



Liberty ID-WSF Interaction Service Specification

Version: 2.0

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

Jonathan Sergent, Sun Microsystems, Inc.

Greg Whitehead, Hewlett-Packard

Abstract:

It is often necessary for providers of identity-based web services to interact with principals (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-v2.0.pdf

1

Notice

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

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

15 Copyright © 2006 Adobe Systems; America Online, Inc.; American Express Company; Amsoft Systems Pvt Ltd.;
16 Avatier Corporation; Axalto; Bank of America Corporation; BIPAC; BMC Software, Inc.; Computer Associates
17 International, Inc.; DataPower Technology, Inc.; Diversinet Corp.; Enosis Group LLC; Entrust, Inc.; Epok, Inc.;
18 Ericsson; Fidelity Investments; Forum Systems, Inc.; France Télécom; French Government Agence pour le
19 développement de l'administration électronique (ADAE); Gamefederation; Gemplus; General Motors; Giesecke &
20 Devrient GmbH; GSA Office of Governmentwide Policy; Hewlett-Packard Company; IBM Corporation; Intel
21 Corporation; Intuit Inc.; Kantega; Kayak Interactive; MasterCard International; Mobile Telephone Networks (Pty)
22 Ltd; NEC Corporation; Netegrity, Inc.; NeuStar, Inc.; Nippon Telegraph and Telephone Corporation; Nokia
23 Corporation; Novell, Inc.; NTT DoCoMo, Inc.; OpenNetwork; Oracle Corporation; Ping Identity Corporation;
24 Reactivity Inc.; Royal Mail Group plc; RSA Security Inc.; SAP AG; Senforce; Sharp Laboratories of America;
25 Sigaba; SmartTrust; Sony Corporation; Sun Microsystems, Inc.; Supremacy Financial Corporation; Symlabs, Inc.;
26 Telecom Italia S.p.A.; Telefónica Móviles, S.A.; Trusted Network Technologies; UTI; VeriSign, Inc.; Vodafone
27 Group Plc.; Wave Systems Corp. All rights reserved.

28 Liberty Alliance Project
29 Licensing Administrator
30 c/o IEEE-ISTO
31 445 Hoes Lane
32 Piscataway, NJ 08855-1331, USA
33 info@projectliberty.org

34 Contents

35	1. Notation and Conventions	4
36	2. Overview	5
37	3. Interaction Service	9
38	3.1. Service Type	9
39	3.2. wsa:Action URIs	9
40	3.3. Interaction Request	9
41	3.3.1. The InteractionRequest Element	10
42	3.3.2. The Inquiry Element	11
43	3.3.3. Example Request	14
44	3.3.4. Processing Rules	15
45	3.4. Interaction Response	15
46	3.4.1. The InteractionResponse Element	16
47	3.4.2. Processing Rules	18
48	4. Security Considerations	19
49	References	20
50	A. Interaction Service XSD	21
51	B. WSDL	24
52	C. Example XSL Stylesheet for HTML Forms (non-normative)	26
53	D. Example XSL Stylesheet for WML Forms (non-normative)	28

54 1. Notation and Conventions

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

57 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT",
58 "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

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

62 The following namespaces are referred to in this document:

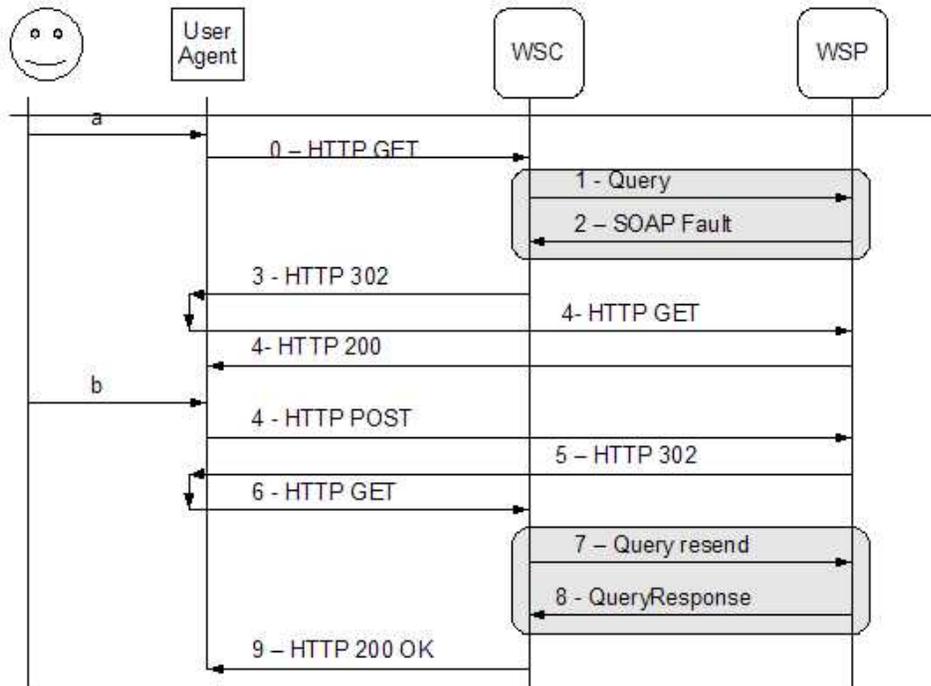
- 63 • The prefix *is*: stands for the ID-WSF working namespace for the interaction service (*urn:liberty:is:2006-08*). This
64 namespace is the default for instance fragments, type names, and element names in this document.
- 65 • The prefix *disco*: stands for the ID-WSF working namespace for the [LibertyDisco] (*urn:liberty:disco:2006-08*).
- 66 • The prefix *sb*: stands for the ID-WSF working namespace for the [LibertySOAPBinding] (*urn:liberty:sb:2006-*
67 *08*).
- 68 • The prefix *S*: stands for the SOAP 1.1 ([SOAPv1.1]) namespace (*http://schemas.xmlsoap.org/soap/envelope/*).
- 69 • The prefix *wsa*: stands for the WS-Addressing [WSAv1.0] namespace (*http://www.w3.org/2005/02/addressing*).
- 70 • The prefix *wSDL*: stands for the primary [WSDLv1.1] namespace (*http://schemas.xmlsoap.org/wSDL/*).
- 71 • The prefix *xs*: stands for the W3C XML schema namespace (*http://www.w3.org/2001/XMLSchema*).
- 72 • The prefix *xsi*: stands for the W3C XML schema instance namespace (*http://www.w3.org/2001/XMLSchema-*
73 *instance*).

