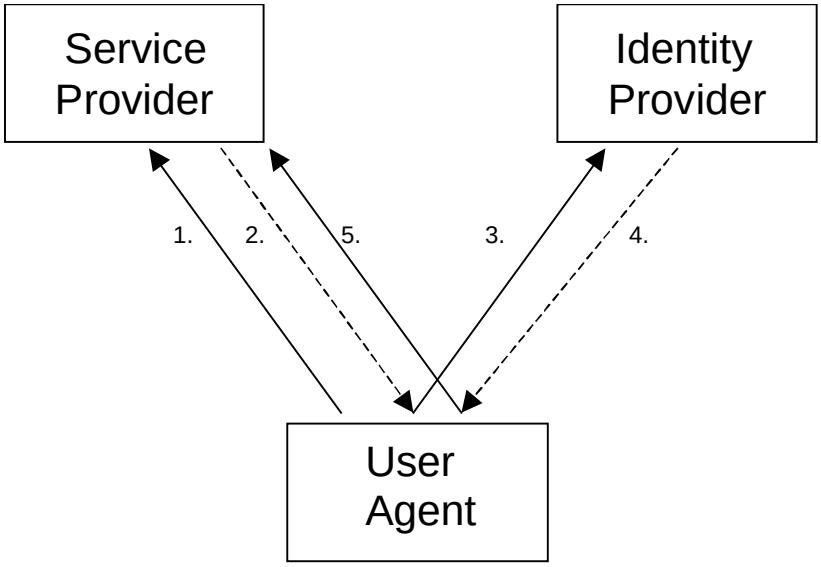


Figure 12: Web redirection between a service provider and an identity provider via the user agent

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

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

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"

5.1.2 Form-POST-Based Redirection

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

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

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

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

5.1.3 Cookies

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

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

5.1.3.1 Why Not Use Cookies in General?

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

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

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

5.1.3.2 Where Cookies are Used

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

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

5.1.4 Web Redirection Summary

Web redirection is not an ideal distributed systems architecture.

> POLICY/SECURITY NOTE: Communications across Web redirection channels as described in 5.1.1 through 5.1.3 have many well-documented security vulnerabilities, which should be given careful consideration when designing protocols utilizing Web redirection. Such consideration was incorporated into the design of the profiles specified in [LibertyBindProf], and specific considerations are called out as appropriate in that document (for example, regarding cleartext transmissions and caching vulnerabilities). Examples of security vulnerabilities include
>
> - **Interception:** Such communications go across the wire in cleartext unless all the steps in 5.1.1 through 5.1.3 are carried out over an SSL or TLS session or across another secured communication transport, for example, an IPsec-based VPN.
> - **User agent leakage:** Because the channel is redirected through the user agent, many opportunities arise for the information to be cached in the user agent and revealed later. This caching is possible even if a secure transport is used because the conveyed information is kept in the clear in the browser. Thus any sensitive information conveyed in this fashion needs to be encrypted on its own before being sent across the channel.
>
> TECHNICAL NOTE: A key limitation of Web redirection is the overall size of URIs passed as arguments of GET requests and as values of the Location field in redirects. These elements have size limitations that vary from browser to browser and are particularly small in some mobile handsets. These limitations were incorporated into the design of the protocols specified in [LibertyProtSchema] and [LibertyBindProf].

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

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

5.2 Web Services Architectural Component

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

5.3 Metadata and Schemas Architectural Component

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

- **Account/Identity:** In Liberty Version 1.0, account/identity is simply the opaque user handle that serves as the name that the service provider and the identity provider use in referring to the user when communicating. In future Liberty phases, it will encompass various attributes.

- **Authentication Context:** Liberty explicitly accommodates identity provider use of arbitrary authentication mechanisms and technologies. Different identity providers will choose different technologies, follow different processes, and be bound by different legal obligations with respect to how they authenticate users. The choices that an identity provider makes here will be driven in large part by the requirements of the service providers with which the identity provider has federated. Those requirements, in turn, will be determined by the nature of the service (that is, the sensitivity of any information exchanged, the associated financial value, the service providers risk tolerance, etc) that the service provider will be providing to the user. Consequently, for anything other than trivial services, if the service provider is to place sufficient confidence in the authentication assertions it receives from an identity provider, the service provider must know which technologies, protocols, and processes were used or followed for the original authentication mechanism on which the authentication assertion is based. The authentication context schema provides a means for service providers and identity providers to communicate such information (see [LibertyAuthnContext]).

- **Provider Metadata:** For identity providers and service providers to communicate with each other, they must a priori have obtained metadata regarding each other. These provider metadata include items such as X.509 certificates and service endpoints. [LibertyProtSchema] defines metadata schemas for identity providers and service providers that may be used for provider metadata exchange. However, provider metadata exchange protocols are outside the scope of the Liberty Version 1.0 specifications.

5.4 Single Sign-On and Identity Federation

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

5.4.1 Identity Federation

The first time that users use an identity provider to log in to a service provider they must be given the option of federating an existing local identity on the service provider with the identity provider login to preserve existing information under the single sign-on. See Figure 13. It is critical that, in a system with multiple identity providers and service providers, a

mechanism exists by which users can be (at their discretion) uniquely identified across the providers. However, it is technically challenging to create a globally unique ID that is not tied to a particular identity provider and a business challenge to ensure the portability of globally unique IDs.

