Liberty ID-WSF Interaction Service Specification

Version: 2.0-errata-v1.0

Editors:
Robert Aarts, Nokia Corporation
Paul Madsen, NTT

Contributors:
Darryl Champagne, IEEE-ISTO
Gael Gourmelen, France Telecom
John Kemp, Nokia Corporation
Eric Lexcellent, France Telecom
Rob Lockhart, IEEE-ISTO
Jonathan Sergent, Sun Microsystems, Inc.
Greg Whitehead, Hewlett-Packard

Abstract:

It is often necessary for providers of identity-based web services to interact with principsls (e.g., the owners of the identity data exposed by such services or the individuals on whose behalf the request is being made). Typically, a principal is not visiting the identity service provider but some other party, known as a *web services consumer*. The web services consumer invokes a service located at the identity service provider. This specification defines an *interaction service*; this is an identity service that allows providers to pose simple questions to principals in order to, for instance, clarify that principal's preferences for data sharing, or to supply some needed attribute. This service can be offered by trusted web services consumers, or by a dedicated interaction service provider that has a reliable means of communication with the relevant principals.

Filename: liberty-idwsf-interaction-svc-2.0-errata-v1.0.pdf

Notice

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

Contents

1. Notation and Conventions

This specification uses schema documents conforming to W3C XML Schema (see [Schema1-2]) and normative text to describe the syntax and semantics of XML-encoded 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 [RFC2119].

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.

The following namespaces are referred to in this document:

- The prefix *is:* stands for the ID-WSF working namespace for the interaction service (*urn:liberty:is:2006-08*). This namespace is the default for instance fragments, type names, and element names in this document.
- The prefix *disco:* stands for the ID-WSF working namespace for the [LibertyDisco] (*urn:liberty:disco:2006-08*).
- The prefix *sb:* stands for the ID-WSF working namespace for the [LibertySOAPBinding] (*urn:liberty:sb:2006-08*).
- The prefix *S:* stands for the SOAP 1.1 ([SOAPv1.1]) namespace ().
- The prefix *wsa:* stands for the WS-Addressing [WSAv1.0] namespace ().
- The prefix *wsdl:* stands for the primary [WSDLv1.1] namespace ().
- The prefix *xs:* stands for the W3C XML schema namespace ().
- The prefix *xsi:* stands for the W3C XML schema instance namespace ().

2. Overview

It may sometimes be necessary for an *identity* service to interact with the owner of the *resource* that it is exposing, to collect attribute values, or to obtain permission to share the data with a *Web Services Consumer* (WSC). Additionally, in situations where the individual on whose behalf the request is being made is not the resource owner (e.g so called "cross-principal" interactions), the *identity* service may need to interact with either or both principals. The interaction service (IS) specification defines schemas and profiles that enable a *Web Services Provider* (WSP) to interact with relevant principals. At the time of service invocation at the WSP by a WSC, various situations are possible. For example, the *resource owner* may have a browser session with the invoking WSC, with the WSC acting as a *service provider* for the user. However, a WSP may need to obtain some information from the resource owner when the resource owner is not browsing at all, perhaps when an invoice needs to be authorized, or the WSP is invoked because another party (perhaps a friend or family member) is using the WSC.

For the case when the resource owner is visiting (where *visiting* is short for having used a HTTP user agent to send a HTTP request) the WSC there are four possible methods that may be used to allow the WSP to interact with the resource owner:

1. The WSC can indicate in the invocation message to the WSP that the resource owner is visiting the WSC and that it is ready to redirect the resource owner to the WSP. The WSP could then, in its response, ask the WSC to redirect the user (user agent) to itself (the WSP). This will cause the resource owner to visit the WSP allowing the WSP to pose its questions. Once the WSP has obtained the information it needed it can redirect the user back to the WSC. The WSC can now re-invoke the WSP which should now be able to serve the request without further interaction with the user.

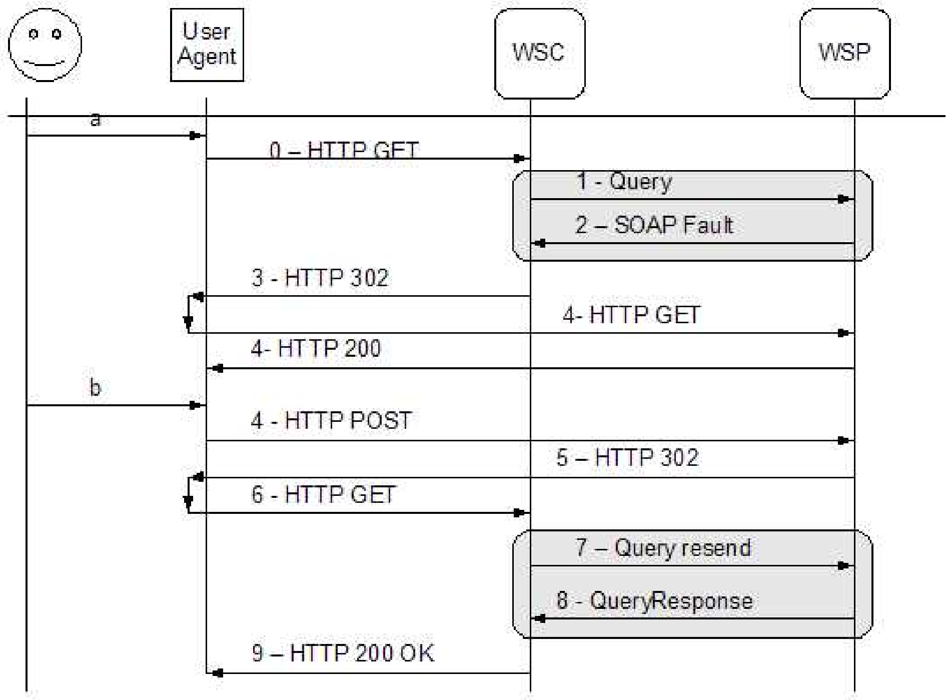

Figure 1. WSP Interacts with Principal by Requesting the WSC to Redirect the User Agent.

2. The WSP can check the resource owner's discovery service ([LibertyDisco]) to see if there is a (permanent) interaction service available for the resource owner. Such a service is, by definition, capable of interaction with the Principal at any time; for example by using special protocols, mechanism and channels such as instant messaging or WAP Push. If such an interaction service is available, the WSP can invoke that IS with a well-defined message that specifies the questions that it wants the IS to pose to the user. The IS would obtain the answers and then respond to the WSP. The WSP now has the information it needs and can respond to the originating invocation from the WSC. In this scenario the WSP and resource owner need to trust the IS to act as proxy.