74 **2. Overview**

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

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

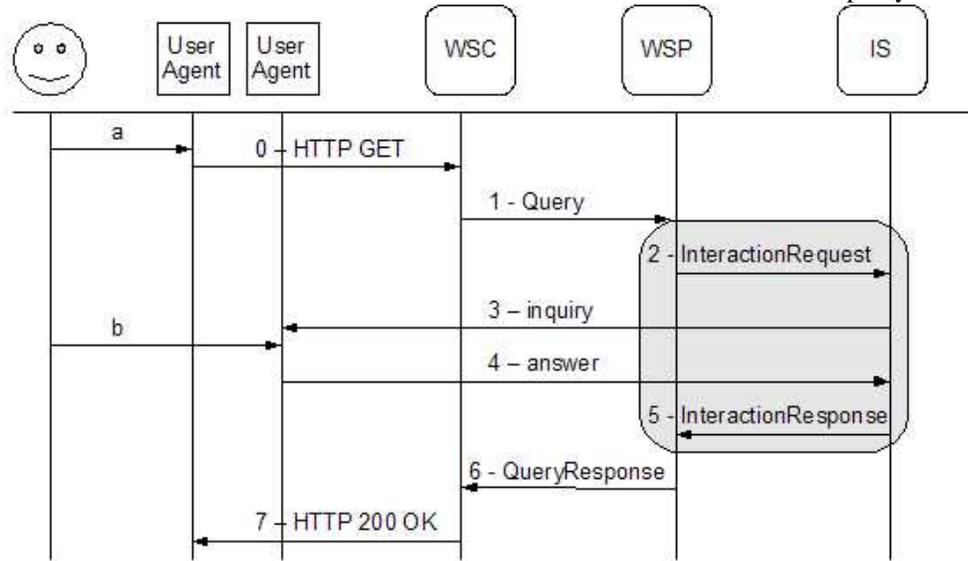
- 88 1. The WSC can indicate in the invocation message to the WSP that the resource owner is visiting the WSC and
 89 that it is ready to redirect the resource owner to the WSP. The WSP could then, in its response, ask the WSC to
 90 redirect the user (user agent) to itself (the WSP). This will cause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



