Liberty ID-WSF: SOAP Binding

Version: 1.0-04

Editors:
Jeff Hodges, Sun Microsystems, Inc
Robert Aarts, Nokia

Contributors:
Marc Hadley, Sun Microsystems, Inc
Jonathan Sergent, Sun Microsystems, Inc

Abstract:

This specification defines the Liberty Identity Web Services Framework (ID-WSF) SOAP binding. It specifies simple SOAP message correlation, consent claims, and usage directives.

Notice

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

Revision History

Revision: 04 **Date:** 14 Apr 2003

Fixes bugs 65, 177, 178 (mostly). Consent claim section added. Redesigned correlation header blocks to be a single block used for every message. updated processing rules accordingly Added examples. Updated "notation and convention" section. Editorially cleaned up UsageDirectives section. Made rev #s on this page be one-based such that they are congruent with coordinating editor's rev#s.

Revision: 03 **Date:** 28 Mar 2003

post IAH. Fixes bugs 40, 64, 65, 66. Various editorial repairs scattered throughout spec.

Revision: 02 **Date:** 28 Mar 2003

post MIA.

Revision: 01 **Date:** 28 Mar 2003

initial, post PHX.

Contents

1. Introduction

Liberty Identity Web Services Framework (ID-WSF) and Service Interface Specification (ID-SIS) messages, collectively referred to as *"ID-* messages"* herein, are designed so that they may be mapped onto various transport or transfer protocols. Thus, they are designed to be conveyed in the data portion of whichever underlying protocol they are mapped onto. ID-* messages do not, unto themselves, address specific aspects of message exchange such as: to whom the message is to be sent, message correlation, the mechanics of message exchange, or security context.

Examples of ID-* messages are: the Discovery Service's `<DiscoveryLookupRequest>` message [LibDiscoSvc], and the Personal Profile's `<Modify>` message [LibPersProf].

This specification defines a mapping of ID-* messages onto SOAP [SOAP], an XML-based messaging protocol.

SOAP itself does not define the specific message exchange aspects mentioned above, but offers an *extensibility model* that may be used to define message components that do address such message exchange specifics. SOAP extensibility is effected by adding message components to the portion of the SOAP message called the *Header*. These components are commonly referred to as *SOAP header blocks*.

This specification defines three SOAP header blocks addressing ID-* message exchange specifics. The functionalities they address are:

- Message Correlation: SOAP does not define a mechanism to correlate one SOAP message with another message, such as in a request-response paradigm.
- Consent Claims: It is a Liberty requirement for ID-WSF-based entities to be able to claim whether they obtained the Principal's consent for carrying out various operations, such as federating the Principal's accounts [LibArchOV].
- Usage Directives: It is also a Liberty requirement for ID-WSF-based entities to be able to specify their policies for handling data at the time of data request, and for entities releasing data to specify their policies for the subsequent use of data at the time of data release.

This specification defines how ID-* messages are mapped into SOAP message bodies, and how SOAP header blocks implementing the above functionalities are mapped into SOAP message headers.

This specification is organized as follows, after the present introductory section:

> Some utility types are presented first. Mapping ID-* messages into SOAP message bodies is discussed. Then, three header blocks addressing the topics of message correlation, consent claims, and usage directives are presented. Each header block section discusses the header block type, and the applicable processing rules for that type.

1.1. Notation and Conventions

This specification uses schema documents conforming to W3C XML Schema (see [Schema1]) and normative text to describe the syntax and semantics of XML-encoded protocol messages.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119:

> "they MUST only be used where it is actually required for interoperation or to limit behavior which has potential for causing harm (e.g., limiting retransmissions)"

These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

This specification references the following XML namespace prefixes:

- The prefix sb: stands for the Liberty (working) namespace, (urn:liberty:wsf:soap-bind:1.0). This namespace is the default for instance fragments, type names, and element names in this document.
  Note: TODO: finalize the namespace URN.
- The prefix S: stands for the SOAP namespace (http://www.w3.org/2001/12/soap-envelope).
- The prefix xs: stands for the W3C XML schema namespace (http://www.w3.org/2001/XMLSchema).
- The prefix saml: stands for the OASIS SAML v1.0 schema namespace (urn:oasis:names:tc:SAML:1.0:assertion).