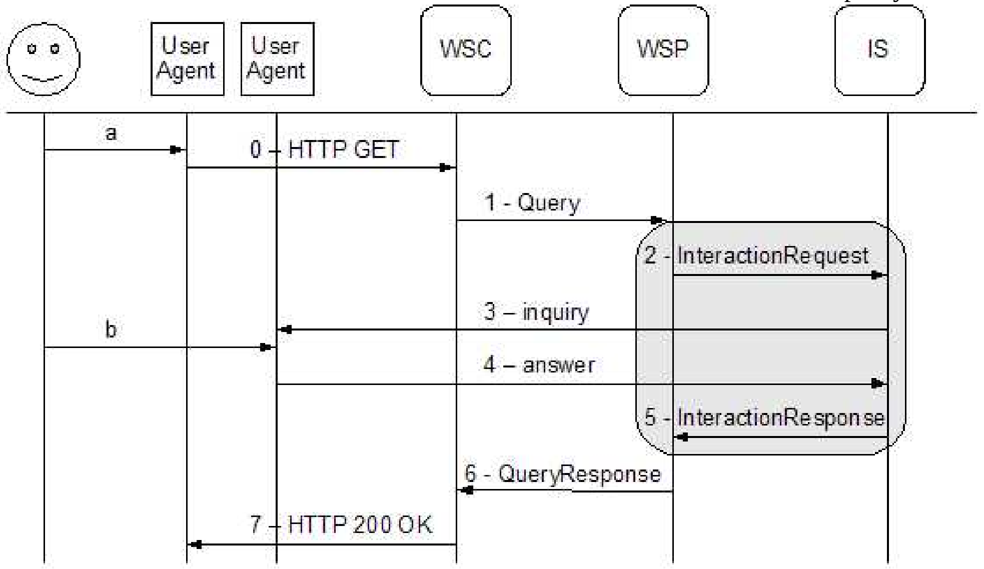

Figure 2. WSP Interacts with Principal by Requesting the Interaction Service to Pose an Inquiry.

3. The WSC can indicate in the invocation message to the WSP that the resource owner is visiting the WSC and that it is willing and able to present questions to the visiting resource owner. The WSC effectively offers an interaction service to the WSP. The WSP could invoke that service with an interaction request that specifies the questions that it wants the WSC to pose to the user. The WSC would obtain the answers and then respond to the WSP. The WSP now has the information it needs and can respond to the original invocation from the WSC. In this scenario the WSP needs to trust the WSC to act as proxy for the resource owner. Similarly, the resource owner needs to trust the WSC in its role as Interaction Service. The IS is almost literally a "man in the middle."

   This method has two variants; depending on how the the WSP's InteractionRequest is bound to the underlying HTTP layer. In the first variant, the WSP sends its InteractionRequest to the WSC/IS on a separate HTTP process than that on which the WSC sent its original invocation. The WSC/IS, after posing the relevant questions to the resource owner, sends an InteractionResponse on an HTTP Response to this second HTTP Request. The WSP can then respond to the original invocation by sending a response to the original HTTP Request. In this variant, the two HTTP Request/Response pairs are nested. In the second variant, the WSP sends its InteractionRequest to the WSC/IS within the HTTP Response to the original HTTP Request from the WSC (using the so-called PAOS Binding). The two variants are shown below.

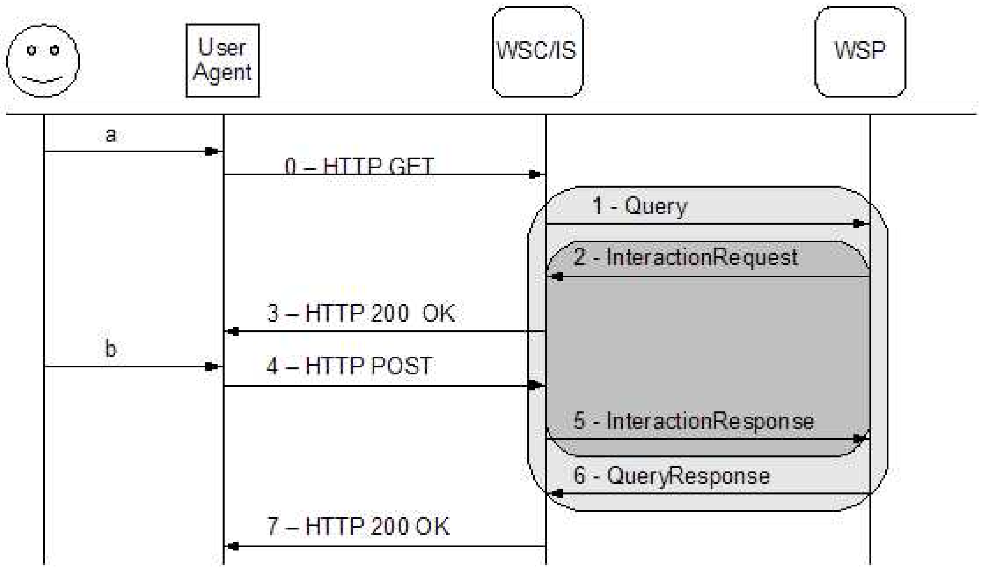

Figure 3. WSP Interacts with Principal by Requesting the WSC Pose an Inquiry through a Nested InteractionRequest

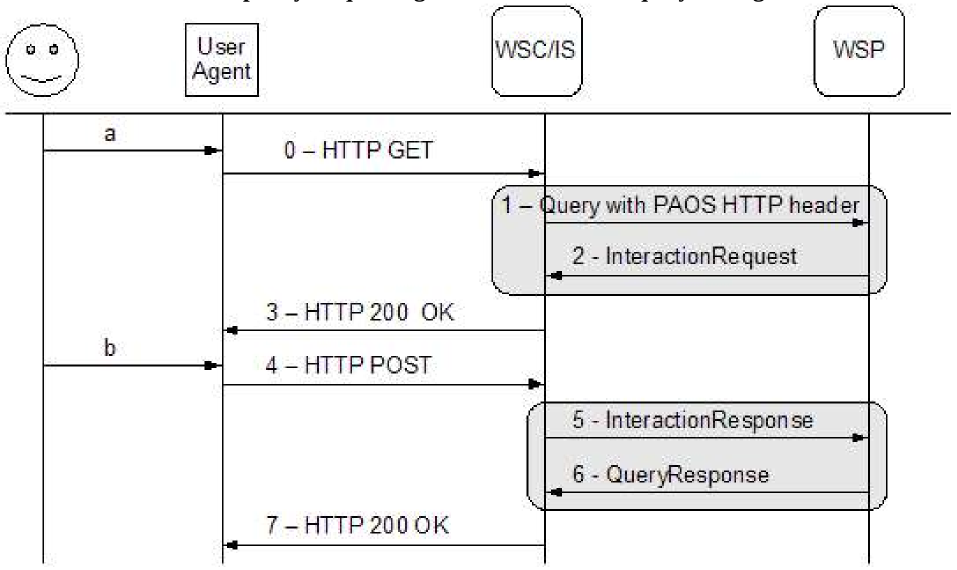

Figure 4. WSP Interacts with Principal by Requesting the WSC Pose an Inquiry through a PAOS-based InteractionRequest

The second variant may simplify the development of a WSC/IS as the interaction request can be handled by the same WSC process that already holds the connection with the user agent (communication channel used to pose questions to the user). This can be compared to the first variant for which complex inter-process communications may have to be implemented on the WSC/IS side to be able to reuse that same connection with the user agent.