94

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

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

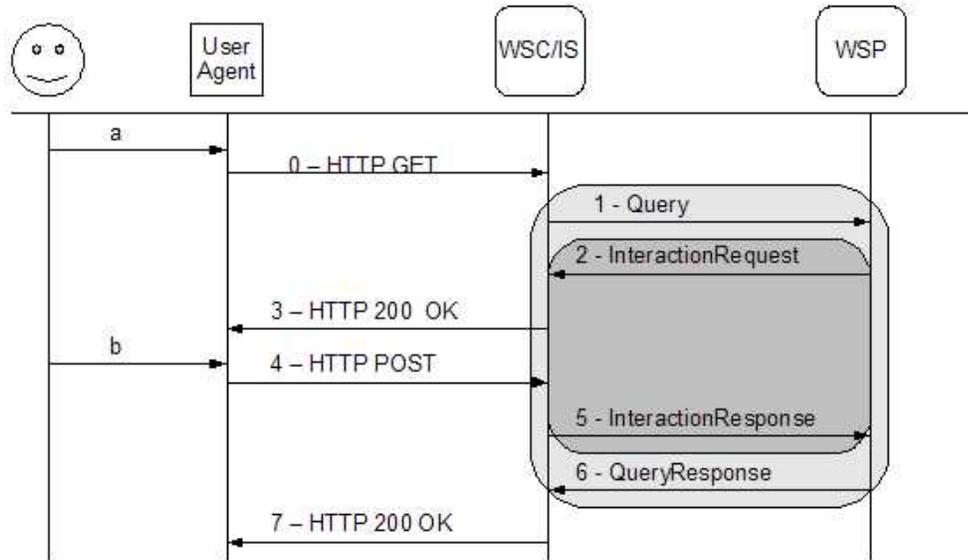


103

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

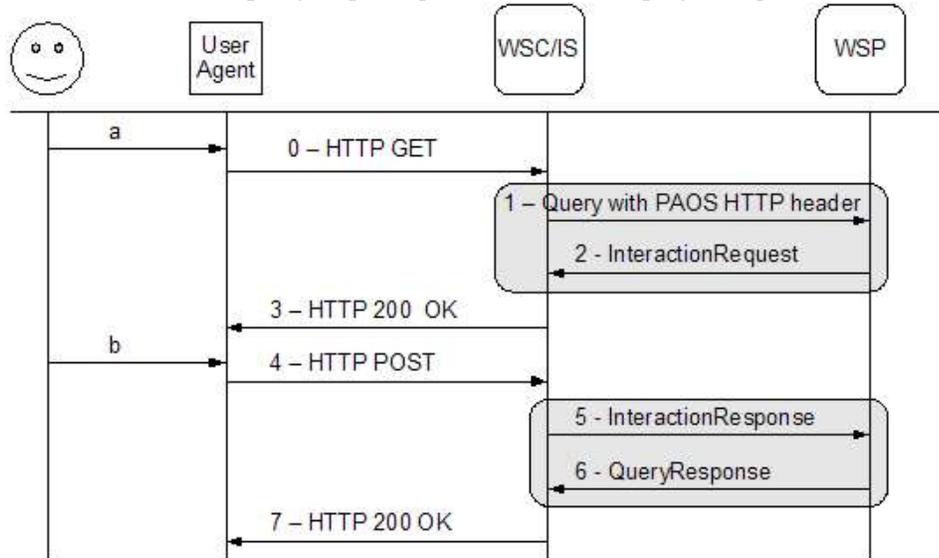
105 3. The WSC can indicate in the invocation message to the WSP that the resource owner is visiting the WSC and that
 106 it is willing and able to present questions to the visiting resource owner. The WSC effectively offers an interaction
 107 service to the WSP. The WSP could invoke that service with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

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



120

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



122

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

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

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

132 To enable the first two models of interaction, the [[LibertySOAPBinding](#)] specification defines a
133 <sb:UserInteraction> SOAP Header block by which a WSC can indicate its preferences and capabilities
134 for interactions with requesting principals and, additionally, a SOAP fault message and HTTP redirect profile that

135 enables the WSC and WSP to cooperate in redirecting the requesting principal to the WSP and, after browser
136 interaction, back to the WSC.

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

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

141 3. Interaction Service

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

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

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

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

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

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

172 3.1. Service Type

173 An Interaction Service is identified by the service type URN:

174 *urn:liberty:is:2006-08*

175 3.2. wsa:Action URIs

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

178 This specification defines the following action identifiers:

- 179 • *urn:liberty:is:2006-08:InteractionRequest*
- 180 • *urn:liberty:is:2006-08:InteractionResponse*

181 3.3. Interaction Request

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

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

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

189 3.3.1. The InteractionRequest Element

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

192 `Inquiry` [Required]

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

195 `ds:KeyInfo` [Optional]

196 This optional element can contain a public signing key that the sender has for the Principal. Presence of this
197 element indicates to the IS that the sender wishes that the Principal sign the response with the associated
198 private key and that the IS should include the signed statement in its response. If this element is present the
199 [signed attribute](#) MUST be present too.

200 `id`

201 Allows the element to be signed according to the rules in [\[LibertySecMech\]](#).

202 `language` [Optional]

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

210 `signed` [Optional]

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

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

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

```
220 <xs:element name="InteractionRequest" type="InteractionRequestType"/>
221 <xs:complexType name="InteractionRequestType">
222   <xs:sequence>
223     <xs:element ref="Inquiry" maxOccurs="unbounded"/>
224     <xs:element ref="ds:KeyInfo" minOccurs="0"/>
225   </xs:sequence>
226   <xs:attribute name="id" type="xs:ID" use="optional"/>
227   <xs:attribute name="language" type="xs:NMTOKENS" use="optional"/>
228   <xs:attribute name="maxInteractTime" type="xs:integer" use="optional"/>
229   <xs:attribute name="signed" type="xs:token" use="optional"/>
230 </xs:complexType>
231
232
```

233 3.3.2. The Inquiry Element

234 The `Inquiry` element contains:

235 `Help` [Optional]
236 Contains informal text regarding the inquiry, which may be presented to the user (See further definition
237 below).

238 Element of type `InquiryElementType` [Zero or more]
239 Elements of this type contain actual query elements to be presented to the user. The type, and its sub-types
240 are defined below.

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

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

248 The schema fragment for the <Inquiry> is:

```
249 <xs:element name="Inquiry" type="InquiryType"/>
250 <xs:complexType name="InquiryType">
251   <xs:sequence>
252     <xs:element ref="Help" minOccurs="0"/>
253     <xs:choice maxOccurs="unbounded">
254       <xs:element ref="Select" minOccurs="0" maxOccurs="unbounded"/>
255       <xs:element name="Confirm" type="InquiryElementType"
256         minOccurs="0" maxOccurs="unbounded"/>
257       <xs:element ref="Text" minOccurs="0" maxOccurs="unbounded"/>
258     </xs:choice>
259   </xs:sequence>
260   <xs:attribute name="id" type="xs:ID" use="optional"/>
261   <xs:attribute name="title" type="xs:string" use="optional"/>
262 </xs:complexType>
263
264
```

265 3.3.2.1. The Help Element

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

271 label [Optional]
272 Specifies a label relating to the help text.

