Liberty ID-FF Implementation Guidelines

Version: 1.2

Editors:
Peter Thompson, IEEE-ISTO
Darryl Champagne, IEEE-ISTO

Contributors:
John Kemp, IEEE-ISTO
Robert Aarts, Nokia Corporation
Nick Bone, Vodafone Group Plc
David Castellanos-Zamora, Ericsson
Jean-Michel Crom, France Telecom
Lena Kannappan, France Telecom
Andrew Lindsay-Stewart, Vodafone Group Plc
Kenichi Maeda, NTT DoCoMo
Mike Meyerstein, Vodafone Group Plc
Alain Nochimowski, France Telecom - Orange SA
Alfredo Gonzalez, Ericsson
Alain Poignet, France Telecom
Xavier Serret, Gemplus SA
James Vanderbeek, Vodafone Group Plc
Juliette Vittu, France Telecom
Alex Walter, Vodafone Group Plc
Jonathan Sergent, Sun Microsystems, Inc.
Paul Madsen, Entrust, Inc.
Conor Cahill, AOL Time Warner, Inc.
John Linn, RSA Security Inc.
Susan Landau, Sun Microsystems, Inc.
Paule Sibieta, France Telecom

Abstract:

This document defines some recommended implementation guidelines for implementors of Liberty-based services.

Filename: liberty-idff-guidelines-v1.2.pdf

Notice

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

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

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

Contents

1. Introduction

This document is non-normative. It provides implementers and deployers specific guidance around the usage of the Liberty Identity Federation Framework (ID-FF) specifications. Although the Liberty specifications provide a basis for interoperability between implementations, it is expected that implementers and deployers will make varied choices with respect to authentication methods, session management and other components of a Liberty implementation. Some of these choices will be dictated by the environment within which a particular implementation may operate (for example, mobile network infrastructure, or an enterprise software environment).

1.1. Recommended knowledge

It is assumed that the reader is familiar with the concept of *Identity Federation*. In addition, it is recommended that implementers should have general familiarity both with the Liberty ID-FF specifications [LibertyProtSchema], [LibertyBindProf], and [LibertyAuthnContext], in addition to SAML ([SAMLCore11]), and the general protocols and systems involved in constructing Internet-based software systems, such as [XML], [SOAPv1.1], SSL/TLS ([SSL], [RFC2246]), and HTTP ([RFC2616]) to name but a few. A recommended reading list is provided at the end of this document (Bibliography).

Additional developer resources may be found on the Liberty Alliance website (see [LibertyDeveloperArea]).

1.2. Liberty architecture

The Liberty ID_FF architecture consists of *service providers* and *identity providers*. In addition, there may be *active intermediaries* present in certain Liberty environments. [LibertyIDFFOverview] explains the basic relationships between these entities. A Liberty-enabled implementation may consist of service providers, identity providers, or a combination. Liberty system entities may be implemented by supporting particular protocols and profiles specified in the Liberty specification set. For example, an identity provider would be required to support the Single-Signon protocol, and one or more profiles that use that protocol. The requirements for protocol and profile support, pertaining to Liberty providers are detailed in [LibertyIDFF12SCR]. Active intermediaries, such as a *Liberty-Enabled Client* or *Liberty-Enabled Proxy* may also be present in a Liberty architecture

For a comprehensive review of the Liberty ID-FF architecture, the reader is directed to [LibertyIDFFOverview].

1.2.1. Liberty environments

Liberty implementations may be designed to operate within specific environments. For example, a Liberty single-signon operation may be conducted differently if it is initiated from a mobile phone rather than from a personal computer. In such a case, the architecture implemented using the Liberty specifications may be quite different than that used to enable single-signon for employees to their corporate network and varied internal company websites. Some environments for which particular Liberty implementations may exist include e-commerce, mobile phone networks, and enterprise network infrastructure. Specific guidelines are presented for some of these environments.

2. General guidance

2.1. Cookies and other user agent considerations

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.

2.1.1. 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.

It should be noted that it is not uncommon for users and/or their organizations to operate their user agents with cookies turned off. Additionally, some mobile phones may not even be capable of accepting cookies.

2.1.2. 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 Common Domain Cookie in [LibertyBindProf]).

Note that use of session or permanent cookies for identity provider introduction should be based on the circle of trust policy, and should be consistent within the common domain.

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.

2.1.3. Other possible user-agent constraints

URL Encoding

A cookie should be URL-encoded (on-the-wire), but no more than once. When receiving a cookie make sure that the cookie is URL-encoded prior to URL-decoding. This is important since it is not possible to URL-decode a cookie more times than it has been URL-encoded, and older implementations may not URL-encode a cookie.

A URL-encoded cookie can be identified by searching the cookie data for a '%'. The reason this check can work is that the unencoded cookie data is a space separated list of base64-encoded SHA-1 hashes. The unencoded data will never have a '%' (that character is not in the base64 output), and will always have a '=' (because SHA-1 hashes are 20 bytes long, and the base64 output of 20 bytes always ends in '='). As a result, the encoded string will always contain the '%' since the '=' character gets escaped with a '%'.

The URL-encoding format is defined in [RFC2854], while the URL specification is defined in [RFC1738].

URL length considerations

Some user-agents are restricted in the size of URL they will support. Implementors should take care to inspect the HTTP *User-Agent* header which they may use to check whether that user-agent cannot support URLs that they construct.

It should be noted that the *User-Agent* header may not accurately describe the user-agent being used to access the service, so making decisions based on the header contents should be carefully considered.

One alternative to cookie-based session management (see above) is to embed a session token in all URLs provided by a service. This may not be an option in environments where URL-limited user-agents may access the service.

If a service detects that a user-agent cannot handle a particular URL using an HTTP GET, then the service may use an HTTP POST to send the message (see [LibertyBindProf] for details).

It should also be noted that a Liberty authentication request may be constructed (by omitting optional elements in favor of default behavior) in such a way as to be acceptable to *most* user agents.

This particular issue will be of more concern in some environments than others. There is further discussion of this in Section 5.2.3 as it relates to a WAP 2.0 proxying mobile scenario.

Use of ECMAScript

In some environments, a Principal may be using a user-agent that is not able (or willing) to run ECMAscript. If implementers are working in such environments, they should be aware that content containing such scripts may not operate optimally.

2.2. Liberty & SAML

The Liberty core protocols are built on top of the Security Assertion Markup Language (SAML - see [SAMLCore11]). Implementers should note the following considerations:

- Liberty and SAML define version numbers for the messages used in their protocols. The `MajorVersion` and `MinorVersion` attributes carried on messages describe their respective version numbers.
  Implementers should take care to ensure that Liberty-defined messages carry a *Liberty* version number, and that pure SAML messages carry a SAML version number ([LibertyProtSchema] defines the current Liberty `MajorVersion` and `MinorVersion` values).

- When a Liberty authentication response includes an `Assertion`, an `InResponseTo` attribute allows correlation to the originally-issued authentication request. However, the `Assertion` is contained in a `samlp:Response` which *also* contains an `InResponseTo` attribute.
  The value of the `InResponseTo` attribute for the `Assertion` should correlate to the value of the `RequestID` of the original `AuthnRequest`, whereas the value of the `InResponseTo` attribute for the `samlp:Response` should correlate to the value of the `RequestID` of the original `samlp:Request`.