When the principal with which interaction is desired is not visiting (as might be the case when the WSC is requesting on behalf of an offline (to it) resource owner or some different principal), then the redirect and WSC IS options are not relevant.

To enable the first two models of interaction, the [LibertySOAPBinding] specification defines a `<sb:UserInteraction>` SOAP Header block by which a WSC can indicate its preferences and capabilities for interactions with requesting principals and, additionally, a SOAP fault message and HTTP redirect profile that enables the WSC and WSP to cooperate in redirecting the requesting principal to the WSP and, after browser interaction, back to the WSC.

To enable the third model of interaction, this document specifies:

- Elements, processing rules and WSDL that together define an identity based interaction service, that can be made temporarily available by the WSC, or offered on a more permanent basis by a party that has the necessary permanent channel to the principal in question.

3. Interaction Service

The *interaction service* (IS) is an ID-WSF service that provides a means for simple interactions between an ID-WSF implementation and a Principal. It allows a client (typically a WSP acting as a WSC towards the interaction service) to query a Principal for consent, authorization decisions, etc. An IS provider accepts requests to present information and requests to a principal. The IS provider is responsible for "rendering" a "form" to the Principal. It is expected that the IS provider knows about the capabilities of the Principal's device and about any preferences he or she may have regarding such interactions. The IS returns the answer(s) of the Principal in a response that contains values for the parameters of the request.

Although an interaction service may exist as an identity service that is registered with a discovery service, the interaction service MAY also (or solely) be provided by a web services consumer that is invoking an identity service, but only when that service provider is engaged in an interactive session with the principal. However, the consumer of such an IS must have great trust in the IS provider as the ns of asserting that the response indeed is based upon principal input. Record keeping by all parties will support resolution of any possible dispute about a breach of such trust.

Only a party that is in principle capable of contacting the Principal *any time* should register a service type URN of *urn:liberty:is:2006-08* with the discovery service (see [LibertyDisco]) of that Principal.

An example deployment of a permanent IS provider could consist of an IS interface on top of a standard WAP Push service. The IS could accept `<InteractionRequest>`messages and create WML pages from such requests. It might then send a WAP Push message to the Principal's device with a temporary URL, that points to the newly created page. Once the WAP client receives the WAP message it will launch a HTTP session and fetch the given URL. The HTTP response will contain the WML page, which will be rendered in a brower on the client. The user would answer the question(s) in the form and submit it. The IS would now send a `<InteractionResponse>` to the invoker (and a "Thank You" page to the Principal). Note that this is just an example; another implementation could use an instant messaging protocol and yet another implementation could do both and switch based upon the users presence information (that it obtains from possibly yet another identity service).

Both a provider, and a client of an interaction service MUST adhere to the processing rules defined for ID-WSF messages in [LibertySOAPBinding] and [LibertySecMech].

An interaction service MAY register an `Option` with the Discovery Service to indicate one or more languages that it prefers for enquiries directed to the Principal. The value of the `Option` element SHOULD be a URI that MUST start with *urn:liberty:is:language* and is concatenated with one or more language identification tags (see [RFC3066]), that are each preceded by a forward slash / character. An example is *urn:liberty:is:language/en-US/fi*

3.1. Service Type

An Interaction Service is identified by the service type URN:

urn:liberty:is:2006-08

3.2. wsa:Action URIs

WS-Addressing defines the `<Action>` header by which the semantics of an input, output, or fault message can be expressed.

This specification defines the following action identifiers:

- *urn:liberty:is:2006-08:InteractionRequest*
- *urn:liberty:is:2006-08:InteractionResponse*

3.3. Interaction Request

A provider that wants to query a Principal sends an `<InteractionRequest>`. This element allows for the sender to define several types of queries. The requester can define text labels, parameters and default values. The response will have values for the supplied parameters. The requester SHOULD NOT assume any particular final format of the query.

The encompassing ID-WSF message MUST NOT contain a `<sb:UserInteraction>` Header block.

`<InteractionRequest>` messages MUST include a `<wsa:Action>` SOAP header with the value of "urn:liberty:is:2006-08:InteractionRequest."

3.3.1. The `InteractionRequest` Element

The `InteractionRequest` element allows the requester to define a "form" that the IS will present to the Principal. It contains:

`Inquiry` [Required]
: This element contains the elements that make up the actual query. There may be more than one `<Inquiry>` but it is RECOMMENDED that an `<InteractionRequest>` contains only one `<Inquiry>`.

`ds:KeyInfo` [Optional]
: This optional element can contain a public signing key that the sender has for the Principal. Presence of this element indicates to the IS that the sender wishes that the Principal sign the response with the associated private key and that the IS should include the signed statement in its response. If this element is present the `signed` attribute MUST be present too.

`id`
: Allows the element to be signed according to the rules in [LibertySecMech].

`language` [Optional]
: Indicates the languages that the user is likely able to process. The sender wishes that the inquiry will be rendered to the Principal using one of these languages. The value of this attribute is a space separated list of language identification Tags ([RFC3066]). The WSC can obtain this information from the HTTP `Accept-Language` header, from a language `Option` URI for the `InteractionService` in the `wsa:EndPointReference` or by other means, for example from a *personal profile service*. It is RECOMMENDED that the value of a `language` attribute does not request a language that was not present in the language `Option` URI, if this was presented to the sender.

`signed` [Optional]
: This attribute indicates that the sender wishes the Principal to sign the response. The value of this attribute can be *strict*, or *lax*. A value of *strict* indicates that the sender wants a positive response only if it will contain a signed statement from the Principal. It this attribute is present a `<ds:Keyinfo>` MAY be present too, and the `<InteractionRequest>` SHOULD NOT contain more than one `<Inquiry>`.

