Liberty ID-WSF – a Web Services Framework

Editor:

John Kemp, Nokia

Contributors:

Carolina Canales-Valenzuela, Ericsson
Britta Glade, RSA Security, Inc.
Paul Madsen, Entrust
Jason Rouault, Hewlett-Packard Company

Abstract:

This Liberty ID-WSF – a Web Services Framework white paper summarize the benefits of the Liberty (ID-WSF) Web Services Framework.

Contents

1. INTRODUCTION TO WEB SERVICES

The technology used in web services can allow businesses to offer new functionality to their partners and customers, ease existing relationships between disparate computing systems, and provide common interfaces to systems, enabling composite computing systems to be built that span geographical locations and business functions.

Web services technologies allow a uniform interface to be built on top of existing computer systems, offering a programming language-neutral and operating environment-neutral model. What this means, concretely, is that your human resources (personnel) system, running on a Unix mainframe, can talk more easily to your insurance benefits provider (a different company), regardless of whether systems are running on the same operating platform. Web services offer a standard external interface to internal computer systems, allowing more automated, and thus cheaper, interactions between computer systems. Such interfaces may be developed and deployed more quickly than traditional computer systems.

Web services are enabled by the passing of standardized messages between systems. A human resources system, for example, might pass a standard message telling an insurance provider to “enroll employee Lois Lane in XYZ insurance plan.” The insurance provider might then respond, telling the human resources system that it had “successfully added Lois Lane to insurance plan XYZ.”

In order to pass such standard messages, it is necessary for businesses to agree on standard ways of communicating these messages. In order to do that, businesses should typically agree on a framework for implementing web services. The Liberty Identity Web Services Framework (ID-WSF) is such a framework.

This overview introduces the components of the Liberty ID-WSF, and shows how these are used to deliver some of the benefits described above.

Web services based on Liberty ID-WSF are available today -- as are the benefits of Liberty ID-WSF to *all* web services.

2. THE ARCHITECTURE OF A WEB SERVICES FRAMEWORK

A web service may be broken down into a couple of different layers. The first of these is considered the *application layer*. This layer is responsible for the actual service provided by the application. If, for example, you have a human resources interface, the application layer would be responsible for defining the standard message that might be used to enroll an employee in an insurance plan, and the response message that should be received upon enrollment. Such messages, when grouped together, form an *application protocol*.

In addition to the application protocol, some other things are necessary to ensure that these messages can be passed back and forth. An application needs to know where (and how, in concrete terms) to send such a message. Additionally, if the human resources application is offered by one business, and the insurance enrollment system is offered by a different business, then messages will be passing over some network connection between the two businesses. The insurance system will need to know that it can trust that the enrollment of Lois message that it was sent actually came from Lois' company! Not only that, but both companies would probably want any messages that they sent to each other to be protected from the prying eyes of some other company that might be nefariously listening on the network connection present between the two companies.

Now, each application may decide to implement its own individual ways of dealing with these functions. However, given that a particular company may be running many such applications, it makes quite a lot of sense to separate such functionality from the applications themselves, and to write application protocols that utilize a common framework. This framework layer is addressed by the Liberty ID-WSF specifications, and these common functions are listed below:

i) Authentication – as the provider of a web service, I might wish to know who is accessing my service in order to ensure that only authorized users are provided service access. This demands that service requesters are authenticated. I might also wish to know that a message purporting to be from some entity I trust is actually from that entity.

   Authentication depends on the notion of *identity* – who is the person accessing my service, and who is this message *about*?

ii) Message protection mechanisms – both clients and providers of web services would like to know that messages they send cannot be intercepted by a malicious entity and then either modified or cached and then replayed.

iii) Service discovery and addressing – some application wishing to make use of a particular web service will need to discover where that web service is concretely located in order to correctly address messages to the service.

iv) Policy – service providers may have particular requirements that apply to service requesters. These requirements, which can be quite varied, can be grouped in the general category of *policy*. Additionally, individual users of software applications, or the service client software applications themselves, may also have particular policies that they must apply in accessing a service. Policies may be related to the privacy requirements of a user, security requirements of a service provider or client, and many other areas.

v) Common data access protocols – multiple applications might define similar operations. For example, a "query" message could equally apply to an insurance plan in the insurance system (who is enrolled in plan XYZ?) or another system at the same company, such as the corporate address book (what is Lois' phone number?). It is possible to define a standard interface that can then be used and extended by application systems.

vi) Transport protocols – generally, web services are made available over some network. Often, these days, that network is the Internet, and services may thus be offered using the HTTP protocol and carried in a standard SOAP message. Liberty provides a binding of application messages to SOAP that may be carried over HTTP.