This specification uses the following typographical conventions in text:

- `<Element>`
- `<ns:ForeignElement>`, where "ns" is a namespace prefix given in the list above.
- `Attribute`
- `ns:ForeignAttribute`, where "ns" is a namespace prefix given in the list above.
- **datatype**.

For readability, when an XML Schema type is specified to be **xsd:boolean**, this document discusses the values as `true` and `false` rather than "1" and "0", which will exist in a document instance conforming to the SOAP Envelope 1.1 schema [SOAP1.1schema].

Definitions for Liberty-specific terms can be found in [LibGloss].

1.2. String and URI Values

All string and URI [RFC2396] values in this specification have the types **string** and **anyURI** respectively, which are built in to the W3C XML Schema Datatypes specification [Schema2]. All strings in ID-WSF messages MUST consist of at least one non-whitespace character (whitespace is defined in the XML Recommendation [XML] section 2.3). Empty and whitespace-only values are disallowed. Also, unless otherwise indicated in this specification, all URI values MUST consist of at least one non-whitespace character.

1.3. Time Values

All time values in this specification have the type **dateTime**, which is built in to the W3C XML Schema Datatypes specification [Schema2] and MUST be expressed in UTC form.

Senders and receivers SHOULD NOT rely on other applications supporting time resolution finer than milliseconds. Implementations MUST NOT generate time instants that specify leap seconds.

2. ID Types

The **IDType** simple type is used to declare message identifiers. The **IDReferenceType** is used to reference identifiers of type **IDType**.

Values declared to be of type **IDType** MUST satisfy the following properties:

- Any party that assigns a message identifier MUST ensure that there is negligible probability that that party or any other party will accidentally assign the same identifier to a different data object.
- Where a data object declares that it has a particular identifier there MUST be exactly one such declaration.

The mechanism by which ID-* senders or receivers ensure that an identifier is unique is left to implementations. In the case that a pseudorandom technique is employed, the probability of two randomly chosen identifiers being identical MUST be less than 2^-128 and SHOULD be less than 2^-160. This requirement MAY be met by applying Base64 encoding to a randomly chosen value 128 or 160 bits in length.

It is OPTIONAL for an identifier based on **IDType** to be resolvable in principle to some resource. In the case that the identifier is resolvable in principle (for example, the identifier is in the form of a URI reference), it is OPTIONAL for the identifier to be dereferenceable.

The following schema fragment defines the **IDType** and **IDReferenceType** simple types:

Figure 1. IDType and IDReferenceType Schema Fragment

```
<simpleType name="IDType">
    <restriction base="string"/>
</simpleType>

<simpleType name="IDReferenceType">
    <restriction base="string"/>
</simpleType>
```

3. SOAP Binding

This section defines the versions of SOAP and WSDL used, and how ID-* messages are mapped into the <Body> element of SOAP messages.

3.1. SOAP and WSDL Versions

This specification normatively depends upon both SOAP1.1, as specified in [SOAP1.1].and upon WSDL 1.1, as specified in [WSDL1.1].

3.2. Mapping ID-* Messages onto SOAP Messages

ID-* messages are concretely bound to SOAP messages by placing the ID-* message of interest in a <S:Body> element, itself a child of a <S:Envelope> element. See the below figure. The message's header will contain at least a <sb:Correlation> header block. Processing rules for constructing and sending, plus receiving and parsing, ID-* SOAP-based messages are given in the next section below.

Example 1. Mapping an ID-* Message into a SOAP Message Body

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:idpp="rn:liberty:idpp:1.0">

 <S:Header>
: <!- ID-WSF header blocks, plus various others, go here. ->
 </S:Header>

 <S:Body>

  <idpp:Modify> <!- This is an ID-PP "Modify" message mapped ->
       :     <!- into the <Body> of a SOAP message.     ->
  </idpp:Modify>