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

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

280 The schema fragment for the Help element is:

```
281 <xs:element name="Help" type="HelpType"/>
282 <xs:complexType name="HelpType">
283   <xs:attribute name="label" type="xs:string" use="optional"/>
284   <xs:attribute name="link" type="xs:anyURI" use="optional"/>
285   <xs:attribute name="moreLink" type="xs:anyURI" use="optional"/>
286 </xs:complexType>
287
288
```

289 3.3.2.2. The InquiryElementType

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

- 292 **Help** [Optional]
293 See definition of the `Help` element above.
- 294 **Hint** [Optional]
295 A `<Hint>` contains short informal text about its parent element. The IS provider is expected to present the
296 text of this element as a hint, according to the conventions of the Principal's user agent. The simple `Hint`
297 element does not contain attributes or children elements.
- 298 **Label** [Optional]
299 An IS provider is expected to present the content of `Label` elements as question labels. Note that the text
300 value of a `<Label>` is normalized.
- 301 **Value** [Optional]
302 Where applicable an IS provider will render the content of `Value` elements as initial values for the parameters
303 (ie. as defaults). Requesters that wish to receive a signed `Statement` in the response **MUST** include a
304 (possibly empty) `<Value>` for each instance of `InquiryElementType`. If multiple items of a `<Select>`
305 are to be pre-selected, the contents of the `<Value>` element is a space separated list of tokens corresponding
306 to the value attributes of the corresponding `<Item>` elements.
- 307 **name** [Required]
308 The name attribute is used as a parameter name. This attribute may not always be presented by the IS service,
309 but in case there is no `<Label>` provided for the parameter, the interaction service **MAY** use the value of
310 the name attribute instead. Note that a single `<InteractionRequest>` may not contain more than one
311 `<InquiryElement>` with the same name, as the type of this attribute is `xs:ID`.

312 The schema fragment for the `InquiryElementType` is:

```
313 <xs:complexType name="InquiryElementType" abstract="true">  
314   <xs:sequence>  
315     <xs:element ref="Help" minOccurs="0"/>  
316     <xs:element ref="Hint" minOccurs="0"/>  
317     <xs:element name="Label" type="xs:normalizedString" minOccurs="0"/>  
318     <xs:element name="Value" type="xs:normalizedString" minOccurs="0"/>  
319   </xs:sequence>  
320   <xs:attribute name="name" type="xs:ID" use="required"/>  
321 </xs:complexType>  
322  
323 <xs:element name="Hint" type="xs:string"/>  
324  
325
```

326 **3.3.2.3. <InquiryElementType> Subtypes**

327 The defined `<InquiryElementType>` subtypes are:

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

333 The schema fragment for the `Select` element is:

```

334
335     <xs:element name="Select" type="SelectType"/>
336     <xs:complexType name="SelectType">
337         <xs:complexContent>
338             <xs:extension base="InquiryElementType">
339                 <xs:sequence>
340                     <xs:element name="Item" minOccurs="2" maxOccurs="unbounded">
341                         <xs:complexType>
342                             <xs:sequence>
343                                 <xs:element ref="Hint" minOccurs="0"/>
344                             </xs:sequence>
345                             <xs:attribute name="label" type="xs:string" use="optional"/>
346                             <xs:attribute name="value" type="xs:NMTOKEN" use="required"/>
347                         </xs:complexType>
348                     </xs:element>
349                 </xs:sequence>
350                 <xs:attribute name="multiple" type="xs:boolean" use="optional" default="false"/>
351             </xs:extension>
352         </xs:complexContent>
353     </xs:complexType>
354
355
  
```

- 356 • The `Confirm` element. This element allows the requester to ask the principal a yes/no question. The resulting
 357 parameter value is *"true"* or *"false"*.
- 358 • The `Text` element. This element allows the requester to ask the principal an open ended question. The requester
 359 may give a recommended minimum and maximum size in characters, and a format input mask. The resulting
 360 parameter value is a text string.

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

366 The schema fragment for the `Text` element is:

```

367     <xs:element name="Text" type="TextType"/>
368     <xs:complexType name="TextType">
369         <xs:complexContent>
370             <xs:extension base="InquiryElementType">
371                 <xs:attribute name="minChars" type="xs:integer" use="optional"/>
372                 <xs:attribute name="maxChars" type="xs:integer" use="optional"/>
373                 <xs:attribute name="format" type="xs:string" use="optional"/>
374             </xs:extension>
375         </xs:complexContent>
376     </xs:complexType>
377
378
  
```

379 3.3.3. Example Request

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

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

403 3.3.4. Processing Rules

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

408 The recipient SHOULD make every attempt to format each `<Inquiry>` according to the expectations defined for the
409 [Inquiry element](#) and its children elements.

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

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

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

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

426 3.4. Interaction Response

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

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

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

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

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

445 **3.4.1. The `InteractionResponse` Element**

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

447 `Parameter` [Optional]

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

451 *or*:

452 `InteractionStatement` [Optional]

453 Contains one or more signed `Inquiry` elements.

454 The `Parameter` element has two attributes:

455 `name` [Required]

456 Contains a value matching the value of the `name` attribute on the corresponding `InquiryElement`.

457 `value` [Required]

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

460 The `<InteractionStatement>` consists of:

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

466 ds:Signature [Optional]
 467 Contains a signature that covers the Inquiry elements (and thus all child elements). The signature
 468 must be constructed by use of the private key associated with the content of the <ds:KeyInfo> of the
 469 <InteractionRequest>.

