



# 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-errata-v1.0.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 © 2007 2FA Technology; Adobe Systems; Agencia Catalana De Certificacio; America Online, Inc.;  
16 American Express Company; Amsoft Systems Pvt Ltd.; Avatier Corporation; BIPAC; BMC Software, Inc.; Bank of  
17 America Corporation; Beta Systems Software AG; 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  
20 Systems LLC; Fidelity Investments; Forum Systems, Inc.; France Télécom; French Government Agence pour le  
21 développement de l'administration électronique (ADAE); Fugen Solutions, Inc; Fulvens Ltd.; GSA Office of  
22 Governmentwide Policy; Gamefederation; Gemalto; General Motors; GeoFederation; Giesecke & Devrient GmbH;  
23 Hewlett-Packard Company; Hochhauser & Co., LLC; IBM Corporation; Intel Corporation; Intuit Inc.; Kantega;  
24 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  
27 Corporation; Novell, Inc.; OpenNetwork; Oracle Corporation; Ping Identity Corporation; RSA Security Inc.;  
28 Reactivity Inc.; Royal Mail Group plc; SAP AG; Senforce; Sharp Laboratories of America; Sigaba; SmartTrust; Sony  
29 Corporation; Sun Microsystems, Inc.; Supremacy Financial Corporation; Symlabs, Inc.; Telecom Italia S.p.A.;  
30 Telefónica Móviles, S.A.; Telenor R&D; Thales e-Security; Trusted Network Technologies; UNINETT AS; UTI;  
31 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

---

## Contents

38		
39	1. Notation and Conventions .....	4
40	2. Overview .....	5
41	3. Interaction Service .....	9
42	3.1. Service Type .....	9
43	3.2. wsa:Action URIs .....	9
44	3.3. Interaction Request .....	9
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 .....	15
49	3.4. Interaction Response .....	15
50	3.4.1. The InteractionResponse Element .....	16
51	3.4.2. Processing Rules .....	18
52	4. Security Considerations .....	19
53	References .....	20
54	A. Interaction Service XSD .....	21
55	B. WSDL .....	23
56	C. Example XSL Stylesheet for HTML Forms (non-normative) .....	25
57	D. Example XSL Stylesheet for WML Forms (non-normative) .....	27

## 1. Notation and Conventions

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

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

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

The following namespaces are referred to in this document:

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

## 2. Overview

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

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

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

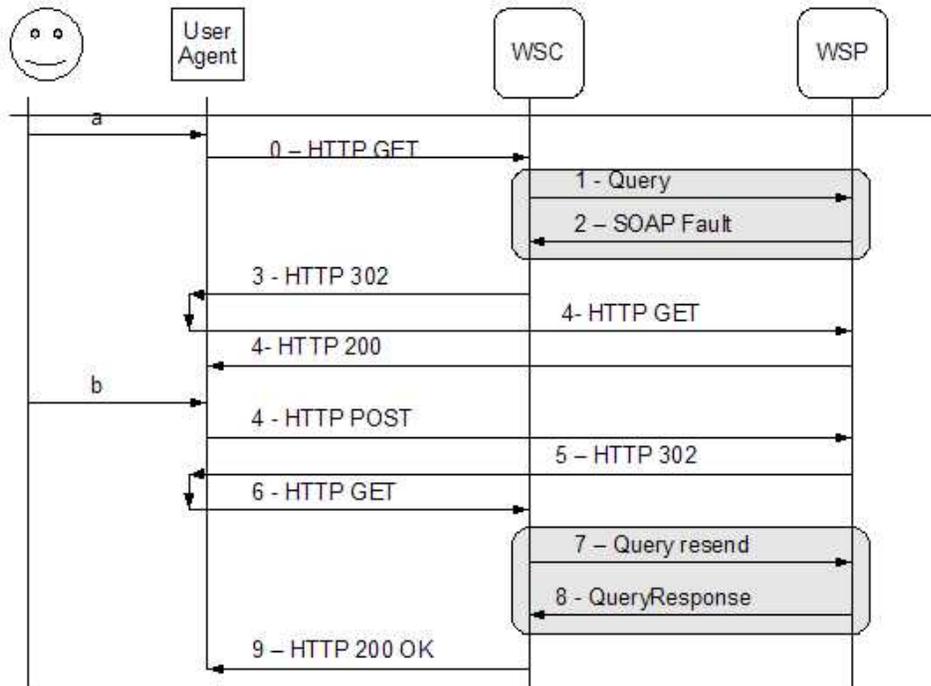
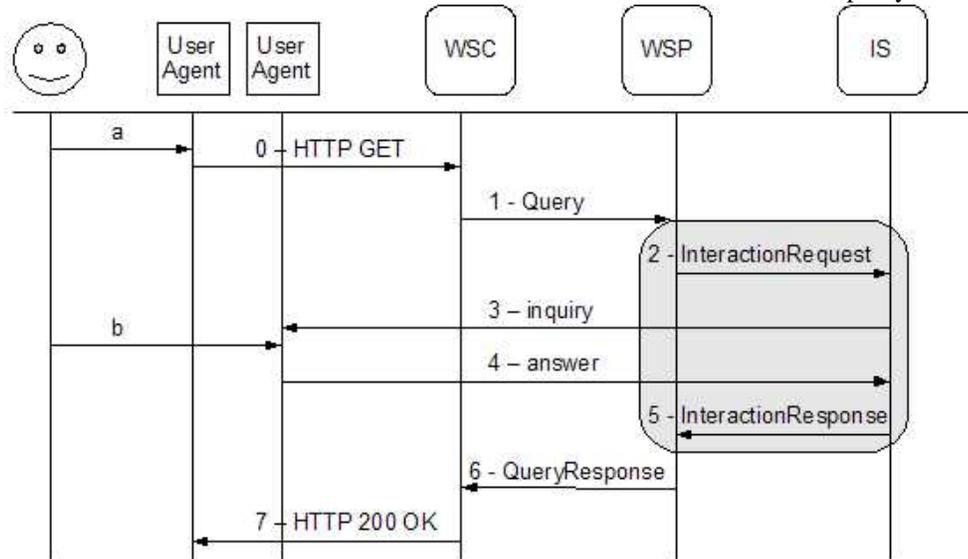


Figure 1. WSP Interacts with Principal by Requesting the WSC to Redirect the User Agent.