 </S:Body>

</S:Envelope>
```

4. Message Correlation

This section defines the ID-* message correlation facilities.

4.1. The `correlationType` Header Block Type

The `correlationType` header block type provides a means for correlating ID-* messages. Message correlation is achieved by inserting a `<Correlation>` element in ID-* message headers and using the `messageId` attribute to identify individual messages. Additionally, a message may refer to another message by setting its `refToMessageId` attribute to the value of the `messageId` of the message of interest.

The `correlationType` header block type has the following attributes:

- `messageId` [Required] – The unique ID of the message.
  Note: The MessageId is intended to act as a *nonce*. Thus it is subject to specific processing rules outlined below in Message Processing Rules, and the value-specific guidance given above in ID Types.
- `refToMessageId` [Optional] – A copy of the `messageId` attribute value of the message being responded to, or being correlated with in some application-specific fashion, if any.
- `timestamp` [Optional] – The time the sender sent the message. Note restrictions on time values outlined in Time Values above.
- `id` [Optional] – identifies a `<Correlation>` header block instance. This attribute MUST be used when the message is signed as described in [LibSecMech], and the element instance is to be included in the signed message components.
- `S:mustUnderstand` [Optional] – The SOAP mustUnderstand attribute [SOAP].
- `S:actor` [Optional] – The SOAP actor attribute [SOAP].

Figure 2. The following schema fragment defines the `Correlation` header block type:

```
<xs:complexType name="correlationType">

  <xs:attribute name="messageId" type="IDType" use="required"/>
  <xs:attribute name="refToMessageId" type="IDType" use="optional"/>
  <xs:attribute name="timestamp" type="xs:dateTime" use="optional"/>
  <xs:attribute name="id" type="xs:ID" use="optional"/>

  <xs:attribute ref="S:mustUnderstand" use="optional"/>
  <xs:attribute ref="S:actor" use="optional"/>

</xs:complexType>
```

4.2. `<Correlation>` Header Block Element

The `<Correlation>` header block implements the features of the `correlationType` header block type, described above.

Figure 3. The following schema fragment defines the `<Correlation>` element:

```
<xs:element name="Correlation" type="correlationType"/>
```

Example 2. A `<Correlation>` Header Block

```
<sb:Correlation s:mustUnderstand="1"
            id="A13454...245"
            actor="http://schemas.../next"
            MessageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
            Timestamp="2112-03-15T11:12:12Z"/>
```

4.3. Correlation Examples

The following example illustrates an ID-* message, containing a `<Correlation>` header block, and conveying a Personal Profile (ID-PP) Modify message [LibPersProf].

Example 3. An ID-* Message with a Correlation Header Block

```
<?xml version="1.0" encoding="UTF-8"?>

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:sb="urn:liberty:wsf:soap-bind:1.0"
        xmlns:idpp="rn:liberty:idpp:1.0">

  <s:Header>
```

```
    <sb:Correlation s:mustUnderstand="1"
                id="A13454...245"
                actor="http://schemas.../next"
                MessageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
                Timestamp="2112-03-15T11:12:12Z"/>

    <!- other header blocks, eg wsse:security, go here ->

  </s:Header>

  <S:Body>

    <idpp:Modify>
         :     <!- this is an ID-PP Modify message ->
    </idpp:Modify>

  </S:Body>

</S:Envelope>
```

The following example illustrates an ID-* message, containing a `<Correlation>` header block referring to the message in the previous example. This message is conveying an ID-PP ModifyResponse message. Note how the `RefToMessageId` attribute is referencing the `MessageId` in the example above.

Example 4. An ID-* Message with a `Correlation` Header Block and Referencing a Another Message

```
<?xml version="1.0" encoding="UTF-8"?>

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:sb="urn:liberty:wsf:soap-bind:1.0"
        xmlns:idpp="rn:liberty:idpp:1.0">

  <s:Header>

    <sb:Correlation s:mustUnderstand="1"
                id="B893483..83736"
                actor="http://schemas..../next"
                MessageId="uuid:aaaaeeee-efef-fefe-efef-abababababab"
                RefToMessageId=
                  "uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
                Timestamp="2112-03-15T11:12:13Z"/>

    <!- other header blocks, eg wsse:security, go here ->

  </s:Header>

  <s:Body>

    <idpp:ModifyResponse>
          :     <!- this is an ID-PP ModifyResponse message ->
    </idpp:ModifyResponse>