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

```

471 <xs:element name="InteractionResponse" type="InteractionResponseType"/>
472 <xs:complexType name="InteractionResponseType">
473   <xs:sequence>
474     <xs:element ref="lu:Status"/>
475     <xs:choice>
476       <xs:element name="InteractionStatement" type="InteractionStatementType"
477         minOccurs="0" maxOccurs="unbounded"/>
478       <xs:element name="Parameter" type="ParameterType" minOccurs="0" maxOccurs="unbounded"/>
479     </xs:choice>
480   </xs:sequence>
481 </xs:complexType>
482 <xs:complexType name="InteractionStatementType">
483   <xs:sequence>
484     <xs:element ref="Inquiry" maxOccurs="unbounded"/>
485     <xs:element ref="ds:Signature"/>
486   </xs:sequence>
487 </xs:complexType>
488 <xs:complexType name="ParameterType">
489   <xs:attribute name="name" type="xs:ID" use="required"/>
490   <xs:attribute name="value" type="xs:string" use="required"/>
491 </xs:complexType>
492
493

```

494 An example of a response to the [example request](#) could look like:

```

495 <InteractionResponse>
496   <Status code="OK" />
497   <Parameter name="addresschoice" value="always"/>
498 </InteractionResponse>

```

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

```

500 <InteractionResponse>
501   <Status code="OK" />
502   <InteractionStatement>
503     <Inquiry title="Profile Provider Question" id="inquiry-3d4e2f8a37213b">
504       <Select name="addresschoice">
505         <Label>Do you want to share your address with service-provider.com?</Label>
506         <Value>always</Value>
507       </Select>
508     </Inquiry>
509     <ds:Signature>
510       .... <ds:Reference>#inquiry-3d4e2f 8a37213b</ds:Reference> ....
511     </ds:Signature>
512   </InteractionStatement>
513 </InteractionResponse>

```

514 An example of an empty, unsuccessful, response to the [example request](#) could look like:

```
515 <InteractionResponse>
516   <Status code="Cancel" />
517 </InteractionResponse>
518
```

519 **3.4.2. Processing Rules**

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

523 4. Security Considerations

524 The interaction service is effectively acting to its client WSCs as a proxy for the Principal. It is therefore important
525 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
526 obtain consent or permissions. There is no general possibility for an IS to proof on-line that it did indeed obtain the
527 response from the Principal. The IS can and should of course authenticate the Principal, and could then save the proof
528 of authentication, such as an assertion. There is little point in forwarding such an assertion to the WSC as proof, as
529 an [authentication assertion](#) will contain the NameID of the Principal as known to the IS, not to the WSC. An IS that is
530 closely associated with an identity provider, i.e. has the same providerID as that identity provider, could actually issue
531 an assertion that states that the Principal as known to the WSC was present. Such statements could be added as SOAP
532 header to the `InteractionResponse` message (see [[LibertySecMech](#)]).

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

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

543 **References**

544 **Normative**

- 545 [RFC2119] S. Bradner "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet
546 Engineering Task Force (March 1997). <http://www.ietf.org/rfc/rfc2119.txt>
- 547 [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., eds. (June
548 1999). "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, The Internet Engineering Task Force
549 <http://www.ietf.org/rfc/rfc2616.txt>
- 550 [RFC3066] Alvestrand, H., eds. (January 2001). "Tags for the Identification of Languages," RFC 3066., Internet
551 Engineering Task Force <http://www.ietf.org/rfc/rfc3066.txt>
- 552 [LibertyDisco] Hodges, Jeff, Cahill, Conor, eds. "Liberty ID-WSF Discovery Service Specification," Version 2.0,
553 Liberty Alliance Project (30 July, 2006). <http://www.projectliberty.org/specs>
- 554 [LibertySecMech] Hirsch, Frederick, eds. "Liberty ID-WSF Security Mechanisms Core," Version v2.0, Liberty
555 Alliance Project (30 July, 2006). <http://www.projectliberty.org/specs>
- 556 [LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, Whitehead, Greg, Madsen, Paul, eds. "Lib-
557 erty ID-WSF SOAP Binding Specification," Version 2.0, Liberty Alliance Project (30 July, 2006).
558 <http://www.projectliberty.org/specs>
- 559 [Schema1-2] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (28 October
560 2004). "XML Schema Part 1: Structures Second Edition," Recommendation, World Wide Web Consortium
561 <http://www.w3.org/TR/xmlschema-1/>
- 562 [SOAPv1.1] "Simple Object Access Protocol (SOAP) 1.1," Box, Don, Ehnebuske, David, Kakivaya, Gopal, Layman,
563 Andrew, Mendelsohn, Noah, Nielsen, Henrik Frystyk, Winer, Dave, eds. World Wide Web Consortium W3C
564 Note (08 May 2000). <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- 565 [WML] "Wireless Markup Language Version 1.3 Specification," Version 1.3, Open Mobile Alliance
566 <http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html>
- 567 [WSAv1.0] "Web Services Addressing (WS-Addressing) 1.0," Gudgin, Martin, Hadley, Marc, Rogers, Tony, eds.
568 World Wide Web Consortium W3C Recommendation (9 May 2006). <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>
- 570 [WSDLv1.1] "Web Services Description Language (WSDL) 1.1," Christensen, Erik, Curbera, Francisco, Meredith,
571 Greg, Weerawarana, Sanjiva, eds. World Wide Web Consortium W3C Note (15 March 2001).
572 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>