99  
 100  
 101  
 102  
 103  
 104  
 105

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



106

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

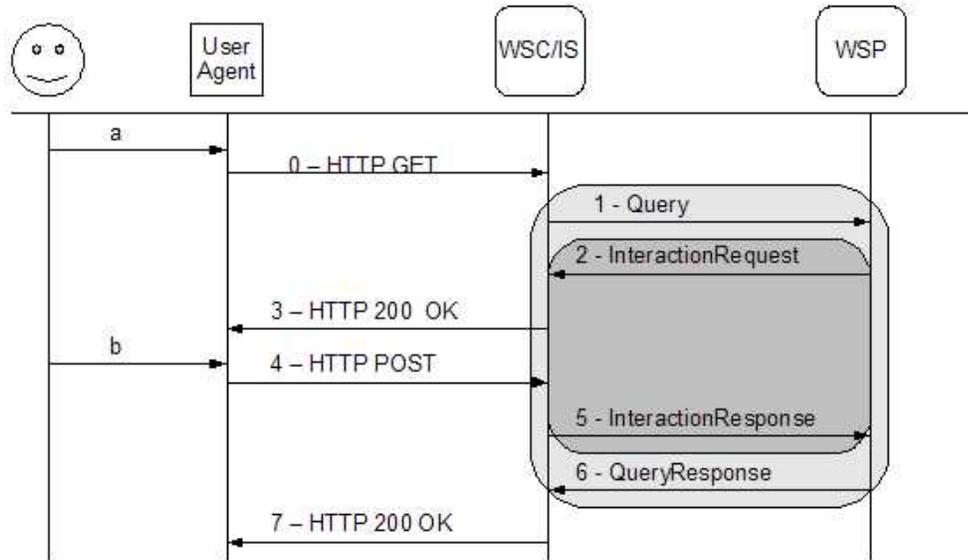
107

108  
 109  
 110  
 111  
 112  
 113  
 114

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

115  
 116  
 117  
 118  
 119  
 120  
 121  
 122

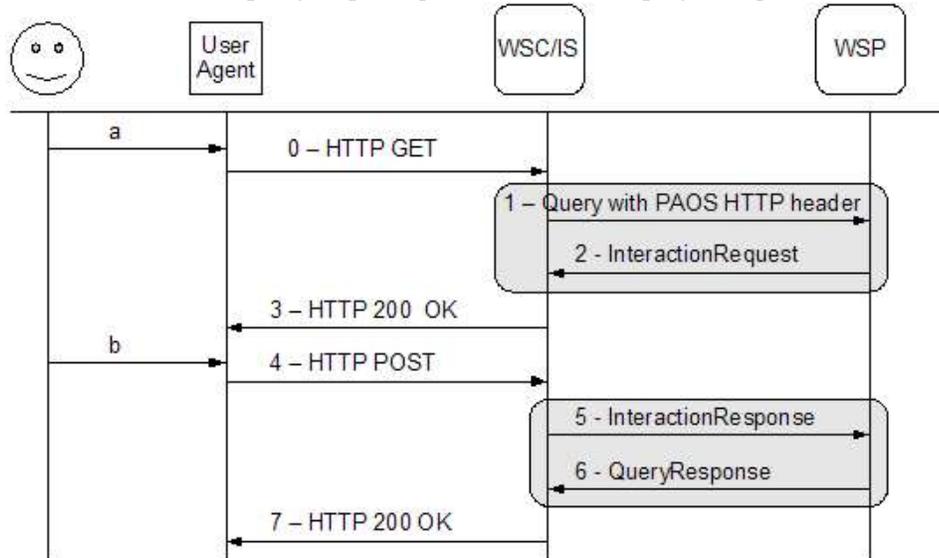
This method has two variants; depending on how the the WSP's InteractionRequest is bound to the underlying HTTP layer. In the first variant, the WSP sends its InteractionRequest to the WSC/IS on a separate HTTP process than that on which the WSC sent its original invocation. The WSC/IS, after posing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



123

124

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



125

126

127

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

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

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

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

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

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

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

## 3. Interaction Service

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

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

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

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

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

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

### 3.1. Service Type

An Interaction Service is identified by the service type URN:

*urn:liberty:is:2006-08*

### 3.2. wsa:Action URIs

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

This specification defines the following action identifiers:

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

## 3.3. Interaction Request

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

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

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

### 3.3.1. The InteractionRequest Element

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

#### Inquiry [Required]

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

#### ds:KeyInfo [Optional]

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

#### id

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

#### language [Optional]

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

#### signed [Optional]

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

218 maxInteractTime [Optional]

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

222 The schema fragment for the <InteractionRequest> is:

```
223 <xs:element name="InteractionRequest" type="InteractionRequestType"/>
224 <xs:complexType name="InteractionRequestType">
225   <xs:sequence>
226     <xs:element ref="Inquiry" maxOccurs="unbounded"/>
227     <xs:element ref="ds:KeyInfo" minOccurs="0"/>
228   </xs:sequence>
229   <xs:attribute name="id" type="xs:ID" use="optional"/>
230   <xs:attribute name="language" type="xs:NMTOKENS" use="optional"/>
231   <xs:attribute name="maxInteractTime" type="xs:integer" use="optional"/>
232   <xs:attribute name="signed" type="xs:token" use="optional"/>
233 </xs:complexType>
234
235
```

### 236 3.3.2. The Inquiry Element

237 The Inquiry element contains:

238 Help [Optional]

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

241 Element of type InquiryElementType [Zero or more]

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

244 id [Optional]