  </s:Body>

</s:Envelope>
```

4.4. Message Processing Rules

Processing of the <Correlation> header block follows the rules of the SOAP processing model described in [SOAP]. The SOAP mustUnderstand and actor attributes MAY be used to mandate processing and target the header block respectively. Where applicable, specific processing rules for these attributes are given below.

4.4.1. Constructing and Sending an ID-* Request Message

An ID-* *request message* is constructed by including a <Correlation> header block in the <S:Header> element of the SOAP message, and omitting the refToMessageId attribute of the <Correlation> header block (hereafter referred to as the *request <Correlation> header block*). The system entity who constructs and sends such a request message is called the *sender*. The sender MAY include the timestamp attribute, whose value SHOULD be set to the time at which the message was sent. See conventions on dateTime values discussed above.

Additionally, the sender MUST include:

- id attribute on the request <Correlation> header block if the sender is signing the message [LibSecMech].
- The S:mustUnderstand attribute MUST be present and its value SHOULD be TRUE.

The soap sender MAY include other header blocks in the message along with the request <Correlation> header block, as required by the overall processing context. For example the sender may include a <Security> header block [LibSecMech].

The sender adds the ID-* service message (see, for example, request messages defined in [LibPersProf, LibDiscoSvc]) to the request message <S:Body>, performs any needed further processing of the message, for example including other header blocks and signing it, and then sends the message to the receiver.

4.4.2. Receiving and Processing an ID-* Request Message

A system entity receiving a ID-* request message is called the *receiver*. The receiver MUST perform the following checks on the request <Correlation> header block and its contents:

- Ensure that the value of the S:mustUnderstand is TRUE. Discard the message otherwise.
- Ensure that the value of the S:actor attribute is either "http://schemas.xmlsoap.org/soap/actor/next" or some other agreed-upon value. Discard the message otherwise.
- Ensure that the incoming inv:messageId attribute has not been seen before. If it has, then this message is likely a duplicate. Receivers SHOULD maintain a list of received inv:messageId values. The length of time values are retained is a matter of local policy. Duplicate messages SHOULD be discarded and the event logged.
- If the message has not been discarded, it SHOULD be dispatched for further processing as appropriate based on the ID-* message contained in the <S:Body> portion of the SOAP message.

4.4.3. Constructing and Sending an ID-* Response Message

A system entity receiving a ID-* request message is called the *receiver*. When a receiver sends a response to a received ID-* request message, the receiver MUST insert a <Correlation> header block in the response message (hereafter referred to as a *response <Correlation> header block*) as follows:

- A messageId attribute MUST be included in the response <Correlation> header block with a new unique value.

- If the response message is not generated in response to a request, or if the `messageId` of a request cannot be determined (e.g. because the request is malformed), then the refToMessageId attribute MUST NOT be present. Otherwise, it MUST be present and match the value found in the corresponding request `messageId` attribute.

- The `S:mustUnderstand` attribute MUST be present and its value SHOULD be TRUE.

- A `timestamp` attribute MAY be included in the response `<Correlation>` header block. Its value SHOULD be set to the time at which the response message was sent. See conventions on dateTime values discussed above.

- The receiver adds the ID-* service response message to the response message `<S:Body>`, performs any needed further processing of the message, for example including other header blocks and signing it, and then sends the message to the original sender.

4.4.4. Receiving and Processing an ID-* Response Message

The original sender receives the response message and performs the following checks on the `<Correlation>` header block and its contents:

- Ensure that the value of the S:mustUnderstand is TRUE. Discard the message otherwise.

- Ensure that the value of the `S:actor` attribute is either "http://schemas.xmlsoap.org/soap/actor/next" or some other agreed-upon value. Discard the message otherwise.

- Ensure that the incoming `refToMessageId` attribute value matches a `messageId` attribute value of a prior message sent by the original sender. If not, then the incoming response message SHOULD be discarded and the event logged. Also, the `messageId` attribute value of the incoming response message should not match that of any prior messages. Receivers of responses SHOULD maintain a list of received `messageId` attribute values. How long values are retained is a matter of local policy. Duplicate messages SHOULD be discarded and the events logged.

- If the message has not been discarded, it SHOULD be dispatched for further processing as appropriate.

5. The <Consent> Header Block

This section defines the <Consent> header block. This

5.1. The consentType Header Block Type

The <Consent> header block element MAY be employed by either a requester or a receiver. For example, the Principal may be using a Liberty-enabled client or proxy (common in the wireless world), and in that sort of environment the mobile operator may cause the Principal's terminal (aka: cell phone) to prompt the principal for consent for some interaction.

The consentType header block type has the following attributes:

- uri [Required] – A URI indicating that the Principal's consent was obtained.
  Optionally, the URI MAY identify a particular *Consent Agreement Statement* defining the specific nature of the consent obtained.
  This specification defines one well-known URI Liberty implementors and deployers MAY use to indicate positive Principal consent was obtained with respect to whatever ID-* interaction is underway or being initiated. This URI is known as the "Principal Consent Obtained" URI (PCO). The value of this URI is:
  D1 urn:liberty:consent:obtained This URI does not correspond to any particular Consent Agreement Statement. Rather, it simply states that consent was obtained. The full meaning and implication of this will need to be derived from the execution context.

- timestamp [Optional] – For announcing the time at which the sender obtained agreement with the POC.

- id [Optional] – identifies a <Consent> header block instance. This attribute MUST be used when the message is signed as described in [LibSecMech], and the element instance is to be included in the signed message components.

- S:mustUnderstand [Optional] – The SOAP mustUnderstand attribute [SOAP].

- S:actor [Optional] – The SOAP actor attribute [SOAP].

Figure 4. The following schema fragment defines the `Consent` header block type:

```
<xs:complexType name="consentType">

  <xs:attribute name="uri" type="xs:anyURI" use="required"/>
  <xs:attribute name="timestamp" type="xs:dateTime" use="optional"/>
  <xs:attribute name="id" type="xs:ID" use="optional"/>

  <xs:attribute ref="S:mustUnderstand" use="optional"/>
  <xs:attribute ref="S:actor" use="optional"/>

</xs:complexType>
```

5.2. <Consent> Header Block Element

Figure 5. The following schema fragment defines the <Consent> element:

```
<xs:element name="Consent" type="consentType"/>
```

Example 5. An example request message with an attached Consent header block

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:idpp="rn:liberty:idpp:1.0">

  <S:Header>

    <sb:Consent id="A124395732495743"
      uri="urn:liberty:consent:obtained"
      timestamp="2112-03-15T11:12:10Z"/>

    <sb:Correlation s:mustUnderstand="1"
                id="B893483..83736"
                actor="http://schemas..../next"
                MessageId="uuid:eeeecccc-efef-fefe-efef-bddbbddb"
                Timestamp="2112-03-15T11:12:13Z"/>

  </S:Header>

  <S:Body>

    <idpp:Modify> <!- This is an ID-PP "Modify" message mapped ->
         :     <!- into the <Body> of a SOAP message.     ->
    </idpp:Modify>


  </S:Body>

</S:Envelope>
```

6. Usage Directives

6.1. Overview

Participants in the ID-WSF framework may need to indicate the privacy policy associated with a message. To this end both ID-WSF servers and clients may add one or more `<UsageDirective>` header blocks to the SOAP Header. A `<UsageDirective>` appearing in a *request* message from a client expresses "intended usage". A `<UsageDirective>` appearing in a *response* expresses "how" data is to be used. A `<UsageDirective>` in a reponse message containing no actual data, a fault response for example, may be used to express acceptable policy.

6.2. UsageDirective Header Type and Element

A participant in the ID-WSF framework can add a `<UsageDirective>` element to the SOAP header. This element is based upon the `UsageDirectiveType` which contains:

- `ref` [Required] – An attribute that refers to the part of the message to which this directive applies.
- `id` [Optional} – An attribute facilitating references to elements of this type.
- `S:mustUnderstand` [Optional] – The SOAP mustUnderstand attribute [SOAP].
- `S:actor` [Optional] – The SOAP actor attribute [SOAP].
- `<element(s)>` [Optional] – Elements, encoded in some policy expression language, that express the actual policy or directive. The `ref` attribute above points at the element in the overall message to which the policy expressed in this element applies.