573 **Informative**

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

578 A. Interaction Service XSD

```

579 <?xml version="1.0" encoding="UTF-8"?>
580 <xs:schema targetNamespace="urn:liberty:is:2006-08"
581   xmlns="urn:liberty:is:2006-08"
582   xmlns:is="urn:liberty:is:2006-08"
583   xmlns:lu="urn:liberty:util:2006-08"
584   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
585   xmlns:ds="http://www.w3.org/2000/09/xml:dsig#"
586   xmlns:xs="http://www.w3.org/2001/XMLSchema"
587   elementFormDefault="qualified"
588   attributeFormDefault="unqualified"
589   version="2.0">
590
591   <xs:import namespace="urn:liberty:util:2006-08"
592     schemaLocation="liberty-idwsf-utility-v2.0.xsd"/>
593
594   <xs:import namespace="http://schemas.xmlsoap.org/soap/envelope/"
595     schemaLocation="http://schemas.xmlsoap.org/soap/envelope/" />
596   <xs:import namespace="http://www.w3.org/2000/09/xml:dsig#"
597     schemaLocation="http://www.w3.org/TR/2002/REC-xml:dsig-core-20020212/xml:dsig-core-sch
598     ema.xsd" />
599
600   <xs:annotation>
601     <xs:documentation>
602       The source code in this XSD file was excerpted verbatim from:
603
604       Liberty ID-WSF Interaction Service Specification
605       Version 2.0
606       30 July, 2006
607
608       Copyright (c) 2006 Liberty Alliance participants, see
609       http://www.projectliberty.org/specs/idwsf_2_0_final_copyrights.php
610     </xs:documentation>
611   </xs:annotation>
612   <xs:element name="InteractionRequest" type="InteractionRequestType"/>
613   <xs:complexType name="InteractionRequestType">
614     <xs:sequence>
615       <xs:element ref="Inquiry" maxOccurs="unbounded"/>
616       <xs:element ref="ds:KeyInfo" minOccurs="0"/>
617     </xs:sequence>
618     <xs:attribute name="id" type="xs:ID" use="optional"/>
619     <xs:attribute name="language" type="xs:NMTOKENS" use="optional"/>
620     <xs:attribute name="maxInteractTime" type="xs:integer" use="optional"/>
621     <xs:attribute name="signed" type="xs:boolean" use="optional"/>
622   </xs:complexType>
623
624   <xs:element name="Inquiry" type="InquiryType"/>
625   <xs:complexType name="InquiryType">
626     <xs:sequence>
627       <xs:element ref="Help" minOccurs="0"/>
628       <xs:choice maxOccurs="unbounded">
629         <xs:element ref="Select" minOccurs="0" maxOccurs="unbounded"/>
630         <xs:element name="Confirm" type="InquiryElementType"
631           minOccurs="0" maxOccurs="unbounded"/>
632         <xs:element ref="Text" minOccurs="0" maxOccurs="unbounded"/>
633       </xs:choice>
634     </xs:sequence>
635     <xs:attribute name="id" type="xs:ID" use="optional"/>
636     <xs:attribute name="title" type="xs:string" use="optional"/>
637   </xs:complexType>
638
639   <xs:element name="Help" type="HelpType"/>
640   <xs:complexType name="HelpType">
641     <xs:attribute name="label" type="xs:string" use="optional"/>
642     <xs:attribute name="link" type="xs:anyURI" use="optional"/>
643     <xs:attribute name="moreLink" type="xs:anyURI" use="optional"/>

```

```

644     </xs:complexType>
645
646     <xs:element name="Hint" type="xs:string" />
647
648     <xs:element name="Select" type="SelectType" />
649     <xs:complexType name="SelectType">
650         <xs:complexContent>
651             <xs:extension base="InquiryElementType">
652                 <xs:sequence>
653                     <xs:element name="Item" minOccurs="2" maxOccurs="unbounded">
654                         <xs:complexType>
655                             <xs:sequence>
656                                 <xs:element ref="Hint" minOccurs="0" />
657                             </xs:sequence>
658                             <xs:attribute name="label" type="xs:string" use="optional" />
659                             <xs:attribute name="value" type="xs:NMTOKEN" use="required" />
660                         </xs:complexType>
661                     </xs:element>
662                 </xs:sequence>
663                 <xs:attribute name="multiple" type="xs:boolean" use="optional" default="false" />
664             </xs:extension>
665         </xs:complexContent>
666     </xs:complexType>
667
668     <xs:element name="Text" type="TextType" />
669     <xs:complexType name="TextType">
670         <xs:complexContent>
671             <xs:extension base="InquiryElementType">
672                 <xs:attribute name="minChars" type="xs:integer" use="optional" />
673                 <xs:attribute name="maxChars" type="xs:integer" use="optional" />
674                 <xs:attribute name="format" type="xs:string" use="optional" />
675             </xs:extension>
676         </xs:complexContent>
677     </xs:complexType>
678
679     <xs:complexType name="InquiryElementType" abstract="true">
680         <xs:sequence>
681             <xs:element ref="Help" minOccurs="0" />
682             <xs:element ref="Hint" minOccurs="0" />
683             <xs:element name="Label" type="xs:normalizedString" minOccurs="0" />
684             <xs:element name="Value" type="xs:normalizedString" minOccurs="0" />
685         </xs:sequence>
686         <xs:attribute name="name" type="xs:ID" use="required" />
687     </xs:complexType>
688
689     <xs:element name="InteractionResponse" type="InteractionResponseType" />
690     <xs:complexType name="InteractionResponseType">
691         <xs:sequence>
692             <xs:element ref="lu:Status" />
693             <xs:choice>
694                 <xs:element name="InteractionStatement" type="InteractionStatementType"
695                     minOccurs="0" maxOccurs="unbounded" />
696                 <xs:element name="Parameter" type="ParameterType" minOccurs="0" maxOccurs="unbounded" />
697             </xs:choice>
698         </xs:sequence>
699     </xs:complexType>
700     <xs:complexType name="InteractionStatementType">
701         <xs:sequence>
702             <xs:element ref="Inquiry" maxOccurs="unbounded" />
703             <xs:element ref="ds:Signature" />
704         </xs:sequence>
705     </xs:complexType>
706     <xs:complexType name="ParameterType">
707         <xs:attribute name="name" type="xs:ID" use="required" />
708         <xs:attribute name="value" type="xs:string" use="required" />
709     </xs:complexType>
710