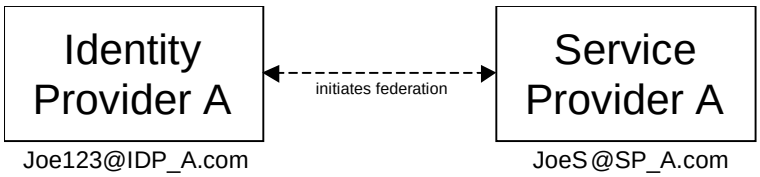


Figure 13: User initiates federation of two identities

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

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

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

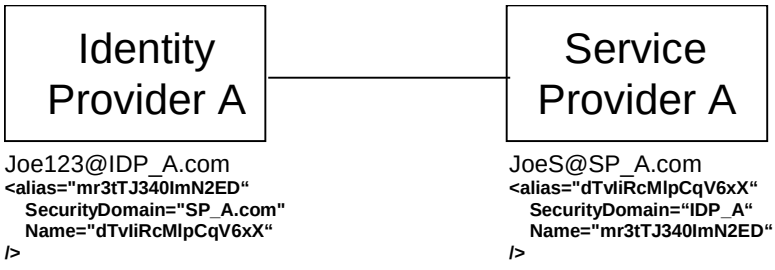


Figure 14: User directories of the identity provider and service provider upon identity federation

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

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

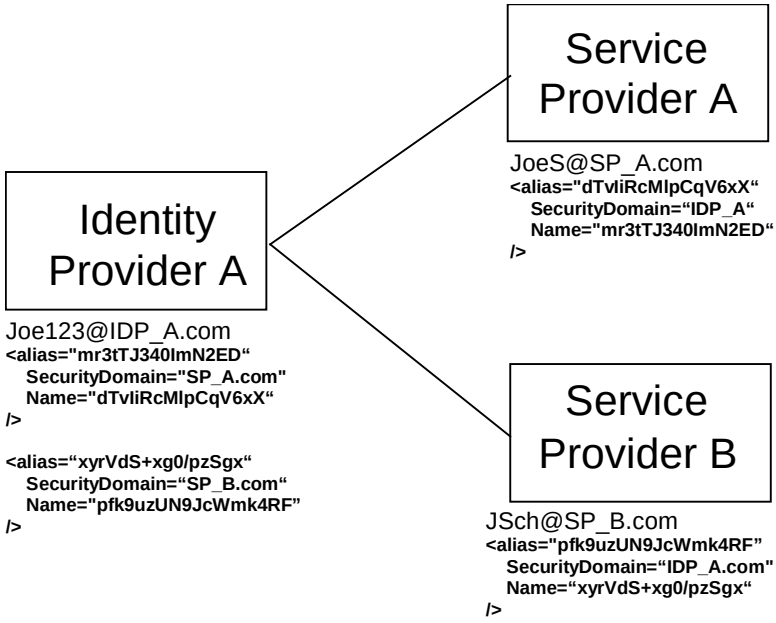


Figure 15: User directories of the identity provider and multiple service providers upon identity federation

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, **mr3tTJ340lmN2ED**, (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 provides.

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.