Finally, web services may be classified, based on their usage of identity-related information, as *identity-based*, *identity-consuming*, or *basic* (which does not depend on or expose any identity-related information).

An identity-based service application is one that exposes an interface on behalf on my (online) identity. For example, Lois may have her business calendar on the corporate intranet – it is *her* calendar – if you want to know what Lois is doing, you have to access *her* calendar, not just some random calendar!

An identity-consuming service application is one that requires, or is enhanced by, knowledge of some data connected with my identity. For example, if Lois wants to find out the weather forecast for her local weather, the weather forecasting service could be enhanced by knowing Lois' postal code – it doesn't need to know who Lois is, nor is it working specifically on her behalf, it just needs some piece of information about her to give her the local weather forecast.

A basic web service is exemplified by a stock quote service that delivers the current price of a particular company's stock. It does not need to know who the person is who is requesting the quote in order to configure the quote – it just needs to know the name or abbreviation of the company whose stock price is being requested.

The Liberty Web Services Framework supports all of these types of service, as described below.

3. LIBERTY ID-WSF – A WEB SERVICE FRAMEWORK

The Liberty Alliance Project has defined a framework that supports the development of identity-based, identity-consuming, and standard web services, in addition to clients of such services.

3.1 Authentication

The Identity Federation Framework (ID-FF), based on the OASIS SSTC SAML standard [SAML] specifies a third-party authentication model, where individual services rely upon assertions generated by an *identity provider*. Thus, the service is not required to directly authenticate the user (although ID-FF does not prevent this). The direct authentication may be performed by an entity whose sole responsibility is to identity the user based on direct authentication. This model, of course, requires that the service provider trusts the identity provider.

Liberty ID-FF defines a protocol that allows a service provider to generate an authentication request and receive an authentication assertion in response from the identity provider. In addition, Liberty specifies bindings for that protocol, which allow the protocol to be performed in a web-based context (either solely over HTTP, or with some communication using SOAP+HTTP).

In addition to the protections available via ID-FF, Liberty provides standard SOAP-based authentication and single-sign-on service interfaces to an identity provider. These can be used by SOAP-based applications to acquire credentials for use at other service provider applications.

The Liberty Authentication Service allows a SOAP client application to authenticate to the service via any of the authentication methods specified by the IETF's Simple Authentication and Security Layer specification, thus standardizing the authentication methods used.

In all of these cases, the authentication results in a SAML assertion being used to communicate the authentication event.

References: [LibertyProtSchema], [LibertyBindProf], [LibertyAuthn]

3.2 Identity

Once an identity provider has authenticated the user requesting service access, they can claim to know the identity of that user. In concrete terms, this means that the user *has an account* with the identity provider. The service provider may or may not know the identity of the user to whom they are providing service (if they have not, themselves, directly authenticated the user, and found them to *have an account* with the service provider) but they will receive a SAML assertion from the identity provider, attesting to the identity provider's knowledge of the user's identity.

It is possible, regardless of whether an account exists for the user at the service provider, for an identifier to be established between the service provider and the identity provider based upon direct authentication of the user by the identity provider and the user's account with that identity provider. If such an identifier is subsequently re-used by the service provider, then a ***federated name identifier*** is said to exist – shared for some period of time by the identity provider and the service provider. A name identifier could be something such as an email address or a string of digits that uniquely identifies a user to either an identity provider or a service provider.

It should be noted that several concerns may apply to name identifiers. Within the context of a single business enterprise, sharing some commonly used identifier among services, such as an email address, may not cause any concern. However, when sharing a globally-known identifier among separate business entities, the user's privacy may be compromised. A user may be fine with the idea that provider A and provider B both know who she is, but does not want provider C to know the same identifier that provider A and B share (for example, she may wish to offer different email addresses to different providers).

Liberty allows the creation of *opaque* (not necessarily visible to all parties) *privacy-protected* name identifiers. These identifiers may cross business entities without compromising the privacy of the user or leaking data (such as his or her email address).

Given that particular resources (a personal profile document or set of location attributes) may be associated with an identity, Liberty also provides an opaque, privacy-protected *resource identifier* – this combines the concept of a user's identity (and name identifier) with the idea of a specific personal profile resource belonging to *that* named user.