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

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

251 The schema fragment for the <Inquiry> is:

```
252 <xs:element name="Inquiry" type="InquiryType"/>
253 <xs:complexType name="InquiryType">
254   <xs:sequence>
255     <xs:element ref="Help" minOccurs="0"/>
256     <xs:choice maxOccurs="unbounded">
257       <xs:element ref="Select" minOccurs="0" maxOccurs="unbounded"/>
258       <xs:element name="Confirm" type="InquiryElementType"
259         minOccurs="0" maxOccurs="unbounded"/>
260       <xs:element ref="Text" minOccurs="0" maxOccurs="unbounded"/>
261     </xs:choice>
262   </xs:sequence>
263   <xs:attribute name="id" type="xs:ID" use="optional"/>
264   <xs:attribute name="title" type="xs:string" use="optional"/>
265 </xs:complexType>
266
267
```

### 268 3.3.2.1. The **Help** Element

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

274 **label** [Optional]  
275       Specifies a label relating to the help text.

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

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

283 The schema fragment for the **Help** element is:

```
284 <xs:element name="Help" type="HelpType"/>
285 <xs:complexType name="HelpType">
286   <xs:attribute name="label" type="xs:string" use="optional"/>
287   <xs:attribute name="link" type="xs:anyURI" use="optional"/>
288   <xs:attribute name="moreLink" type="xs:anyURI" use="optional"/>
289 </xs:complexType>
290
291
```

### 292 3.3.2.2. The **InquiryElementType**

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

295 **Help [Optional]**

296 See definition of the `Help` element above.

297 **Hint [Optional]**

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

301 **Label [Optional]**

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

304 **Value [Optional]**

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

310 **name [Required]**

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

315 The schema fragment for the `InquiryElementType` is:

```
316 <xs:complexType name="InquiryElementType" abstract="true">
317   <xs:sequence>
318     <xs:element ref="Help" minOccurs="0"/>
319     <xs:element ref="Hint" minOccurs="0"/>
320     <xs:element name="Label" type="xs:normalizedString" minOccurs="0"/>
321     <xs:element name="Value" type="xs:normalizedString" minOccurs="0"/>
322   </xs:sequence>
323   <xs:attribute name="name" type="xs:ID" use="required"/>
324 </xs:complexType>
325
326 <xs:element name="Hint" type="xs:string"/>
327
328
```

329 **3.3.2.3. `<InquiryElementType>` Subtypes**

330 The defined `<InquiryElementType>` subtypes are:

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

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

```

337     <xs:element name="Select" type="SelectType"/>
338     <xs:complexType name="SelectType">
339       <xs:complexContent>
340         <xs:extension base="InquiryElementType">
341           <xs:sequence>
342             <xs:element name="Item" minOccurs="2" maxOccurs="unbounded">
343               <xs:complexType>
344                 <xs:sequence>
345                   <xs:element ref="Hint" minOccurs="0"/>
346                 </xs:sequence>
347                 <xs:attribute name="label" type="xs:string" use="optional"/>
348                 <xs:attribute name="value" type="xs:NMTOKEN" use="required"/>
349               </xs:complexType>
350             </xs:element>
351           </xs:sequence>
352           <xs:attribute name="multiple" type="xs:boolean" use="optional" default="false"/>
353         </xs:extension>
354       </xs:complexContent>
355     </xs:complexType>
356
357
358
  
```

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

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

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

```

370     <xs:element name="Text" type="TextType"/>
371     <xs:complexType name="TextType">
372       <xs:complexContent>
373         <xs:extension base="InquiryElementType">
374           <xs:attribute name="minChars" type="xs:integer" use="optional"/>
375           <xs:attribute name="maxChars" type="xs:integer" use="optional"/>
376           <xs:attribute name="format" type="xs:string" use="optional"/>
377         </xs:extension>
378       </xs:complexContent>
379     </xs:complexType>
380
381
  
```

### 3.3.3. Example Request

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

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

### 3.3.4. Processing Rules

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

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

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

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

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

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

## 3.4. Interaction Response

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

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

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

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

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

### 448 **3.4.1. The `InteractionResponse` Element**

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

450 `Parameter` [Optional]

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

454 *OR:*

455 `InteractionStatement` [Optional]