`maxInteractTime` [Optional]

Indicates the maximum time in seconds that the sender regards as reasonable for the principal interaction. A WSP MUST NOT set the value of this attribute to a greater value than the value of a possibly received `maxInteractTime` attribute in a `<sb:UserInteraction>` Header block.

The schema fragment for the `<InteractionRequest>` is:

```
<xs:element name="InteractionRequest" type="InteractionRequestType"/>
<xs:complexType name="InteractionRequestType">
   <xs:sequence>
      <xs:element ref="Inquiry" maxOccurs="unbounded"/>
      <xs:element ref="ds:KeyInfo" minOccurs="0"/>
   </xs:sequence>
   <xs:attribute name="id" type="xs:ID" use="optional"/>
   <xs:attribute name="language" type="xs:NMTOKENS" use="optional"/>
   <xs:attribute name="maxInteractTime" type="xs:integer" use="optional"/>
   <xs:attribute name="signed" type="xs:token" use="optional"/>
</xs:complexType>


```

3.3.2. The `Inquiry` Element

The `Inquiry` element contains:

`Help` [Optional]

Contains informal text regarding the inquiry, which may be presented to the user (See further definition below).

Element of type `InquiryElementType` [Zero or more]

Elements of this type contain actual query elements to be presented to the user. The type, and its sub-types are defined below.

`id` [Optional]

The `id` attribute MUST be present if the encompassing `<InteractionRequest>` contains the `signed` attribute and then its value MUST have the properties of a nonce; i.e., the uniqueness properties defined for a `messageID` in [LibertySOAPBinding].

`title` [Optional]
: The interaction service SHOULD present the value of the `title` attribute in accordance with the conventions of the user agent used to present the inquiry to the Principal.

The schema fragment for the `<Inquiry>` is:

```
<xs:element name="Inquiry" type="InquiryType"/>
<xs:complexType name="InquiryType">
   <xs:sequence>
      <xs:element ref="Help" minOccurs="0"/>
      <xs:choice maxOccurs="unbounded">
         <xs:element ref="Select" minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="Confirm" type="InquiryElementType"
                  minOccurs="0" maxOccurs="unbounded"/>
         <xs:element ref="Text" minOccurs="0" maxOccurs="unbounded"/>
      </xs:choice>
   </xs:sequence>
   <xs:attribute name="id" type="xs:ID" use="optional"/>
   <xs:attribute name="title" type="xs:string" use="optional"/>
</xs:complexType>
```

3.3.2.1. The `Help` Element

The `Help` element contains informal text about its parent element. Whitespace in this element is significant in that the IS provider is expected to attempt to respect newline characters. The IS provider is not expected to render the text of this element, but rather provide the Principal with an option to view the text. The IS provider is expected to realize such option according to the conventions of the user agent of the Principal. Apart from the help text this element may have:

`label` [Optional]
: Specifies a label relating to the help text.

`link` [Optional]
: This element MUST contain a resolvable URL to information about the inquiry. If the `link` attribute is present then the `Help` element MUST NOT contain text.

`moreLink` [Optional]
: An optional attribute whose value MUST be a resolvable URL to *additional* information about the inquiry. The IS provider is expected to present the Principal with an appropriate means such as a button, link or menu-item for obtaining this additional information.

The schema fragment for the `Help` element is:

```
<xs:element name="Help" type="HelpType"/>
<xs:complexType name="HelpType">
   <xs:attribute name="label" type="xs:string" use="optional"/>
   <xs:attribute name="link" type="xs:anyURI" use="optional"/>
   <xs:attribute name="moreLink" type="xs:anyURI" use="optional"/>
</xs:complexType>
```

3.3.2.2. The `InquiryElementType`

The `InquiryElementType` is an abstract type that defines the common content for query elements. The type contains:

`Help` [Optional]
: See definition of the `Help` element above.

`Hint` [Optional]
: A `<Hint>` contains short informal text about its parent element. The IS provider is expected to present the text of this element as a hint, according to the conventions of the Principal's user agent. The simple `Hint` element does not contain attributes or children elements.

`Label` [Optional]
: An IS provider is expected to present the content of `Label` elements as question labels. Note that the text value of a `<Label>` is normalized.

`Value` [Optional]
: Where applicable an IS provider will render the content of `Value` elements as initial values for the parameters (ie. as defaults). Requesters that wish to receive a signed `Statement` in the response MUST include a (possibly empty) `<Value>` for each instance of `InquiryElementType`. If multiple items of a `<Select>` are to be pre-selected, the contents of the `<Value>` element is a space separated list of tokens corresponding to the value attributes of the corresponding `<Item>` elements.

`name` [Required]
: The `name` attribute is used as a parameter name. This attribute may not always be presented by the IS service, but in case there is no `<Label>` provided for the parameter, the interaction service MAY use the value of the `name` attribute instead. Note that a single `<InteractionRequest>` may not contain more than one `<InquiryElement>` with the same name, as the `type` of this attribute is `xs:ID`.

The schema fragment for the `InquiryElementType` is:

```
<xs:complexType name="InquiryElementType" abstract="true">
   <xs:sequence>
      <xs:element ref="Help" minOccurs="0"/>
      <xs:element ref="Hint" minOccurs="0"/>
      <xs:element name="Label" type="xs:normalizedString" minOccurs="0"/>
      <xs:element name="Value" type="xs:normalizedString" minOccurs="0"/>
   </xs:sequence>
   <xs:attribute name="name" type="xs:ID" use="required"/>
</xs:complexType>

<xs:element name="Hint" type="xs:string"/>
```

3.3.2.3. `<InquiryElementType>` Subtypes

The defined `<InquiryElementType>` subtypes are:

- The `Select` element. This element allows the requester to ask the principal to select one (or more) items out of a given set of values. The resulting parameter value is a string with space separated tokens. This element contains `Item` elements that contain label and value *attributes*. The content of the optional `<Value>` MUST match the `value` of one of the children `Item` elements. The `Select` element has a boolean `multiple` attribute to indicate if more than one item can be selected; the default is *false*.

  The schema fragment for the `Select` element is:

```

<xs:element name="Select" type="SelectType"/>
<xs:complexType name="SelectType">
   <xs:complexContent>
      <xs:extension base="InquiryElementType">
         <xs:sequence>
            <xs:element name="Item" minOccurs="2" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="Hint" minOccurs="0"/>
                  </xs:sequence>
                  <xs:attribute name="label" type="xs:string" use="optional"/>
                  <xs:attribute name="value" type="xs:NMTOKEN" use="required"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
         <xs:attribute name="multiple" type="xs:boolean" use="optional" default="false"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>


```

- The `Confirm` element. This element allows the requester to ask the principal a yes/no question. The resulting parameter value is *"true"* or *"false"*.