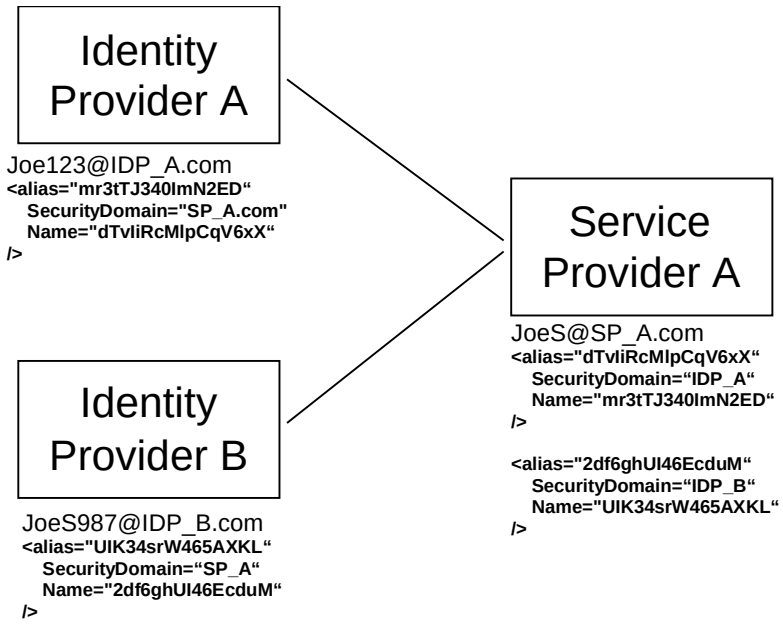


Figure 16: A user with two identity providers federated to a service provider

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

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

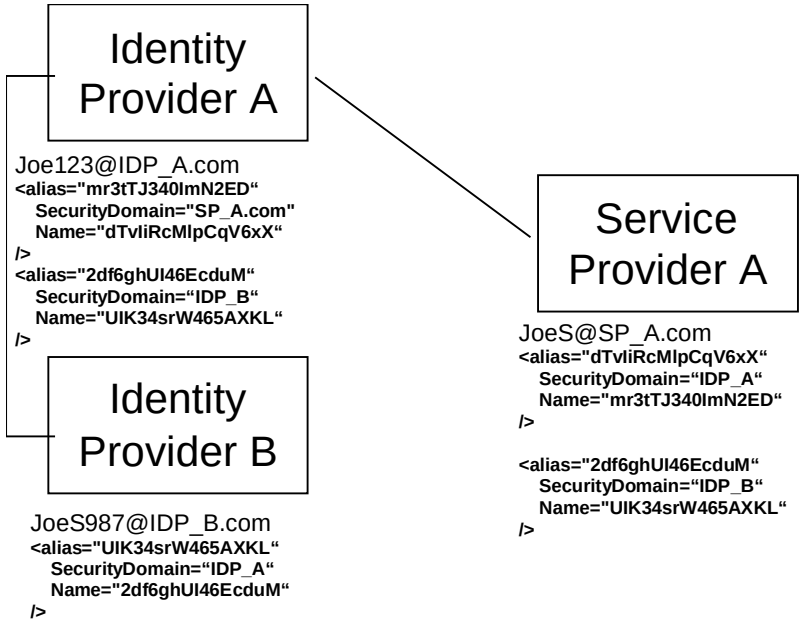


Figure 17: A user with two identity providers federated

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

POLICY/SECURITY NOTE:

1. The semantics of such a federated relationship (Figure 17) between identity providers are not dictated by the underlying Liberty protocols, nor are they precluded. These semantics need to be addressed by the agreements between the identity providers and supported by the capabilities of the deployed Liberty-enabled implementations.

2. Additionally, how trust relationships between identity providers are established, and how those relationships are represented to service providers, are unspecified. Identity providers enabling relationships such as that illustrated in Figure 17 must mutually define governing policies and means of representing such trust relationships to relying service providers (for example Service Provider A in Figure 17).

3. Circle of trust agreements should address how federation failures are materialized to users.

4. Appropriate portions of the assertions passed between the identity provider and the service provider to effect federation should be logged.

5. By creating many local identities with many service providers and/or identity providers and then federating them, users possess many sets of local credentials that may be used as a basis to authenticate with many service providers via single sign-on. This situation constitutes a risk. For example, every identity provider that possesses reusable user credentials, for example, a username and password, can impersonate the user at every service provider federated with that account.

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

5.4.1.1 No Need for Global Account/Identity Namespace

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

5.4.1.2 Federation Management: Defederation

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

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

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

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

- The user should be authenticated by the provider at which identity defederation is being initiated.

- Providers should ask the user for confirmation before performing defederation and appropriately log the event and appropriate portions of the user's authentication information.

- It is recommended that the service provider, after initiating or receiving a federation termination notification for a Principal, check whether that Principal is presently logged in to the service provider on the basis of an assertion from the identity provider with which the federation termination notification was exchanged. If so, then the local session information that was based on the identity provider"s assertion should be invalidated.

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

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

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

5.4.2 Single Sign-on

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

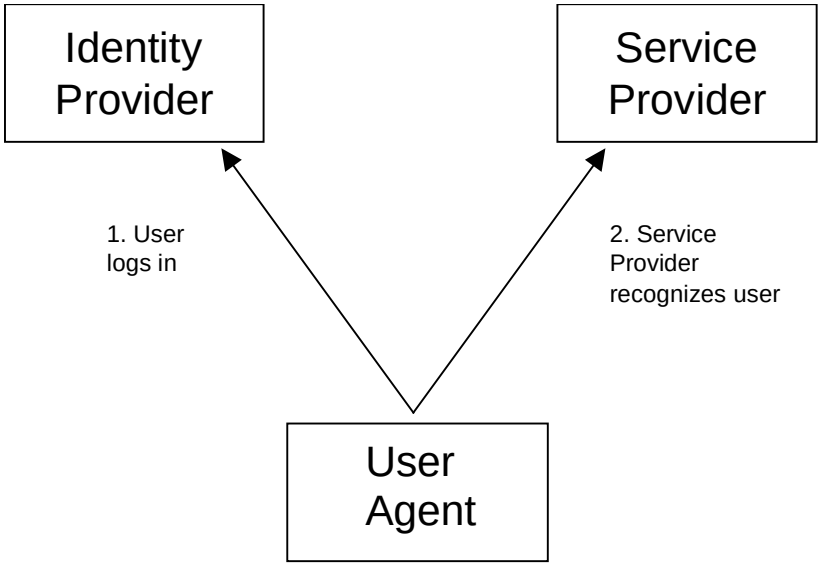


Figure 18: User logs in at identity provider and is recognized by service provider

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

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

Authentication Mechanisms are Orthogonal to Single Sign-On

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

Identity Provider Session State Maintenance

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

provider an <AuthnResponse> containing an authentication assertion if its local session state information indicates the user agent's session with the identity provider is presently active.

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.