456 Contains one or more signed `Inquiry` elements.

457 The `Parameter` element has two attributes:

458 `name` [Required]

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

460 `value` [Required]

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

463 The `<InteractionStatement>` consists of:

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

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

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

```
474 <xs:element name="InteractionResponse" type="InteractionResponseType"/>
475 <xs:complexType name="InteractionResponseType">
476   <xs:sequence>
477     <xs:element ref="lu:Status"/>
478     <xs:choice>
479       <xs:element name="InteractionStatement" type="InteractionStatementType"
480         minOccurs="0" maxOccurs="unbounded"/>
481       <xs:element name="Parameter" type="ParameterType" minOccurs="0"
482         maxOccurs="unbounded"/>
483     </xs:choice>
484   </xs:sequence>
485 </xs:complexType>
486 <xs:complexType name="InteractionStatementType">
487   <xs:sequence>
488     <xs:element ref="Inquiry" maxOccurs="unbounded"/>
489     <xs:element ref="ds:Signature"/>
490   </xs:sequence>
491 </xs:complexType>
492 <xs:complexType name="ParameterType">
493   <xs:attribute name="name" type="xs:ID" use="required"/>
494   <xs:attribute name="value" type="xs:string" use="required"/>
495 </xs:complexType>
496
497
```

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

```
499 <InteractionResponse>
500   <Status code="OK"/>
501   <Parameter name="addresschoice" value="always"/>
502 </InteractionResponse>
```

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

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

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

```
519     <InteractionResponse>  
520         <Status code="Cancel" />  
521     </InteractionResponse>  
522
```

### 523 **3.4.2. Processing Rules**

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

## 527 4. Security Considerations

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

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

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

## References

### Normative

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., eds. (June 1999). "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, The Internet Engineering Task Force <http://www.ietf.org/rfc/rfc2616.txt>

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

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

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

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

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

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

[SOAPv1.1] "Simple Object Access Protocol (SOAP) 1.1," Box, Don, Ehnebuske, David, Kakivaya, Gopal, Layman, Andrew, Mendelsohn, Noah, Nielsen, Henrik Frystyk, Winer, Dave, eds. World Wide Web Consortium W3C Note (08 May 2000). <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

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

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

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

### Informative

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

## A. Interaction Service XSD

584

```

585 <?xml version="1.0" encoding="UTF-8"?>
586 <xs:schema targetNamespace="urn:liberty:is:2006-08"
587     xmlns="urn:liberty:is:2006-08"
588     xmlns:is="urn:liberty:is:2006-08"
589     xmlns:lu="urn:liberty:util:2006-08"
590     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
591     xmlns:ds="http://www.w3.org/2000/09/xml:dsig#"
592     xmlns:xs="http://www.w3.org/2001/XMLSchema"
593     elementFormDefault="qualified"
594     attributeFormDefault="unqualified"
595     version="2.0">
596
597     <xs:import namespace="urn:liberty:util:2006-08"
598         schemaLocation="liberty-idwsf-utility-v2.0.xsd"/>
599
600     <xs:import namespace="http://schemas.xmlsoap.org/soap/envelope/"
601         schemaLocation="http://schemas.xmlsoap.org/soap/envelope/" />
602     <xs:import namespace="http://www.w3.org/2000/09/xml:dsig#"
603         schemaLocation="http://www.w3.org/TR/2002/REC-xml:dsig-core-20020212/xml:dsig-core-sch
604 ema.xsd" />
605
606     <xs:element name="InteractionRequest" type="InteractionRequestType" />
607     <xs:complexType name="InteractionRequestType">
608         <xs:sequence>
609             <xs:element ref="Inquiry" maxOccurs="unbounded" />
610             <xs:element ref="ds:KeyInfo" minOccurs="0" />
611         </xs:sequence>
612         <xs:attribute name="id" type="xs:ID" use="optional" />
613         <xs:attribute name="language" type="xs:NMTOKENS" use="optional" />
614         <xs:attribute name="maxInteractTime" type="xs:integer" use="optional" />
615         <xs:attribute name="signed" type="xs:token" use="optional" />
616     </xs:complexType>
617
618     <xs:element name="Inquiry" type="InquiryType" />
619     <xs:complexType name="InquiryType">
620         <xs:sequence>
621             <xs:element ref="Help" minOccurs="0" />
622             <xs:choice maxOccurs="unbounded">
623                 <xs:element ref="Select" minOccurs="0" maxOccurs="unbounded" />
624                 <xs:element name="Confirm" type="InquiryElementType"
625                     minOccurs="0" maxOccurs="unbounded" />
626                 <xs:element ref="Text" minOccurs="0" maxOccurs="unbounded" />
627             </xs:choice>
628         </xs:sequence>
629         <xs:attribute name="id" type="xs:ID" use="optional" />
630         <xs:attribute name="title" type="xs:string" use="optional" />
631     </xs:complexType>
632
633     <xs:element name="Help" type="HelpType" />
634     <xs:complexType name="HelpType">
635         <xs:attribute name="label" type="xs:string" use="optional" />
636         <xs:attribute name="link" type="xs:anyURI" use="optional" />
637         <xs:attribute name="moreLink" type="xs:anyURI" use="optional" />
638     </xs:complexType>
639
640     <xs:element name="Hint" type="xs:string" />
641
642     <xs:element name="Select" type="SelectType" />
643     <xs:complexType name="SelectType">
644         <xs:complexContent>
645             <xs:extension base="InquiryElementType">
646                 <xs:sequence>
647                     <xs:element name="Item" minOccurs="2" maxOccurs="unbounded">
648                         <xs:complexType>
649                             <xs:sequence>