2.2.1. Use of SAML AttributeStatement in an Authentication Response

The Liberty ID-FF single-signon protocol utilizes SAML (see [SAMLCore11] assertions to carry authentication information (in the form of AuthenticationStatement elements). SAML assertions may also carry other statements (see [SAMLCore11] and section 2.4 for more information). Although this functionality is not used directly in the Liberty ID-FF specifications, it is possible that such statements may appear in authentication responses. In particular, a SAML AttributeStatement may be used to convey information used to discover a Principal's discovery service (see [LibertyDisco] section 6). An example of such an interaction is as follows:

UA = User Agent

A = service provider a.com

B = identity provider b.com

UA -> A:

```
GET /some-resource HTTP/1.0
Host: a.com
<...>
```

A -> UA:

```
HTTP/1.0 302 Found
Location: http://b.com/ authn?RequestID=RMvY34pg%2FV9aGJ5yw0HL0Ac
jcqQF&MajorVersion=1&MinorVersion=2&IssueInstant=2002-05-15T00%3A58%3A19&ProviderID=http%3
A%2F%2Fa.com%2Fliberty%2F&ForceAuthn=true&IsPassive=false&NameIDPolicy=federated&ProtocolP
rofile=http%3A%2F%2Fprojectliberty.org%2Fprofiles%2Fbrws-art
<...>
```

UA -> B:

```
GET / authn?RequestID=RMvY34pg%2FV9aGJ5yw0HL0AcjcqQF&MajorVersion=1&Mino
rVersion=2&IssueInstant=2002-05-15T00%3A58%3A19&ProviderID=http%3A%2F%2Fa.com%2Fliberty%2F
&ForceAuthn=true&IsPassive=false&NameIDPolicy=federated&ProtocolProfile=http%3A%2F%2Fproje
ctliberty.org%2Fprofiles%2Fbrws-art HTTP/1.0
Host: b.com
Cookie: <...>
<...>
```

B -> UA:

```
HTTP/1.0 302 Found
Location: http://a.com/ assertion?SAMLart=AAPkJDxAECC7Fp8Hm1lXksv
c41gR10hlbGxvLCB0aGVyZSwgcGVvcGxl
<...>
```

UA -> A:

```
GET / assertion?SAMLart=AAPkJDxAECC7Fp8Hm1lXksvc41gR10hlbGxvLCB0aGVyZSwg
cGVvcGxl HTTP/1.0
Host: a.com
<...>
```

A -> B:

```
POST /soap HTTP/1.0
Host: b.com
Content-length: ...
Content-type: text/xml

<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
<soap-env:Header/>
<soap-env:Body>
   <Request xmlns="urn:oasis:names:tc:SAML:1.0:protocol"
      IssueInstant="2003-10-28T19:23:46Z"
      MajorVersion="1"
      MinorVersion="1"
      RequestID="e4d71c43-c89a-426b-853e-a2b0c14a5ed8">
   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
```

```
                ...
              </ds:Signature>
              <AssertionArtifact>AAPkJDxAECC7Fp8Hm1lXksvc41gR10hlbGxvLCB0aG
VyZSwgcGVvcGxl</AssertionArtifact>
              </Request>
            </soap-env:Body>
            </soap-env:Envelope>
```

B -> A:

```
            HTTP/1.0 200 OK
            Content-length: ...
            Content-type: text/xml

            <soap-env:Envelope  xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
            <soap-env:Header/>
            <soap-env:Body>
              <Response xmlns="urn:oasis:names:tc:SAML:1.0:protocol"
                InResponseTo="e4d71c43-c89a-426b-853e-a2b0c14a5ed8"
                   IssueInstant="2003-10-28T19:23:47Z" MajorVersion="1" MinorVersion="1"
                   Recipient="http://b.com/liberty/"
                   ResponseID="LANWfL2xLybnc+BCwgY+p1/vIVAj">
              <Status>
                <StatusCode xmlns:samlp="urn:oasis:names:tc:SA
ML:1.0:protocol"  Value="samlp:Success"/>
              </samlp:Status>
              <Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
                    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
                    xmlns:lib="urn:liberty:iff:2003-08"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    AssertionID="SqMC8Hs2vJ7Z+t4UiLSmhKOSUO0U"
                 InResponseTo="e4d71c43-c89a-426b-853e-a2b0c14a5ed8"
                 IssueInstant="2003-10-28T19:23:47Z"
                 Issuer="http://localhost:8080/idp"
                 MajorVersion="1" MinorVersion="2"
                 xsi:type="lib:AssertionType">
              <Conditions NotBefore="2002-10-31T21:42:12Z"
                    NotOnOrAfter="2002-10-31T21:42:43Z">
                <AudienceRestrictionCondition>
                  <Audience>http://localhost:8080/sp</saml:Audience>
                </AudienceRestrictionCondition>
              </Conditions>
              <AuthenticationStatement AuthenticationInstant="2003-10-28T19:22:07Z"
                 AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
                 xsi:type="lib:AuthenticationStatementType">
                <Subject xsi:type="lib:SubjectType">
                  <NameIdentifier  Format="urn:liberty:iff:nameid:federated">C
9FfGouQdBJ7bpkismYgd8ygeVb3Pl WK</saml:NameIdentifier>
                <SubjectConfirmation>
                  <ConfirmationMethod>
                    urn:oasis:names:tc:SAML:1.0:cm:artifact-01
                  </ConfirmationMethod>
                </SubjectConfirmation>
                <lib:IDPProvidedNameIdentifier>
                  C9FfGouQdBJ7bpkismYgd8ygeVb3PlWK
                </lib:IDPProvidedNameIdentifier>
                </Subject>
              </AuthenticationStatement>
              <AttributeStatement>
                <Subject>
                  <NameIdentifier  Format="urn:liberty:iff:nameid:federat
ed">C9FfGouQdBJ7bpkismYgd8ygeVb3Pl WK</saml:NameIdentifier>
                </Subject>
                <Attribute AttributeName="DiscoveryResourceOffering"
                     AttributeNamespace="urn:liberty:disco:2003-08">
```

```
            <AttributeValue>
           <ResourceOffering xmlns="urn:liberty:disco:2003-08">
            <ResourceID>http://b.com/disco/d0CQF8elJTDLmzEo</ResourceID>
            <ServiceInstance>
              <ServiceType>urn:liberty:disco:2003-08</ServiceType>
              <ProviderID>http://b.com/liberty/</ProviderID>
              <Description>
                <SecurityMechID>urn:liberty:security:2003-08:TLS:X509</SecurityMechID>
                <SecurityMechID>urn:liberty:security:2003-08:TLS:null</SecurityMechID>
               <Endpoint>https://b.com/soap</Endpoint>
               <SoapAction>urn:liberty:disco:2003-08</SoapAction>
              </Description>
            </ServiceInstance>
            <Abstract>Discovery service</Abstract>
           </ResourceOffering>
            </AttributeValue>
           </Attribute>
          </AttributeStatement>
          <ds:Signature>
          ...
          </ds:Signature>
            </saml:Assertion>
          </samlp:Response>
        </soap-env:Body>
        </soap-env:Envelope>
```

2.2.2. Nested SAML Status

It is possible to return nested status SAML status codes. See Example 1 for a detailed example. Note that returning status detail may be a security risk - providers should take care not to return so much information in the status that they compromise their security.

```
        <lib:AuthnResponse ResponseID="hhuuja1bc744hGJn5Q9A5yvEIgS"
           InResponseTo="Zon3WjJ2KL7j+bJu7MuIr 4Pt2go5" MajorVersion="1" MinorVersion="2"
           IssueInstant="2002-10-31T21:55:41Z">
           <samlp:Status>
              <samlp:StatusCode Value="samlp:Responder">
                 <samlp:StatusCode Value="lib:ProxyCountExceeded"/>
              </samlp:StatusCode>
           </samlp:Status>
           <lib:ProviderID>http://IdentityProvider.com</lib:ProviderID>
           <RelayState>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</RelayState>
        </lib:AuthnResponse>
```

Example 1.

2.3. Pseudonym Access by Principal

Providers should consider whether or not to grant Principal's access to the actual values of the opaque identifiers associated with that Principal. Beyond the fact that, as random strings, the values themselves will be meaningless to the principal, their visibility, perhaps as part of some 'federation management' interface, could pose a privacy vulnerability. If a Principal's account at one provider were to be compromised - the scope of the breach could be magnified if the impostor were able to access the pseudonyms for that Principal as known by other providers.

3. 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
Non-repudiation	—	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 passing between identity and service providers, and some of the message components are generally required to be digitally signed and verified. The Liberty ID-FF does not specify any method for the digital signing of messages at the Principal's client terminal, although some Liberty-enabled services may provide such a method. Signing and verifying messages provide data integrity, data origin authentication, and a basis for non-repudiation. 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.

4. Authentication Mechanisms and Liberty

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.

User authentication methods are commonly classified as being one-factor, two-factor, or three-factor. Generally, the more factors used, the more reliable the authentication method. The first factor may be "something you have" such as a smart card. The second factor might then be a PIN for the smart card, or a password - "something you know". A possible third factor might be "something you are", such as a body characteristic (fingerprint, for example).

Authentication mechanisms consist of combinations of these factors. Examples of authentication mechanisms include username with password (*not* HTTP Basic Auth), certificate-based (for example, via SSL or TLS), or Kerberos.