References: [LibertyProtSchema], [LibertyBindProf], [LibertyDisco]

3.3 Message Protection Mechanisms

When a service accepts a request, it will most likely be interested in knowing that the request is a genuine one from some party that it trusts to deliver the request. Liberty specifies ways in which this can be assured. These range from transport security mechanisms, ensuring that the underlying transport is secure (for example, by use of TLS [RFC2246]), to token-based mechanisms (such as the propagation of a SAML assertion in a WS-Security [wss-sms, wss-saml] SOAP header block). In addition, Liberty specifies a SOAP binding ([LibertySOAPBinding]) that includes header blocks that provide *message threading* (so that a message received may be correlated to a message that was sent) and the ability for a message sender to make a claim about the sender's identity, which can be confirmed by the message recipient.

References: [RFC2246], [LibertySecMech], [LibertySOAPBinding], [wss-sms], [wss-saml]

3.4 Service Discovery and Addressing

There are a number of ways in which a service may be discovered. Liberty specifies a discovery service [LibertyDisco] and a protocol and profile by which a discovery service may be accessed by a service requestor. The Liberty framework itself does not require explicit discovery, and other methods (such as UDDI service registry) may be employed, particularly for the discovery of basic web services.

It should be noted that the Liberty discovery service has a special property – it is available to discover services belonging to a particular user, so it is ideal for the discovery of identity-based web services. And, of course, it uses privacy-protected name and resource identifiers to provide that functionality. Ultimately, all such methods must result in a service requester having a) the service *endpoint* to which they should direct a service request, b) a credential that will convince the service provider that the requester should be granted access, and c) any indications of the policy of the service provider that would be required for them to gain access (does the service provider require a particular secure transport, such as SSL/TLS, for example?).

References: [LibertyDisco]

3.5 Policy

The Liberty WSF provides a number of places where policy may be both specified and enforced. Specifically, we provide a *usage directive* SOAP header block so that a particular service request may be handled according to the requested policy, placeholders for policy information pertaining to service access in both the Liberty discovery service, and individual metadata documents related to particular service providers. In addition, policy may be indicated in WSDL [WSDLv1.1] documents associated with a particular application service provider.

Liberty thus does not specify any particular policy language, but provides placeholders where such policy languages may be employed.

References: [WSDLv1.1], [LibertySOAPBinding], [LibertyMetadata], [LibertyDisco]

3.6 Common Data Access Protocols

The Liberty Data Services Template Specification (DST) defines common data access protocols to allow the querying and modification of arbitrary data items according to the application. So, an application may simply use or extend the DST protocol to provide a basic query/modify interface to application clients without having to design or code such functionality itself.

References: [LibertyDST]

3.7 Liberty ID-WSF and the Various Classes of Web Service

As noted above, we classify the general idea of a web service into three classes – identity-based, identity-consuming, and basic. The Liberty framework may be applied in all three cases.

3.7.1 Basic Web Services

Although the Liberty framework is most useful in providing a framework for identity services, it can still be used very effectively to provide framework-level features that are very useful to web service applications. All but the very simplest web services will benefit greatly from the following pieces of the Liberty ID-WSF:

i) Liberty ID-WSF SOAP Binding

SOAP represents the meeting point between standardized messages and the particular method of transporting those messages over a network. SOAP specifies a message envelope, which contains *message headers* that may be used for conveying non-application data to a service, and a *message body* that contains the application message.

The Liberty ID-WSF SOAP Binding Specification provides a number of SOAP headers that may be used to provide facilities such as message threading (the idea that an application may want to know that a message it receives is related to one that it sent) and the securing of a SOAP message via the OASIS WS-Security specification. There are additional SOAP headers provided to indicate particular policies that might apply to the usage of a service and that also allow a service application to perform load-balancing operations at the SOAP level.

The Liberty ID-WSF SOAP Binding provides basic messaging infrastructure to your application, *regardless of whether the application requires identity*! Thus, the Liberty SOAP Binding may be used equally by both the stock quote service and Lois' calendar service, and can be used solely to help scale a robust web services application as well.

ii) Liberty Security Mechanisms

The security mechanisms offered by the Liberty framework may be used to protect SOAP messages in a standard, interoperable form. Application service providers can standardize on the use of such mechanisms and communicate those mechanisms to service clients. They offer standard profiles of SOAP message security, utilizing SAML assertions and XML digital signatures to secure messages.