```

```

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

```

## B. WSDL

708

```
709 <?xml version="1.0"?>
710 <definitions
711   name="id-wsf-is_2006-08_wsd_interface"
712   targetNamespace="urn:liberty:is:2006-08"
713   xmlns:tns="urn:liberty:is:2006-08"
714   xmlns="http://schemas.xmlsoap.org/wsdl/"
715   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
716   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
717   xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl"
718   xmlns:is="urn:liberty:is:2006-08"
719   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
720   xsi:schemaLocation="http://schemas.xmlsoap.org/wsdl/
721     http://schemas.xmlsoap.org/wsdl/
722     http://www.w3.org/2006/02/addressing/wsdl
723     http://www.w3.org/2006/02/addressing/wsdl/ws-addr-wsd.xsd">
724
725   <types>
726     <xsd:import namespace="urn:liberty:is:2006-08"
727       schemaLocation="liberty-idwsf-interaction-svc-v2.0.xsd"/>
728   </types>
729
730   <!-- Messages -->
731
732   <message name="InteractionRequest">
733     <part name="body" type="is:InteractionRequest"/>
734   </message>
735
736   <message name="InteractionResponse">
737     <part name="body" type="is:InteractionResponse"/>
738   </message>
739
740   <!-- Port Type -->
741
742   <portType name="ISPort">
743     <operation name="ISInteraction">
744       <input message="tns:InteractionRequest"
745         wsaw:Action="urn:liberty:is:2006-08:InteractionRequest"/>
746       <output message="tns:InteractionResponse"
747         wsaw:Action="urn:liberty:is:2006-08:InteractionResponse"/>
748     </operation>
749   </portType>
750
751   <!--
752   An example of a binding and service that can be used with this
753   abstract service description is provided below.
754   -->
755
756   <binding name="ISBinding" type="tns:ISPort">
757
758     <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
759
760     <operation name="Interaction">
761       <soap:operation soapAction="urn:liberty:is:2006-08:Interaction"/>
762       <input> <soap:body use="literal"/> </input>
763       <output> <soap:body use="literal"/> </output>
764     </operation>
765   </binding>
766
767   <service name="InteractionService">
768     <port name="ISPort" binding="tns:ISBinding">
769
770       <!-- Modify with the REAL SOAP endpoint -->
771
772       <soap:address location="http://example.com/id-wsf/is"/>
773     </port>
```

```
774     </service>  
775  
776 </definitions>  
777
```