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

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

Mutual Authentication

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

Validating Liveness

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

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

Communication Security

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

5.4.3 Profiles of the Single Sign-On and Federation Protocol

The Single Sign-On and Federation Protocol, as specified in [LibertyProtSchema], defines messages exchanged between service providers and identity providers. The concrete mapping of these messages to particular transfer (for example, HTTP) and/or messaging (for example,

SOAP) protocols and precise protocol flows are specified in [LibertyBindProf]. These mappings are called *profiles*. The Single Sign-On and Federation Protocol specifies four profiles. The following sections summarize each profile. For a detailed discussion of the common interactions and processing rules of these profiles and for details about each profile, see [LibertyBindProf].

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

5.4.3.1 Liberty Browser Artifact Profile

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

5.4.3.2 Liberty Browser POST Profile

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

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 accomodate one.. See Figure 19.

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

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

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

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

5.4.3.3 Liberty WML POST Profile

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

TECHNICAL NOTE: The primary difference between this profile and the Liberty browser POST profile is that certain responses from the service provider and identity provider to the user agent contain WML rather than HTML.

The difference between this profile and the Liberty-enabled client and proxy profile is that this profile is designed to accommodate standard, unmodified WML browsers, while the Liberty-enabled client and proxy profile assumes a browser and/or proxy with built-in Liberty protocol capabilities.

5.4.3.4 Liberty-Enabled Client and Proxy Profile

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

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

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

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

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

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

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

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

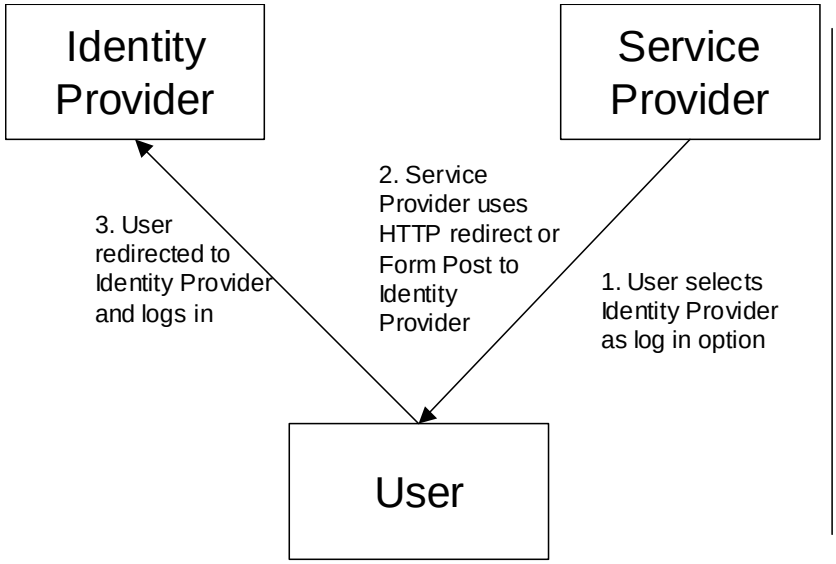


Figure 20: Single sign-on using HTTP redirect / form POST (1 of 2)

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

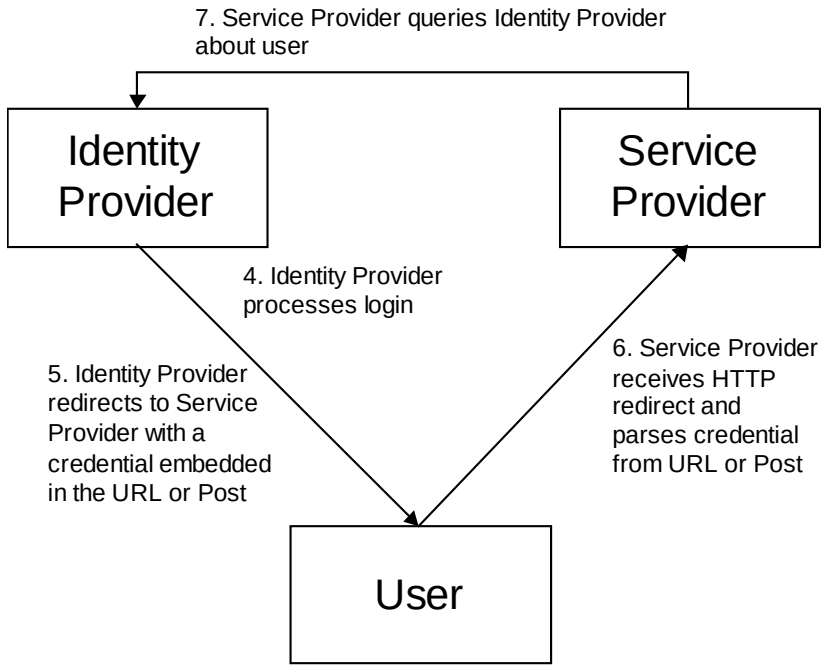


Figure 21: Single sign-on using HTTP redirect / form POST (2 of 2)

5.5 Identity Provider Introduction

In circle of trusts having more than one identity provider, service providers need a means to discover which identity providers a user is using. Ideally, an identity provider could write a cookie that a service provider could read. However, due to the cookie constraint outlined in 5.1.3, an identity provider in one DNS domain has no standardized way to write a cookie that a service provider in another DNS domain can read.