- The `Text` element. This element allows the requester to ask the principal an open ended question. The requester may give a recommended minimum and maximum size in characters, and a format input mask. The resulting parameter value is a text string.

  The `format` string SHOULD adhere to the specification for format input masks for WML 1.3 `input` elements (see [WML]). However note that it is the interaction service that SHOULD attempt to obtain a `value` for the `Text` element that matches with the requested format input mask. It is up to the recipient of the `<InteractionResponse>` to verify the format of values as an interaction service MAY ignore a `format` attribute. The format input mask may help speed up entry of the value by the Principal.

  The schema fragment for the `Text` element is:

```
<xs:element name="Text" type="TextType"/>
<xs:complexType name="TextType">
   <xs:complexContent>
      <xs:extension base="InquiryElementType">
         <xs:attribute name="minChars" type="xs:integer" use="optional"/>
         <xs:attribute name="maxChars" type="xs:integer" use="optional"/>
         <xs:attribute name="format" type="xs:string" use="optional"/>
   </xs:extension>
   </xs:complexContent>
</xs:complexType>


```

3.3.3. Example Request

An example of a interaction request that asks for consent to share the owner's address with a WSC might look like:

```
<InteractionRequest xmlns="urn:liberty:is:2006-08">
   <Inquiry title="Profile Provider Question">
      <Help moreLink="http://pip.example.com/help/attribute/re ad/consent">
      example.com is requesting your address.  We do not have a rule that
      instructs us how you want us to process this request.  Please pick one of
      the given options.  Note that the last two options ensure you will not be prompted again
       should example.com ask for your address again in the future.
      </Help>
      <Select name="addresschoice">
         <Label>Do you want to share your address with service-provider.com?</Label>
         <Value>no</Value>
         <Item label="Not this time" value="no"/>
         <Item label="Yes, once" value="yes"/>
         <Item label="No, never" value="never">
            <Hint>We won't give out your address and won't ask you again</Hint>
         </Item>
         <Item label="Yes, always" value="always">
            <Hint>We will share your address now and in the future with example.com</Hint>
         </Item>
      </Select>
   </Inquiry>
</InteractionRequest>
```

3.3.4. Processing Rules

The recipient of an `<InteractionRequest>` MUST pose the first `<Inquiry>` to the principal. The recipient MUST NOT pose any `<Inquiry>` if the `<InteractionRequest>` has a `<maxInteractTime>` attribute with a value smaller than the time that the recipient *expects* to be required to process that `<Inquiry>`. The recipient MAY pose *all* the `Inquiry` elements, if it is able to do so in a manner that is both efficient as well as user friendly.

The recipient SHOULD make every attempt to format each `<Inquiry>` according to the expectations defined for the `Inquiry` element and its children elements.

The recipient SHOULD attempt to present user interface elements such a buttons, labels etc., in one of the languages given in the language attribute, if present. Nevertheless, the recipient SHOULD NOT attempt to translate any of the texts given by the sender for elements of the interaction request. For example, a `Confirm` element could be rendered on a web page with links for "Yes" and "No, but if the language indicated "fi" (for Finnish) the IS could render "KyllÃÂ¤" and "Ei."

If the `<InteractionRequest>` includes a `signed` attribute then the recipient SHOULD attempt to obtain a signed `<InteractionStatement>` from the Principal. If the value of the `signed` attribute is *strict* the recipient MUST respond with an `<InteractionResponse>` that contains either an `<InteractionStatement>`, or a `Status` element with its `code` attribute set to *NotSigned*. Further, if the `<InteractionRequest>` includes a `ds:KeyInfo` element then the recipient SHOULD attempt to obtain an `<InteractionStatement>` signed with the (private) key associated with the key described in the `ds:KeyInfo` element. In this case the recipient MUST verify that the signature was constructed with the indicated key and if this was not the case the response SHOULD include a `Status` of *KeyNotUsed*.

If processing is successful, the recipient MUST respond with a message containing an `<InteractionResponse>` with a `<Status>` element holding a `code` attribute of *OK*.

Additional values for the `code` attribute are specified below.

3.4. Interaction Response

The IS Service responds with an ID-WSF message that contains either an `InteractionResponse` element, or a SOAP fault (see [LibertySOAPBinding].

All responses will contain a `Status` element and, upon success, the `InteractionResponse` will contain values for all the parameters in the query of the corresponding `<InteractionRequest>`.

The `code` attribute of the `Status` element can take one of the values listed below:

- *OK* when the Principal answered the query and the message contains an `<InteractionResponse>`.
- *Cancel* when the Principal canceled the query.
- *NotSigned* when the request indicates `signed="strict"` but no signed statement could be obtained.
- *KeyNotUsed* when the Principal signed the inquiry with a key other than indicated in the `<ds:KeyInfo>` of the request.
- *InteractionTimeOut* when the Principal did not answer the query in a timely manner, or the connection to the Principals user agent was lost.
- *InteractionTimeNotSufficient* when the IS provider expects that the Principal cannot answer the inquiry within the `maxInteractTime` number of seconds, e.g., due to the fact that it takes time than allowed to establish a connection.
- *NotConnected* when the IS provider can currently not contact the Principal.

`<InteractionResponse>` messages MUST include a `<wsa:Action>` SOAP header with the value of "urn:liberty:is:2006-08:InteractionResponse."

3.4.1. The `InteractionResponse` Element

The `InteractionResponse` element contains a `Status` element and, upon success, either:

`Parameter` [Optional]
: The `InteractionResponse` will contain `Parameter` elements corresponding to each element supplied in the `<Inquiry>` that is of the `InquiryElementType`. Each `<Parameter>` MUST have its `name` attribute match the value of the `name` attribute of the corresponding `InquiryElement`.

or:

`InteractionStatement` [Optional]
: Contains one or more signed `Inquiry` elements.

The `Parameter` element has two attributes:

`name` [Required]
: Contains a value matching the value of the `name` attribute on the corresponding `InquiryElement`.

`value` [Required]
: The answer that was obtained from the principal, or the unchanged default supplied. For `<Select>` query elements the `value` may be a space separated list of `tokens`.

The `<InteractionStatement>` consists of:

`Inquiry` [Optional]

This is a copy of the element (or elements) submitted in the request, but with the `value` attributes of each `InquiryElement` set (or left blank) by the Principal. The `<Inquiry>` in an `<InteractionStatement>` MUST include *all* `InquiryElements` of **`InquiryElementType`** specified in the request; but other elements, such as `<Help>`, `<Hint>` and `<Item>`, MAY be omitted.