An example of an authenticated stock quote request carried in a Liberty-compliant SOAP message might look like this:

```
<?xml version="1.0" encoding="utf-8" ?>
<S11:Envelope>

xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/";
        xmlns:sb="urn:liberty:sb:2003-08"
        xmlns:wsse="…">

  <S11:Header>

    <!-- Liberty ID-WSF SOAP Binding Correlation header -->

    <sb:Correlation S:mustUnderstand="1"
      messageID="uuid:76514281-38298263-290739173"
      timestamp="2004-05-06T11:14:21Z"/>

    <!-- Liberty-specified security token for securing the
message -see Liberty Security Mechanisms for examples -->

    <wsse:Security>
       …
    </wsse:Security>

  </S11:Header>

  <S11:Body>

    <!-- The actual (secure) stock quote request -->

    <GetQuote xmlns="http://sq.example.com/">
        <Symbol>Liberty</Symbol>
    </GetQuote>

  </S11:Body>

</S11:Envelope>
```

3.7.2 Identity-Based and Identity-Consuming Web Services

In addition to the basic framework provided for all web services, identity services require that messages be delivered to a service offered on behalf of some identity, such as my personal profile where my address is stored. Thus, they have to identify me in some way. Of course, they should identify me in some way such that only the application client and the application service know that it is me whose profile is being requested or modified. Liberty uses the ***resource identifier*** to label that identity. Service clients may also need to discover the location of *my* profile service, in which case, they could contact *my* discovery service in order to acquire a resource identifier for that profile service. As, in many cases, such services may be accessed via query and/or modify type operations, the Liberty DST protocol might be used to access an identity-based service, as shown below, where the client, an identity-consuming weather-forecasting web service, wishes to determine my postal code in order to deliver me a weather forecast for the area where I live (***note:*** a UsageDirective header block is shown in this example, which indicates the requester's privacy policy regarding this data access):

```
<?xml version="1.0" encoding="utf-8" ?>
<S11:Envelope>

xmlns:S="http://schemas.xmlsoap.org/soap/envelope/";
        xmlns:idpp="urn:liberty:id-sis-pp:2003-08"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#";
        xmlns:sb="urn:liberty:sb:2003-08"
        xmlns:wsse="…"
        xmlns:saml="…">

  <S11:Header>

    <!-- Liberty ID-WSF SOAP Binding Correlation header -->

    <sb:Correlation S:mustUnderstand="1"
```

```
      messageID="uuid:76514281-38298263-290739173"
      timestamp="2004-05-06T11:14:21Z"/>

    <!-- Liberty UsageDirective, indicating the requester’s
policy regarding the requested data -->

    <sb:UsageDirective mustUnderstand="1"
ref="#postalCode">
      <weather:PrivacyPolicy xmlns:weather="…">
        http://weather.example.com/policy/data-privacy.html
      </weather:PrivacyPolicy>
    </sb:UsageDirective>

    <!-- Liberty assertion carried in a WS-Security header
block -->

    <wsse:Security>

      <!—The following authenticates the WSC to the profile
service provider -->
      <!-- In this case, the WSC is representing some user who
was authenticated at some IdP -->

      <saml:Assertion MajorVersion="1" MinorVersion="1"
        AssertionID="ad42093f-4899-9b3a-5471-124ba3e4e8f0"
        Issuer="http://idp.example.com/";
        IssueInstant="2004-05-06T10:10:09Z"
        InResponseTo="uuid:00823431-72625431-23839056">

        … <!-- additional authentication details -->

      </saml:Assertion>

    </wsse:Security>
  </S11:Header>

  <S11:Body xmlns="urn:liberty:id-sis-pp:2003-08">

    <Query>

      <!-- Liberty ResourceID -->

      <ResourceID>
        http://profile.example.com/659gft565
      </ResourceID>

      <QueryItem>
        <Select id="postalCode">
          /IDPP/AddressCard/Address/PostalCode
        </Select>
      </QueryItem>

    </Query>

  </S11:Body>

</S11:Envelope>
```

Of course, in addition to the benefits detailed above, web services that conform to these various Liberty standards can also be implemented as conformant to the WS-I Basic Profile specification, and have WSDL service description documents.

It should be noted that Liberty provides a working framework that can be used for access to all web service applications. However, certain services may require other capabilities not provided directly by the Liberty ID-WSF specifications. In such cases, an application may, of course, implement the Liberty framework to provide the functionality detailed in this paper, in addition to any other functionality necessary for that application. As such, the Liberty specifications may be combined with other specifications.

