



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 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-2.0-diff-v1.0.pdf

1

Notice

2 This document has been prepared by Sponsors of the Liberty Alliance. Permission is hereby granted to use the document
3 solely for the purpose of implementing the Specification. No rights are granted to prepare derivative works of this
4 Specification. Entities seeking permission to reproduce portions of this document for other uses must contact the Liberty
5 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 rights,
7 including without limitation, patent rights. The Sponsors of and any other contributors to the Specification are not and
8 shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual
9 property rights. **This Specification is provided "AS IS", and no participant in the Liberty Alliance makes any**
10 **warranty of any kind, express or implied, including any implied warranties of merchantability, non-infringe-**
11 **ment of third party intellectual property rights, and fitness for a particular purpose.** Implementers of this
12 Specification are advised to review the Liberty Alliance Project's website (<http://www.projectliberty.org/>) for infor-
13 mation concerning any Necessary Claims Disclosure Notices that have been received by the Liberty Alliance
14 Management Board.

15 Copyright © 2007 2FA Technology; Adobe Systems; Agencia Catalana De Certificacio; America Online, Inc.; Amer-
16 ican Express Company; Amsoft Systems Pvt Ltd.; Avatier Corporation; BIPAC; BMC Software, Inc.; Axalto; Bank of
17 America Corporation; Beta Systems Software AG; BIPAC; British Telecommunications plc; Computer Associates
18 International, Inc.; Credentica; DataPower Technology, Inc.; Deutsche Telekom AG, T-Com; Diamelle Technologies,
19 Inc.; Diversinet Corp.; Drummond Group Inc.; Enosis Group LLC; Entrust, Inc.; Epok, Inc.; Ericsson; Falkin Systems
20 LLC; Fidelity Investments; Forum Systems, Inc.; France Télécom; French Government Agence pour le développement
21 de l'administration électronique (ADAE); Fugen Solutions, Inc; Fulvens Ltd.; GSA Office of Governmentwide Policy;
22 Gamefederation; Gemalto; General Motors; GeoFederation; Giesecke & Devrient GmbH; Hewlett-Packard GSA Office
23 Company; Hochhauser & Co., Policy; Hewlett-Packard LLC; IBM Corporation; Intel Corporation; Intuit Inc.; Kant-
24 ega; Kayak Interactive; Livo Technologies; Luminance Consulting Services; MasterCard International; MedCommons
25 Inc.; Mobile Telephone Networks (Pty) Ltd; NEC Corporation; NTT DoCoMo, Inc.; Netegrity, Inc.; Neustar, Inc.;
26 New Zealand Government State Services Commission; Nippon Telegraph and Telephone Corporation; Nokia Corpo-
27 ration; Novell, Inc.; NTT DoCoMo, Inc.; OpenNetwork; Oracle Corporation; Ping Identity Corporation; RSA Security
28 Inc.; Reactivity Inc.; Royal Mail Group plc; RSA Security Inc.; SAP AG; Senforce; Sharp Laboratories of America;
29 Sigaba; SmartTrust; Sony Corporation; Sun Microsystems, Inc.; Supremacy Financial Corporation; Symlabs, Inc.;
30 Telecom Italia S.p.A.; Telefónica Móviles, S.A.; Telenor R&D; Thales e-Security; Trusted Network Technologies;
31 UNINETT AS; UTI; VeriSign, Inc.; Vodafone Group Plc.; Wave Systems Corp. All rights reserved.

32 Liberty Alliance Project
33 Licensing Administrator
34 c/o IEEE-ISTO
35 445 Hoes Lane
36 Piscataway, NJ 08855-1331, USA
37 info@projectliberty.org

38 Contents

39	1. Notation and Conventions	5
40	2. Overview	6
41	3. Interaction Service	9
42	3.1. Service Type	9
43	3.2. wsa:Action URIs.....	9
44	3.3. Interaction Request.....	10
45	3.3.1. The InteractionRequest Element.....	10
46	3.3.2. The Inquiry Element.....	11
47	3.3.3. Example Request.....	14
48	3.3.4. Processing Rules	14
49	3.4. Interaction Response.....	15
50	3.4.1. The InteractionResponse Element	15
51	3.4.2. Processing Rules	17
52	4. Security Considerations	18
53	References.....	19
54	A. Interaction Service XSD.....	20
55	B. WSDL	22
56	C. Example XSL Stylesheet for HTML Forms (non-normative)	24
57	D. Example XSL Stylesheet for WML Forms (non-normative)	26