`ds:Signature` [Optional]

Contains a signature that covers the `Inquiry` elements (and thus all child elements). The signature must be constructed by use of the private key associated with the content of the `<ds:KeyInfo>` of the `<InteractionRequest>`.

The schema fragment for the `<InteractionResponse>`element is:

```
<xs:element name="InteractionResponse" type="InteractionResponseType"/>
<xs:complexType name="InteractionResponseType">
   <xs:sequence>
      <xs:element ref="lu:Status"/>
      <xs:choice>
         <xs:element name="InteractionStatement" type="InteractionStatementType"
            minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="Parameter" type="ParameterType" minOccurs="0"
            maxOccurs="unbounded"/>
      </xs:choice>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="InteractionStatementType">
   <xs:sequence>
      <xs:element ref="Inquiry" maxOccurs="unbounded"/>
      <xs:element ref="ds:Signature"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="ParameterType">
   <xs:attribute name="name" type="xs:ID" use="required"/>
   <xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>
```

An example of a response to the example request could look like:

```
<InteractionResponse>
   <Status code="OK"/>
   <Parameter name="addresschoice" value="always"/>
</InteractionResponse>
```

The same example as a response to an `<InteractionRequest>` with the `signed` attribute could look like:

```
<InteractionResponse>
   <Status code="OK"/>
   <InteractionStatement>
      <Inquiry title="Profile Provider Question" id="inquiry-3d4e2f8a37213b">
         <Select name="addresschoice">
            <Label>Do you want to share your address with service-provider.com?</Label>
            <Value>always</Value>
         </Select>
      </Inquiry>
      <ds:Signature>
      ....  <ds:Reference>#inquiry-3d4e2f8 a37213b</ds:Reference> ....
      </ds:Signature>
   </InteractionStatement>
</InteractionResponse>
```

An example of an empty, unsuccessful, response to the example request could look like:

```
<InteractionResponse>
   <Status code="Cancel"/>
</InteractionResponse>

```

3.4.2. Processing Rules

The recipient of an `<InteractionResponse>` that contains a signed `<InteractionStatement>` MUST verify the signature, and discard the response if the signature cannot be verified. That recipient MUST verify that the `id` attribute of the signed `<Inquiry>` corresponds with the `id` of the corresponding request `<Inquiry>`.

4. Security Considerations

The interaction service is effectively acting to its client WSCs as a proxy for the Principal. It is therefore important that the IS can be trusted by those clients. This is especially the case when such a WSC is itself a WSP that needs to obtain consent or permissions. There is no general possibility for an IS to proof on-line that it did indeed obtain the response from the Principal. The IS can and should of course authenticate the Principal, and could then save the proof of authentication, such as an assertion. There is little point in forwarding such an assertion to the WSC as proof, as an authentication assertion will contain the `NameID` of the Principal as known to the IS, not to the WSC. An IS that is closely associated with an identity provider, i.e., has the same providerID as that identity provider, could actually issue an assertion that states that the Principal as known to the WSC was present. Such statements could be added as SOAP header to the `InteractionResponse` message (see [LibertySecMech].

It is not sufficient to know that a Principal was present at the IS. There is still the possibility that a rogue IS created or changed the Principal's answers in the `<InteractionResponse>`. The interaction service client can verify the integrity of the response if the answered `Inquiry` is signed with a key that is: either shared between the Principal and the WSC, or is the private key of the Principal and the WSC knows that the associated public key is bound to the Principal. To this end the WSC can include such public asymmetric key in the `<InteractionRequest>`. Naturally the WSC should have consent from the Principal to share that key with the IS. Use of a private key is preferred for a more provable audit trail of the Principals answers to the inquiry.

The Principal has a risk that an IS, or for that matter any WSP, may misrepresent him. IS providers should make efforts to induce trust in the Principal, for example by offering transaction logs, deploying sufficiently strong authentication methods, etc.

References

Normative

[RFC2119] S. Bradner "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet Engineering Task Force (March 1997). *http://www.ietf.org/rfc/rfc2119.txt*

[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*

[RFC3066] Alvestrand, H., eds. (January 2001). "Tags for the Identification of Languages," RFC 3066., Internet Engineering Task Force *http://www.ietf.org/rfc/rfc3066.txt*

[LibertyDisco] Cahill, Conor, Hodges, Jeff, eds. "Liberty ID-WSF Discovery Service Specification," Version 2.0-errata-v1.0, Liberty Alliance Project (29 November, 2006). *http://www.projectliberty.org/specs*

[LibertySecMech] Hirsch, Frederick, eds. "Liberty ID-WSF Security Mechanisms Core," Version 2.0-errata-v1.0, Liberty Alliance Project (21 April, 2007). *http://www.projectliberty.org/specs*

[LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, Whitehead, Greg, Madsen, Paul, eds. "Liberty ID-WSF SOAP Binding Specification," Version 2.0-errata-v1.0, Liberty Alliance Project (21 April, 2007). *http://www.projectliberty.org/specs*

[LibertyIDWSFv20Errata] Champagne, Darryl, Lockhart, Rob, Tiffany, Eric, eds. "Liberty ID-WSF 2.0 Errata," Version 1.0, Liberty Alliance Project (13 April, 2007). *http://www.projectliberty.org/specs*

[Schema1-2] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (28 October 2004). "XML Schema Part 1: Structures Second Edition," Recommendation, World Wide Web Consortium *http://www.w3.org/TR/xmlschema-1/*

[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/*

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

[WSAv1.0] "Web Services Addressing (WS-Addressing) 1.0," Gudgin, Martin, Hadley, Marc, Rogers, Tony, eds. World Wide Web Consortium W3C Recommendation (9 May 2006). *http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/*

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

Informative

[SAMLCore2] Cantor, Scott, Kemp, John, Philpott, Rob, Maler, Eve, eds. (15 March 2005). "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0," SAML V2.0, OASIS Standard, Organization for the Advancement of Structured Information Standards *http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf*

A. Interaction Service XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:liberty:is:2006-08"
   xmlns="urn:liberty:is:2006-08"
   xmlns:is="urn:liberty:is:2006-08"
   xmlns:lu="urn:liberty:util:2006-08"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified"
   version="2.0">

   <xs:import namespace="urn:liberty:util:2006-08"
      schemaLocation="liberty-idwsf-utility-v2.0.xsd"/>

   <xs:import namespace="http://schemas.xmlsoap.org/soap/envelope/"
      schemaLocation="http://schemas.xmlsoap.org/soap/envelope/"/>
   <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
      schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-sch
ema.xsd"/>