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

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

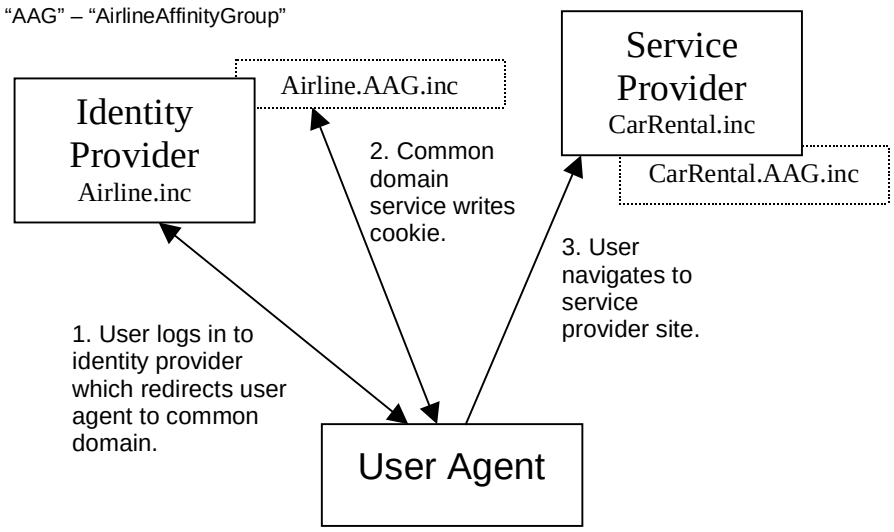


Figure 22: Using a common domain to facilitate introductions (1 of 2)

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

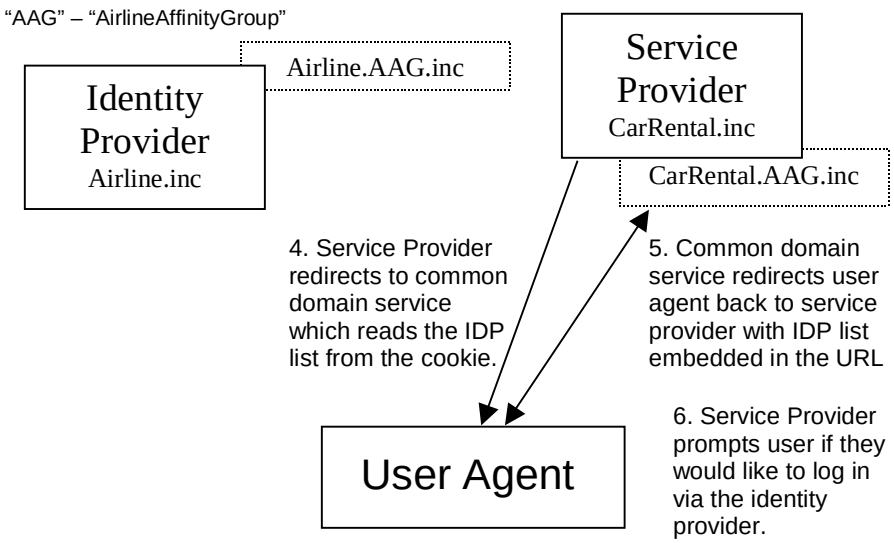


Figure 23: Using a common domain to facilitate introductions (2 of 2)

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).