58 1. Notation and Conventions

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

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

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

66 The following namespaces are referred to in this document:

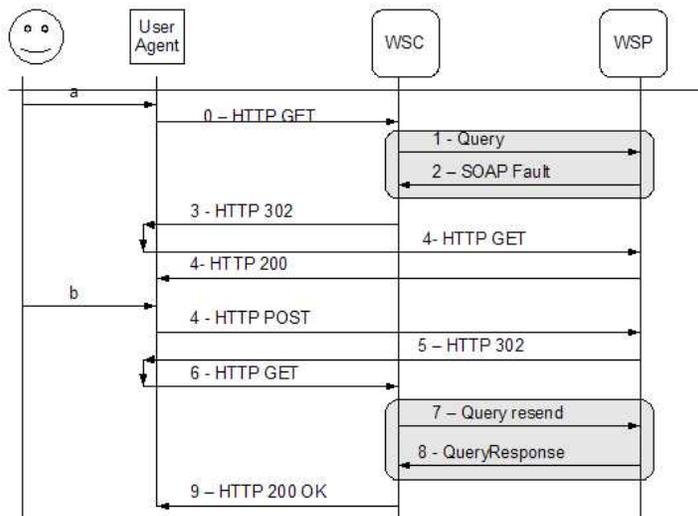
- 67 • The prefix *is*: stands for the ID-WSF working namespace for the interaction service (*urn:liberty:is:2006-08*). This
68 namespace is the default for instance fragments, type names, and element names in this document.
- 69 • The prefix *disco*: stands for the ID-WSF working namespace for the [LibertyDisco]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

78 2. Overview

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

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

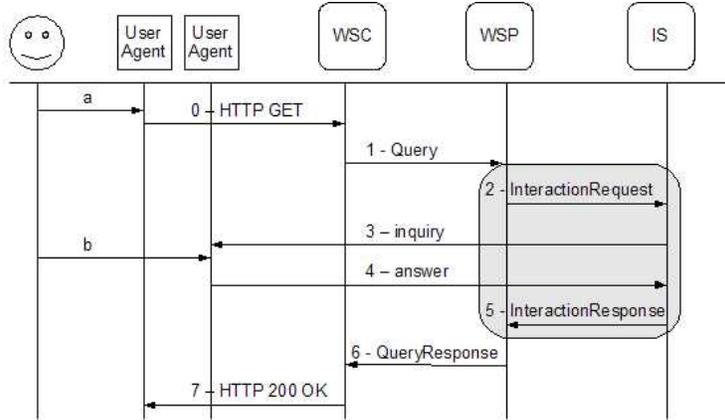
- 92 1. The WSC can indicate in the invocation message to the WSP that the resource owner is visiting the WSC and that
 93 it is ready to redirect the resource owner to the WSP. The WSP could then, in its response, ask the WSC to redirect
 94 the user (user agent) to itself (the WSP). This will cause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