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 1), 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.

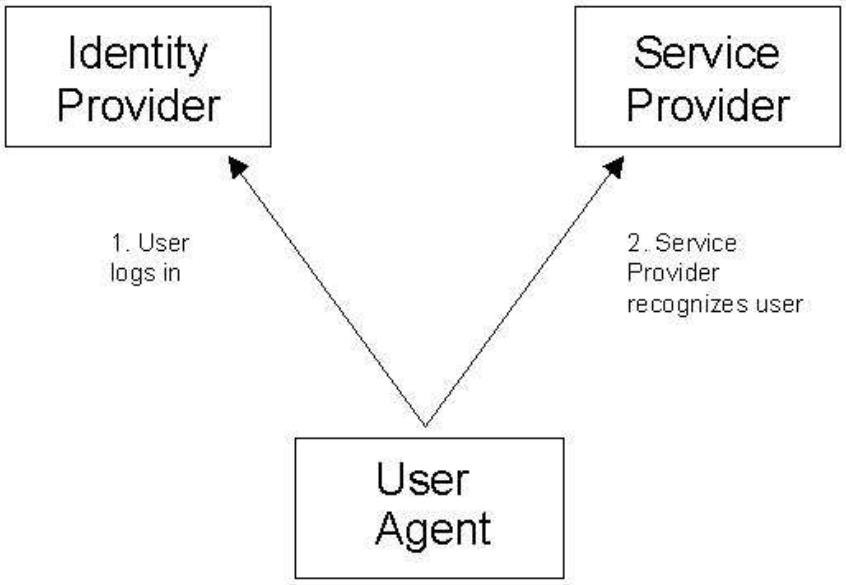

Figure 1. User logs in at identity provider and is recognized by service provider

Note:

In most cases, a service provider will submit an authentication request to an identity provider, and the recipient will return an authentication response. It is, however, possible for an identity provider and service provider

to agree out-of-band to support a model whereby the identity provider proactively creates an authentication response without first having received an authentication request. This allows single-signon for a Principal at a service provider to occur without the Principal first visiting that service provider, with the following provisions:

1. An SP should drop an unsolicited AuthnResponse, unless it has negotiated some out-of-band agreement with the IdP sending the response.

2. An SP can determine that an AuthnResponse is unsolicited by looking for the InResponseTo attribute. If not found, the response is unsolicited, and they should examine the assertion to determine whether they have an out-of-band agreement with the message sender.

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 must have the properties of a nonce. A nonce is a random or non-repeating value that is included in data exchanged by a protocol, usually for guaranteeing liveness and thus detecting and protecting against replay attacks.

Authentication Type, Multi-tiered 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 birth date, mother's maiden name, etc. This act is *reauthentication* and the overall functionality is *multi-tiered authentication*. Wielding multi-tiered 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, smart-card, 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. For instance, the server may be wielding a bogus certificate, and the user may not adequately inspect it or understand the significance. It may also be the case that the server does not know if the real user is present for the authentication unless some PIN or password is bound into the authentication protocol.

Validating Liveness

Liveness refers to whether the user who authenticated at time *t0* is the same user who is about to perform a given operation at time *t1*. 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.

Compromised Credentials

A service provider may discover that a user's credentials have been compromised. A preferred solution to this is to send a single logout message for that user. To aid in this, service providers should expose a SOAP single logout endpoint.

5. Guidelines for Mobile Environments

The Liberty specifications recognize that the mobile environment requires specific treatment, and define specific methods which may be used by mobile devices and other systems operating in mobile networks. In particular, mobile devices may offer a more constrained environment for identity management than do other devices such as personal computers on an enterprise network. Liberty defines one specific profile for mobile environments (the LECP Profile - see [LibertyBindProf]) and an adaptation of the Liberty Artifact Profile ([LibertyBindProf]) which enables the transmission of an artifact using WAP/WML ([WML]).

It should be noted, however, that there are specific items that should be considered when implementing or deploying in a mobile environment. Some of these are noted below. The guidelines in this section relate to these specific deployment issues.

5.1. Some mobile-specific deployment considerations

Use of the radio resource
: In some mobile environments, radio bandwidth may be restricted or costly, and in such environments, it may be desirable to limit the number of exchanges with a mobile device over the radio link.

Reliability/Latency
: Mobile devices may have poor network connectivity over a radio link, causing unreliable network connections, or latency over such connections.

Ease of deployment
: WAP 2.0 provides a different architecture for browsing from WAP 1.x. However, there remain many mobile devices deployed that are equipped with WAP 1.x-compliant user-agents. To enable handset usage of the Liberty profiles may in some cases require the deployment of handsets that utilize additional or improved software.

Presence of SIM card
: GSM-based networks make use of the Subscriber Identity Module (SIM) card. Such cards may provide enhanced security for identity-based transactions.

Network roaming
: Mobile roaming business agreements established between network operators provide an important basis for trust between Liberty providers.

Link security
: WAP 1.x does not allow for secure, encrypted links at the transport layer between a mobile device and a service provider. WAP 2.0 introduced TLS which does allow for such links. It is strongly recommended in the Liberty specifications that adequate security measures be taken to protect data transmitted via the Liberty protocols, including the use of transport-layer security.

5.2. Using the Liberty single-signon profiles in mobile environments

5.2.1. Summary of the Liberty profiles in mobile environments

5.2.1.1. The artifact profile

The Liberty artifact profile may be used to enable WML redirects to perform single-signon (see [LibertyBindProf]). The following list summarizes this profile with respect to some of the deployment issues noted above.

Latency

The artifact profile will force the user-agent through a minimum of three redirects. If such redirects are performed on an unreliable network, then there may be significant delays in processing of the single-signon request. It should be noted, however, that the benefits that single-signon provides (ie. removing extra login pages at Liberty-enabled sites) will mitigate the effect of such redirects.

Reliability

As noted above, conducting three redirects over an unreliable network may prove to be a less-than desirable user experience.

Ease of deployment

The artifact profile is compatible with any WAP 1.x or 2.x (x >= 0) user-agent, which implies that there is no concern about deploying new software or hardware.

Radio consumption

The redirects necessary for use of this profile will use the radio link. The benefits afforded the user by use of the single-signon protocols should mitigate any concern about additional consumption of radio resources.

Link Security

WAP 1.x does not allow for end-to-end secure transport-layer link between the device and the service provider. WAP 2.0 does provide this possibility.

5.2.1.2. The LECP profile

The LECP profile ([LibertyBindProf]) is an alternative to WML usage of the artifact profile. A Liberty-enabled client or proxy may initiate this profile by specifying a *Liberty-Enabled* header in a request to a service provider.

Latency

If a device is performing the LECP profile, then a number of redirects are still needed for the single-signon to occur. Once again, in practice, Liberty deployments will cut down on the overall number of logins required, and mitigate the performance effect of these redirects. A useful optimization, however, would be to deploy a LECP proxy in environments where network latency is an issue. This may further mitigate the effect of network latency by minimizing use of the radio network.

Reliability

As mentioned in the previous section, the user may experience difficulties when conducting several redirects over an unreliable network. Deploying a LECP proxy may again mitigate this issue by restricting redirects to the more reliable wired network.

Ease of deployment

If a proxy server is deployed to perform the LECP profile on behalf of mobile devices, then there are no concerns about device deployment. However, if the device is performing the LECP profile itself, it may be necessary for additional software to be deployed, or upgraded on the mobile device.

Radio consumption

The redirects necessary for use of this profile will be made over the radio link in the case of a mobile device performing the LECP profile. Deploying a proxy for the device may ease this concern, as the LECP redirects will be performed on the wired network.

Link Security

In the case of a WAP 2.0-compliant mobile device, end-to-end encrypted connections are possible between the device and the service provider at the transport layer. It should be noted that WAP 1.x does not allow for such connections. In environments where a proxy server is deployed in order to perform the LECP profile, however, it may be necessary to create two separate secure sessions - one between the mobile device and the

proxy, and another between the proxy and the service provider. See Section 5.2.3 for more information about this scenario.