```

711 </xs: schema>
712

713 B. WSDL

```

714 <?xml version="1.0"?>
715 <definitions
716   name="id-wsf-is_2006-08_wsd_interface"
717   targetNamespace="urn:liberty:is:2006-08"
718   xmlns:tns="urn:liberty:is:2006-08"
719   xmlns="http://schemas.xmlsoap.org/wsdl/"
720   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
721   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
722   xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl"
723   xmlns:is="urn:liberty:is:2006-08"
724   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
725   xsi:schemaLocation="http://schemas.xmlsoap.org/wsdl/
726     http://schemas.xmlsoap.org/wsdl/
727     http://www.w3.org/2006/02/addressing/wsdl
728     http://www.w3.org/2006/02/addressing/wsdl/ws-addr-wsd.xsd">
729
730   <xsd:documentation>
731
732     The source code in this XSD file was excerpted verbatim from:
733
734     Liberty ID-WSF Interaction Service Specification
735
736     Copyright (c) 2006 Liberty Alliance participants, see
737     http://www.projectliberty.org/specs/idwsf_2_0_final_copyrights.php
738
739   </xsd:documentation>
740
741   <types>
742     <xsd:import namespace="urn:liberty:is:2006-08"
743       schemaLocation="liberty-idwsf-interaction-svc-v2.0.xsd"/>
744   </types>
745
746   <!-- Messages -->
747
748   <message name="InteractionRequest">
749     <part name="body" type="is:InteractionRequest"/>
750   </message>
751
752   <message name="InteractionResponse">
753     <part name="body" type="is:InteractionResponse"/>
754   </message>
755
756   <!-- Port Type -->
757
758   <portType name="ISPort">
759     <operation name="ISInteraction">
760       <input message="tns:InteractionRequest"
761 wsaw:Action="urn:liberty:is:2006-08:InteractionRequest" />
762       <output message="tns:InteractionResponse"
763 wsaw:Action="urn:liberty:is:2006-08:InteractionResponse" />
764     </operation>
765   </portType>
766
767   <!--
768   An example of a binding and service that can be used with this
769   abstract service description is provided below.
770   -->
771
772   <binding name="ISBinding" type="tns:ISPort">
773
774     <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
775
776     <operation name="Interaction">
777       <soap:operation soapAction="urn:liberty:is:2006-08:Interaction"/>
778   <input> <soap:body use="literal" /> </input>

```

```
779 <output> <soap:body use="literal" /> </output>
780   </operation>
781 </binding>
782
783 <service name="InteractionService">
784   <port name="ISPort" binding="tns:ISBinding">
785
786     <!-- Modify with the REAL SOAP endpoint -->
787
788     <soap:address location="http://example.com/id-wsf/is"/>
789   </port>
790 </service>
791
792 </definitions>
793
```

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

```

795 <?xml version="1.0" encoding="UTF-8"?>
796 <!-- This stylesheet converts an is:Inquiry into an HTML form.
797 Note that this is just a simple example that does not render all required elements.
798 Note the use of xsl:parameters to insert some session information, obviously other
799 techniques can be used.
800 Note for Hints this stylesheet adds a reference to a "showHint" script, but such script
801 is not defined here.
802 -->
803 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
804   xmlns:is="urn:liberty:is:2006-08" exclude-result-prefixes="is">
805   <xsl:output method="xml" version="4.0" encoding="UTF-8" omit-xml-declaration="yes" />
806   <xsl:param name="jsessionId">null</xsl:param>
807   <xsl:param name="messageID">null</xsl:param>
808   <xsl:template match="/">
809     <xsl:apply-templates select="//is:Inquiry" />
810   </xsl:template>
811
812   <xsl:template match="is:Inquiry">
813     <html>
814       <head>
815         <title>
816           <xsl:value-of select="@title" />
817         </title>
818       </head>
819       <body>
820         <h2>
821           <xsl:value-of select="@title" />
822         </h2>
823         <xsl:element name="form">
824           <xsl:attribute name="method">get</xsl:attribute>
825           <xsl:attribute name="action">
826             submit;jsessionId=<xsl:value-of select="$jsessionId" />
827           </xsl:attribute>
828           <xsl:element name="input">
829             <xsl:attribute name="type">hidden</xsl:attribute>
830             <xsl:attribute name="name">msg</xsl:attribute>
831             <xsl:attribute name="value"><xsl:value-of select="$messageID" /></xsl:attribute>
832           </xsl:element>
833           <xsl:apply-templates select="is:Confirm" /><br/>
834           <xsl:apply-templates select="is:Select" /><br/>
835           <br/>
836           <input type="submit" value="Submit" />
837         </xsl:element>
838       </body>
839     </html>
840     <xsl:apply-templates select="is:Help" />
841   </p>
842 </body>
843 </html>
844 </xsl:template>
845
846   <xsl:template match="is:Confirm">
847     <xsl:value-of select="is:Label" />
848     <xsl:element name="label">
849       <xsl:attribute name="for">isid-<xsl:value-of select="@name" />-yes</xsl:attribute>
850       Yes
851     </xsl:element>
852     <xsl:element name="input">
853       <xsl:attribute name="type">radio</xsl:attribute>
854       <xsl:attribute name="checked"></xsl:attribute>
855       <xsl:attribute name="name">is-confirm-yes-<xsl:value-of select="@name" /></xsl:attribute>
856       <xsl:attribute name="id">isid-<xsl:value-of select="@name" />-yes</xsl:attribute>
857     </xsl:element>
858     <xsl:element name="label">
859       <xsl:attribute name="for">isid-<xsl:value-of select="@name" />-no</xsl:attribute>