98

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

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



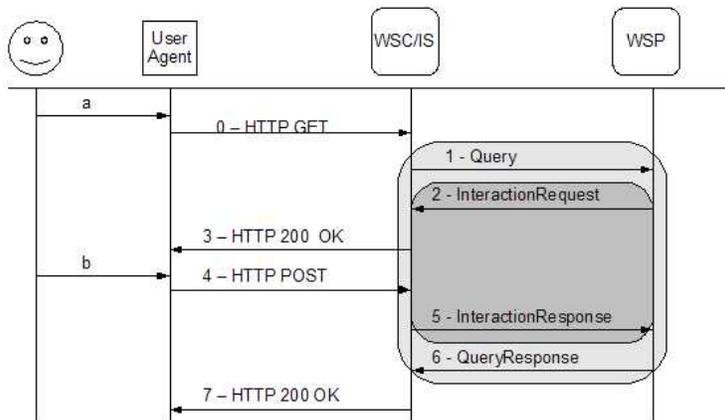
107

108

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

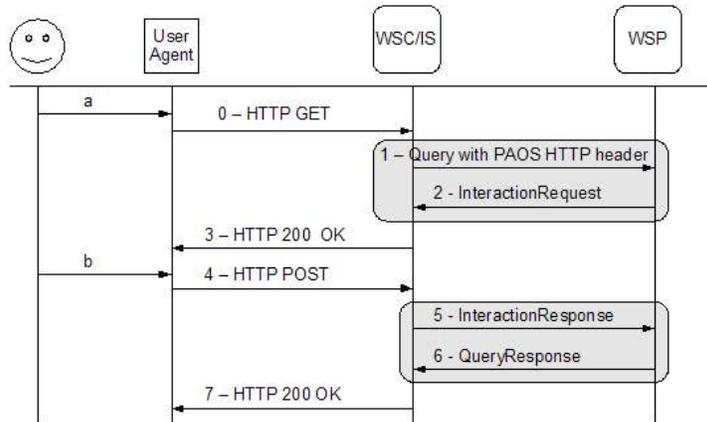
109 3. The WSC can indicate in the invocation message to the WSP that the resource owner is visiting the WSC and that
 110 it is willing and able to present questions to the visiting resource owner. The WSC effectively offers an interaction
 111 service to the WSP. The WSP could invoke that service with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

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



124

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



126

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

127
128

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

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

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

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

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

145 3. Interaction Service

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

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

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

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

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

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

175 3.1. Service Type

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

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

178 3.2. wsa:Action URIs

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

181 This specification defines the following action identifiers:

- 182 • *urn:liberty:is:2006-08:InteractionRequest*
- 183 • *urn:liberty:is:2006-08:InteractionResponse*

184 3.3. Interaction Request

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

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

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

191 3.3.1. The InteractionRequest Element

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

194 `Inquiry` [Required]

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

197 `ds:KeyInfo` [Optional]

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

202 `id`

203 Allows the element to be signed according to the rules in [LibertySecMech].

204 `language` [Optional]

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

212 `signed` [Optional]

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

217 `maxInteractTime` [Optional]

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

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

```
222 <xs:element name="InteractionRequest" type="InteractionRequestType"/>
223 <xs:complexType name="InteractionRequestType">
224   <xs:sequence>
225     <xs:element ref="Inquiry" maxOccurs="unbounded"/>
226     <xs:element ref="ds:KeyInfo" minOccurs="0"/>
```

```

227     </xs:sequence>
228     <xs:attribute name="id" type="xs:ID" use="optional"/>
229     <xs:attribute name="language" type="xs:NMTOKENS" use="optional"/>
230     <xs:attribute name="maxInteractTime" type="xs:integer" use="optional"/>
231     <xs:attribute name="signed" type="xs:token" use="optional"/>
232 </xs:complexType>
233

```

234 3.3.2. The Inquiry Element

235 The Inquiry element contains:

236 `Help` [Optional]

237 Contains informal text regarding the inquiry, which may be presented to the user (See further definition 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 are
240 defined below.

241 `id` [Optional]

242 The `id` attribute MUST be present if the encompassing `<InteractionRequest>` contains the `signed` attribute
243 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 of
247 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 3.3.2.1. The Help Element

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

270 `label` [Optional]

271 Specifies a label relating to the help text.

272 `link` [Optional]

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

275 moreLink [Optional]

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

279 The schema fragment for the Help element is:

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

287 3.3.2.2. The InquiryElementType

288 The InquiryElementType is an abstract type that defines the common content for query elements. The type con-
289 tains:

290 Help [Optional]

291 See definition of the Help element above.

292 Hint [Optional]

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

296 Label [Optional]

297 An IS provider is expected to present the content of Label elements as question labels. Note that the text value
298 of a <Label> is normalized.