4. CONCLUSIONS

Broadly speaking, there are three classes of web service – identity-based (my profile service), identity-consuming (your localized weather forecast) and basic (a general stock quote service).

The Liberty Identity Web Services Framework provides functionality to address the security and basic reliable messaging functions of any such web service. Additionally, a general framework is available for indicating policy that might apply to service access. Finally, and perhaps most usefully, a mechanism (the opaque name and resource identifier) is provided to offer privacy-protected identity-based services, and access to such services, providing the opportunity to create personalized services without compromising the privacy of individuals.

In today's marketplace, web services play a critical role in enabling companies to easily and cost-effectively do business with trusted partners and customers, without compromising security, visibility or control over identity information. The Liberty Identity Web Services Framework is currently helping many companies to successfully implement federated identity-management projects. Already, AOL, Nokia, and Vodafone are among several member companies with plans to support ID-WSF under the Phase 2 Liberty specifications in existing or new products and services.

Under a new arrangement with D-Link, Radio@AOL -- the No. 1 Internet radio broadcaster -- and You've Got Pictures will be available to AOL's 31.5 million subscribers as well as non-AOL subscribers.

A prototype developed jointly between AOL and Nokia demonstrates how AOL employs ID-WSF specifications - particularly the authentication, discovery, and permission-based attribute sharing and security features, to enable any consumer to access and personalize the Radio@AOL service using their Nokia mobile handset.

Vodafone, among the world's largest mobile telecommunications network companies, has collaborated with Trustgenix and Gamefederation to build a Liberty-enabled multiplayer mobile gaming proof-of-concept. Using Liberty as the authentication mechanism, a user can discover a game site over Vodafone's network, access it, and personalize his or her experience.

These projects illuminate Liberty ID-WSF's ability to bridge fixed and mobile Internet services. ID-WSF also underscores that an identity-aware framework allows communities to provide highly personalized and attractive services while improving the service's usability. Deployments such as those by AOL, Nokia, and Vodafone illustrate how Liberty supports the entire web services ecosystem, including mobile devices.

For more information on ID-WSF, developers should visit the Liberty Alliance Web site at http://www.projectliberty.org.

5. BIBLIOGRAPHY

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

[LibertySOAPBinding] Hodges, Jeff, Aarts, Robert, eds. “Liberty ID-WSF SOAP Binding Specification,” Version 1.0, Liberty Alliance Project (12 November 2003). http://www.projectliberty.org/specs

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

[LibertyDisco] Sergent, Jonathan, ed. “Liberty ID-WSF Discovery Service Specification,” Version 1.0-08, Liberty Alliance Project (24 July 2003). http://www.projectliberty.org/specs

[LibertyDST] Kainulainen, Jukka, Ranganathan, Aravindan, eds. “Liberty ID-WSF Data Services Template Specification,” Version 1.01-03-errata, Liberty Alliance Project (15 January 2004. http://www.projectliberty.org/specs

[LibertySecMech] Ellison, Gary, ed. “Liberty ID-WSF Security Mechanisms,” Version 1.0, Liberty Alliance Project (12 November 2003). http://www.projectliberty.org/specs

[LibertyAuthn] Hodges, Jeff, Aarts, Robert, eds. “Liberty ID-WSF Authentication and Single Sign-on Services Specification,” Version 1.0-18, Liberty Alliance Project (12 May 2004). http://www.projectliberty.org/specs

[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

[wss-sms] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (June 30, 2003). Organization for the Advancement of Structured Information Standards “Web Services Security: Draft WSS-SOAPMessageSecurity-14-063003” http://www.oasis-open.org/committees/download.php/2757/WSS-SOAPMessageSecurity-14-063003.pdf

[wss-saml] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (May 5, 2003). Organization for the Advancement of Structured Information Standards “Web Services Security: SAML Token Profile,” Draft WSS-SAML-07.pdf” http://www.oasis-open.org/committees/download.php/1911/WSS-SAML-07.pdf

[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.

[WSDLv1.1] Christensen, Erik, Curbera, Francisco, Meredith, Greg, Weerawarana, Sanjiva, eds. (15 March 2001) "Web Services Description Language (WSDL) 1.1," World Wide Web Consortium W3C Note http://www.w3.org/TR/2001/NOTE-wsdl-20010315.