5.6 Single Logout

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

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

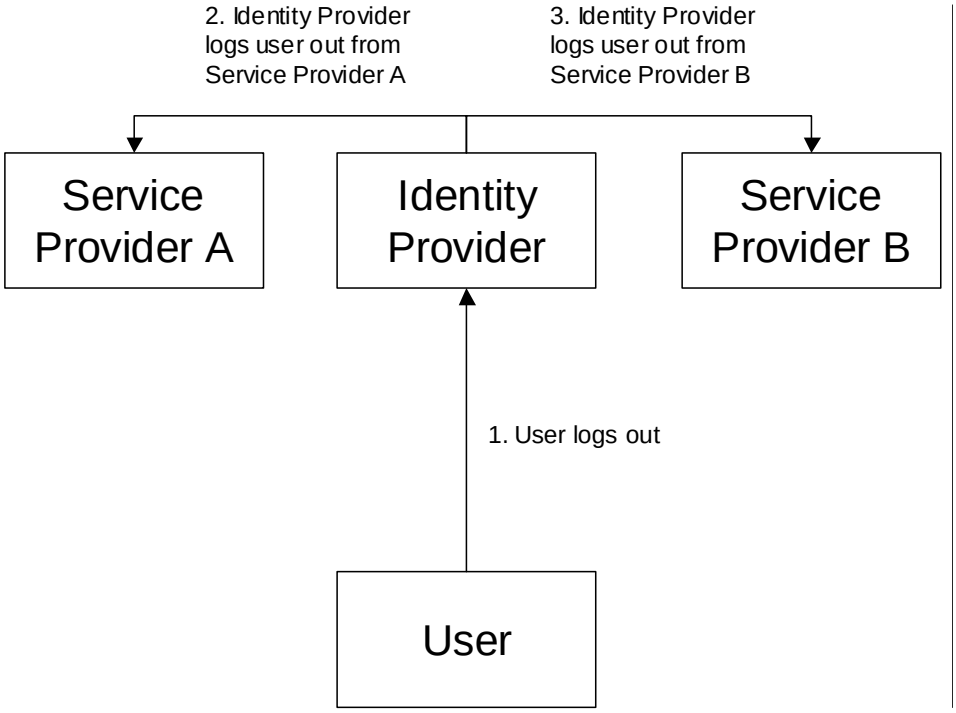


Figure 24: Single logout from an identity provider

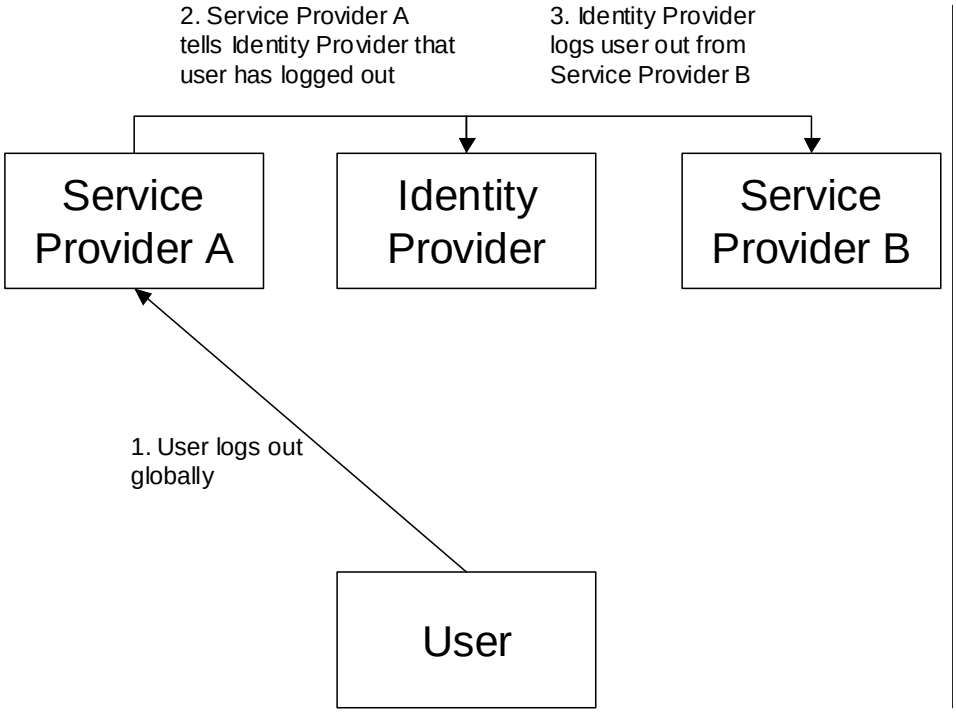


Figure 25: Single logout from a service provider

5.6.1 Single Logout Profiles

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

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

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

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

5.7 Example User Experience Scenarios

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

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

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

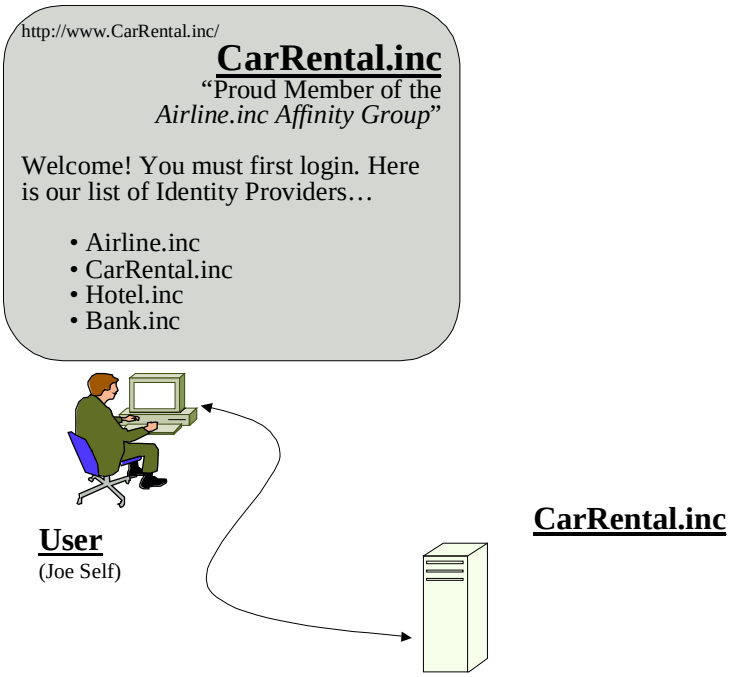


Figure 26: User arrives at service provider's Website without any authentication evidence or common domain cookie