5.2.2. Methods of roaming using the Liberty protocols

As the user of a mobile device moves from place to place, they may roam from one mobile operators network to another. These operators may have in place business trust agreements that allow a user to access services from service providers that are not on the users home network. Authentication of the user may be necessary in order to access such services on the visited network (see [LibertyProtSchema] section on Dynamic Proxying). It may be necessary for Liberty identity and service providers to establish dynamic trust agreements. These allow authentication of the user by an identity provider on their home network, or for an identity provider on the roaming network to perform authentication of the user as a proxy for the home network identity provider. These methods are summarized below.

Liberty-enabled roaming methods:

Proxying

An identity provider on the visited (roaming) network proxies Liberty authentication messages from the service provider on the visited network to the appropriate identity provider on the users home network, based on existing roaming contracts.

Direct

The service provider on the visited network sends authentication requests directly to the appropriate identity provider on the users home network, based on an existing trust chain (such as PKI infrastructure or roaming agreements) and directed via the existing (General Packet Radio Service (GPRS)) technical infrastructure (ie. not using any specific Liberty mechanism to direct requests to the correct identity provider).

Using identity provider proxying may reduce the total number of federation and/or redirect operations that occur, thus providing a better user experience.

5.2.2.1. Roaming using the proxy method

Even with roaming agreements in place between the user's home network and the visited network, a service provider on the visited network may not be able to accept authentication assertions directly generated by an identity provider on the users home network. The service provider may not even know about the identity provider on the home network. This situation might necessitate the intervention of an identity provider on the visited network to bridge the two networks. Such an identity provider may act as a proxy for the identity provider on the user's home network.

5.2.2.1.1. Guidelines for roaming using proxying methods

For the proxying methods, the following guidelines may be followed by those deploying Liberty services in a mobile environment.

A service provider will not know *a priori* that the Principal may be roaming on the visited network, and may also not have an account with the local identity provider.

In order to provide for such users, a service provider on the visited network should issue authentication requests with a `<NameIDPolicy>` of *any* or *onetime* when operating in a mobile environment. This would ensure that federation of the user's identity would not be required in order for the roaming user to use services on the visited network.

A Liberty-enabled client "has, or knows how to obtain, knowledge about the identity provider that a Principal wishes to use with the service provider". In certain environments, a Liberty-enabled proxy may also be present, which is "an HTTP proxy (typically a WAP gateway) that emulates a Liberty-enabled client" (definitions excerpted from [LibertyGlossary]).

The following guidelines pertain to LEP and LEC deployments in a mobile roaming through proxying situation:

- Any identity provider on the visited network may be provisioned with information that enables them to have some knowledge of their network's roaming partners.

- Any Liberty-enabled proxy or client present on the home network should be provisioned with a list of identity providers who are partners of the user's home identity provider. It should be noted that how such provisioning is carried out is not governed by the Liberty specifications.

In addition, some specific guidelines may be used with the proxy method of mobile roaming.

- In order to ensure that proxying is considered by the recipient of an authentication request, the `<ProxyCount>` element should be set to a value greater than 0. In order to minimize network latency caused by multiple proxying steps, however, it is recommended that `<ProxyCount>` be set no higher than 1.

5.2.3. WAP and the Liberty-enabled Proxy

In version 1 of WAP, a *WAP Gateway* translates between WAP protocol flows and HTTP protocol flows. There is no end-to-end TLS-secured session directly between the user-agent and the (web-based) service provider. The user-agent maintains one session with the WAP gateway, and the gateway maintains a separate session with the web server providing the service to the end-user.

Such an environment allows the LECP profile to be managed entirely on the WAP gateway, allowing that gateway to become a proxy for the mobile device, providing a performance benefit.

In an environment where the user-agent accessing a service may be severely constrained, unable to process URLs over a certain size; ECMAScript (thus preventing HTTP POST without user interaction), and cookie-based sessions, then using a WAP gateway to perform LECP interactions is a useful method of circumventing such user-agent deficiencies, without requiring new mobile handset deployments.

In WAP version 2, the user-agent is XHTML/HTTP-compliant, and thus an end-to-end TLS-secured session is possible directly between the user-agent and the web server, without the necessity for a gateway to intermediate between the HTTP and WAP protocol flows. The LEC may perform the LECP profile directly, without the need for an additional intermediary. In this situation, the WAP gateway becomes an HTTP proxy server, merely forwarding HTTP requests and responses. However, in certain environments a proxy for the LECP profile may still make sense as an optimization.

In such an environment, any proxy deployed between a service provider and the user-agent will see only an encrypted HTTP protocol flow, and will not be able to perform the LECP profile, as specified, on behalf of the user-agent.

One solution to this issue is for the service provider to redirect the user-agent to the proxy, allowing the gateway to perform the LECP profile. In this situation, the LECP profile is performed as specified in [LibertyBindProf] with an additional step, which allows the service provider to determine that usage of a LEP has been requested by the Principal, to obtain metadata for the LEP, and finally to redirect the user-agent to the WAP proxy. At this point, the LECP profile will be performed by the proxy, until the final response is received by the WAP proxy and forwarded to the user-agent.

Security note:

It is strongly recommended that the LEP and the identity provider in this adaptation of the LECP profile are hosted by the same entity, in the same secure facility, to avoid the possibility of a man-in-the-middle attack being launched.

5.2.3.1. LECP Profile with Liberty Enabled Proxy

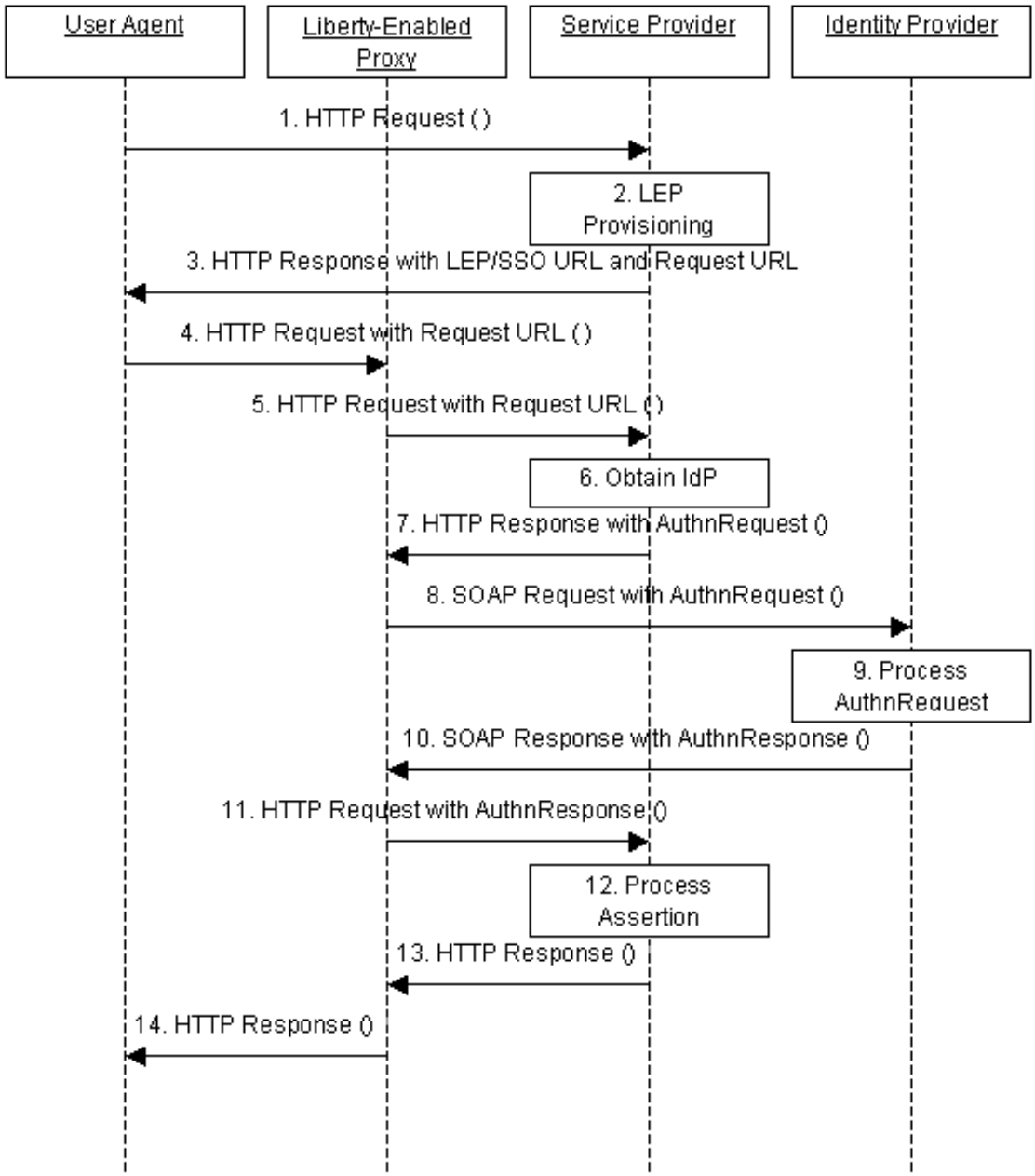

Figure 2. LECP Profile with Liberty Enabled Proxy

1) Step 1 == LECP Step 1.