Figure 6. The following schema fragment defines the <UsageDirective> header type and element:

```
<element name="UsageDirective" type="UsageDirectiveType" />
<complexType name="UsageDirectiveType">
   <sequence>
      <xs:any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="ref" type="reference" use="required"/>
   <attribute name="id" type="id" use="optional"/>
   <attribute ref="S:mustUnderstand" use="optional"/>
   <attribute ref="S:actor" use="optional"/>
</complexType>
```

Example 6. An example of a request for the address of a principal:

```
 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
         xmlns:idpp="rn:liberty:idpp:1.0">

   <S:Header>

      <UsageDirective
            id="directive1000"
            ref="#datarequest001"
            mustUnderstand="1">

         <cot:PrivacyPolicyReference
             xmlns:cot="http://circle-of-trust.com/isf">
             http://circle-of-trust.com/policies/eu-compliant
         </cot:PrivacyPolicyReference>

      </UsageDirective>

   </S:Header>

   <S:Body>
      <Query id="datarequest001" xmlns="urn:liberty:pp:1.0">
         <Resource>data:d8ddw6dd7m28v628</Resource>
         <QueryItem>
            <Select>/pp:IDPP/pp:IDPPAddressCard</Select>
         </QueryItem>
      </Query>
   </S:Body>

</S:Envelope>
```

6.3. Processing Rules

The constructor of a <UsageDirective> header block MUST ensure that the value of the ref attribute is set to the ID of an appropiate element in the message. The constructor SHOULD ensure that the <UsageDirective> is integrity-protected. The protection mechanism must be in accordance with the *ID-WSF Security Profiles*.

A recipient of a <UsageDirective> header block MUST check the actor attribute and determine if it is in the role that requires processing the header. If so, the recipient MUST check the mustUnderstandattribute. If set to true the recipient MUST process the contents. If the attribute is absent or set to false the recipient SHOULD attempt to process the content of the <UsageDirective>.

A recipient that processes the contents of a <UsageDirective> header block SHOULD verify the integrity of the header block (e.g. verify any digital signatures). The recipient MUST verify that the ref attribute refers to an element in the message. That recipient must further process the message according to the policy expressed by the children elements of the <UsageDirective> header block. Those children elements will be imported from a foreign namespace, and MUST be parsed and interpreted according to the applicable schema and processing rules of that foreign namespace.

A recipient that cannot process a <UsageDirective> with mustUnderstand="true" MUST respond with a <S:Fault>. The S:Fault MUST contain a <S:Detail> element which in turn MUST contain a <Status> element with its <StatusCode> set to sb:CannotHonourUsageDirective. The <Status> element SHOULD include a ref attribute with its value set to the idof the offending lt;UsageDirective>header block.

Note:

TODO: add the ref attribute to the utility Status code element.

A recipient that cannot honour a non-mandatory (with mustUnderstand="false") <UsageDirective> must respond according to the contained policy. In addition, in this case the recipient MAY respond with a ID-WSF message that includes a <Status element with its <StatusCode> set to sb:CannotHonourUsageDirective. This <Status> element instance SHOULD include a ref attribute with its value set to the idof the <UsageDirective> that could not be honoured.

In its response message, the recipient MAY include one or more new <UsageDirective> headers that each express a policy that the recipient would have been able to honour. The ref attribute of these headers SHOULD be set to the messageID of the <Response> element.

7. References

Bibliography

[LibArchOV] "Liberty Architecture Overview,"

[LibDiscoSvc] "Liberty ID-WSF: Discovery Service,"

[LibGloss] "Liberty Architecture Glossary,"

[LibPersProf] "Liberty ID-WSF: Personal Profile,"

[LibSecMech] "Liberty ID-WSF: Security Mechanisms,"

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels,"

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax,"

[Schema1] "XML Schema Part 1: Structures,"

[Schema2] "XML Schema Part 2: Datatypes,"

[SOAP] "Simple Object Access Protocol (SOAP) 1.1," W3C Note

[SOAP1.1schema] "SOAP 1.1 Envelope schema," W3C Note http://schemas.xmlsoap.org/soap/envelope/,

[WSDL] "Web Services Description Language (WSDL) 1.1," W3C Note

[XML] "Extensible Markup Language (XML) 1.0 (Second Edition),"