```

```
860         No
861     </xsl:element>
862     <xsl:element name="input">
863         <xsl:attribute name="type">radio</xsl:attribute>
864         <xsl:attribute name="name">is-confirm-no-<xsl:value-of select="@name"/></xsl:attribute>
865         <xsl:attribute name="id">isid-<xsl:value-of select="@name"/>-no</xsl:attribute>
866     </xsl:element>
867 </xsl:template>
868
869 <xsl:template match="is:Select">
870     <xsl:element name="label">
871         <xsl:value-of select="is:Label"/>
872         <xsl:attribute name="for">isid-<xsl:value-of select="@name"/></xsl:attribute>
873     </xsl:element>
874     <xsl:element name="select">
875         <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
876         <xsl:attribute name="id">isid-<xsl:value-of select="@name"/></xsl:attribute>
877         <xsl:apply-templates select="is:Item"/>
878     </xsl:element>
879 </xsl:template>
880
881 <xsl:template match="is:Item">
882     <xsl:element name="option">
883         <xsl:attribute name="label"><xsl:value-of select="@label"/></xsl:attribute>
884         <xsl:attribute name="value"><xsl:value-of select="@value"/></xsl:attribute>
885         <xsl:value-of select="@label"/>
886         <xsl:apply-templates select="is:Hint"/>
887     </xsl:element>
888 </xsl:template>
889 <xsl:template match="is:Hint">
890     <xsl:attribute name="onmouseover">showHint(<xsl:value-of select="."/>)</xsl:attribute>
891 </xsl:template>
892 <xsl:template match="is:Help">
893     <p id="help"><b>Help</b><br/>
894         <xsl:value-of select="."/>
895         <xsl:element name="a">
896             <xsl:attribute name="href">
897                 <xsl:value-of select="@morelink"/>
898             </xsl:attribute>
899             More information
900         </xsl:element>
901     </p>
902 </xsl:template>
903 </xsl:stylesheet>
904
```

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

```

906 <?xml version="1.0" encoding="UTF-8"?>
907 <!-- This stylesheet converts an is:Inquiry into a WML deck.
908      This is only an example stylesheet that does not render all required elements.
909      In fact it only renders Confirm elements, and hence is barely sufficient to handle
910      the example in the specification.
911      Note the use of xsl:parameters to insert some session information, obviously other
912      techniques can be used.
913      TODO: add a least support for Help elements. -->
914
915 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
916       xmlns:is="urn:liberty:is:2006-08" exclude-result-prefixes="is">
917
918   <xsl:output
919     method="xml"
920     version="1.0"
921     encoding="UTF-8"
922     omit-xml-declaration="no"
923     doctype-public="-//WAPFORUM//DTD WML 1.1//EN"
924     doctype-system="http://www.wapforum.org/DTD/wml_1.1.xml"
925     media-type="text/vnd.wap.wml" />
926
927   <xsl:param name="jsessionId">null</xsl:param>
928   <xsl:param name="messageID">null</xsl:param>
929   <xsl:param name="card-index">1</xsl:param>
930
931   <xsl:template match="/">
932     <wml>
933       <template>
934         <do type="prev">
935           <prev/>
936         </do>
937       </template>
938       <xsl:apply-templates select="//is:Inquiry" />
939     </wml>
940   </xsl:template>
941
942   <xsl:template match="is:Inquiry">
943     <xsl:element name="card">
944       <xsl:attribute name="id">inquiry-<xsl:value-of select="$card-index"/></xsl:attribute>
945       <xsl:attribute name="title"><xsl:value-of select="@title"/></xsl:attribute>
946       <xsl:apply-templates select="is:Confirm"/>
947     </xsl:element>
948   </xsl:template>
949
950   <xsl:template match="is:Confirm">
951     <p><xsl:value-of select="is:Label"/><br/>
952     <anchor>
953       <xsl:element name="go">
954         <xsl:attribute name="href">
955           submit;jsessionId=<xsl:value-of select="$jsessionId"/>
956         </xsl:attribute>
957         <xsl:attribute name="method">get</xsl:attribute>
958         <xsl:element name="postfield">
959           <xsl:attribute name="name">msg</xsl:attribute>
960           <xsl:attribute name="value">
961             <xsl:value-of select="$messageID"/>
962           </xsl:attribute>
963         </xsl:element>
964         <xsl:element name="postfield">
965           <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
966           <xsl:attribute name="value">1</xsl:attribute>
967         </xsl:element>
968       </xsl:element>>Yes</anchor><br/>
969     </anchor>
970     <xsl:element name="go">
  
```

```
971     <xsl:attribute name="href">
972         submit;jsessionId=<xsl:value-of select="$jsessionid"/>
973     </xsl:attribute>
974     <xsl:attribute name="method">get</xsl:attribute>
975     <xsl:element name="postfield">
976         <xsl:attribute name="name">msg</xsl:attribute>
977         <xsl:attribute name="value"><xsl:value-of select="$messageID"/></xsl:attribute>
978     </xsl:element>
979     <xsl:element name="postfield">
980         <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
981         <xsl:attribute name="value">0</xsl:attribute>
982     </xsl:element>
983 </xsl:element>No</anchor><br/>
984 </p>
985 </xsl:template>
986
987 </xsl:stylesheet>
988
```