2) Step 2 is non-LECP.

The SP should recognize whether the UA uses LEP or not by metadata. The metadata could be encoded in the extension of HTTP header added by LEP (the Step 1 HTTP request from LEP to SP). It may look as follows.

```
GET /somewhere.html HTTP/1.1
...
Liberty-Enabled: LIBV=urn:liberty:iff:1.2
X-LEP: http://lep.operator.com/lep/
```

Example 2. Example of LEP Extension.

This method would require additional work for the SP. When the UA first requests an HTTPS session to the SP, the SP should read the User Agent in the HTTP header and should judge if the UA uses LEP or not. The SP then should lead the UA to non-secure HTTP connection where the LEP can add the above extension on the HTTP header bound for SP.

3) Step 3 is non-LECP.

If the Principal uses LEP, redirect Principal to the LEP/SSO URL written in the LEP metadata. SP's URL should also be encoded in the URL.

4) Step 4 is non-LECP.

The User Agent is now redirected to LEP.

5) Step 5 is non-LECP.

The LEP decodes the requested URL, and redirects the User Agent back to SP.

Steps 6-14 are the usual LECP steps.

5.3. Mobile Provisioning using tamper-resistant smart cards

A Liberty-enabled client (LEC) should be able to make decisions regarding the handling of the Liberty protocols. In order to do so, the LEC should know with which identity providers it should communicate and thus needs certain information about them. Given certain constraints of the mobile environment, it may not be possible for the LEC to rely exclusively on acquiring metadata about identity providers using the standard Liberty methods for metadata acquisition (see [LibertyMetadata]). Better performance and security may be achieved by provisioning the LEC at some time prior to use of the Liberty protocols, using a tamper-resistant, cryptography-enabled device with protected memory (such as a GSM SIM card). In such cases, we recommend the use of the following schema to provide a minimal set of configuration parameters that will enable the LEC to be provisioned with necessary data. The metadata in this schema incorporates actual values such as the URLs of the default and partner identity providers that would be used by the mobile device.

Note:

Work is underway to standardize data structures, and methods of access for such provisioning data in the ETSI Project Smart Card Platform. When this work is complete, the following section will reference that work.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
   targetNamespace="urn:liberty:lecprov:2003-08"
   xmlns="urn:liberty:lecprov:2003-08"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">

   <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
      schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
   <xs:element name="LecProv" type="LecProvType"/>
   <xs:complexType name="LecProvType">
      <xs:sequence>
```

```
         <xs:element name="IdPConfig" type="IdPConfigType"/>
         <xs:element name="Ext" type="ExtType" minOccurs="0"/>
         <xs:element ref="ds:Signature" minOccurs="0" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="optional"/>
      <xs:attribute name="release" type="xs:integer" use="required"/>
      <xs:attribute name="date" type="xs:date" use="optional"/>
   </xs:complexType>
   <xs:complexType name="IdPConfigType">
      <xs:sequence>
         <xs:element name="DefaultIdP" type="IdPEntryType"/>
         <xs:element name="PartnerIdP" type="IdPEntryType"
            maxOccurs="unbounded" minOccurs="0" />
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="IdPEntryType">
      <xs:sequence>
         <xs:element name="ProviderID" type="entityIDType"/>
         <xs:element name="ProviderDisplayName" type="xs:string"
            minOccurs="0" />
         <xs:element name="SingleSignOnServiceURL" type="versionedEndPoint"
            minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="KeyInfo" type="ds:KeyInfoType" minOccurs="0"/>
         <xs:element name="MetadataConfKey" type="ds:KeyInfoType" minOccurs="0" />
         <xs:element name="Ext" type="ExtType" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="versionedEndPoint">
      <xs:simpleContent>
         <xs:extension base="xs:anyURI">
            <xs:attribute name="version" type="xs:anyURI"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <xs:simpleType name="entityIDType">
      <xs:restriction base="xs:anyURI">
         <xs:maxLength id="maxlengid" value="1024"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:complexType name="ExtType">
      <xs:sequence>
         <xs:any maxOccurs="unbounded" namespace="##other"
            processContents="lax"/>
      </xs:sequence>
   </xs:complexType>
```

Some guidelines apply to the use of this schema.

- Document instances should be canonicalized using XML Exclusive Canonicalization ([XMLCanon]).
- Instances of this schema should only use `<LecProv>` as a root element.
- Document instances using this schema should be constructed such that other namespaces are not imported into the LEC provisioning schema, unless absolutely necessary.
- The `<ProviderID>` may be used to acquire additional metadata about the provider specified in instances of this schema.

- An extension point (element <Ext>) is provided to allow elements from other namespaces to be included in this schema. Given the storage constraints of this environment, implementors should take care only to add extensions that are absolutely necessary.

- The <DefaultIdP> should be considered by the LEC to be the identity provider that should be used whenever possible.

- The <PartnerIdP> elements should be listed in order of preference in document instances, and should be used whenever the <DefaultIdP> is not available.

- It should be noted that an instance of this schema may reach a considerable size if a number of identity providers are included in the instance. In particular, the key values supplied in the <KeyInfo> and <MetadataConfKey> elements may be quite sizeable.
  It is suggested that when supplying key value information for these elements, that the key value portion of the XML be hashed (using SHA-1 for example) and then encoded base64. This value would then be placed in the KeyName element of the KeyInfo element.
  When the LEC must compare this stored key value against that obtained from a provider, it may simply encode the received key appropriately (for example as RSAKeyValue), hash it, encode base64 and then compare to the value stored.
  D1

```
<RSAKeyValue>
   <Modulus>
      xA7SEU+e0yQH5rm9kbCDN9o3aPIo7HbP7tX6WOocLZAtNfyxSZDU16ksL6W
      jubafOqNEpcwR3RdFsT7bCqnXPBe5ELh5u4VEy19MzxkXRgrMvavz yBpVRgBUwUlV
      5foK5hhmbktQhyNdy/6LpQRhDUDsTvK+g9Ucj47es 9AQJ3U=
   </Modulus>
   <Exponent>AQAB</Exponent>
</RSAKeyValue>
```

  Could be encoded as described above, to create the following key information in the document instance:

```
<KeyInfo>
   <KeyName>
      reyebww23rADSaddfDFSe34ncaksdcnsd lcnk=
   </KeyName>
</KeyInfo>
```

Example 3. Key encoding

Additionally, there are the following guidelines for the usage of the IdPEntry type:

- <ProviderId> is the identifier that the LEC must use to validate the IdP as liberty provider.

- If <MetadataConfKey> is present, the <ProviderID> may be used to acquire additional metadata about the provider specified in instances of this schema using the well-known location mechanism. Fetched metadata instance documents should be signed by the key specified by this element.

- If <SingleSignOnServiceURL> is present, it denotes the URL that the LEC must use when issuing <AuthnRequest> to the IdP. Several entries of this element can appear denoting different entry points listening for different protocol version messages.

- If <KeyInfo> is present, IdP authentication must be achieved following the contents of this element. If not present, IdP-authentication may be achieved following any other existing mechanism, such as TLS Root-CA-List based authentication.

- If <ProviderDisplayName>is present, the LEC should display this value to user when interacting with this IdP.

5.3.1. Obtaining LEC-Provisioning document instances using a (U)SIM

It is recommended that when hosted by a (U)SIM, LEC-applications (or the handset firmware) should obtain the LEC-Provisioning document instances through simple file I/O. A single document instance will be hosted in a read-only file (denoted herein S-EF) updated unilaterally by the (U)SIM. The (U)SIM itself will obtain the LEC-Provisioning information at "personalization time" and by the means described in Section 5.3.2.

The LEC application should look for the existence of this file and retrieve its contents each time the application is launched. It should also poll the (U)SIM periodically for modification or leverage the existing "SIM-REFRESH" mechanism by which the (U)SIM notifies the handset of any modification of its contents.

It should be noted that from the LEC-Application perspective, the authenticity and integrity of the LEC-Provisioning document is guaranteed by the fact that it is obtained directly from the U(SIM). Therefore, LEC applications are not required to apply any further integrity check. For "Smart-Phone" class devices, the handset firmware and hardware must ensure that data retrieved from this source has not been tampered with by any local process or potentially malicious component.

Beyond setting the S-EF file to be read-only, it is not necessary to enforce PIN protection or any other specific access control measure on the file.

5.3.2. Updating LEC-Provisioning document instances using a (U)SIM

Initially, the (U)SIM will obtain the LEC-Provisioning document instance during its "personalization" at manufacturing time. Later on, it would be possible to update the information by using one of two methods described below.

Independently of the mechanism used to physically obtain the updated LEC- Provisioning document instances, all configuration data will be source-authenticated and integrity checked by the (U)SIM. It is recommended that the existing SIM Application Toolkit (SAT) integrity and authentication mechanisms are used for that purpose.

5.3.2.1. SAT-based (U)SIM LEC-Provisioning Document Update

The first and preferred method for updating the LEC-Provisioning document leverages the SAT, using SMS or a different bearer, depending on specific handset support. It is recommended that if this method is used, a complementary SIM-REFRESH command be issued to signal to the LEC application that the LEC-Provisioning document instance has changed.

5.3.2.2. File-based (U)SIM LEC-Provisioning Document Update

The second method of provisioning may be performed by the LEC via input to a specific write-only input file (denoted herein I-EF). The format of this file will be determined by the (U)SIM provisioning authority, and it is for now out of scope for these guidelines. Once the LEC application has finished writing to the I-EF, it should re-read the S-EF to check for an updated version of the LEC-Provisioning document instance. Beyond setting up the I-EF file as write-only, it is not necessary to enforce PIN protection or any other specific access control measure on it.

The Provisioning-source entity will deliver the following to the LEC application embedded in the appropriate application-level protocol flow (such as within a AuthnResponseEnvelope if the provisioning source is also an IdP):

```
<xs:element name="newLECProvInfo">
   <xs:complexType>
      <xs:complexContent>
      <xs:extension base="xs:base64">
      <xs:attribute name="release" type="xs:integer" use="required"/>
      <xs:attribute name="date" type="xs:date" use="optional"/>
      </xs:extention>
      </xs:complexContent>
   </xs:complexType>
```

```
</xs:element>
```

These elements would be in a separate schema.

The provisioning information itself will be in a binary format that has been base64 encoded. After base64 decoding, the LEC will write the contents into the I-EF. When the LEC-application finishes writing to the I-EF, the SIM will issue a SIM-REFRESH command (if the handset platform supports this). Then the LEC application should read the new provisioning information from the S-EF and check that its release version is the same as the one conveyed through the container. If the SIM-REFRESH command is not supported by the Handset, the LEC-application should know about this limitation and instead perform a poll for a change.

If after a reasonable amount of time the contents of the S-EF are not updated accordingly, the LEC-application should assume that the operation failed and signal back to the provisioning source the condition whenever the opportunity arises with:

```
<xs:element name="newLECProvInfoFailed">
   <xs:complexType>
      <xs:simpleContent>
         <xs:attribute name="release" type="xs:integer" use="required"/>
         <xs:attribute name="date" type="xs:date" use="optional"/>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

These elements would be in a separate schema.

A potential liberty-bound candidate protocol message for hosting this could be the extension point of LEC-to-IdP AuthenRequest. But we could also create a specific SOAP call. As we are just crafting this as guidelines, we don't have to be completely specific for now on how this should be implemented.

5.3.3. External standardization dependencies

(U)SIM file identifiers (S-EF and I-EF) will eventually be registered with ETSI.

5.4. Liberty authentication context in mobile environments

The Liberty Authentication Context Specification ([LibertyAuthnContext]) provides a schema for describing the context surrounding the authentication of an individual. Such contextual information may be required for an organization to decide whether a particular Principal's authentication was rigorous enough for their purposes. For example, a payment made via credit card, authorized with the signature of the Principal may be seen as more "trustworthy" because of the presence of the signature.

Liberty provides a number of authentication contexts that are specific to the mobile environment:

MobileTwoFactorContract

Use of this context would involve contract registration procedures for mobile subscribers and the protocols used to control access to digital mobile networks. The two authentication factors would be a physical device, normally containing a secret key and executing symmetric cryptography, plus a mechanism that verifies that the rightful user of the device is present, e.g. the input of a CHV (Card-Holder Verification) value such as a PIN or some biometric data. Requirements for PKI-based authentication, for access to Liberty Alliance services or for securing transactions, can be met if private signing keys are deployed (e.g. using WAP's WIM), with corresponding functionality in mobile devices. Best established practice in the mobile industry is to hold secret or private key material and execute cryptographic functions present in mobile network operators' (MNO) tamper-resistant smart cards (commonly called SIM cards or USIM cards).

MobileOneFactorContract

Use of this context would involve contract registration procedures for mobile subscribers and the protocols used to control access to digital mobile networks. The single authentication factor would be a physical device, normally containing a secret key and executing symmetric cryptography. Requirements for PKI-based authentication, for access to Liberty Alliance services or for securing transactions, can be met if private keys are deployed (e.g. using WAP's WIM), with corresponding functionality in mobile devices. Best established practice in the mobile industry is to hold secret or private key material and execute cryptographic functions in MNOs' tamper-resistant smart cards (commonly called SIM cards or USIM cards)

MobileTwoFactorUnregistered

This context may be useful when a service other than that of the MNO wishes to link their customer ID to a mobile-supplied two-factor authentication service, by capturing mobile data such as a phone number or device id at enrolment. This context would involve the protocols used to control access to digital mobile networks, but with no subscriber registration procedure employed by the MNO. The two authentication factors would be a physical device, normally containing a secret key and executing symmetric cryptography, plus a mechanism that verifies that the rightful user of the device is present, e.g. the input of a CHV (Card-Holder Verification) value such as a PIN or some biometric data . Requirements for PKI-based authentication, for access to Liberty Alliance services or for securing transactions, can be met if private signing keys are deployed (e.g. using WAP's WIM), with corresponding functionality in mobile devices. Best established practice in the mobile industry is to hold secret or private key material and execute cryptographic functions in MNOs' tamper-resistant smart cards (commonly called SIM cards or USIM cards)

MobileOneFactorUnregistered

This context may be useful when a service other than that of the MNO wishes to link their customer ID to a mobile-supplied one-factor authentication service, by capturing mobile data such as a phone number or device id at enrolment. This context would involve the protocols used to control access to digital mobile networks, but with no subscriber registration procedure by MNO. The single authentication factor would be a physical device, normally containing a secret key and executing symmetric cryptography. Requirements for PKI-based authentication, for access to Liberty Alliance services or for securing transactions, can be met if private keys are deployed (e.g. using WAP's WIM), with corresponding functionality in mobile devices. Best established practice in the mobile industry is to hold secret or private key material and execute cryptographic functions in MNOs' tamper-resistant smart cards (commonly called SIM cards or USIM cards)

The above-noted authentication contexts may be appropriate in different situations. For example, two-factor authentication either with or without a MNO contract (i.e. unregistered) might be required for authorization of high-value mobile payments, whereas one-factor unregistered authentication may be sufficient for low-value transactions, or access to a corporate network. It should be noted, that other information (such as the presence of keying information on the mobile device) will also factor in to the authentication context.

6. Privacy Principles

Federated authentication systems have the potential to either enhance or hinder an individual's privacy. By simplifying authentication, such systems will ensure that a user's personal data can be better protected, and not compromised by the typical cribs that users currently typically use for simplification (e.g. reuse of password, weak passwords, stored passwords, etc). Nevertheless, if inappropriately implemented, federated authentication systems have the potential to compromise a user's privacy. While the Liberty Alliance has defined normative mechanisms for enhancing privacy into its specifications, other aspects of federated authentication that cannot be normatively defined can still impact privacy. This section lists guidance for such aspects.