299 Value [Optional]

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

305 name [Required]

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

310 The schema fragment for the InquiryElementType is:

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

323 3.3.2.3. <InquiryElementType> Subtypes

324 The defined <InquiryElementType> subtypes are:

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

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

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

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

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

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

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

```

363     <xs:element name="Text" type="TextType"/>
364     <xs:complexType name="TextType">
365       <xs:complexContent>
366         <xs:extension base="InquiryElementType">
367           <xs:attribute name="minChars" type="xs:integer" use="optional"/>
368           <xs:attribute name="maxChars" type="xs:integer" use="optional"/>
369           <xs:attribute name="format" type="xs:string" use="optional"/>
370         </xs:extension>
371       </xs:complexContent>
372     </xs:complexType>
373 
```

374 3.3.3. Example Request

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

```

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

```

398 3.3.4. Processing Rules

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

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

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

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

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

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

421 3.4. Interaction Response

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

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

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

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

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

439 3.4.1. The InteractionResponse Element

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

441 Parameter [Optional]

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

445 or:

446 InteractionStatement [Optional]

447 Contains one or more signed Inquiry elements.

448 The Parameter element has two attributes:

449 name [Required]

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

451 value [Required]

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

454 The <InteractionStatement> consists of:

455 Inquiry [Optional]

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

460 ds:Signature [Optional]

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

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

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

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

```
489     <InteractionResponse>
490         <Status code="OK" />eode="OK" />
491         <Parameter name="addresschoice" value="always" />
492     </InteractionResponse>
```

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

```
494     <InteractionResponse>
495         <Status code="OK" />eode="OK" />
496         <InteractionStatement>
497             <Inquiry title="Profile Provider Question" id="inquiry-3d4e2f8a37213b">
498                 <Select name="addresschoice">
499                     <Label>Do you want to share your address with service-provider.com?</Label>
500                     <Value>always</Value>
501                 </Select>
502             </Inquiry>
503             <ds:Signature>
504                 .... <ds:Reference>#inquiry-3d4e2f8a37213b</ds:Reference> ....
505             </ds:Signature>
506         </InteractionStatement>
507     </InteractionResponse>
```

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

```
509     <InteractionResponse>
510         <Status code="Cancel" />eode="Cancel" />
511     </InteractionResponse>
512
```

513 3.4.2. Processing Rules

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

517 4. Security Considerations

518 The interaction service is effectively acting to its client WSCs as a proxy for the Principal. It is therefore important
519 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
520 obtain consent or permissions. There is no general possibility for an IS to proof on-line that it did indeed obtain the
521 response from the Principal. The IS can and should of course authenticate the Principal, and could then save the proof
522 of authentication, such as an assertion. There is little point in forwarding such an assertion to the WSC as proof, as an
523 [authentication assertion](#) will contain the NameID of the Principal as known to the IS, not to the WSC. An IS that is
524 closely associated with an identity provider, *i.e.*, has the same providerID as that identity provider, could actually issue
525 an assertion that states that the Principal as known to the WSC was present. Such statements could be added as SOAP
526 header to the `InteractionResponse` message (see [[LibertySecMech](#)]).

527 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
528 changed the Principal's answers in the `<InteractionResponse>`. The interaction service client can verify the in-
529 tegrity of the response if the answered `Inquiry` is signed with a key that is: either shared between the Principal and
530 the WSC, or is the private key of the Principal and the WSC knows that the associated public key is bound to the
531 Principal. To this end the WSC can include such public asymmetric key in the `<InteractionRequest>`. Naturally
532 the WSC should have consent from the Principal to share that key with the IS. Use of a private key is preferred for a
533 more provable audit trail of the Principals answers to the inquiry.