   <xs:element name="InteractionRequest" type="InteractionRequestType"/>
   <xs:complexType name="InteractionRequestType">
      <xs:sequence>
         <xs:element ref="Inquiry" maxOccurs="unbounded"/>
         <xs:element ref="ds:KeyInfo" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="optional"/>
      <xs:attribute name="language" type="xs:NMTOKENS" use="optional"/>
      <xs:attribute name="maxInteractTime" type="xs:integer" use="optional"/>
      <xs:attribute name="signed" type="xs:token" use="optional"/>
   </xs:complexType>

   <xs:element name="Inquiry" type="InquiryType"/>
   <xs:complexType name="InquiryType">
      <xs:sequence>
         <xs:element ref="Help" minOccurs="0"/>
         <xs:choice maxOccurs="unbounded">
            <xs:element ref="Select" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Confirm" type="InquiryElementType"
                     minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="Text" minOccurs="0" maxOccurs="unbounded"/>
         </xs:choice>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="optional"/>
      <xs:attribute name="title" type="xs:string" use="optional"/>
   </xs:complexType>

   <xs:element name="Help" type="HelpType"/>
   <xs:complexType name="HelpType">
      <xs:attribute name="label" type="xs:string" use="optional"/>
      <xs:attribute name="link" type="xs:anyURI" use="optional"/>
      <xs:attribute name="moreLink" type="xs:anyURI" use="optional"/>
   </xs:complexType>

   <xs:element name="Hint" type="xs:string"/>

   <xs:element name="Select" type="SelectType"/>
   <xs:complexType name="SelectType">
      <xs:complexContent>
         <xs:extension base="InquiryElementType">
            <xs:sequence>
               <xs:element name="Item" minOccurs="2" maxOccurs="unbounded">
                  <xs:complexType>
                     <xs:sequence>
```

```
                        <xs:element ref="Hint" minOccurs="0"/>
                     </xs:sequence>
                     <xs:attribute name="label" type="xs:string" use="optional"/>
                     <xs:attribute name="value" type="xs:NMTOKEN" use="required"/>
                  </xs:complexType>
               </xs:element>
            </xs:sequence>
            <xs:attribute name="multiple" type="xs:boolean" use="optional" default="false"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

   <xs:element name="Text" type="TextType"/>
   <xs:complexType name="TextType">
      <xs:complexContent>
         <xs:extension base="InquiryElementType">
            <xs:attribute name="minChars" type="xs:integer" use="optional"/>
            <xs:attribute name="maxChars" type="xs:integer" use="optional"/>
            <xs:attribute name="format" type="xs:string" use="optional"/>
      </xs:extension>
      </xs:complexContent>
   </xs:complexType>

   <xs:complexType name="InquiryElementType" abstract="true">
      <xs:sequence>
         <xs:element ref="Help" minOccurs="0"/>
         <xs:element ref="Hint" minOccurs="0"/>
         <xs:element name="Label" type="xs:normalizedString" minOccurs="0"/>
         <xs:element name="Value" type="xs:normalizedString" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:ID" use="required"/>
   </xs:complexType>

<xs:element name="InteractionResponse" type="InteractionResponseType"/>
   <xs:complexType name="InteractionResponseType">
      <xs:sequence>
         <xs:element ref="lu:Status"/>
         <xs:choice>
            <xs:element name="InteractionStatement" type="InteractionStatementType"
               minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Parameter" type="ParameterType" minOccurs="0"
               maxOccurs="unbounded"/>
         </xs:choice>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="InteractionStatementType">
      <xs:sequence>
         <xs:element ref="Inquiry" maxOccurs="unbounded"/>
         <xs:element ref="ds:Signature"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="ParameterType">
      <xs:attribute name="name" type="xs:ID" use="required"/>
      <xs:attribute name="value" type="xs:string" use="required"/>
   </xs:complexType>

</xs:schema>
```

B. WSDL

```
<?xml version="1.0"?>
<definitions
   name="id-wsf-is_2006-08_wsdl_interface"
   targetNamespace="urn:liberty:is:2006-08"
   xmlns:tns="urn:liberty:is:2006-08"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl"
   xmlns:is="urn:liberty:is:2006-08"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://schemas.xmlsoap.org/wsdl/
                  http://schemas.xmlsoap.org/wsdl/
      http://www.w3.org/2006/02/addressing/wsdl
      http://www.w3.org/2006/02/addressing/wsdl/ws-addr-wsdl.xsd">

   <types>
      <xsd:import namespace="urn:liberty:is:2006-08"
              schemaLocation="liberty-idwsf-interaction-svc-v2.0.xsd"/>
   </types>

   <!-- Messages -->

   <message name="InteractionRequest">
      <part name="body" type="is:InteractionRequest"/>
   </message>

   <message name="InteractionResponse">
      <part name="body" type="is:InteractionResponse"/>
   </message>

   <!-- Port Type -->

   <portType name="ISPort">
      <operation name="ISInteraction">
         <input message="tns:InteractionRequest"
            wsaw:Action="urn:liberty:is:2006-08:InteractionRequest"/>
         <output message="tns:InteractionResponse"
            wsaw:Action="urn:liberty:is:2006-08:InteractionResponse"/>
      </operation>
   </portType>

  <!--
  An example of a binding and service that can be used with this
  abstract service description is provided below.
  -->

   <binding name="ISBinding" type="tns:ISPort">

      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

      <operation name="Interaction">
         <soap:operation soapAction="urn:liberty:is:2006-08:Interaction"/>
         <input>  <soap:body use="literal"/> </input>
         <output> <soap:body use="literal"/> </output>
      </operation>
   </binding>

   <service name="InteractionService">
      <port name="ISPort" binding="tns:ISBinding">