6.1. Privacy Principles for Authentication

In a sense, there is an inherent conflict between authentication and privacy. A common definition for privacy is 'the ability for individuals to control how information about themselves is collected and shared'. As authentication systems often require that some information about the authenticating entity be collected, for a user to opt out of this collection (as is their privacy 'right') implies an inability for them to subsequently use that particular authentication system and, presumably, access those resources being protected by that system. Consequently, for every authentication system, a user must weigh the advantages afforded by accepting the requirements (with respect to collection of personal information about themselves) of the system against the potential privacy risk inherent in any such personal information sharing.

Recently, a variety of organizations have drafted "best-practice" guidelines for creating authentication systems that will respect the privacy rights of those individuals who use the systems. For example, the Center for Democracy & Technology (CDT) explored the privacy issues associated with authentication systems:

> *New technologies for authentication make possible greater realization of the Internet's potential by making online transactions more seamless, tying together information on multiple devices, and enabling yet unimagined services. However, many authentication systems will collect and share personally identifiable information, creating privacy and security risks. To mitigate these risks, it is essential that authentication systems be designed to support effective privacy practices and offer individuals greater control over their personal information.*

The CDT Working Group drafted basic privacy principles [CDT] that should be considered in the design and implementation of authentication systems. These principles are:

- Provide User Control - The informed consent of the individual should be obtained before information is used for enrollment, authentication and any subsequent uses.
- Support a Diversity of Services - Individuals should have a choice of authentication tools and providers in the marketplace. While convenient authentication mechanisms should be available, privacy is put at risk if individuals are forced to use one single identifier for various purposes.
- Use Individual Authentication Only When Appropriate -Authentication systems should be designed to authenticate individuals by use of identity only when such information is needed to complete the transaction. Individual identity need not and should not be a part of all forms of authentication.
- Provide Notice - Individuals should be provided with a clear statement about the collection and use of information upon which to make informed decisions.
- Minimize Collection and Storage- Institutions deploying or using authentication systems should collect only the information necessary to complete the intended authentication function.
- Provide Accountability - Authentication providers should be able to verify that they are complying with applicable privacy practices.

The CDT Working Group did not specifically address the privacy issues introduced by federated authentication. Their basic guidelines can be extended to address these new issues.

Similarly, the EU Article 29 Data Protection Working Party published guidance on online authentication systems [Art29]. The Working Party's conclusions are as follows:

- Both those who design and those who actually implement on-line authentication systems (authentication providers) bear responsibility for the data protection aspects, although at different levels. Websites making use of these schemes (service providers) also have their own responsibility in the process. It is advisable for the different players to have clear contractual agreements between them where the obligations of each party are made explicit.

- All possible efforts should be made to allow anonymous or pseudonymous use of online authentication systems. Where this would inhibit full functionality, the system should be built to require minimal information only for the authentication of the user and to give the user full control over decisions concerning additional information (such as profile data). This choice should exist both at the level of the authentication provider and of the service providers (the sites making use of the system).

- It is vital to provide adequate information to the users concerning the data protection implications of the system (controller identity, purposes, data collected, recipients and so on). This information should be provided in an easily accessible and user-friendly way, preferably through the collection form or via a prompt box that would automatically open on the screen of the user, and in all the languages in which the service is offered.

- When personal data are to be transferred to third countries, authentication providers should work with service providers who take all necessary measures to provide adequate protection or that put in place sufficient safeguards to ensure the protection of the personal data of the users of the system, by using contracts or binding corporate rules. This should be the general rule. If in particular cases consent is used as a basis for the transfer, sufficient information and choice should be given to the users. They should have the option to agree or not to the transfer on a case by case basis.

- The use of identifiers, whatever form they take, entails data protection risks. Full consideration should be given to all possible alternatives. If user identifiers are indispensable, the possibility of allowing the user to refresh the identifier should be considered.

- The adoption of software architecture that minimizes the centralization of personal data of the Internet users would be appreciated and encouraged as a means of increasing the fault-tolerance properties of the authentication system, and of avoiding the creation of high added-value databases owned and managed by a single company or by a small set of companies and organizations.

- Users should have an easy means to exercise their rights (including their right to opt-out) and to have all their data deleted if they decide to stop using an on-line authentication system. They should also be adequately informed about the procedure they should follow if they have enquiries or complaints.

- Security plays a fundamental role in this context. Organizational and technical measures that are appropriate to the risks at stake should be taken.

6.2. Privacy Principles for Federated Authentication

In this section, we interpret the basic privacy principles listed in the previous section for federated architectures.

6.2.1. Passive SSO

Federated authentication makes possible 'invisible' authentication, i.e. a Principal can be logged in at a Service Provider without explicit action on their part but rather based on a previous authentication event performed at an IDP. While a valuable usability mechanism, the possibility of a principal being logged in without their knowledge could pose a privacy concern if that principal was not adequately informed of, and consented to, the process. The principal might believe they are surfing anonymously at the SP while in fact they are not.

Authentication at a (non-proxying) IDP will always be overt and active, federated authentication at an SP can be either overt/active (i.e. they click on a 'Sign in at IDP' link or button to instigate the process) or passive. Passive SSO includes both when the SP automatically directs a Principal to the IDP for authentication and when the IDP sends the Principal to an SP along with an unsolicited authentication assertion.

As the Principal is unknown to the SP until after it has received an authentication assertion from the IDP, it will be unable to apply policy for passive SSO at anything other than a site-wide level (e.g. always notify users). Nevertheless, SPs can enable principal-specific passive SSO policy to be applied at the IDP through the inclusion of the passive SSO details in the AuthnRequest.

Note:
An IDP would be unable to determine the validity of the SP's claims with respect to passive SSO - this model relies on the SP acting in good faith. In some situations it may be the case that the IDP is 'more trusted' by the Principal than the SP. In these situations, the Principal's confidence that their policy with respect to passive SSO should be based on the trust they have in the SP, not the more trusted IDP.

- When instigated by the Principal, an SP must include a value of 'urn:liberty:consent:obtained:current:explicit' in the Consent attribute on the AuthnRequest to indicate that the Principal has explicitly given consent to the sending of the AuthnRequest message.

- When instigated by the SP, the SP must include a value of either 'urn:liberty:consent:obtained:current:implicit' or 'urn:liberty:consent:obtained:prior' in the Consent attribute on the AuthnRequest.

- After recieving an AuthnRequest with a value of either 'urn:liberty:consent:obtained:current:implicit' or 'urn:liberty:consent:obtained:prior' in the Consent attribute, an IDP must ensure that the relevant Principal's policy allows this.

- When sending a Principal to an SP along with an unsolicited AuthnResponse, an IDP must ensure that the Principal's policy with regard to passive SSO is satisfied.

- An IDP must provide Principals the opportunity to specify their policy with regard to passive SSO.
  An IDP should allow Principal's to specify any of:

  - They consent to authorizing passive SSO.
  - They wish to be notified of passive SSO.
  - They wish to be prompted for authorization for passive SSO.

- An IDP may give the principal the ability to specify their preferences for passive SSO on a per SP basis.

Past experience with SSO systems has shown that systems that rely upon active SSO by requiring the User to instigate the SSO process can cause confusion (the User may provide any SP credentials they have to the IDP or vice versa) and hinder adoption. Additionally, a Principal who has consented to federation has presumably done so in order to enjoy a more streamlined browsing experience, this value diminished if passive SSO is ruled out.

- An IDP should implement a default policy of 'allow passive SSO', this default modified by individual Principals.
- If the Principal sets their policy at the IDP such that they choose to be notified/prompted for authorization for passive SSO, the IDP should display the name of the relevant SP at which they will be authenticated.

In general, the advantages that the different mechanisms for Principal control of the SSO process will need to be weighed against the issues increased control would present for usability. This decision should however be the Principals to make and IDPs must provide simple and intuitive interfaces to assist the principal in asserting their choices.

6.2.2. Authentication Status

Passive SSO, even when consented to, creates the possibility of a Principal being logged in at a Provider without explicit action on their part. It will sometimes be the case that, through the available access to account details, it will be readily obvious to the principal that they are logged in at the SP. However, it will also be the case that sometimes the Principal will be accessing generic resources that would provide no such clue. It may be the case that a Principal's behavior will be different if they believe are surfing anonymously or pseudonymously ratther than as a verified individual. Consequently, in general, but especially true for passive SSO, a Principal should always be able to determine their current authentication status at both Identity and Service Providers.