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

537 References

538 Normative

- 539 [RFC2119] S. Bradner "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet Engi-
540 neering Task Force (March 1997). <http://www.ietf.org/rfc/rfc2119.txt>
- 541 [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., eds. (June 1999).
542 "Hypertext Transfer Protocol -- HTTP/1.1," RFC 2616, The Internet Engineering Task Force <http://>
543 www.ietf.org/rfc/rfc2616.txt
- 544 [RFC3066] Alvestrand, H., eds. (January 2001). "Tags for the Identification of Languages," RFC 3066., Internet En-
545 gineering Task Force <http://www.ietf.org/rfc/rfc3066.txt>
- 546 [LibertyDisco] Cahill, Conor, Hodges, Jeff, eds. "Liberty ID-WSF Discovery Service Specification," Version 2.0-
547 errata-v1.0, Liberty Alliance Project (29 November, 2006). <http://www.projectliberty.org/specs>
- 548 [LibertySecMech] Hirsch, Frederick, eds. "Liberty ID-WSF Security Mechanisms Core," Version 2.0-errata-v1.0,
549 Liberty Alliance Project (21 April, 2007). <http://www.projectliberty.org/specs>
- 550 [LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, Whitehead, Greg, Madsen, Paul, eds. "Liberty ID-
551 WSF SOAP Binding Specification," Version 2.0-errata-v1.0, Liberty Alliance Project (21 April, 2007). <http://>
552 www.projectliberty.org/specs
- 553 [LibertyIDWSFv20Errata] Champagne, Darryl, Lockhart, Rob, Tiffany, Eric, eds. "Liberty ID-WSF 2.0 Errata," Ver-
554 sion 1.0, Liberty Alliance Project (13 April, 2007). <http://www.projectliberty.org/specs>
- 555 [Schema1-2] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (28 October 2004).
556 "XML Schema Part 1: Structures Second Edition," Recommendation, World Wide Web Consortium <http://>
557 www.w3.org/TR/xmlschema-1/
- 558 [SOAPv1.1] "Simple Object Access Protocol (SOAP) 1.1," Box, Don, Ehnebuske, David, Kakivaya, Gopal, Layman,
559 Andrew, Mendelsohn, Noah, Nielsen, Henrik Frystyk, Winer, Dave, eds. World Wide Web Consortium W3C
560 Note (08 May 2000). <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- 561 [WML] "Wireless Markup Language Version 1.3 Specification," Version 1.3, Open Mobile Alliance <http://www.open->
562 [mobilealliance.org/tech/affiliates/wap/wapindex.html](http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html)
- 563 [WSAv1.0] "Web Services Addressing (WS-Addressing) 1.0," Gudgin, Martin, Hadley, Marc, Rogers, Tony, eds.
564 World Wide Web Consortium W3C Recommendation (9 May 2006). <http://www.w3.org/TR/2006/REC-ws->
565 [addr-core-20060509/](http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/)
- 566 [WSDLv1.1] "Web Services Description Language (WSDL) 1.1," Christensen, Erik, Curbera, Francisco, Meredith,
567 Greg, Weerawarana, Sanjiva, eds. World Wide Web Consortium W3C Note (15 March 2001). <http://>
568 www.w3.org/TR/2001/NOTE-wsdl-20010315

569 Informative

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

574 **A. Interaction Service XSD**

```

575 <?xml version="1.0" encoding="UTF-8"?>
576 <xs:schema targetNamespace="urn:liberty:is:2006-08"
577     xmlns="urn:liberty:is:2006-08"
578     xmlns:is="urn:liberty:is:2006-08"
579     xmlns:lu="urn:liberty:util:2006-08"
580     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
581     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
582     xmlns:xs="http://www.w3.org/2001/XMLSchema"
583     elementFormDefault="qualified"
584     attributeFormDefault="unqualified"
585     version="2.0">
586
587     <xs:import namespace="urn:liberty:util:2006-08"
588         schemaLocation="liberty-idwsf-utility-v2.0.xsd"/>
589
590     <xs:import namespace="http://schemas.xmlsoap.org/soap/envelope/"
591         schemaLocation="http://schemas.xmlsoap.org/soap/envelope/" />
592     <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
593         schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd" />
594
595     <xs:element name="InteractionRequest" type="InteractionRequestType"/>
596     <xs:complexType name="InteractionRequestType">
597         <xs:sequence>
598             <xs:element ref="Inquiry" maxOccurs="unbounded"/>
599             <xs:element ref="ds:KeyInfo" minOccurs="0"/>
600         </xs:sequence>
601         <xs:attribute name="id" type="xs:ID" use="optional"/>
602         <xs:attribute name="language" type="xs:NMTOKENS" use="optional"/>
603         <xs:attribute name="maxInteractTime" type="xs:integer" use="optional"/>
604         <xs:attribute name="signed" type="xs:token" use="optional"/>
605     </xs:complexType>
606
607     <xs:element name="Inquiry" type="InquiryType"/>
608     <xs:complexType name="InquiryType">
609         <xs:sequence>
610             <xs:element ref="Help" minOccurs="0"/>
611             <xs:choice maxOccurs="unbounded">
612                 <xs:element ref="Select" minOccurs="0" maxOccurs="unbounded"/>
613                 <xs:element name="Confirm" type="InquiryElementType"
614                     minOccurs="0" maxOccurs="unbounded"/>
615                 <xs:element ref="Text" minOccurs="0" maxOccurs="unbounded"/>
616             </xs:choice>
617         </xs:sequence>
618         <xs:attribute name="id" type="xs:ID" use="optional"/>
619         <xs:attribute name="title" type="xs:string" use="optional"/>
620     </xs:complexType>
621
622     <xs:element name="Help" type="HelpType"/>
623     <xs:complexType name="HelpType">
624         <xs:attribute name="label" type="xs:string" use="optional"/>
625         <xs:attribute name="link" type="xs:anyURI" use="optional"/>
626         <xs:attribute name="moreLink" type="xs:anyURI" use="optional"/>
627     </xs:complexType>
628
629     <xs:element name="Hint" type="xs:string"/>
630
631     <xs:element name="Select" type="SelectType"/>
632     <xs:complexType name="SelectType">
633         <xs:complexContent>
634             <xs:extension base="InquiryElementType">
635                 <xs:sequence>
636                     <xs:element name="Item" minOccurs="2" maxOccurs="unbounded">
637                         <xs:complexType>
638                             <xs:sequence>
639                                 <xs:element ref="Hint" minOccurs="0"/>