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

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

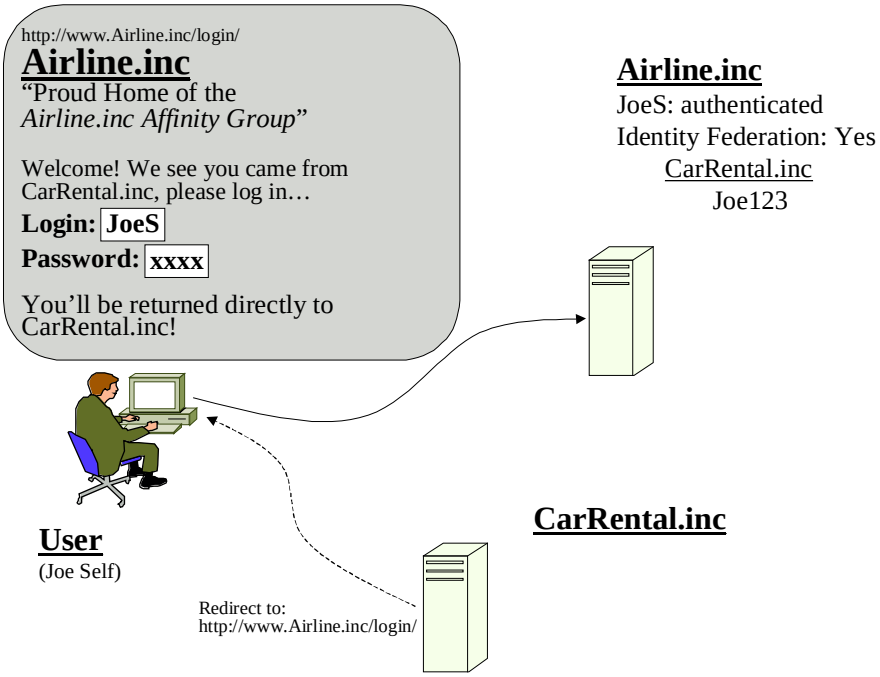
- Redirect to identity provider Website
- Identity provider dialog box
- Embedded form

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

5.7.1.1 Login via Redirect to Identity Provider Website

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

1253



1254

1255 **Figure 27: Service provider redirects to identity provider's login page.**

1256

1257 POLICY/SECURITY NOTE: Login via redirect to the identity provider's Website is relatively secure in
1258 that the user reveals his credentials directly to the identity provider. Of course, the usual security
1259 considerations surrounding login and authentication events apply.

1260 ### 5.7.1.2 Login via Identity Provider Dialog Box

1261 With login via a dialog box from the identity provider, the links on the service provider's
1262 Webpage invoke a dialog or popup box. Joe Self's browser will display an identity provider
1263 popup (see Figure 28); and upon successful login, the popup box will close, and Joe Self will
1264 be provided access at the service provider's Website (see Figure 30).
1265

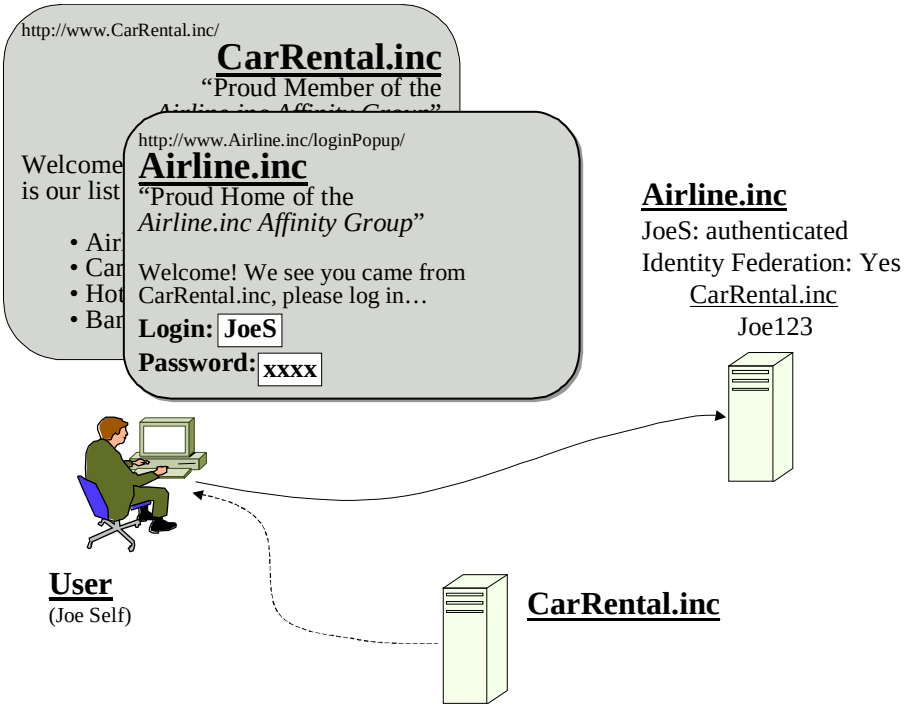


Figure 28: Service provider invokes dialog or popup box from identity provider.

POLICY/SECURITY NOTE: Login via a dialog box from the identity provider 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.