- Providers must provide a mechanism to allow a Principal to determine their current status. Such status should include the following information:

  - Their current authentication status.
  - Their current session history.
  - The authenticating IDP(s) (if an SP status display).
  - The chain of IDPs (if an SP status display).
  - The current authentication context.
  - The SP's to which an assertion has been sent (if an IDP status display).

6.2.3. Authentication Strength

Liberty's protocols allow an SP to indicate their preferred mechanism(s) that the IDP should use to authenticate the Principal by specifying the relevant preferred Authentication Context(s). An IDP, in its response, indicates the actual relevant authentication context used. Different authentication technologies differ in the degree to which they bind the authenticating entity to the identity claims they are making. It is expected that SPs will request stronger authentication mechanisms for more sensitive resources.

The general principle that a choice of authentication technology should be commensurate with the value of the resource being accessed can be refined for federated authentication through the following guidelines:

- An SP should request non-anonymous authentication of a Principal only when it is necessary to know the Principal's real identity.
- An SP should request the minimally acceptable authentication context that its access control policies define as sufficient.

6.2.4. IDP Proxying

A proxying IDP mediates SSO between another IDP (that to which the principal actually authenticates) and an SP (which, for a variety of reasons, is unable to deal directly with the first IDP). The proxying IDP consumes the authentication assertion of the original IDP and then creates another for delivery on to the SP. As the Principal's browser will be temporarily sent to the proxying IDP as part of the normal SSO protocol, the proxying IDP has the opportunity to establish an authenticated session for that Principal and set a cookie on their browser to maintain session state.

Although the Principal will necessarily have had to federate their accounts at the authenticating and proxying IDPs (with the implied consent that the proxying IDP can consume authentication assertions issued by the authenticating IDP), as well as those at the proxying IDP and the SP (with the implied consent that the proxying IDP can perform authentication for the SP), this does not necessarily mean that they consent to proxying.

- An IDP must allow Principals to specify their policy for proxying.
  An IDP should allow Principals to specify any of:
  - consent to authorizing proxying.
  - a policy to be notified of proxying.
  - a policy to be prompted for authorization of proxying at run-time.
  - a policy to limit proxying by specifying maximum hops.
- An IDP may give the principal the ability to specify their preferences for proxying on a per provider basis.

In principle, a Principal can specify policy at either the Authenticating IDP or the Proxying IDP. For the former, once the Authenticating IDP determined who the Principal was, it would do a policy lookup and determine whether or not it should proceed with proxying. For the latter, the proxying IDP would do a lookup after it had received the authentication assertion from the authenticating IDP. Consequently, a proxying IDP (like the original SP) would only be able to apply policy once it had sent a request to the authenticating IDP and received the appropriate response.

- An authenticating IDP must apply any applicable proxying policy after authenticating that Principal. If the Principal's policy forbids proxying, the authenticating IDP must not return the requested assertion to the proxying IDP.

- A proxying IDP must apply any applicable proxying policy after receiving the authentication assertion for the Principal from the authenticating IDP. If the Principal's policy forbids proxying, the proxying IDP must not return the requested assertion to the SP (or other proxying IDP).

From the Principal's point of view, they are trying to access a resource at the SP and they log-in at the authenticating IDP in order to do so - being logged in at the proxying IDP through a session being established there is by product of the 'visible' part of the process. Such unforeseen and unrequested authentication at the proxying IDP poses a potential privacy concern if the principal were to subsequently visit the proxying IDP site and assume that they were surfing anonymously. It may however be necessary for the proxying IDP to establish a session for the Principal in order to maintain sufficient state to enable any subsequent SLO.

- If a proxying IDP sets a cookie in the Principal's browser to maintain a session for subsequent SSO, the cookie should not be bound to the Principal's local identity, i.e. it should be anonymous.

- If a proxyingIDP does set a cookie bound to the Principal's local identity in order to establish a session, they should provide mechanisms to allow that principal to determine their SSO status if they visit the site during the lifetime of the session.

Bibliography

[LibertyDeveloperArea] eds. (11 November 2003). "Project Liberty Privacy Public Developer Area," Release 1.0, Liberty Alliance Project *http://www.projectliberty.org/resources/resources.html*

[LibertyProtSchema] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Protocols and Schema Specification," Version 1.2, Liberty Alliance Project (20 January 2004). *http://www.projectliberty.org/specs*

[LibertyBindProf] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Bindings and Profiles Specification," Version 1.2-errata-v1.0, (18 April 2004). *http://www.projectliberty.org/specs*

[LibertyAuthnContext] Madsen, Paul, eds. "Liberty ID-FF Authentication Context Specification," Version 1.2, Liberty Alliance Project (12 November 2003). *http://www.projectliberty.org/specs*

[LibertyMetadata] Davis, Peter, eds. "Liberty Metadata Description and Discovery Specification," Version 1.0, Liberty Alliance Project (12 November 2003). *http://www.projectliberty.org/specs*

[LibertyIDFFOverview] Wason, Thomas, eds. "Liberty ID-FF Architecture Overview," Version 1.2, Liberty Alliance Project (12 November 2003). *http://www.projectliberty.org/specs*

[LibertyGlossary] "Liberty Technical Glossary," Version 1.2-21, v1.2-21 Liberty Alliance Project (27 Feb 2004). *http://www.projectliberty.org/specs* Hodges, Jeff, Wason, Thomas, eds.

[LibertyIDFF12SCR] Tiffany, Eric, eds. Version 1.0-04, Liberty Alliance Project (). *http://www.projectliberty.org/specs*

[SAMLCore11] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (27 May 2003). "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1," OASIS Committee Specification, version 1.1, Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/documents.php?wg_abbrev=security*

[RFC1738] Berners-Lee, T., Masinter, L., McCahill, M., eds. (December 1994). "Uniform Resource Locators (URL)," RFC 1738., Internet Engineering Task Force *http://www.ietf.org/rfc/rfc1738.txt [December 1994]*.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., eds. (June 1999). "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2616.txt [June 1999]*.

[RFC2854] Connolly, Daniel., Masinter, Larry, eds. (June 2000). "The 'text/html' Media Type," RFC 2854, The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2854.txt [June 2000]*.

[RFC2965] Kristol, D., Montulli, L., eds. (October 2000). "HTTP State Management Mechanism," RFC 2965., Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2965.txt [October 2000]*.

[SOAPv1.1] "Simple Object Access Protocol (SOAP) 1.1," Box, Don, Ehnebuske, David , Kakivaya, Gopal, Layman, Andrew, Mendelsohn, Noah, Nielsen, Henrik Frystyk, Winer, Dave, eds. World Wide Web Consortium W3C Note (08 May 2000). *http://www.w3.org/TR/2000/NOTE-SOAP-20000508/*

[XML] Bray, Tim, Paoli, Jean, Sperberg-McQueen, C.M., Maler, Eve, eds. (Oct 2000). "Extensible Markup Language (XML) 1.0 (Second Edition)," Recommendation, World Wide Web Consortium *http://www.w3.org/TR/2000/REC-xml-20001006*

[XMLCanon] Boyer, John, Eastlake, Donald, Reagle, Joseph, eds. (18 July 2002). "Exclusive XML Canonicalization," Recommendation, World Wide Web Consortium *http://www.w3.org/TR/xml-exc-c14n*

[WML] "Wireless Markup Language Version 1.3 Specification," Version 1.3, Open Mobile Alliance *http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html*

[SSL] Frier, A., Karlton, P., Kocher, P., eds. (November 1996). Netscape Communications Corporation "The SSL 3.0 Protocol," *http://www.netscape.com/eng/ssl3/*

[RFC2246] Dierks, T., Allen, C., eds. (January 1999). "The TLS Protocol," Version 1.0 RFC 2246, Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2246.txt [January 1999]*.

[CDT] Schwartz, Ari, eds. (2003). Center for Democracy & Technology "Privacy Principles for Authentication Systems," http://www.cdt.org/privacy/authentication/030513interim.pdf,

[Art29] Rodota, Stefano , eds. (2003). ARTICLE 29 Data Protection Working Party "Working Document on on-line authentication services," http://europa.eu.int/comm/internal_market/privacy/docs/wpdocs/2003/wp68_en.pdf,