       <!-- Modify with the REAL SOAP endpoint -->

         <soap:address location="http://example.com/id-wsf/is"/>
      </port>
```

```
    </service>

</definitions>

```

C. Example XSL Stylesheet for HTML Forms (non-normative)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This stylesheet converts an is:Inquiry into an HTML form.
    Note that this is just a simple example that does not render all required elements.
    Note the use of xsl:parameters to insert some session information, obviously other
    techniques can be used.
    Note for Hints this stylesheet adds a reference to a "showHint" script, but such script
    is not defined here.

-->

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
   xmlns:is="urn:liberty:is:2006-08" exclude-result-prefixes="is">
   <xsl:output method="xml" version="4.0" encoding="UTF-8" omit-xml-declaration="yes" />
   <xsl:param name="jsessionid">null</xsl:param>
   <xsl:param name="messageID">null</xsl:param>
   <xsl:template match="/">
      <xsl:apply-templates select="//is:Inquiry" />
   </xsl:template>

   <xsl:template match="is:Inquiry">
      <html>
         <head>
            <title>
               <xsl:value-of select="@title"/>
            </title>
         </head>
         <body>
            <h2>
               <xsl:value-of select="@title"/>
            </h2>
            <xsl:element name="form">
               <xsl:attribute name="method">get</xsl:attribute>
               <xsl:attribute name="action">
                  submit;jsessionid=<xsl:value-of select="$jsessionid"/>
               </xsl:attribute>
                  <xsl:element name="input">
                     <xsl:attribute name="type">hidden</xsl:attribute>
                     <xsl:attribute name="name">msg</xsl:attribute>
                     <xsl:attribute name="value"><xsl:value-of select="$messageID"/></xsl:attribute>

                  </xsl:element>
               <xsl:apply-templates select="is:Confirm"/><br/>
               <xsl:apply-templates select="is:Select"/><br/>
               <br/>
               <input type="submit" value="Submit"/>
            </xsl:element>
            <p>
               <xsl:apply-templates select="is:Help"/>
            </p>
         </body>
      </html>
   </xsl:template>

   <xsl:template match="is:Confirm">
      <xsl:value-of select="is:Label"/>
      <xsl:element name="label">
         <xsl:attribute name="for">isid-<xsl:value-of select="@name"/>-yes</xsl:attribute>
         Yes
      </xsl:element>
      <xsl:element name="input">
         <xsl:attribute name="type">radio</xsl:attribute>
         <xsl:attribute name="checked"></xsl:attribute>
         <xsl:attribute name="name">is-confirm-yes-<xsl: value-of select="@name"/></xsl:attribute>
         <xsl:attribute name="id">isid-<xsl:value-of select="@name"/>-yes</xsl:attribute>
      </xsl:element>
      <xsl:element name="label">
         <xsl:attribute name="for">isid-<xsl:value-of select="@name"/>-no</xsl:attribute>
         No
      </xsl:element>
      <xsl:element name="input">
         <xsl:attribute name="type">radio</xsl:attribute>
         <xsl:attribute name="name">is-confirm-no-<xsl:value-of select="@name"/></xsl:attribute>
         <xsl:attribute name="id">isid-<xsl:value-of select="@name"/>-no</xsl:attribute>
      </xsl:element>
   </xsl:template>

   <xsl:template match="is:Select">
      <xsl:element name="label">
         <xsl:value-of select="is:Label"/>
         <xsl:attribute name="for">isid-<xsl:value-of select="@name"/></xsl:attribute>
      </xsl:element>
      <xsl:element name="select">
         <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
         <xsl:attribute name="id">isid-<xsl:value-of select="@name"/></xsl:attribute>
         <xsl:apply-templates select="is:Item"/>
      </xsl:element>
   </xsl:template>

   <xsl:template match="is:Item">
      <xsl:element name="option">
         <xsl:attribute name="label"><xsl:value-of select="@label"/></xsl:attribute>
         <xsl:attribute name="value"><xsl:value-of select="@value"/></xsl:attribute>
         <xsl:value-of select="@label"/>
         <xsl:apply-templates select="is:Hint"/>
      </xsl:element>
   </xsl:template>
   <xsl:template match="is:Hint">
      <xsl:attribute name="onmouseover">showHint(<xsl:value-of select="."/>)</xsl:attribute>
   </xsl:template>
   <xsl:template match="is:Help">
      <p id="help"><b>Help</b><br/>
         <xsl:value-of select="."/>
         <xsl:element name="a">
            <xsl:attribute name="href">
               <xsl:value-of select="@morelink"/>
            </xsl:attribute>
            More information
         </xsl:element>
      </p>
   </xsl:template>
</xsl:stylesheet>
```

D. Example XSL Stylesheet for WML Forms (non-normative)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This stylesheet converts an is:Inquiry into a WML deck.
    This is only an example stylesheet that does not render all required elements.
    In fact it only renders Confirm elements, and hence is barely sufficient to handle
    the example in the specification.
    Note the use of xsl:parameters to insert some session information, obviously other
    techniques can be used.

-->

<!-- TODO: add a least support for Help elements. -->

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
      xmlns:is="urn:liberty:is:2006-08" exclude-result-prefixes="is">

   <xsl:output
      method="xml"
      version="1.0"
      encoding = "UTF-8"
      omit-xml-declaration="no"
      doctype-public="-//WAPFORUM//DTD WML 1.1//EN"
      doctype-system="http://www.wapforum.org/DTD/wml_1.1.xml"
      media-type="text/vnd.wap.wml" />

   <xsl:param name="jsessionid">null</xsl:param>
   <xsl:param name="messageID">null</xsl:param>
   <xsl:param name="card-index">1</xsl:param>

   <xsl:template match="/">
      <wml>
         <template>
         <do type="prev">
            <prev/>
         </do>
         </template>
         <xsl:apply-templates select="//is:Inquiry" />
      </wml>
   </xsl:template>

   <xsl:template match="is:Inquiry">
      <xsl:element name="card">
         <xsl:attribute name="id">inquiry-<xsl:value-of select="$card-index"/></xsl:attribute>
         <xsl:attribute name="title"><xsl:value-of select="@title"/></xsl:attribute>
         <xsl:apply-templates select="is:Confirm"/>
      </xsl:element>
   </xsl:template>

   <xsl:template match="is:Confirm">
      <p><xsl:value-of select="is:Label"/><br/>
      <anchor>
         <xsl:element name="go">
            <xsl:attribute name="href">
               submit;jsessionid=<xsl:value-of select="$jsessionid"/>
            </xsl:attribute>
            <xsl:attribute name="method">get</xsl:attribute>
            <xsl:element name="postfield">
               <xsl:attribute name="name">msg</xsl:attribute>
               <xsl:attribute name="value">
                  <xsl:value-of select="$messageID"/>
               </xsl:attribute>
            </xsl:element>
            <xsl:element name="postfield">
               <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
               <xsl:attribute name="value">1</xsl:attribute>
            </xsl:element>
         </xsl:element>Yes</anchor><br/>
      <anchor>
         <xsl:element name="go">
            <xsl:attribute name="href">
               submit;jsessionid=<xsl:value-of select="$jsessionid"/>
            </xsl:attribute>
            <xsl:attribute name="method">get</xsl:attribute>
            <xsl:element name="postfield">
               <xsl:attribute name="name">msg</xsl:attribute>
               <xsl:attribute name="value"><xsl:value-of select="$messageID"/></xsl:attribute>
            </xsl:element>
            <xsl:element name="postfield">
               <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
               <xsl:attribute name="value">0</xsl:attribute>
            </xsl:element>
         </xsl:element>No</anchor><br/>
      </p>
   </xsl:template>

</xsl:stylesheet>
```