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

```

778
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
819 </xsl:element>
820 <xsl:apply-templates select="is:Confirm"/><br/>
821 <xsl:apply-templates select="is:Select"/><br/>
822 <br/>
823 <input type="submit" value="Submit"/>
824 </xsl:element>
825 <p>
826 <xsl:apply-templates select="is:Help"/>
827 </p>
828 </body>
829 </html>
830 </xsl:template>
831
832 <xsl:template match="is:Confirm">
833 <xsl:value-of select="is:Label"/>
834 <xsl:element name="label">
835 <xsl:attribute name="for">isid-<xsl:value-of select="@name"/>-yes</xsl:attribute>
836 Yes
837 </xsl:element>
838 <xsl:element name="input">
839 <xsl:attribute name="type">radio</xsl:attribute>
840 <xsl:attribute name="checked"></xsl:attribute>
841 <xsl:attribute name="name">is-confirm-yes-<xsl:value-of select="@name"/></xsl:attribute>
842 <xsl:attribute name="id">isid-<xsl:value-of select="@name"/>-yes</xsl:attribute>
843 </xsl:element>
    
```

```
844     <xsl:element name="label">
845         <xsl:attribute name="for">isid-<xsl:value-of select="@name"/>-no</xsl:attribute>
846         No
847     </xsl:element>
848     <xsl:element name="input">
849         <xsl:attribute name="type">radio</xsl:attribute>
850         <xsl:attribute name="name">is-confirm-no-<xsl:value-of select="@name"/></xsl:attribute>
851         <xsl:attribute name="id">isid-<xsl:value-of select="@name"/>-no</xsl:attribute>
852     </xsl:element>
853 </xsl:template>
854
855 <xsl:template match="is:Select">
856     <xsl:element name="label">
857         <xsl:value-of select="is:Label"/>
858         <xsl:attribute name="for">isid-<xsl:value-of select="@name"/></xsl:attribute>
859     </xsl:element>
860     <xsl:element name="select">
861         <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
862         <xsl:attribute name="id">isid-<xsl:value-of select="@name"/></xsl:attribute>
863         <xsl:apply-templates select="is:Item"/>
864     </xsl:element>
865 </xsl:template>
866
867 <xsl:template match="is:Item">
868     <xsl:element name="option">
869         <xsl:attribute name="label"><xsl:value-of select="@label"/></xsl:attribute>
870         <xsl:attribute name="value"><xsl:value-of select="@value"/></xsl:attribute>
871         <xsl:value-of select="@label"/>
872         <xsl:apply-templates select="is:Hint"/>
873     </xsl:element>
874 </xsl:template>
875 <xsl:template match="is:Hint">
876     <xsl:attribute name="onmouseover">showHint(<xsl:value-of select="."/>)</xsl:attribute>
877 </xsl:template>
878 <xsl:template match="is:Help">
879     <p id="help"><b>Help</b><br/>
880         <xsl:value-of select="."/>
881         <xsl:element name="a">
882             <xsl:attribute name="href">
883                 <xsl:value-of select="@morelink"/>
884             </xsl:attribute>
885             More information
886         </xsl:element>
887     </p>
888 </xsl:template>
889 </xsl:stylesheet>
890
```

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

```

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

```

```
957     </xsl:element>Yes</anchor><br/>
958 <anchor>
959   <xsl:element name="go">
960     <xsl:attribute name="href">
961       submit;jsessionid=<xsl:value-of select="$jsessionid" />
962     </xsl:attribute>
963     <xsl:attribute name="method">get</xsl:attribute>
964     <xsl:element name="postfield">
965       <xsl:attribute name="name">msg</xsl:attribute>
966       <xsl:attribute name="value"><xsl:value-of select="$messageID" /></xsl:attribute>
967     </xsl:element>
968     <xsl:element name="postfield">
969       <xsl:attribute name="name"><xsl:value-of select="@name" /></xsl:attribute>
970       <xsl:attribute name="value">0</xsl:attribute>
971     </xsl:element>
972   </xsl:element>No</anchor><br/>
973 </p>
974 </xsl:template>
975
976 </xsl:stylesheet>
977
```