```

```

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

```

696 **B. WSDL**

```

697 <?xml version="1.0"?>
698 <definitions
699     name="id-wsf-is_2006-08_wsdll_interface"
700     targetNamespace="urn:liberty:is:2006-08"
701     xmlns:tns="urn:liberty:is:2006-08"
702     xmlns="http://schemas.xmlsoap.org/wsdl/"
703     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
704     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
705     xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl"
706     xmlns:is="urn:liberty:is:2006-08"
707     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
708     xsi:schemaLocation="http://schemas.xmlsoap.org/wsdl/
709         http://schemas.xmlsoap.org/wsdl/
710         http://www.w3.org/2006/02/addressing/wsdl
711         http://www.w3.org/2006/02/addressing/wsdl/ws-addr-wsdl.xsd">
712
713     <xsd:documentation>
714
715 The source code in this XSD file was excerpted verbatim from:
716
717 Liberty ID-WSF Interaction Service Specification
718
719 Copyright (c) 2006 Liberty Alliance participants, see
720 http://www.projectliberty.org/specs/idwsf_2_0_final_copyrights.php
721
722     </xsd:documentation>
723
724     <types>
725         <xsd:import namespace="urn:liberty:is:2006-08"
726             schemaLocation="liberty-idwsf-interaction-svc-v2.0.xsd"/>
727     </types>
728
729     <!-- Messages -->
730
731     <message name="InteractionRequest">
732         <part name="body" type="is:InteractionRequest"/>
733     </message>
734
735     <message name="InteractionResponse">
736         <part name="body" type="is:InteractionResponse"/>
737     </message>
738
739     <!-- Port Type -->
740
741     <portType name="ISPort">
742         <operation name="ISInteraction">
743             <input message="tns:InteractionRequest"
744                 wsaw:Action="urn:liberty:is:2006-08:InteractionRequest"/>
745             <output message="tns:InteractionResponse"
746                 wsaw:Action="urn:liberty:is:2006-08:InteractionResponse"/>
747             </operation>
748         </portType>
749
750     <!--
751     An example of a binding and service that can be used with this
752     abstract service description is provided below.
753     -->
754
755     <binding name="ISBinding" type="tns:ISPort">
756
757         <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
758
759         <operation name="Interaction">

```

```
762         <soap:operation soapAction="urn:liberty:is:2006-08:Interaction"/>
763         <input> <soap:body use="literal"/></input>
764         <output> <soap:body use="literal"/></output>
765     </operation>
766 </binding>
767
768 <service name="InteractionService">
769     <port name="ISPort" binding="tns:ISBinding">
770
771         <!-- Modify with the REAL SOAP endpoint -->
772
773         <soap:address location="http://example.com/id-wsf/is"/>
774     </port>
775 </service>
776
777 </definitions>
```

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

```

779 <?xml version="1.0" encoding="UTF-8"?>
780 <!-- This stylesheet converts an is:Inquiry into an HTML form.
781     Note that this is just a simple example that does not render all required elements.
782     Note the use of xsl:parameters to insert some session information, obviously other
783     techniques can be used.
784     Note for Hints this stylesheet adds a reference to a "showHint" script, but such script
785     is not defined here.
786
787 -->
788
789 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
790     xmlns:is="urn:liberty:is:2006-08" exclude-result-prefixes="is">
791     <xsl:output method="xml" version="4.0" encoding="UTF-8" omit-xml-declaration="yes" />
792     <xsl:param name="jsessionId">null</xsl:param>
793     <xsl:param name="messageID">null</xsl:param>
794     <xsl:template match="/">
795         <xsl:apply-templates select="//is:Inquiry" />
796     </xsl:template>
797
798     <xsl:template match="is:Inquiry">
799         <html>
800             <head>
801                 <title>
802                     <xsl:value-of select="@title"/>
803                 </title>
804             </head>
805             <body>
806                 <h2>
807                     <xsl:value-of select="@title"/>
808                 </h2>
809                 <xsl:element name="form">
810                     <xsl:attribute name="method">get</xsl:attribute>
811                     <xsl:attribute name="action">
812                         submit;jsessionId=<xsl:value-of select="$jsessionId"/>
813                     </xsl:attribute>
814                     <xsl:element name="input">
815                         <xsl:attribute name="type">hidden</xsl:attribute>
816                         <xsl:attribute name="name">msg</xsl:attribute>
817                         <xsl:attribute name="value"><xsl:value-of select="$messageID"/></xsl:attribute>
818                     </xsl:element>
819                     <xsl:apply-templates select="is:Confirm"/><br/>
820                     <xsl:apply-templates select="is:Select"/><br/>
821                     <br/>
822                     <input type="submit" value="Submit"/>
823                 </xsl:element>
824             <p>
825                 <xsl:apply-templates select="is:Help"/>
826             </p>
827         </body>
828     </html>
829 </xsl:template>
830
831     <xsl:template match="is:Confirm">
832         <xsl:value-of select="is:Label"/>
833         <xsl:element name="label">
834             <xsl:attribute name="for">isid-<xsl:value-of select="@name"/>-yes</xsl:attribute>
835             Yes
836         </xsl:element>
837         <xsl:element name="input">
838             <xsl:attribute name="type">radio</xsl:attribute>
839             <xsl:attribute name="checked"></xsl:attribute>
840             <xsl:attribute name="name">is-confirm-yes-<xsl:value-of select="@name"/></xsl:attribute>
841             <xsl:attribute name="id">isid-<xsl:value-of select="@name"/>-yes</xsl:attribute>
842         </xsl:element>
843         <xsl:element name="label">

```

```

844         <xsl:attribute name="for">isid-<xsl:value-of select="@name"/>-no</xsl:attribute>
845         No
846     </xsl:element>
847     <xsl:element name="input">
848         <xsl:attribute name="type">radio</xsl:attribute>
849         <xsl:attribute name="name">is-confirm-no-<xsl:value-of select="@name"/></xsl:attribute>
850         <xsl:attribute name="id">isid-<xsl:value-of select="@name"/>-no</xsl:attribute>
851     </xsl:element>
852 </xsl:template>
853
854 <xsl:template match="is:Select">
855     <xsl:element name="label">
856         <xsl:value-of select="is:Label"/>
857         <xsl:attribute name="for">isid-<xsl:value-of select="@name"/></xsl:attribute>
858     </xsl:element>
859     <xsl:element name="select">
860         <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
861         <xsl:attribute name="id">isid-<xsl:value-of select="@name"/></xsl:attribute>
862         <xsl:apply-templates select="is:Item"/>
863     </xsl:element>
864 </xsl:template>
865
866 <xsl:template match="is:Item">
867     <xsl:element name="option">
868         <xsl:attribute name="label"><xsl:value-of select="@label"/></xsl:attribute>
869         <xsl:attribute name="value"><xsl:value-of select="@value"/></xsl:attribute>
870         <xsl:value-of select="@label"/>
871         <xsl:apply-templates select="is:Hint"/>
872     </xsl:element>
873 </xsl:template>
874 <xsl:template match="is:Hint">
875     <xsl:attribute name="onmouseover">showHint(<xsl:value-of select="."/>)</xsl:attribute>
876 </xsl:template>
877 <xsl:template match="is:Help">
878     <p id="help"><b>Help</b><br/>
879         <xsl:value-of select="."/>
880         <xsl:element name="a">
881             <xsl:attribute name="href">
882                 <xsl:value-of select="@morelink"/>
883             </xsl:attribute>
884             More information
885         </xsl:element>
886     </p>
887 </xsl:template>
888 </xsl:stylesheet>

```

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

```

890 <?xml version="1.0" encoding="UTF-8"?>
891 <!-- This stylesheet converts an is:Inquiry into a WML deck.
892      This is only an example stylesheet that does not render all required elements.
893      In fact it only renders Confirm elements, and hence is barely sufficient to handle
894      the example in the specification.
895      Note the use of xsl:parameters to insert some session information, obviously other
896      techniques can be used.
897
898 -->
899
900 <!--
901      TODO: add a least support for Help elements. -->
902
903 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
904      xmlns:is="urn:liberty:is:2006-08" exclude-result-prefixes="is">
905
906   <xsl:output
907     method="xml"
908     version="1.0"
909     encoding = "UTF-8"
910     omit-xml-declaration="no"
911     doctype-public="-//WAPFORUM//DTD WML 1.1//EN"
912     doctype-system="http://www.wapforum.org/DTD/wml_1.1.xml"
913     media-type="text/vnd.wap.wml" />
914
915   <xsl:param name="jsessionId">null</xsl:param>
916   <xsl:param name="messageID">null</xsl:param>
917   <xsl:param name="card-index">1</xsl:param>
918
919   <xsl:template match="/">
920     <wml>
921       <template>
922         <do type="prev">
923           <prev/>
924         </do>
925       </template>
926       <xsl:apply-templates select="//is:Inquiry" />
927     </wml>
928   </xsl:template>
929
930   <xsl:template match="is:Inquiry">
931     <xsl:element name="card">
932       <xsl:attribute name="id">inquiry-<xsl:value-of select="$card-index"/></xsl:attribute>
933       <xsl:attribute name="title"><xsl:value-of select="@title"/></xsl:attribute>
934       <xsl:apply-templates select="is:Confirm"/>
935     </xsl:element>
936   </xsl:template>
937
938   <xsl:template match="is:Confirm">
939     <p><xsl:value-of select="is:Label"/><br/>
940     <anchor>
941       <xsl:element name="go">
942         <xsl:attribute name="href">
943           submit;jsessionId=<xsl:value-of select="$jsessionId"/>
944         </xsl:attribute>
945         <xsl:attribute name="method">get</xsl:attribute>
946         <xsl:element name="postfield">
947           <xsl:attribute name="name">msg</xsl:attribute>
948           <xsl:attribute name="value">
949             <xsl:value-of select="$messageID"/>
950           </xsl:attribute>
951         </xsl:element>
952         <xsl:element name="postfield">
953           <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
954           <xsl:attribute name="value">1</xsl:attribute>

```

```
955         </xsl:element>
956     </xsl:element>Yes</anchor><br/>
957 <anchor>
958     <xsl:element name="go">
959         <xsl:attribute name="href">
960             submit;jsessionId=<xsl:value-of select="$jsessionid"/>
961         </xsl:attribute>
962         <xsl:attribute name="method">get</xsl:attribute>
963         <xsl:element name="postfield">
964             <xsl:attribute name="name">msg</xsl:attribute>
965             <xsl:attribute name="value"><xsl:value-of select="$messageID"/></xsl:attribute>
966         </xsl:element>
967         <xsl:element name="postfield">
968             <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
969             <xsl:attribute name="value">0</xsl:attribute>
970         </xsl:element>
971     </xsl:element>No</anchor><br/>
972 </p>
973 </xsl:template>
974
975 </xsl:stylesheet>
```