5.7.1.3 Login via Embedded Form

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

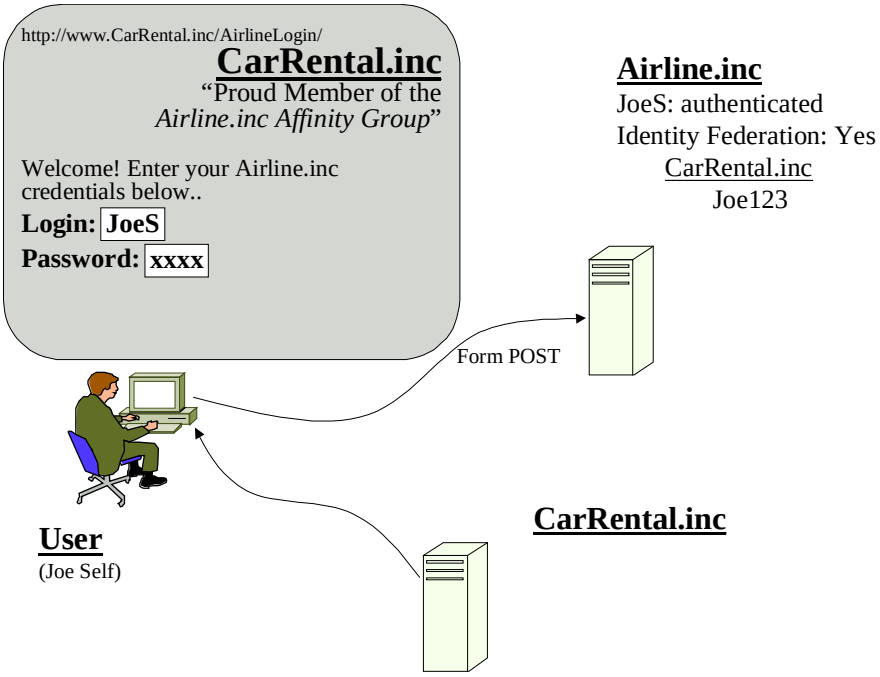


Figure 29: Login via embedded form

.

[1107] 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.

5.7.1.4 The User is Logged in at CarRental.inc

CarRental.inc and Airline.inc then work in conjunction to effect login, and the CarRental.inc Website establishes a session based upon Joe Self's identity federation with Airline.inc (see Figure 30).

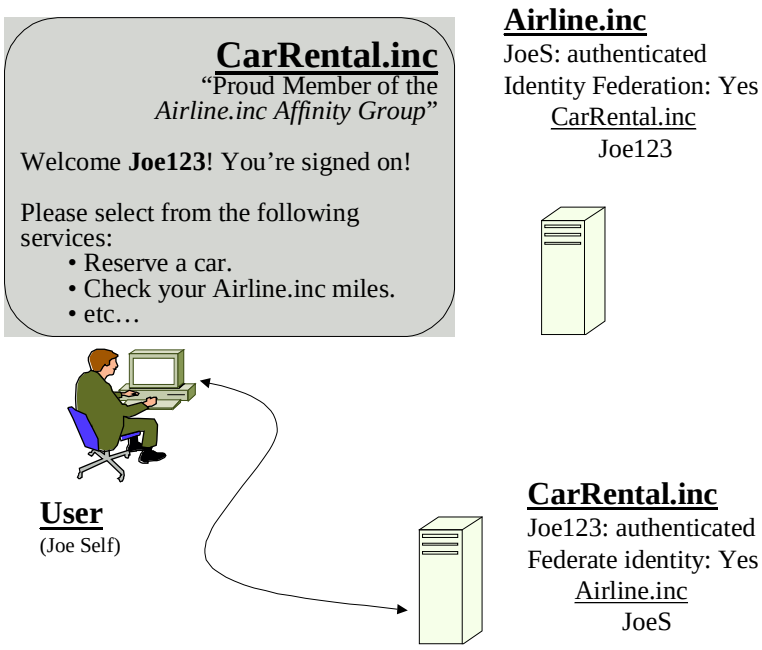


Figure 30: Service provider's Website delivers services on basis of federated identity.

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., "Liberty Architecture Implementation Guidelines."

[LibertyAuthnContext] Madsen, P., "Liberty Authentication Context Specification."

[LibertyBindProf] Rouault, J., "Liberty Bindings and Profiles Specification."

[LibertyGloss] Mauldin, H., "Liberty Glossary."

[LibertyProtSchema] Beatty, J., "Liberty Protocols and Schemas Specification."

[RFC1738] "Uniform Resource Locators (URL)," http://www.ietf.org/rfc/rfc1738.txt

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[RFC2246] "The TLS Protocol Version 1.0," http://www.ietf.org/rfcs/rfc2246.html.

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax," http://www.ietf.org/rfc/rfc2396.txt.

[RFC2616] "Hypertext Transfer Protocol — HTTP/1.1," http://www.ietf.org/rfc/rfc2616.txt.

[RFC2617] "HTTP Authentication," http://www.ietf.org/rfc/rfc2617.txt.

[RFC2965] "HTTP State Management Mechanism," http://www.ietf.org/rfc/rfc2965.txt.

[SAMLBind] P. Mishra et al., "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)," http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-11.pdf, OASIS, January 2002.

[SOAP1.1] D. Box et al., "Simple Object Access Protocol (SOAP) 1.1," http://www.w3.org/TR/SOAP, World Wide Web Consortium Note, May 2000.

[SSLv3] "The SSL Protocol Version 3.0," http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt.