



Liberty ID-WSF SOAP Binding Specification

Version: 1.2

Editors:

Jeff Hodges, Sun Microsystems, Inc.
John Kemp, Nokia Corporation
Robert Aarts, Nokia Corporation

Contributors:

Conor Cahill, AOL Time Warner, Inc.
Marc Hadley, Sun Microsystems, Inc.
Jukka Kainulainen, Nokia Corporation
Jonathan Sergent, Sun Microsystems, Inc.
Greg Whitehead, Trustgenix, Inc.

Abstract:

This specification defines a SOAP binding for the Liberty Identity Web Services Framework (ID-WSF) and the Liberty Identity Services Interface Specifications (ID-SIS). It specifies simple message correlation, as well as provider declaration, processing context, consent claims, usage directives and a number of other optional headers.

Filename: liberty-idwsf-soap-binding-v1.2.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.** Implementors
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 © 2004-2005 ADAE; Adobe Systems; America Online, Inc.; American Express Company; Avatier
16 Corporation; Axalto; Bank of America Corporation; BIPAC; Computer Associates International, Inc.; DataPower
17 Technology, Inc.; Diversinet Corp.; Enosis Group LLC; Entrust, Inc.; Epok, Inc.; Ericsson; Fidelity Investments;
18 Forum Systems, Inc. ; France Telecom; Gamefederation; Gemplus; General Motors; Giesecke & Devrient GmbH;
19 Hewlett-Packard Company; IBM Corporation; Intel Corporation; Intuit Inc.; Kantega; Kayak Interactive; MasterCard
20 International; Mobile Telephone Networks (Pty) Ltd; NEC Corporation; Netegrity, Inc.; NeuStar, Inc.; Nippon
21 Telegraph and Telephone Corporation; Nokia Corporation; Novell, Inc.; NTT DoCoMo, Inc.; OpenNetwork; Oracle
22 Corporation; Ping Identity Corporation; Royal Mail Group plc; RSA Security Inc.; SAP AG; Senforce; Sharp
23 Laboratories of America; Sigaba; SmartTrust; Sony Corporation; Sun Microsystems, Inc.; Telefonica Moviles, S.A.;
24 Trusted Network Technologies.; Trustgenix; UTI; VeriSign, Inc.; Vodafone Group Plc. All rights reserved.

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

31 **Contents**

32 1. Introduction 4
33 2. Notation and Conventions 6
34 3. Schema Particulars 10
35 4. SOAP Binding 14
36 5. Messaging-specific Header Blocks 18
37 6. Optional Header Blocks 29
38 7. Security Considerations 48
39 8. Acknowledgements 49
40 Bibliography
41 A. Liberty ID-WSF SOAP Binding Schema 52
42 B. Liberty ID-WSF SOAP Binding Extension Schema (April 2004) 54
43 C. Liberty ID-WSF Utility Schema Listing 56
44 D. SOAP Envelope Schema Listing 58

45 1. Introduction

46 Liberty Identity Web Services Framework (ID-WSF) and Service Interface Specification (ID-SIS) messages [LibertyIDWSFOverview], collectively referred to as *ID-* messages* in this specification, are designed so that they may be
47 mapped onto various transport or transfer protocols. Thus, they are designed to be conveyed in the data portion of the
48 underlying protocol's messages. ID-* messages do not intrinsically address specific aspects of message exchange
49 such as: to which system entity the message is to be sent, message correlation, the mechanics of message exchange,
50 or security context.
51

52 Examples of ID-* messages include the <DiscoveryLookupRequest> message of [LibertyDisco], and the
53 <Modify> message of [LibertyIDPP].

54 This specification defines a mapping of ID-* messages onto SOAP [SOAPv1.1], an XML-based [XML] messaging
55 protocol.

56 SOAP itself does not define the specific message exchange aspects mentioned above, but offers an *extensibility model*
57 that may be used to define message components that do address such message exchange specifics. SOAP extensibility
58 is effected by adding message components to the portion of the SOAP message called the *Header*. These message
59 components are referred to as *SOAP header blocks* [SOAPv1.2].

60 This specification defines a number of SOAP header blocks, addressing specific aspects of ID-* message exchange
61 functionality. They are:

62 • Message Correlation:

63 SOAP does not define a mechanism to correlate one SOAP message with another message, such as in a request-
64 response paradigm. This specification defines the <Correlation> header block for this purpose.

65 • Provider and affiliation declaration:

66 Participants in ID-* interactions may declare themselves by their Provider ID, as well as their Affiliation ID if
67 appropriate. This specification defines the <Provider> header block for this purpose.

68 • Processing context:

69 An ID-* requester may need to express additional context for a given request, for example indicating that the
70 requester expects to make such requests in the future when the Principal may or may not be online. This
71 specification defines the <ProcessingContext> header block for this purpose.

72 • Consent Claims:

73 ID-WSF-based entities may wish to claim whether they obtained the Principal's consent for carrying out any
74 given operation, such as updating a Principal's Personal Profile entry [LibertyIDPP]. This specification defines the
75 <Consent> header block for this purpose.

76 • Usage Directives:

77 ID-WSF-based entities may wish to indicate their policies for handling data at the time of data request, and entities
78 releasing data may wish to specify their policies for the subsequent use of data at the time of data release. This
79 specification defines the <UsageDirective> header block for this purpose.

80 • Timeout:

81 The <Timeout> header block is defined in this specification to allow the receiver of an ID-* message to indicate
82 that processing of the received message failed due to a timeout condition.

83 • Credentials Context:

84 The receiver of an ID-* message might indicate that credentials supplied in the request did not meet its policy in
85 allowing access to the requested resource. The <CredentialsContext> header block allows such policies to be
86 expressed to the requester.

- 87 • Service Instance Update:
88 The <ServiceInstanceUpdate> header block allows a service to indicate that requesters should contact it on a
89 different endpoint or use a different security mechanism and credentials to access the requested resource.
- 90 Additionally, this specification defines how ID-* messages are bound into SOAP message bodies, and how the SOAP
91 header blocks implementing the above functionalities are bound into SOAP message headers.
- 92 Note that other specifications in the ID-WSF specification suite also define SOAP header blocks, for example
93 [\[LibertySecMech\]](#) and [\[LibertyInteract\]](#), which may be used concurrently with the header blocks defined in this
94 specification. Header blocks specified in specifications outside of the ID-WSF specification suite may also be
95 composed with ID-WSF header blocks. An example is the <wsse:Security> header block as discussed in
96 [\[LibertySecMech\]](#). However no further mention of doing such is made in this specification.

97 2. Notation and Conventions

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

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

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

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

107 2.1. XML Namespaces

108 This specification makes normative use of the XML namespace prefixes noted in Table 1.

109 **Table 1. XML Namespaces and Prefixes**

Prefix	Namespace
sb:	Represents the Liberty SOAP Binding namespace: <code>urn:liberty:sb:2003-08</code> Note: This is the point of definition of this namespace. This namespace is the default for instance fragments, type names, and element names in this document when a namespace is not explicitly noted.
sb-ext:	Represents a Liberty SOAP Binding Extensions namespace: <code>urn:liberty:sb:2004-04</code>
idpp:	Represents the namespace defined in [LibertyIDPP].
is:	Represents the namespace defined in [LibertyInteract].
S:	Represents the SOAP namespace: <code>http://schemas.xmlsoap.org/soap/envelope</code> This namespace is defined in [SOAPv1.1].
wsse:	Represents the SOAP Message Security namespace: <code>http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-secext-1.0.xsd</code> This namespace is defined in [LibertySecMech].
xs:	Represents the W3C XML schema namespace: <code>http://www.w3.org/2001/XMLSchema</code> This namespace is defined in [Schema1].

110 2.2. Terminology

111 This section defines key terminology used in this specification. Definitions for other Liberty-specific terms can be
112 found in [LibertyGlossary]. See also [RFC2828] for overall definitions of security-related terms.

113 affiliation

114 An *affiliation* is a set of one or more entities, described by *Provider IDs*, who may perform Liberty interactions
115 as a member of the set. An affiliation is referenced by exactly one *Affiliation ID*, and is administered by exactly
116 one entity identified by their Provider ID. Members of an affiliation may invoke services either as a member of
117 the affiliation—by virtue of their Affiliation ID, or individually by virtue of their Provider ID [LibertyGlossary].

-
- 118 Affiliation ID
- 119 An *Affiliation ID* identifies an *affiliation*. It is schematically represented by the `affiliationID` attribute of the
- 120 `<AffiliationDescriptor>` metadata element [[LibertyMetadata](#)].
- 121 client
- 122 A *role* assumed by a *system entity* who makes a request of another system entity, often termed a *server* [[RFC2828](#)],
- 123 i.e. a client is also a *sender*.
- 124 ID-*
- 125 A shorthand designator referring to the Liberty ID-WSF, ID-FF, and ID-SIS specification sets. For example, one
- 126 might say that the former specification sets are all part of the Liberty ID-* specification suite.
- 127 ID-* header block
- 128 One of the header blocks defined in this specification, or defined in any of the other Liberty ID-* specification
- 129 suite.
- 130 ID-* message
- 131 Equivalent to *ordinary ID-* message*.
- 132 ID-* fault message
- 133 An *ID-* fault message* is a SOAP `<S:Fault>` element containing a `<Status>` element, with the attributes and
- 134 attribute values of both elements configured as specified herein, or as specified in other specification(s) in the
- 135 ID-WSF or ID-SIS specification sets.
- 136 ID-SIS
- 137 Liberty Identity Service Interface specification set.
- 138 ID-WSF
- 139 Liberty Identity Web Services Framework specification set.
- 140 MEP
- 141 see Message Exchange Pattern.
- 142 Message Exchange Pattern
- 143 A [[SOAPv1.2](#)] term for the overall notion of various patterns of message exchange between SOAP nodes. For
- 144 example, request-reply and one-way are two *MEPs* used in this specification.
- 145 message thread
- 146 A *message thread* is a synchronous exchange of messages in a request-response *MEP* between two *SOAP nodes*.
- 147 All the messages of a given message thread are "linked" via each message's `<Correlation>` header block
- 148 `refToMessageID` attribute value being set, by the sender, from the previous successfully received message's
- 149 `<Correlation>` header block `messageID` attribute value.
- 150 ordinary ID-* message
- 151 An *ordinary ID-* message* is a Liberty Identity Web Services Framework (ID-WSF) or Service Interface
- 152 Specification (ID-SIS) message as defined in the [[LibertyDST](#)], [[LibertyDisco](#)], and [[LibertyIDPP](#)] specifications
- 153 and others. It has the characteristics of being designed to be conveyed by essentially any transport or transfer
- 154 protocol, notably SOAP [[SOAPv1.1](#)]. It is also known among the ID-* specifications as a *service request*, or an
- 155 ID-WSF (service) request, or an ID-SIS (service) request.
- 156 processing context
- 157 A *processing context* is the collection of specific circumstances under which a particular processing step or set of
- 158 steps take place.
- 159 processing context facet
- 160 A *processing context facet* is an identified aspect, inherent or additive, of a *processing context*.

- 161 provider
162 A *provider* is a Liberty-enabled entity that performs one or more of the provider roles in the Liberty architecture,
163 for example Service Provider or Identity Provider. See also *Liberty-enabled Provider* in [\[LibertyGlossary\]](#).
164 Providers are identified in Liberty protocol interactions by their *Provider IDs* or optionally their *Affiliation ID*
165 if they are a member of an affiliation(s) and are acting in that capacity.
- 166 Provider ID
167 A *Provider ID* identifies an entity known as a *provider*. It is schematically represented by the `providerID`
168 attribute of the `<EntityDescriptor>` metadata element [\[LibertyMetadata\]](#).
- 169 receiver
170 A *role* taken by a *system entity* when it receives a message sent by another system entity. See also *SOAP receiver*
171 in [\[SOAPv1.2\]](#).
- 172 role
173 A function or part performed, especially in a particular operation or process [\[Merriam-Webster\]](#).
- 174 sender
175 A *role* donned by a *system entity* when it constructs and sends a message to another system entity. See also *SOAP*
176 *sender* in [\[SOAPv1.2\]](#).
- 177 server
178 A *role* performed by a *system entity* that provides a service in response to requests from other system entities
179 called *clients* [\[RFC2828\]](#). Note that in order to provide a service to clients, a server will often be both a *sender*
180 and a *receiver*.
- 181 service request
182 A *service request* is another term for an *ordinary ID-* message*. Service request is also loosely equivalent to a
183 "SOAP-bound (ordinary) ID-* message".
- 184 SOAP-bound ID-* message
185 A *SOAP message* conveying ID-WSF and perhaps ID-SIS header blocks and conveying either an *ordinary ID-**
186 *message* or an *ID-* fault message*. After being bound to SOAP, the resultant composite messages are referred to
187 as an *Ordinary SOAP-bound ID-* Message* and a *SOAP-bound ID-* Fault Message*, respectively.
- 188 SOAP header block
189 A [\[SOAPv1.2\]](#) term whose definition is: An [element] used to delimit data that logically constitutes a single
190 computational unit within the SOAP header. In [\[SOAPv1.1\]](#) these are known as simply *SOAP headers*, or simply
191 *headers*. This specification uses the SOAPv1.2 terminology.
- 192 SOAP message
193 In this specification, the term *SOAP message* refers to a message consisting of only a `<S:Envelope>` element as
194 defined in [\[SOAPv1.1\]](#). It contains two top-level subelements: `<S:Header>` and `<S:Body>`. This message is in
195 turn mapped onto a lower-layer transport or transfer protocol, typically HTTP [\[RFC2616\]](#).
- 196 SOAP node
197 A [\[SOAPv1.2\]](#) term describing *system entities* who are parties to SOAP-based message exchanges that are, for
198 purposes of this specification, also the ultimate destination of the exchanged messages, i.e. *SOAP endpoints*. In
199 [\[SOAPv1.1\]](#), SOAP nodes are referred to as *SOAP endpoints*, or simply *endpoints*. This specification uses the
200 SOAPv1.2 terminology.
- 201 system entity
202 An active element of a computer/network system. For example, an automated process or set of processes, a
203 subsystem, a person or group of persons that incorporates a distinct set of functionality [\[SAMLGloss\]](#).

204 **2.3. Treatment of Boolean Values**

205 For readability, when an XML Schema type is specified to be `xsd:boolean`, this document discusses the values as
206 `TRUE` and `FALSE` rather than "1" and "0", which will exist in a document instance conforming to the SOAP Envelope
207 1.1 schema [\[SOAPv1.1-Schema\]](#).

208 **2.4. String and URI Values**

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

214 **Note:**

215 The ID types described in [Section 3.2: "ID" Types and Message Identifiers](#) are `string`-based.

216 Also, various element and/or attribute components of the schema described by this specification (see
217 [Appendix A: SOAP Binding Schema](#) , below) may have further requirements placed on the values they
218 may take on. For example, see [Section 5.1.2: messageID Value Requirements](#) .

219 **2.5. Time Values**

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

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

224 3. Schema Particulars

225 This section addresses schema particulars such as which schemas this specification defines, describes, and depends
226 upon, as well as various underlying schema types.

227 3.1. Schema Declarations

228 This specification normatively defines and describes an XML schema which is constituted in the XML Schema
229 [Schema1] files ("Liberty ID-WSF SOAP Binding Schema v1.0, v1.1", reproduced in [Appendix A](#), [Appendix B](#)). In
230 addition, the Liberty ID-WSF SOAP Binding Schema files explicitly include, in the XML Schema sense, the Liberty
231 ID-WSF utility schema file (reproduced in [Appendix C](#)).

232 Also, the Liberty ID-WSF SOAP Binding Schema files explicitly depend upon the SOAPv1.1 schema [[SOAPv1.1-](#)
233 [Schema](#)] (reproduced in [Appendix D](#)).

234 3.2. "ID" Types and Message Identifiers

235 The IDType simple type is used in this specification to declare *message identifiers*. The IDReferenceType is used
236 to reference message identifiers.

237 The schema fragment in [Figure 1](#), from the ID-WSF Utility schema ([Appendix C](#)), defines the IDType and
238 IDReferenceType simple types.

```
239 <xs:simpleType name="IDType">  
240 <xs:annotation>  
241 <xs:documentation>This type should be used to provided IDs to components that have IDs that  
242 may not be scoped within the local xml instance document. </xs:documentation>  
243 </xs:annotation>  
244 <xs:restriction base="xs:string"/>  
245 </xs:simpleType>  
246 <xs:simpleType name="IDReferenceType">  
247 <xs:annotation>  
248 <xs:documentation> This type can be used when referring to elements that are  
249 identified using an IDType </xs:documentation>  
250 </xs:annotation>  
251 <xs:restriction base="xs:string"/>  
252 </xs:simpleType>  
253  
254
```

255 **Figure 1. IDType and IDReferenceType Schema**

256 3.3. Status Types

257 The <Status> element, of type StatusType complex type, is used in this specification to convey status codes and
258 related information. The schema fragment in [Figure 2](#), from the ID-WSF Utility schema ([Appendix C](#)), shows both
259 the <Status> element and StatusType complex type.

```
260 <xs:element name="Status" type="StatusType">
261   <xs:annotation>
262     <xs:documentation> A standard Status type</xs:documentation>
263   </xs:annotation>
264 </xs:element>
265
266 <xs:complexType name="StatusType">
267   <xs:annotation>
268     <xs:documentation> A type that may be used for status codes. </xs:documentation>
269   </xs:annotation>
270   <xs:sequence>
271     <xs:element ref="Status" minOccurs="0" maxOccurs="unbounded"/>
272   </xs:sequence>
273   <xs:attribute name="code" type="xs:QName" use="required"/>
274   <xs:attribute name="ref" type="IDReferenceType" use="optional"/>
275   <xs:attribute name="comment" type="xs:string" use="optional"/>
276 </xs:complexType>
277
278
```

279 **Figure 2. Status and StatusType Schema**

280 3.3.1. Status Codes

281 This section lists, in [Table 2](#), the values defined in this specification for the code attribute of the <Status> element.

282

Table 2. Status Codes

Code	Semantics	Suggested Fault Source
sb:InvalidActor	There is an issue with the <code>actor</code> attribute on the indicated header block in the indicated message.	S:Client
sb:InvalidMustUnderstand	There is an issue with the <code>mustUnderstand</code> attribute on the indicated header block in the indicated message.	S:Client
sb:StaleMsg	The indicated inbound SOAP-bound ID-* message has a correlation timestamp value outside of the receivers allowable time window.	S:Client
sb:DuplicateMsg	The indicated inbound SOAP-bound ID-* message appears to be a duplicate.	S:Client
sb:InvalidRefToMsgID	The indicated inbound SOAP-bound ID-* message appears to incorrectly refer to the preceding message in the message thread.	S:Client
sb:ProviderIDNotValid	The receiver does not consider the claimed Provider ID to be valid.	S:Client
sb:AffiliationIDNotValid	The receiver does not consider the claimed Affiliation ID to be valid.	S:Server
sb:IDStarMsgNotUnderstood	There was a problem with understanding/parsing the conveyed ID-* message.	S:Client
sb:ProcCtxURINotUnderstood	The receiver did not understand the processing context facet URI.	S:Server
sb:ProcCtxUnwilling	The receiver is unwilling to apply the sender's stipulated processing context.	S:Server
sb:CannotHonourUsageDirective	The receiver is unable or unwilling to honor the stipulated usage directive.	S:Server
sb-ext:EndpointMoved	The request cannot be processed at this endpoint. This is typically used in conjunction with the <code><ServiceInstanceUpdate></code> header block to indicate the endpoint to which the request should be re-submitted.	S:Server
sb-ext:InappropriateCredentials	The sender has submitted a request that does not meet the needs of the receiver. The receiver may indicate credentials that are acceptable to them via a <code><CredentialsContext></code> or <code><ServiceInstanceUpdate></code> header block.	S:Client
sb-ext:ProcessingTimeout	The sender is indicating that processing of the request has failed due to the processing taking longer than the <code>maxProcessingTime</code> specified on the request <code><Timeout></code> header block.	S:Server

283 3.4. SOAP Fault Types

284 The SOAPv1.1 `Fault` and `Detail` complex types are used in this specification to convey processing exceptions.

285 The schema fragment in [Figure 3](#), extracted from [\[SOAPv1.1-Schema\]](#), defines the SOAPv1.1 `Fault` and `detail`
286 complex types, which define the `<S:Fault>` and `<detail>` elements, respectively.

287 **Note:**

288 The `<S:Fault>` element is **not** intended to be used as a SOAP header block. Rather, it is designed to be
289 conveyed in the `<S:Body>` of a SOAP message.

```
290 <xs:element name="Fault" type="tns:Fault"/>
291
292 <xs:complexType name="Fault" final="extension">
293   <xs:annotation>
294     <xs:documentation>
295       Fault reporting structure
296     </xs:documentation>
297   </xs:annotation>
298   <xs:sequence>
299     <xs:element name="faultcode" type="xs:QName"/>
300     <xs:element name="faultstring" type="xs:string"/>
301     <xs:element name="faultactor" type="xs:anyURI" minOccurs="0"/>
302     <xs:element name="detail" type="tns:detail" minOccurs="0"/>
303   </xs:sequence>
304 </xs:complexType>
305
306 <xs:complexType name="detail">
307   <xs:sequence>
308     <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
309   </xs:sequence>
310   <xs:anyAttribute namespace="##any" processContents="lax"/>
311 </xs:complexType>
312
```

313

Figure 3. SOAP Fault and detail Types Schema

314 4. SOAP Binding

315 This section defines the notion of *ID-* messages* and the overall, high-level considerations with respect to *binding*
316 them into *SOAP messages* for subsequent conveyance. The detailed processing rules are then given in
317 [Section 5.3: Messaging Processing Rules](#) .

318 4.1. SOAP Version

319 This specification normatively depends upon SOAP version 1.1, as specified in [\[SOAPv1.1\]](#) . Messages conformant
320 to this specification MUST also be conformant to [\[SOAPv1.1\]](#).

321 4.2. The SOAPAction HTTP Header

322 [\[SOAPv1.1\]](#) defines the SOAPAction HTTP header, and requires its usage on HTTP-bound SOAP messages. This
323 header may be used to indicate the "intent" of a SOAP message to the recipient. When a SOAP message conformant
324 to this specification is emitted bound to HTTP, according to the binding specified in [\[SOAPv1.1\]](#), the SOAPAction
325 HTTP header SHOULD be set to "" (an empty string).

326 **Note:**

327 It should be noted that [\[WSDLv1.1\]](#) documents may be defined that specify the value of the SOAPAction
328 header to be included on messages sent to the service defined in WSDL. A tool that generates message-
329 producing code might thus auto-generate code that adds a SOAPAction header of a different value than that
330 specified above.

331 4.3. Ordinary ID-* Messages

332 *Ordinary ID-* Messages* are messages of the forms defined in Liberty ID-WSF and ID-SIS specification sets (see
333 [Section 2.2: Terminology](#), above). For example, these are the class of messages defined in [\[LibertyDST\]](#), or
334 the Discovery Service specification [\[LibertyDisco\]](#), or in the ID-PersonalProfile specification [\[LibertyIDPP\]](#). See
335 [Example 1](#). These messages must be mapped onto an underlying transport or transfer protocol in order for them
336 to be communicated between system entities.

```
337  
338  
339     <idpp:Query>  
340     :  
341     <!-- various message-specific subelements may go here -->  
342     :  
343     </idpp:Query>  
344  
345
```

346 **Example 1. A Specific ID-* Message: The <idpp:Query> Message**

347 4.4. ID-* Fault Messages

348 An *ID-* Fault Message* consists of a SOAP <S:Fault> element (see [Section 3.4: SOAP Fault Types](#)) containing
349 a <Status> element, with the attributes and attribute values of both elements configured as specified herein, or as
350 specified in other specification(s) in the ID-WSF or ID-SIS specification sets.

351 The <S:Fault> element's attributes and child elements MUST be tailored according to these rules:

352 1. The <S:Fault> element:

- 353 A. SHOULD contain a `<faultcode>` element whose value SHOULD be either "S:server" or "S:client".
354 **Note:**
355 A `<faultcode>` of "S:server" indicates that the receiver believes that it has erred, whereas "S:client"
356 is intended for cases in which the receiver believes that the sender has erred. Such an indication should be
357 considered merely informational.
- 358 B. SHOULD contain a `<faultstring>` element. This string value MAY be localized.
- 359 C. SHOULD NOT contain a `<S:faultactor>` element.
- 360 2. The `<S:Fault>` element's `<detail>` child element MUST contain a `<Status>` element (see [Section 3.3:](#)
361 `Status Types`). The `<Status>` element:
- 362 A. MUST contain a `code` attribute set to the value as specified when the issuance of a ID-* Fault message is
363 indicated.
- 364 B. MAY contain a `ref` attribute set to the value as specified in this specification when the issuance of a ID-*
365 Fault message is indicated.
- 366 C. MAY contain a `comment` attribute set to the value as specified in this specification when the issuance of a
367 ID-* Fault message is indicated. This string value MAY be localized.
- 368 3. The `<S:Fault>` element's `<detail>` child element SHOULD also contain an indicator of which header block
369 or ID-* message body element, in the message being processed, is being referred to by the fault. The indicator
370 is constructed by including in the `<detail>` element, after the `<Status>` element, an element of the same type
371 as the one referred to by the fault. This element SHOULD have only one attribute, the `id` attribute whose value
372 MUST be the value of the `id` attribute of the element referred to by the fault.

373 4.5. SOAP-bound ID-* Messages

374 ID-* messages are bound into SOAP messages, yielding *SOAP-bound ID-* messages*. This binding thus provides a
375 concrete means for ID-* message conveyance since [\[SOAPv1.1\]](#) specifies a binding to HTTP [\[RFC2616\]](#), which is
376 itself layered onto the ubiquitous [\[TLS/SSL\]/TCP/IP](#) protocol stack.

377 Although this binding is the only one given in this specification, other protocols could be used to convey ID-*
378 messages, with appropriateness depending on the protocol selected and the target operational context. This is not
379 discussed further in this specification.

380 **A SOAP-bound ID-* message is defined as:**

- 381 • having a `<Correlation>` header block, and possibly a `<Provider>` header block in its `<S:Header>` element,
382 and,
- 383 • perhaps having other Liberty ID-WSF or ID-SIS header blocks in its `<S:Header>` element, and,
- 384 • containing either an ordinary ID-* message, or an ID-* fault message, in its `<S:Body>` element. The former is
385 known as an *ordinary SOAP-bound ID-* message* (see [Example 2](#)), and the latter is known as a *SOAP-bound ID-**
386 *fault message* (see [Example 3](#)).

387 [Section 5.3: Messaging Processing Rules](#) specifies the detailed normative processing rules for constructing, sending,
388 and receiving SOAP-bound ID-* messages.

```
389
390
391 <S:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
392   xmlns:idpp="urn:liberty:id-sis-pp:2003-08">
393
394   <S:Header>
395
396     <!-- other header blocks, eg wsse:security, may go here -->
397
398     <Correlation S:mustUnderstand="1"
399       id="A13454...245"
400       actor="http://schemas.../next"
401       messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbbb"
402       timestamp="2112-03-15T11:12:12Z"/>
403
404     <Provider providerID="example.com"
405       affiliationID="affiliation.example.com"
406       S:mustUnderstand="1"
407       id="A9kendan...542"
408       actor="http://schemas.../next"/>
409
410     <!-- other header blocks, eg wsse:security, may go here -->
411
412   </S:Header>
413
414   <S:Body>
415
416     <idpp:Query> <!-- This is an ID-PP "Query" message bound -->
417       :         <!-- into the <S:Body> of a SOAP message.   -->
418       :
419     </idpp:Query>
420
421   </S:Body>
422
423 </S:Envelope>
424
```


425

Example 2. An Ordinary SOAP-bound ID-* Message

426

427

428

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"  
  xmlns:sb="urn:liberty:sb:2003-08"  
  xmlns:pp="urn:liberty:id-sis-pp:2003-08">
```

431

```
<S:Header>
```

433

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

435

```
<Correlation S:mustUnderstand="1"
```

437

```
  id="B6432...466"
```

438

```
  actor="http://schemas.../next"
```

439

```
  messageID="uuid:efefefef-aaaa-ffff-cc cc-eeeeffffbbbb"
```

440

```
  timestamp="2112-03-15T11:12:12Z"/>
```

441

```
<Provider providerID="example.com"
```

443

```
  affiliationID="affiliation.example.com"
```

444

```
  S:mustUnderstand="1"
```

445

```
  id="A9kendan...542"
```

446

```
  actor="http://schemas.../next"/>
```

447

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

449

```
</S:Header>
```

451

```
<S:Body>
```

453

```
<S:Fault>
```

455

```
  <faultcode>S:server</faultcode>
```

456

```
  <faultstring>Server Error</faultstring>
```

457

```
  <!-- <S:faultactor> should be absent -->
```

458

```
  <detail namespace="urn:liberty:sb:2003-08" >
```

459

```
    <Status code="sb:"
```

461

```
      ref="Foo"
```

462

```
      comment="Bar" />
```

463

```
    <Correlation id="A13454...245"/>
```

464

```
  </detail>
```

465

```
</S:Fault>
```

466

```
</S:Body>
```

468

```
</S:Envelope>
```

470

471

472

Example 3. A SOAP-bound ID-* Fault Message

473 5. Messaging-specific Header Blocks

474 The header blocks defined in this section implement the ID-* message exchange model. The first, the <Correlation>
475 header block, addresses message correlation. The second, the <Provider> header block, addresses providerID and
476 affiliationID declaration.

477 The messaging processing rules associated with these two header blocks are given in
478 [Section 5.3: Messaging Processing Rules](#).

479 Additional ID-* header blocks and their processing rules are defined below in [Section 6: Optional Header Blocks](#).

480 **Note:**

481 Other ID-* specifications MAY define additional ID-* header blocks. [\[LibertyInteract\]](#) defines a header
482 block, for example.

483 5.1. Message Correlation: The <Correlation> Header Block

484 This section defines the SOAP-bound ID-* message correlation facilities.

485 5.1.1. The correlationType Header Block Type

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

490 The correlationType header block type defines the following attributes:

491 • messageID [Required] – The unique ID of the message.

492 **Note:**

493 The messageID is intended to act as a *nonce*. Thus it is subject to specific processing rules defined below in
494 Messaging Processing Rules, and the value-specific guidance given below in [Section 5.1.2: messageID Value](#)
495 Requirements.

496 • refToMessageID [Optional] – A copy of the messageID attribute value of the message being responded to, or
497 being correlated with in some application-specific fashion, if any.

498 • timestamp [Required] – The time the sender sent the message. Note guidance on time values given in [Time](#)
499 Values above.

500 • id [Optional] – identifies a <Correlation> header block instance. This attribute MUST be used when the
501 message is signed as described in [\[LibertySecMech\]](#), and the element instance is to be included as one of the set
502 of signed message components.

503 • S:mustUnderstand [Optional] – The SOAP mustUnderstand attribute [\[SOAPv1.1\]](#).

504 • S:actor [Optional] – The SOAP actor attribute [\[SOAPv1.1\]](#).

505 The schema fragment in [Figure 4](#) defines the `Correlation` header block type.

```
506
507
508     <xs:complexType name="correlationType">
509         <xs:attribute name="messageID" type="IDType" use="required"/>
510         <xs:attribute name="refToMessageID" type="IDType" use="optional"/>
511         <xs:attribute name="timestamp" type="xs:dateTime" use="required"/>
512         <xs:attribute name="id" type="xs:ID" use="optional"/>
513         <xs:attribute ref="S:mustUnderstand" use="optional"/>
514         <xs:attribute ref="S:actor" use="optional"/>
515     </xs:complexType>
516
517
```

518 **Figure 4. The `Correlation` Header Block Type Schema**

519 **5.1.2. messageID Value Requirements**

520 Values of the `messageID` attribute of objects of type `correlationType` MUST satisfy the following property:

521 Any party that assigns a value to a `messageID` MUST ensure that there is negligible probability that
522 that party or any other party will accidentally assign the same identifier to any other message.

523 The mechanism by which SOAP-based ID-* senders or receivers ensure that an identifier is unique is left to
524 implementations. In the case that a pseudorandom technique is employed, the probability of two randomly chosen
525 identifiers being identical MUST be less than 2^{-128} and SHOULD be less than 2^{-160} . The above requirement MAY
526 be met by applying Base64 [\[RFC2045\]](#) encoding to a randomly chosen value [\[RFC1750\]](#) 128 or 160 bits in length.

527 It is OPTIONAL for a `messageID` value to be resolvable in principle to some resource. In the case that a `messageID`
528 is resolvable in principle (for example, it is in the form of a URI reference [\[RFC2396\]](#)), it is OPTIONAL for the
529 identifier to be dereferenceable.

530 **5.1.3. <Correlation> Header Block Element**

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

533 The schema fragment illustrated in [Figure 5](#) defines the `<Correlation>` element, and [Example 4](#) illustrates an
534 instantiated one.

```
535
536
537     <xs:element name="Correlation" type="correlationType"/>
538
539
```

540 **Figure 5. The `<Correlation>` Header Block Element Schema**

```
541
542
543     <Correlation S:mustUnderstand="1"
544         id="A13454...245"
545         actor="http://schemas.../next"
546         messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
547         timestamp="2112-03-15T11:12:12Z" />
548
549
```

550 **Example 4. An Instantiated <Correlation> Header Block**

551 **5.2. Provider ID / Affiliation ID Declaration: The <Provider> Header** 552 **Block**

553 This section defines the <Provider> header block. This header block provides a means for a sender to claim that it
554 is represented by a given providerID value. The sender may also claim that it is a member of a given affiliation. Such
555 claims are generally verifiable by receivers by looking up these values in the sender's metadata [[LibertyMetadata](#)].

556 **5.2.1. The ProviderType Header Block Type**

557 The ProviderType header block type defines the following attributes:

- 558 • providerID [Required] – The Provider ID of the sender.
- 559 • affiliationID [Optional] – The Affiliation ID of the sender, if any.
- 560 • id [Optional] – identifies a <Provider> header block instance. This attribute **MUST** be used when the message
561 is signed as described in [[LibertySecMech](#)], and the element instance is to be included as one of the set of signed
562 message components.
- 563 • S:mustUnderstand [Optional] – The SOAP mustUnderstand attribute [[SOAPv1.1](#)].
- 564 • S:actor [Optional] – The SOAP actor attribute [[SOAPv1.1](#)].

565 The schema fragment in [Figure 6](#) defines the ProviderType header block type.

```
566 <xs:complexType name="ProviderType">  
567   <xs:attribute name="providerID" type="xs:anyURI" use="required"/>  
568   <xs:attribute name="affiliationID" type="xs:anyURI" use="optional"/>  
569   <xs:attribute name="id" type="xs:ID" use="optional"/>  
570   <xs:attribute ref="S:mustUnderstand" use="optional"/>  
571   <xs:attribute ref="S:actor" use="optional"/>  
572 </xs:complexType>
```

577 **Figure 6. The ProviderType Header Block Type Schema**

578 **5.2.2. <Provider> Header Block Element**

579 The <Provider> header block element implements the features of the providerType header block type.

580 The schema fragment shown in [Figure 7](#) defines the <Provider> element, and [Example 5](#) illustrates an instantiated
581 <Provider> element.

```
582 <xs:element name="Provider" type="providerType"/>
```

587 **Figure 7. The <Provider> Header Block Element Schema**

```
588  
589  
590 <Provider providerID="example.com"  
591   affiliationID="affiliation.example.com"  
592   S:mustUnderstand="1"  
593   id="A9kendan...542"  
594   actor="http://schemas.../next"/>  
595  
596
```

597 **Example 5. An instantiated <Provider> Header Block**

598 5.3. Messaging Processing Rules

599 Overall processing of SOAP-bound ID-* messages, which by definition contain the <Correlation> and the
600 <Provider> header blocks, follows the rules of the SOAP processing model described in [SOAPv1.1]; specifically,
601 the SOAP `mustUnderstand` and `actor` attributes MAY be used to mandate header block processing and target
602 header blocks, respectively. Where applicable, specific processing rules for these attributes are given in the overall
603 processing rules defined below.

604 The system entity constructing and sending a SOAP-bound ID-* message is called the *sender* in the context of the act
605 of sending the message. The entity receiving this message is called the *receiver* in the context of the act of receiving
606 an individual message (see [Section 2.2: Terminology](#)).

607 Two *Message Exchange Patterns* (MEPs) are supported: one-way, and request-response. One-way is simply where a
608 sender sends a message to a receiver without necessarily expecting to receive an explicit response to the sent message.
609 Request-response is where a sender sends a message to a receiver and expects to receive an explicit response.

610 The processing rules are described below in terms of [Constructing and Sending a SOAP-bound ID-* Message](#) and
611 [Receiving and Processing a SOAP-bound ID-* Message](#). A sender instigating a one-way message exchange will
612 perform only the steps outlined in the former section. A sender participating in a request-response message exchange
613 will perform the steps in the former section when sending a message, and the steps in the latter section when receiving
614 and processing the response. A receiver participating in a request-response exchange will do the reverse. Note that a
615 receiver of an asynchronous one-way message will perform the steps in the latter section.

616 **Note:**

617 The label "ID-* header block(s)" is used to refer to at least one of, or all of, the following set of header blocks
618 (the first nine are defined in this specification, the remainder are defined in the cited specifications):

- 619 • <Correlation>
- 620 • <Provider>
- 621 • <ProcessingContext>
- 622 • <Consent>
- 623 • <UsageDirective>
- 624 • <ServiceInstanceUpdate>
- 625 • <Timeout>

- 626 • <CredentialsContext>
- 627 • <wsse:Security> [[LibertySecMech](#)]
- 628 • <is:UserInteraction> [[LibertyInteract](#)]

629 Other specifications in the Liberty ID-* specification suite MAY define header block(s) not listed above.
630 Nevertheless, they should generally be considered a member of the above list when interpreting the processing
631 rules in this section, and explicitly considered where the processing rules refer to "ID-* header blocks" (see
632 [Section 2.2: Terminology](#)).

633 **5.3.1. Constructing and Sending a SOAP-bound ID-* Message**

634 The sender MUST follow these processing rules when constructing and sending an outgoing SOAP-bound ID-*
635 message (hereafter referred to as the *outgoing message*):

- 636 1. The outgoing message MUST satisfy the rules given in [Section 4: SOAP Binding](#).
- 637 2. The outgoing message MUST include exactly one <Correlation> header block (see [Section 5.1](#)) in the
638 <S:Header> child element of the <S:Envelope> element.
639 The values of the <Correlation> header block attributes MUST be set as follows:
 - 640 A. `messageID` MUST be present, and its value SHOULD be set according to the rules presented in
641 [Section 5.1.2: messageID Value Requirements](#) .
 - 642 B. `refToMessageID` MUST be present if the sender is both participating in a request-response MEP and is
643 responding to a prior-received message. The `refToMessageID` attribute value MUST be set to the value
644 of the `messageID` attribute from the prior-received message. Additionally, the `refToMessageID` value
645 SHOULD be set according to the rules presented in [Section 5.1.2: messageID Value Requirements](#) .
646 **Note:**
647 If the outgoing message will convey an ID-* fault message ([Section 4.4](#)), then the `refToMessageID`
648 attribute of its <Correlation> header block SHOULD NOT be present.
 - 649 C. `timestamp` MUST be present. Its value SHOULD be set to the time at which the message is sent. The
650 `timestamp` value MUST conform to the rules presented in [Section 2.5: Time Values](#)
 - 651 D. `id` MUST be present if the sender is signing the message [[LibertySecMech](#)] and including the
652 <Correlation> header block among the signed elements. Its value SHOULD be set according to
653 the rules presented in [Section 5.1.2: messageID Value Requirements](#)
 - 654 E. `S:actor` SHOULD be present. Its value SHOULD be "http://schemas.xmlsoap.org/soap/actor/next".
 - 655 F. `S:mustUnderstand` SHOULD be present. Its value SHOULD be TRUE.
- 656 3. The message SHOULD include exactly one, and MUST include no more than one <Provider> header block (see
657 [Section 5.2](#)) in the <S:Header> child element of the <S:Envelope> element. If this header block is present,
658 the sender is claiming to be a Liberty provider with the specified `providerID`.
659 The values of the <Provider> header block attributes MUST be set as follows:
 - 660 A. `providerID` MUST be present and SHOULD be set to a value appropriate for the sender to claim
661 [\[LibertyMetadata\]](#).

- 662 B. `affiliationID` MAY be present. If so, it SHOULD be set to a value appropriate for the sender to claim
663 [\[LibertyMetadata\]](#).
- 664 C. `id` MUST be present if the sender is signing the message [\[LibertySecMech\]](#) and including the `<Provider>`
665 header block among the signed elements. Its value SHOULD be set according to the rules presented in
666 [Section 5.1.2: messageID Value Requirements](#)
- 667 D. `S:actor` SHOULD be present. Its value SHOULD be `"http://schemas.xmlsoap.org/soap/actor/next"`.
- 668 E. `S:mustUnderstand` SHOULD be present. Its value SHOULD be `TRUE`.
- 669 4. The sender MAY include other ID-* header blocks in the message, in addition to the two enumerated
670 above, as required by the overall messaging and processing context. For example, the sender may include a
671 `<wsse:Security>` header block [\[LibertySecMech\]](#).
- 672 5. The sender adds either:
- 673 A. an ordinary ID-* message (as described in [Section 4.3: Ordinary ID-* Messages](#) ; see [Example 2](#)), or,
674 B. an ID-* fault message (as **prescribed** in [Section 4.4: ID-* Fault Messages](#) ; see [Example 3](#)),
675 to the SOAP-bound ID-* message's `<S:Body>` element.
- 676 6. The sender also performs any needed additional preparation of the message, for example including other header
677 blocks, and signing some or all of the message elements, and then sends the message to the receiver. See
678 [Section 5.4: Correlation and Provider ID / Affiliation ID Examples](#) .

679 5.3.2. Receiving and Processing a SOAP-bound ID-* Message

680 The receiver of a SOAP-bound ID-* message, either ordinary or fault, MUST perform the following processing steps
681 on the ID-* header blocks of the incoming SOAP-bound ID-* message.

682 **Note:**

683 Although the steps below are explicitly arranged and numbered sequentially, the intent is **not** to strictly define
684 a specific overall processing algorithm in terms of having implementations follow these steps in exactly the
685 same sequence on a per-header-block basis. However, all specified tests MUST be applied as appropriate to
686 all ID-* header blocks in the inbound SOAP-bound ID-* message.

687 1. Processing common to **all** received ID-* header blocks:

- 688 A. The `S:actor` attribute MAY be present. If present, its value SHOULD be
689 `"http://schemas.xmlsoap.org/soap/actor/next"` or some other previously agreed upon (out-of-
690 band) value.
- 691 B. The `S:mustUnderstand` attribute MAY be present. If present, its value SHOULD be `TRUE`.
- 692 C. A single `<Correlation>` header block should be present in the header of the message.

- 693 D. If the foregoing tests (1.A and 1.B and 1.C) hold true, processing continues with step 2.
- 694 E. Otherwise, the receiver MAY respond to the sender with a SOAP-bound ID-* Fault message (per Section 4.4:
695 ID-* Fault Messages) with the <Status> element configured with:
- 696 • a code attribute with a value of:
 - 697 • "sb:InvalidActor" if the failed test is 1.A,
 - 698 • "sb:InvalidMustUnderstand" if the failed test is 1.B,
 - 699 • "sb:IDStarMsgNotUnderstood" if the failed test is 1.C.
 - 700 • and a ref attribute with its value taken from the messageID value of the incoming message.
- 701 The <S:Fault> SHOULD contain a <S:faultcode> of S:Client.
702 Additionally, a header block element of the same type as the offending header block SHOULD be included
703 in the ID-* Fault message's "<detail>" element, and the id attribute value of this header block element
704 SHOULD be set to the same value as the id attribute of the offending header block in the incoming message.
705 The receiver MAY discard the incoming message. The receiver is finished processing this incoming message
706 at this point.
- 707 2. Processing specific to the <Correlation> header block:
- 708 A. Ensure that the value of the timestamp attribute is within an appropriate offset from local time expressed
709 in UTC. Absent other guidance, a value of 5 minutes MAY be used.
 - 710 B. For messages within the previous step's time window, ensure that the messageID attribute has not been
711 seen before. If it has, then this message is likely a duplicate. Receivers SHOULD maintain a list of received
712 messageID values that, at least, fall within the previous step's time window.
 - 713 C. If the refToMessageID attribute is present, and its value is non-null, and if the receiver is participating in
714 a request-response MEP with the sending party, then the value of the refToMessageID attribute SHOULD
715 match the value of the messageID attribute of a message previously sent by the receiver to the sender of the
716 now incoming message.
 - 717 D. If the foregoing tests (2.A through 2.C) hold true, processing continues with step 3 .
 - 718 E. Otherwise, the receiver MAY respond to the sender with a SOAP-bound ID-* Fault message (per Section 4.4)
719 with the <Status> element configured with:
 - 720 • a code attribute with a value of:
 - 721 • "sb:StaleMsg" if the failed test is 2.A,
 - 722 • "sb:DuplicateMsg" if the failed test is 2.B,
 - 723 • "sb:InvalidRefToMsgID" if the failed test is 2.C,
 - 724 • and a ref attribute with its value taken from the messageID value of the incoming message.

725 The <S:Fault> SHOULD contain a <S:faultcode> of S:Client.
726 Additionally, a header block element of the same type as the offending header block SHOULD be included
727 in the ID-* Fault message's "s:detail" element, and the id attribute value of this header block element
728 SHOULD be set to the same value as the id attribute of the offending header block in the incoming message.
729 The receiver MAY discard the incoming message. The receiver is finished processing this incoming message
730 at this point.

731 3. At this point, the receiver of the message MAY cease processing the message, and indicate to the requester that
732 the message should be re-submitted to a different endpoint, according to the rules specified in [Section 6.4.4.2](#)

733 4. Processing specific to the <Provider> header block:

734 A. Verify that any declared providerID or affiliationID, are valid. The receiver SHOULD perform this
735 verification and validation against metadata obtained via methods specified by [\[LibertyMetadata\]](#).

736 B. If the foregoing test (4.A) holds true, processing continues with step 5.

737 C. Otherwise, the receiver MAY respond to the sender with a SOAP-bound ID-* Fault message (per [Section 4.4](#))
738 with the <Status> element configured with:

739 • a code attribute with a value of:

740 • "sb:ProviderIDNotValid", or,

741 • "sb:AffiliationIDNotValid", as appropriate (if both the claimed Provider ID and the Affilia-
742 tion ID
743 are deemed invalid, then the returned code SHOULD be "sb:AffiliationIDNotValid"),

744 • and a ref attribute with its value taken from the messageID value of the incoming message.

745 The <S:Fault> SHOULD contain a <S:faultcode> of S:Client.
746 Additionally, a header block element of the same type as the offending header block SHOULD be included
747 in the ID-* Fault message's "<detail>" element, and the id attribute value of this header block element
748 SHOULD be set to the same value as the id attribute of the offending header block in the incoming message.
749 The receiver MAY discard the incoming message. The receiver is finished processing this incoming message
750 at this point.

751 5. Processing specific to ID-* header blocks **other than** <Correlation> and <Provider> – see appropriate
752 sections in this and other specifications:

753 • For <wsse:Security>, see [\[LibertySecMech\]](#).

754 **Note:**

755 It should be noted that the receiver MAY return an sb-ext:InappropriateCredentials based on their
756 processing of the <wsse:Security> header block, under conditions specified below in [Section 6.4.4](#) and
757 [Section 6.3.4](#), in addition to other conditions such as an expired credential (see [\[LibertySecMech\]](#)).

758 • For <ProcessingContext>, see [Section 6.1](#).

759 • For <Consent>, see [Section 6.2](#).

- 760 • For <UsageDirective>, see [Section 6.6](#).
- 761 • For <ServiceInstanceUpdate>, see [Section 6.4](#).
- 762 • For <Timeout>, see [Section 6.5](#).
- 763 • For <CredentialsContext>, see [Section 6.3](#).
- 764 • For <is:UserInteraction>, see [\[LibertyInteract\]](#).

765 The manner of reporting any issues with these header blocks is specified in the respective section of this
766 specification or in other specification(s) as noted above. If there are no issues with these header blocks, then
767 processing continues with step 6 below, otherwise the receiver is finished processing this incoming message at
768 this point.

769 6. If the incoming message's applicable header blocks have passed all specified and applicable tests, the incoming
770 message SHOULD be dispatched for further processing as appropriate, based on the ordinary ID-* message or
771 ID-* fault message contained in the encompassing SOAP message's <S:Body> element.
772 Or, if the message contained in the encompassing SOAP message's <S:Body> element is not recognizable as
773 either an ordinary ID-* message or an ID-* fault message, and thus is not dispatchable, the receiver MAY respond
774 to the sender with a SOAP-bound ID-* Fault message (per [Section 4.4](#)) with the <Status> element configured
775 with:

776 • a code attribute with a value of:

777 • "sb:IDStarMsgNotUnderstood"

778 • and a ref attribute with its value taken from the messageID value of the incoming message.

779 Receivers MUST be able to avoid ID-* fault message "loops". For example, if the incoming message is conveying
780 an ID-* fault message, and there is some issue with one or more of its ID-* header blocks, the receiver should not
781 issue a SOAP-bound ID-* Fault message in response.

782 **Note:**

783 Other specifications conforming to this binding that specify ordinary ID-* messages and their processing,
784 such as [LibertyIDPP] or [LibertyDisco], MAY define <Status> element code attribute values in addition to
785 the ones defined in Section 3.3.1 of this document. These code attribute values SHOULD be used to signal to
786 the sender any issues with the incoming ordinary ID-* message found by the receiver. This specification does
787 not define any such conditions other than the one described above in 6, and they are not further discussed in
788 this document.

789 **5.4. Correlation and Provider ID / Affiliation ID Examples**

790 **Example 6** illustrates a SOAP-bound ID-* message, containing both <Correlation> and <Provider> header
791 blocks, and conveying a Personal Profile (ID-PP) Modify message [LibertyIDPP].

```
792
793
794 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
795   xmlns:sb="urn:liberty:sb:2003-08"
796   xmlns:idpp="urn:liberty:id-sis-pp:2003-08">
797
798   <S:Header>
799
800     <!-- other header blocks, eg wsse:security, may go here -->
801
802     <Correlation S:mustUnderstand="1"
803       id="A13454...245"
804       actor="http://schemas.../next"
805       messageID="uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbbb"
806       timestamp="2112-03-15T11:12:12Z"/>
807
808     <Provider providerID="example.com"
809       affiliationID="affiliation.example.com"
810       S:mustUnderstand="1"
811       id="A9kendan...542"
812       actor="http://schemas.../next"/>
813
814     <!-- other header blocks, eg wsse:security, may go here -->
815
816   </S:Header>
817
818   <S:Body>
819
820     <idpp:Modify>
821       : <!-- this is an ID-PP Modify message -->
822     </idpp:Modify>
823
824   </S:Body>
825
826 </S:Envelope>
827
828
```

829 **Example 6. A SOAP-bound ID-* Message with a <Correlation> Header Block**

830 **Example 7** illustrates a SOAP-bound ID-* message, containing both <Correlation> and <Provider> header
831 blocks. The <Correlation> header block is referring to the message in the previous example, which conveyed an ID-
832 PP Modify message, and the message in **Example 7** is conveying an ID-PP ModifyResponse message in response to the
833 former. Note how the refToMessageID attribute is referencing the messageID in the example above. Additionally,
834 note that the <Provider> header block is claiming that its Provider ID is example579.com, which is different than
835 that claimed by the sender of the message in **Example 6**. Note that both system entities are claiming to belong to have
836 the same Affiliation ID, affiliation.example.com.

```
837
838
839 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
840   xmlns:sb="urn:liberty:sb:2003-08"
841   xmlns:idpp="urn:liberty:id-sis-pp:2003-08">
842
843   <S:Header>
844
845     <!-- other header blocks, eg wsse:security, go here -->
846
847     <Correlation S:mustUnderstand="1"
848       id="B893483..83736"
849       actor="http://schemas.../next"
850       messageID="uuid:aaaaeeee-efef-fefe-efef-abababababab"
851       refToMessageID=
852       "uuid:efefefef-aaaa-ffff-cccc-eeeeffffbbbb"
853       timestamp="2112-03-15T11:12:13Z"/>
854
855     <Provider providerID="example579.com"
856       affiliationID="affiliation.example.com"
857       S:mustUnderstand="1"
858       id="A0brijas...975"
859       actor="http://schemas.../next"/>
860
861     <!-- other header blocks, eg wsse:security, go here -->
862
863   </S:Header>
864
865   <S:Body>
866
867     <idpp:ModifyResponse>
868       : <!-- this is an ID-PP ModifyResponse message -->
869     </idpp:ModifyResponse>
870
871   </S:Body>
872
873 </S:Envelope>
874
875
```

876 **Example 7. A SOAP-bound ID-* Message with a <Correlation> Header Block Referencing a Another Message**

877 6. Optional Header Blocks

878 This section describes ID-* header blocks that are "optional" in the sense that they are not integral to basic ID-*
879 messaging, per se. For example, the <Correlation> header block is minimally necessary to realize the simple
880 one-way and request-reply MEPs defined above. Plus, the <Provider> header block facilitates receiver retrieval of
881 provider metadata, which is also minimally necessary for communication between Liberty-enabled system entities.

882 In contrast, the header blocks described in this section are not strictly necessary to realize such basic messaging
883 notions. However, the optional header blocks described below are essential to realizing several of the higher-level
884 Project Liberty notions—for example: Principal consent and policy-based interactions.

885 The optional header blocks described in this specification are:

- 886 • <ProcessingContext>
- 887 • <Consent>
- 888 • <UsageDirective>
- 889 • <ServiceInstanceUpdate>
- 890 • <Timeout>
- 891 • <CredentialsContext>

892 The following sections describe these optional ID-* header blocks along with their specific processing rules.

893 **Note:**

894 Whenever an optional header block appears in a SOAP-bound ID-* message, the processing rules specific to
895 that header block (which are given in this section, below) **MUST** be used in combination with the messaging
896 processing rules given above in [Section 5.3: Messaging Processing Rules](#) . This applies whether the message
897 is being constructed and sent, or being received and processed.

898 6.1. The <ProcessingContext> Header Block

899 This section defines the <ProcessingContext> header block. This header block may be employed by a sender to
900 signal to a receiver that the latter should add a specific additional facet to the overall *processing context* in which
901 any action(s) are invoked as a result of processing any ID-* message also conveyed in the overall SOAP-bound ID-*
902 message. The full semantics of this header block are described below in [Section 6.1.3: <ProcessingContext>](#)
903 Header Block Semantics and Processing Rules .

904 Processing context facets are denoted by URIs. URIs are assigned to denote specific processing context facets. This
905 specification defines several such URIs below in [Section 6.1.3.2](#).

906 6.1.1. The ProcessingContextType Header Block Type

907 The `ProcessingContextType` content model is `anyURI`. It defines the following attributes:

- 908 • `id` [Optional] – identifies a <ProcessingContext> header block instance. This attribute **MUST** be used when
909 the message is signed as described in [\[LibertySecMech\]](#), and the element instance is to be included as one of the
910 set of signed message components.
- 911 • `S:mustUnderstand` [Optional] – The SOAP `mustUnderstand` attribute [\[SOAPv1.1\]](#).

- 912 • S:actor [Optional] – The SOAP actor attribute [SOAPv1.1].

913 The following schema fragment defines the ProcessingContext header block type:

```
914
915
916     <xs:complexType name="ProcessingContextType">
917         <xs:simpleContent>
918             <xs:extension base="xs:anyURI">
919                 <xs:attribute name="id" type="xs:ID" use="optional"/>
920                 <xs:attribute ref="S:mustUnderstand" use="optional"/>
921                 <xs:attribute ref="S:actor" use="optional"/>
922             </xs:extension>
923         </xs:simpleContent>
924     </xs:complexType>
925
926
```

927 **Figure 8. The ProcessingContext Header Block Type Schema**

928 6.1.2. <ProcessingContext> Header Block Element

929 The <ProcessingContext> schema element is given in [Figure 9](#).

```
930
931
932     <xs:element name="ProcessingContext" type="ProcessingContextType"/>
933
934
```

935

Figure 9. The <ProcessingContext> Element Schema

936

937

938

```
<S:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:idpp="urn:liberty:id-sis-pp:2003-08">
```

940

```
<S:Header>
```

942

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

944

```
<ProcessingContext S:mustUnderstand="1">
  urn:liberty:sb:2003-08:ProcessingContext:PrincipalOffline
</ProcessingContext>
```

948

```
<Correlation S:mustUnderstand="1"
  id="B893483..83736"
  actor="http://schemas.../next"
  messageID="uuid:eeeecccc-efef-fefe-efef-bddbdbdb"
  timestamp="2112-03-15T11:12:13Z"/>
```

954

```
<Provider providerID="example.com"
  affiliationID="affiliation.example.com"
  S:mustUnderstand="1"
  id="A9kendan...542"
  actor="http://schemas.../next"/>
```

959

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

962

```
</S:Header>
```

964

```
<S:Body>
```

966

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

969

```
</S:Body>
```

971

```
</S:Envelope>
```

974

975

976

Example 8. A SOAP-bound ID-* Message with an Attached <ProcessingContext> Header Block**977 6.1.3. <ProcessingContext> Header Block Semantics and Processing Rules**

978 This section first describes the overall semantics of the <ProcessingContext> header block, then defines two
979 *processing context facet URIs*, and concludes with defining specific processing rules.

980 6.1.3.1. <ProcessingContext> Header Block Semantics

981 The overall semantic of the <ProcessingContext> header block is:

982 The <ProcessingContext> header block MAY be employed by a sender, who is acting in a web
983 services client (WSC) role, to signal to a receiver, who is acting in a web services provider (WSP)
984 role, that the latter should add a specific *processing context facet* to the overall *processing context*
985 (see [Section 2.2: Terminology](#)) in which the *service request* is evaluated.

986 The specific processing context facet being conveyed by the <ProcessingContext> header block is identified by
987 the header block's URI element value.

988 Such URIs are known as *processing context facet URIs*. An example of a processing context facet that may be signaled
989 by such a URI is whether the principal should be considered to be online or not.

990 An ID-WSF or ID-SIS WSP receiving a service request containing a <ProcessingContext> header block with
991 one of the above processing context facet URIs SHOULD process the conveyed ID-* message **with the indicated**
992 **processing context facet in force**. Thus the ID-* message's processing as well as any applicable access management
993 policies are exercised within an overall processing context which includes the processing context facet. Finally, an
994 indication of success or failure of the ID-* message processing is returned to the sender, in the same manner as would
995 be done if the ID-* message had been sent without the attendant <ProcessingContext> header block.

996 The above completely describes the semantic of this header block itself, and further description of particular effects
997 on processing must be made in descriptions of processing context facet URIs. Such a description is given in the next
998 section.

999 **Note:**

1000 Whether or not a receiver honors a <ProcessingContext> header block is a matter of local policy at the
1001 receiver, as is whether or not a receiver honors any given request from any given sender. For example,
1002 the <ProcessingContext> header block functionality has security implications in the sense of possibly
1003 facilitating an adversary to probe a receiver's behavior given adversary-chosen inputs. For these reasons,
1004 whether or not the <ProcessingContext> header block functionality is enabled on the part of a receiver
1005 with respect to a particular sender, should be a matter of business-level agreement between the receiver and
1006 the sender.

1007 **6.1.3.2. Processing Context Facet URIs: PrincipalOnline, PrincipalOffline, and Simulate**

1008 Three processing context facet URIs are defined below for use with the <ProcessingContext> header block:

1009 urn:liberty:sb:2003-08:ProcessingContext:PrincipalOffline
1010 Conduct the processing of the ID-* message as if the Principal is offline.

1011 urn:liberty:sb:2003-08:ProcessingContext:PrincipalOnline
1012 Conduct the processing of the ID-* message as if the Principal is online.

1013 urn:liberty:sb:2003-08:ProcessingContext:Simulate
1014 *Simulate* the processing of the ID-* message.

1015 If the sender includes a <UserInteraction> header block in addition to the <ProcessingContext> header block
1016 in the SOAP-bound ID-* request message, the receiver and sender MUST appropriately initiate the indicated user
1017 interaction (see [\[LibertyInteract\]](#)), and incorporate information supplied by the user as a part of the resultant user
1018 interaction, into the appropriate data and/or policy stores.

1019 **Note:**

1020 Any processing context facet that was conveyed in the <ProcessingContext> header block MUST NOT
1021 be enforced during such a user interaction. Rather, it applies only to the processing of the ID-* message itself.

1022 In summary, the overall intended side-effect of using the above-defined processing context facets is for the receiver to
1023 evaluate applicable policy, and return a putative indication of success or failure to the sender. This provides WSCs the
1024 capability to make an ID-WSF or ID-SIS service request and ascertain whether it will be successful or not—without
1025 the service request actually being carried out. Additionally, it facilitates carrying out any user interaction that may be
1026 indicated by the current combination of service request context and applicable policy. This will, for example, facilitate
1027 some WSCs to fashion more "user friendly" experiences.

1028 **6.1.3.3. Defining New Processing Context Facet URIs**

1029 The rightmost portions of the processing context facet URIs after the "ProcessingContext:" component are referred
1030 to as *processing context facet identifiers*. For example, whether the Principal is online or not is a facet of a request
1031 context. New processing context facet identifiers MAY be defined in other specifications, for example in ID-SIS data
1032 service specifications. An ID-SIS data service may define as many levels of request context identifiers as necessary to
1033 address the application's needs.

1034 **6.1.3.4. Sender Processing Rules**

1035 A sender MAY include a <ProcessingContext> header block in a SOAP-bound ID-* message along with other
1036 header blocks besides <Correlation> and <Provider> such as <UserInteraction>. The sender MUST include
1037 a processing context facet URI in the <ProcessingContext> header block. The sender then sends the ID-* SOAP-
1038 bound message to an ID-WSF or ID-SIS service-hosting node (AKA the receiver).

1039 A sender MAY indicate that it believes either that the Principal is currently "online" or "offline" when it sends a
1040 message by specifying one of the two processing context facet URIs:

1041 • urn:liberty:sb:2003-08:ProcessingContext:PrincipalOnline

1042 • urn:liberty:sb:2003-08:ProcessingContext:PrincipalOffline

1043 The sender will typically receive a response from the receiver indicating success or failure or will receive a SOAP
1044 fault indicating a processing error with the SOAP-bound ID-* message. Note that in the case of a "successful" request
1045 *simulation*, the service will not return any result data other than an indication of success or failure to the sender.

1046 **6.1.3.5. Receiver Processing Rules**

1047 The receiver of a request containing a <ProcessingContext> header block MUST examine the included processing
1048 context facet URI. If it is known to the data service, then the data service MUST attempt to process the data service
1049 request, represented by the ID-* message, in an overall processing context including the processing context facet as
1050 indicated by the conveyed processing context facet URI, and return an indication of success or failure to the sender.

1051 If the data service request is malformed or has some other issue that would normally cause the receiver to issue a
1052 SOAP fault, the receiver SHOULD do so.

1053 If the receiver is asked to simulate processing of the request (by the inclusion of the
1054 urn:liberty:sb:2003-08:ProcessingContext:Simulate facet URI), and they are both able and willing to
1055 honour that processing context, then the receiver MUST evaluate the conveyed ID-* message, but MUST NOT actually
1056 perform the operation. That is, the receiver MUST NOT make actual changes to underlying ID-* service datastore,
1057 and it MUST NOT return any data as a result of evaluating the ID-* message.

1058 If the sender includes a <UserInteraction> header block, in addition to the <ProcessingContext> header block,
1059 then both participants MUST initiate the indicated user interaction (see [\[LibertyInteract\]](#) appropriately, and incorporate
1060 information supplied by the user as a part of the interaction into appropriate data and/or policy stores, even if
1061 the urn:liberty:sb:2003-08:ProcessingContext:Simulate URI is specified in a <ProcessingContext>
1062 header.

1063 In the event the receiver does not understand the included processing context facet URI, the receiver MAY respond with
1064 a SOAP-bound ID-* fault message (per [Section 4.4: ID-* Fault Messages](#)) with the <Status> element configured
1065 with:

1066 • a code attribute with a value of:

1067 • "sb:ProcCtxURINotUnderstood"

1068 • and a `ref` attribute with its value taken from the `messageID` value of the incoming message.

1069 In the event the receiver is not willing to enforce a stipulated processing context, the receiver MAY respond with a
1070 SOAP-bound ID-* fault message (per [Section 4.4: ID-* Fault Messages](#)) with the `<Status>` element configured
1071 with:

1072 • a `code` attribute with a value of:

1073 • `"sb:ProcCtxUnwilling"`

1074 • and a `ref` attribute with its value taken from the `messageID` value of the incoming message.

1075 **Note:**

1076 The receiver MAY reference multiple `<ProcessingContext>` headers in the `<detail>` of the fault
1077 response (in accordance with the rules specified in [Section 4.4](#)).

1078 6.2. The `<Consent>` Header Block

1079 This section defines the `<Consent>` header block. This header block is used to explicitly claim that the Principal
1080 consented to the present interaction.

1081 6.2.1. The `consentType` Header Block Type

1082 The `<Consent>` header block element MAY be employed by either a requester or a receiver. For example, the Principal
1083 may be using a Liberty-enabled client or proxy (common in the wireless world), and in that sort of environment the
1084 mobile operator may cause the Principal's terminal (AKA: cell phone) to prompt the principal for consent for some
1085 interaction.

1086 The `consentType` header block type has the following attributes:

1087 • `uri` [Required] – A URI indicating that the Principal's consent was obtained.

1088 Optionally, the URI MAY identify a particular *Consent Agreement Statement* defining the specific nature of the
1089 consent obtained.

1090 This specification defines one well-known URI Liberty implementors and deployers MAY use to indicate positive
1091 Principal consent was obtained with respect to whatever ID-* interaction is underway or being initiated. This URI
1092 is known as the "Principal Consent Obtained" URI (PCO). The value of this URI is:

1093 `urn:liberty:consent:obtained` This URI does not correspond to any particular Consent Agreement
1094 Statement. Rather, it simply states that consent was obtained. The full meaning and implication of this will
1095 need to be derived from the execution context.

1096 • `timestamp` [Optional] – For denoting the time at which the sender obtained Principal consent with the POC.

1097 • `id` [Optional] – identifies a `<Consent>` header block instance. This attribute MUST be used when the message is
1098 signed as described in [\[LibSecMech\]](#), and the element instance is to be included in the signed message components.

1099 • `S:mustUnderstand` [Optional] – The SOAP `mustUnderstand` attribute [\[SOAPv1.1\]](#).

1100 • `S:actor` [Optional] – The SOAP `actor` attribute [\[SOAPv1.1\]](#).

1101 The schema fragment in [Figure 10](#) defines the Consent header block type.

```
1102
1103
1104     <xs:complexType name="consentType">
1105     <xs:attribute name="uri" type="xs:anyURI" use="required"/>
1106     <xs:attribute name="timestamp" type="xs:dateTime" use="optional"/>
1107     <xs:attribute name="id" type="xs:ID" use="optional"/>
1108     <xs:attribute ref="S:mustUnderstand" use="optional"/>
1109     <xs:attribute ref="S:actor" use="optional"/>
1110 </xs:complexType>
1111
1112
```

1113 **Figure 10. The Consent Header Block Type Schema**

1114 6.2.2. <Consent> Header Block Element

1115 The schema fragment in [Figure 11](#) defines the <Consent> element:

```
1116
1117
1118     <xs:element name="Consent" type="consentType"/>
1119
1120
```

1121 **Figure 11. The <Consent> Element Schema**

```
1122
1123
1124     <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
1125     xmlns:idpp="urn:liberty:id-sis-pp:2003-08">
1126
1127     <S:Header>
1128
1129         <!-- other header blocks, eg wsse:security, may go here -->
1130
1131         <Consent id="A124395732495743"
1132         uri="urn:liberty:consent:obtained"
1133         timestamp="2112-03-15T11:12:10Z"/>
1134
1135         <Correlation S:mustUnderstand="1"
1136         id="B893483..83736"
1137         actor="http://schemas.../next"
1138         messageID="uuid:eeeecccc-efef-fefe-efef-bddbdbdb"
1139         timestamp="2112-03-15T11:12:13Z"/>
1140
1141         <!-- other header blocks, eg wsse:security, may go here -->
1142
1143     </S:Header>
1144
1145     <S:Body>
1146
1147         <idpp:Modify> <!-- This is an ID-PP "Modify" message bound -->
1148         : <!-- into the <Body> of a SOAP message. -->
1149         </idpp:Modify>
1150
1151     </S:Body>
1152
1153 </S:Envelope>
1154
1155
```

1156 **Example 9. A SOAP-bound ID-* Message with an Attached <Consent> Header Block**

1157 **6.3. The <CredentialsContext> Header Block**

1158 **6.3.1. Overview**

1159 It may be necessary for an entity receiving an ID-* message to indicate the type of credentials that should be used by
1160 the requester in submitting a message.

1161 **6.3.2. CredentialsContext Type and Element**

1162 Receivers of an ID-* message MAY add <CredentialsContext> elements to the SOAP header of their response.

1163 The element is based upon the `CredentialsContextType` which is defined as:

- 1164 • `lib:RequestAuthnContext` [Optional] – a container that allows the expression of a requested authentication
1165 context (see [\[LibertyAuthnContext\]](#)).
- 1166 • `SecurityMechID` [Optional] – A set of elements that specify ID-WSF security mechanism URIs (see [\[Liberty-](#)
1167 `SecMech]).`
- 1168 • `id` [Optional] – An attribute facilitating references to elements of this type.
- 1169 • `S:mustUnderstand` [Optional] – The SOAP `mustUnderstand` attribute [\[SOAPv1.1\]](#).
- 1170 • `S:actor` [Optional] – The SOAP `actor` attribute [\[SOAPv1.1\]](#).

1171 The following schema fragment describes the <CredentialsContext> header block.

```
1172 <xs:complexType name="CredentialsContextType">
1173   <xs:sequence>
1174     <xs:element ref="lib:RequestAuthnContext" minOccurs="0" />
1175     <xs:element name="SecurityMechID" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded" />
1176   </xs:sequence>
1177   <xs:attribute name="id" type="xs:ID" use="optional" />
1178   <xs:attribute ref="S:mustUnderstand" use="optional" />
1179   <xs:attribute ref="S:actor" use="optional" />
1180 </xs:complexType>
1181
1182 <xs:element name="CredentialsContext" type="CredentialsContextType" />
1183
1184
1185
```

1186 **Figure 12. The <CredentialsContext> Header Block Element and Type Schema**

1187 **6.3.3. CredentialsContext Example**

```

1188
1189
1190 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
1191   xmlns:sb="urn:liberty:sb:2003-08"
1192   xmlns:lib="urn:liberty:iff:2003-08">
1193
1194   <S:Header>
1195
1196     <sb:Correlation S:mustUnderstand="1"
1197       messageID="uuid:0023923-28329023-238239023"
1198       refToMessageID="uuid:00123-ad32-23-09923-88239"
1199       timestamp="2003-06-06T12:10:10Z" />
1200
1201     <!-- Says that the sender would like credentials that include RequestAuthnContext
1202          as specified -->
1203
1204     <sb-ext:CredentialsContext mustUnderstand="1">
1205
1206       <lib:RequestAuthnContext>
1207         ...
1208       </lib:RequestAuthnContext>
1209
1210     </sb-ext:CredentialsContext>
1211
1212   </S:Header>
1213
1214   <S:Body>
1215
1216   <!-- a fault in the body indicates that the WSP's policy requires
1217        different (perhaps "stronger") credentials than were originally
1218        provided in the request -->
1219
1220     <S:Fault>
1221       <faultcode>S:Server</faultcode>
1222       <faultstring>
1223         Your request contained inappropriate credentials.
1224       </faultstring>
1225       <detail namespace="urn:liberty:sb:2003-08">
1226         <Status code="sb-ext:InappropriateCredentials"/>
1227         <wsse:Security id="a6352...564"/>
1228       </detail>
1229     </S:Fault>
1230   </S:Body>
1231
1232 </S:Envelope>
1233
1234

```

1235 **Example 10. A CredentialsContext Header Offered in Response to a Request with Inappropriate Credentials.**

1236 6.3.4. Processing Rules

1237 A sender including this header MUST specify at least one RequestAuthnContext or one SecurityMechID.

1238 The SecurityMechID elements SHOULD be listed in order of preference by the sender. The receiver SHOULD use
1239 the highest-listed SecurityMechID that it supports in future requests to the sender of this header.

1240 6.4. The <ServiceInstanceUpdate> Header Block

1241 6.4.1. Overview

1242 It may be necessary for an entity receiving an ID-* message to indicate that messages from the sender should be
1243 directed to a different endpoint, or that they wish a different credential to be used than was originally specified by the

1244 entity for access to the requested resource. The <ServiceInstanceUpdate> allows a message receiver to indicate
1245 that a new SOAP endpoint, new credentials, or new security mechanisms should be employed by the sender of the
1246 message. This header block may be used in conjunction with the <sb-ext:InappropriateCredentials> and
1247 <sb-ext:EndpointMoved> faults, to indicate that the current message processing failed for those reasons, and
1248 should be submitted with the changes noted in any accompanying <ServiceInstanceUpdate> header block.

1249 **Note:**

1250 The use of this header block allows the sender of the message to convey updates to security tokens, essentially
1251 providing a token renewal mechanism. This is not discussed further in this specification.

1252 6.4.2. ServiceInstanceUpdate Type and Element

1253 Receivers of an ID-* message may add a <ServiceInstanceUpdate> element to the SOAP header of their response.

1254 This element is based upon the ServiceInstanceUpdateType which is defined as:

- 1255 • SecurityMechID [Optional] – a URI indicating a new security mechanism that should be used in place of those
1256 currently employed in the sending of this message.
- 1257 • Credential [Optional] – a container for arbitrary credentials that are supplied by the message sender. This
1258 contains an optional notOnOrAfter attribute, used to indicate an expiration time for the supplied credential.
- 1259 • id [Optional] – An attribute facilitating references to elements of this type.
- 1260 • S:mustUnderstand [Optional] – The SOAP mustUnderstand attribute [\[SOAPv1.1\]](#).
- 1261 • S:actor [Optional] – The SOAP actor attribute [\[SOAPv1.1\]](#).

1262 The following schema fragment describes the <ServiceInstanceUpdate> header block.

```
1263 <xs:complexType name="ServiceInstanceUpdateType">
1264   <xs:sequence>
1265     <xs:element name="SecurityMechID" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
1266     <xs:element name="Credential" minOccurs="0" maxOccurs="unbounded">
1267       <xs:complexType>
1268         <xs:sequence>
1269           <xs:any namespace="##any" processContents="lax"/>
1270         </xs:sequence>
1271         <xs:attribute name="notOnOrAfter" type="xs:dateTime" use="optional"/>
1272       </xs:complexType>
1273     </xs:element>
1274     <xs:element name="Endpoint" type="xs:anyURI" minOccurs="0"/>
1275   </xs:sequence>
1276   <xs:attribute name="id" type="xs:ID" use="optional"/>
1277   <xs:attribute ref="S:mustUnderstand" use="optional"/>
1278   <xs:attribute ref="S:actor" use="optional"/>
1279 </xs:complexType>
1280
1281 <xs:element name="ServiceInstanceUpdate" type="ServiceInstanceUpdateType"/>
1282
1283
1284
```

1285 **Figure 13. The <ServiceInstanceUpdate> Header Block Element and Type Schema**

1286 6.4.3. ServiceInstanceUpdate Examples

```
1287
1288
1289
1290     1. Service responds to a request, indicating a new security mechanism and credential
1291
1292     <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
1293       xmlns:sb="urn:liberty:sb:2003-08"
1294       xmlns:idpp="urn:liberty:id-sis-pp:2003-08">
1295
1296       <S:Header>
1297         <sb:Correlation S:mustUnderstand="1"
1298           messageID="uuid:0023923-28329023-238239023"
1299           refToMessageID="uuid:00123-ad32-23-09923-88239"
1300           timestamp="2003-06-06T12:10:10Z" />
1301
1302         <sb-ext:ServiceInstanceUpdate mustUnderstand="1">
1303
1304           <sb-ext:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</sb-ext:SecurityMe
1305 chID>
1306
1307           <sb-ext:Credential notOnOrAfter="2003-06-06T15:10:10Z">
1308
1309             <wsse:BinarySecurityToken xmlns:wsse="..." wsu:Id="..."
1310 ValueType="anyNSprefix:ServiceSessionContext" >
1311               ZjgzOWZlNzgyZTk1ZWU3OWEyMTRlODVmNGZkYzE4MmQ2ZDNhMzc3Nwo=
1312             </wsse:BinarySecurityToken>
1313
1314           </sb-ext:Credential>
1315
1316         </sb-ext:ServiceInstanceUpdate>
1317
1318       </S:Header>
1319
1320       <S:Body>
1321
1322         <idpp:QueryResponse>
1323           ...
1324         </idpp:QueryResponse>
1325
1326       </S:Body>
1327
1328     </S:Envelope>
1329
```

```
1330
1331
1332      2. The client sends a new request, using the contents of the ServiceInstanceUpdate
1333
1334      <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
1335      xmlns:sb="urn:liberty:sb:2003-08 "
1336      xmlns:idpp="urn:liberty:id-sis-pp:2003-08">
1337
1338      <S:Header>
1339      <sb:Correlation S:mustUnderstand="1"
1340      messageID="uuid:3736251-21532023-774625253"
1341      refToMessageID="uuid:0023923-28329023-238239023"
1342      timestamp="2003-06-06T12:15:04Z" />
1343
1344      <wsse:Security xmlns:wsse="...">
1345
1346      <wsse:BinarySecurityToken xmlns:wsse="..." wsu:Id="bst"
1347      ValueType="anyNSprefix:ServiceSessionContext" >
1348      ZjgzOWZlNzgyZTk1ZWU3OWEYMTRlODVmNGZkYzE4MmQ2ZDZhMzc3Nwo=
1349      </wsse:BinarySecurityToken>
1350
1351      </wsse:Security>
1352
1353      </S:Header>
1354
1355      <S:Body>
1356
1357      <idpp:Query>
1358      ...
1359      </idpp:Query>
1360
1361      </S:Body>
1362
1363      </S:Envelope>
1364
1365
1366
```


1367 **Example 11. A ServiceInstanceUpdate Specifying a ServiceSessionContext Token, and the TLS Bearer Security**
 1368 **Mechanism.**

```

1369
1370
1371 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
1372   xmlns:sb="urn:liberty:sb:2003-08">
1373
1374   <S:Header>
1375     <sb:Correlation S:mustUnderstand="1"
1376       messageID="uuid:0023923-28329023-238239023"
1377       refToMessageID="uuid:00123-ad32-23-09923-88239"
1378       timestamp="2003-06-06T12:10:10Z" />
1379
1380     <sb-ext:ServiceInstanceUpdate mustUnderstand="1">
1381       <sb-ext:Endpoint>http://example.com:443/soap</sb-ext:Endpoint>
1382     </sb-ext:ServiceInstanceUpdate>
1383
1384   </S:Header>
1385
1386   <S:Body>
1387
1388     <!-- a fault in the body indicates that the request corresponding
1389     to this response should be re-submitted to the endpoint -->
1390
1391     <S:Fault>
1392
1393       <faultcode>S:Server</faultcode>
1394
1395       <faultstring>
1396         You must resubmit this request to the new endpoint.
1397       </faultstring>
1398
1399       <detail namespace="urn:liberty:sb:2003-08">
1400         <Status code="sb-ext:EndpointMoved"/>
1401       </detail>
1402
1403     </S:Fault>
1404
1405   </S:Body>
1406
1407 </S:Envelope>
1408
1409
  
```

1410 **Example 12. A ServiceInstanceUpdate Specifying an Updated Endpoint.**

1411 **6.4.4. Processing Rules**

1412 **6.4.4.1. Processing Rules for the ServiceInstanceUpdate header**

1413 The receiver of an ID-* message MAY add a <ServiceInstanceUpdate> header block to their response.

1414 The sender of this header block MUST include at least one of the optional elements contained in the header block
 1415 (<Endpoint>, <SecurityMechID>, or <Credential>).

1416 Any <Endpoint> specified SHOULD be set to a complete URI.

1417 This header block may contain one or more <SecurityMechID> elements. In this case, the service is stating that
 1418 the security mechanisms denoted by these elements SHOULD be used in preference to any that had previously been
 1419 specified (for example, upon registration of the service at a discovery service [[LibertyDisco](#)]). Any mechanisms

1420 supplied SHOULD be listed in order of preference by the service. <SecurityMechID> elements MUST be specified
1421 according to [\[LibertySecMech\]](#).

1422 If the message sender includes one or more <SecurityMechID> elements in their response, that were not previously
1423 specified (for example during registration at a discovery service), then they MUST supply appropriate credentials for
1424 the message recipient to use in accordance with the supplied mechanisms. This implies, for example, that if a bearer
1425 (see [\[LibertySecMech\]](#)) mechanism is supplied in this header block, but was not previously specified for the requester,
1426 that the sender should supply a bearer token that the requester may use with this mechanism.

1427 The <Credential> element of the header block may contain a notOnOrAfter attribute, which the sender SHOULD
1428 use to indicate the latest time at which the supplied credential is valid, unless the credential being supplied contains
1429 some other indication of such a value.

1430 The entity constructing a <Credential> element SHOULD specify a valid credential for use by the recipient. For
1431 example, if the token constructed is of type wsse:BinarySecurityToken then it should conform to the rules
1432 specified in [\[LibertySecMech\]](#) and supporting specifications.

1433 **6.4.4.2. Processing Rules for the EndpointMoved SOAP Fault**

1434 The receiver of an ID-* message MAY issue a SOAP Fault indicating that the endpoint to which this message was
1435 submitted has permanently changed.

1436 Once the receiver has sent this fault response, no further processing of the message should take place.

1437 If the receiver chooses to send the fault response, then it SHOULD also include a <ServiceInstanceUpdate>
1438 header, indicating the new endpoint which should be used to re-submit this message, and any further messages directed
1439 to the responding service.

1440 If the receiver of this fault response also received a <ServiceInstanceUpdate> header in the response, it MAY
1441 re-submit the failed request to any endpoint specified in that header, but it SHOULD provide a different messageID
1442 in the <Correlation> header block of the request.

1443 **6.5. The <Timeout> Header Block**

1444 **6.5.1. Overview**

1445 A requesting entity may wish to indicate that they would like a request to be processed within some specified amount
1446 of time. Such an entity would indicate their wish via the <Timeout> header block.

1447 **6.5.2. Timeout Type and Element**

1448 Senders of ID-* messages MAY add a <Timeout> element to the SOAP header of their request.

1449 This element is based upon the TimeoutType which is defined as:

1450 • maxProcessingTime [Required] – an integer specifying (in seconds) the maximum amount of time the requester
1451 wishes the receiver to spend in processing their request

1452 • id [Optional] – An attribute facilitating references to elements of this type.

1453 • S:mustUnderstand [Optional] – The SOAP mustUnderstand attribute [\[SOAPv1.1\]](#).

1454 • S:actor [Optional] – The SOAP actor attribute [\[SOAPv1.1\]](#).

1455 The following schema fragment describes the <Timeout> header block.

```
1456 <xs:complexType name="TimeoutType">
1457   <xs:attribute name="maxProcessingTime" type="xs:integer" use="required"/>
1458   <xs:attribute name="id" type="xs:ID" use="optional"/>
1459   <xs:attribute ref="S:mustUnderstand" use="optional"/>
1460   <xs:attribute ref="S:actor" use="optional"/>
1461 </xs:complexType>
1462
1463 <xs:element name="Timeout" type="TimeoutType"/>
1464
1465
1466
```

1467 **Figure 14. The <Timeout> Header Block Element and Type Schema**

1468 6.5.3. Timeout Example

```
1469
1470
1471 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
1472   xmlns:sb="urn:liberty:sb:2003-08"
1473   xmlns:idpp="urn:liberty:id-sis-pp:2003-08">
1474
1475   <S:Header>
1476
1477     <sb-ext:Timeout mustUnderstand="1" id="timeout.123"
1478       maxProcessingTime="7"/>
1479
1480   </S:Header>
1481
1482   <S:Body>
1483
1484     <idpp:Query>
1485       ...
1486     </idpp:Query>
1487
1488   </S:Body>
1489
1490 </S:Envelope>
1491
1492
```

1493 **Example 13. Example of a Request with Timeout Specified**

```
1494
1495
1496 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
1497   xmlns:sb="urn:liberty:sb:2003-08">
1498
1499   <S:Header>
1500
1501     ...
1502
1503   </S:Header>
1504
1505   <S:Body>
1506
1507     <S:Fault>
1508
1509       <S:faultcode>
1510         S:Server
1511       </S:faultcode>
1512
1513       <S:detail>
1514
1515         <sb:Status code="sb-ext:ProcessingTimeout"/>
1516
1517         <!-- Reference the specified Timeout header, if it was supplied
1518            by the requester -->
1519
1520         <sb-ext:Timeout id="timeout.123"/>
1521
1522       </S:Fault>
1523
1524     </S:Body>
1525
1526   </S:Envelope>
1527
1528
```

1529 **Example 14. Example of a Timed-out Response**

1530 **6.5.4. Processing Rules**

1531 The receiver of a Timeout header SHOULD NOT begin processing of a message (beyond processing of the SOAP
1532 headers as noted in this specification) if it expects that such processing would exceed the value specified in the
1533 maxProcessingTime attribute.

1534 The receiver MUST respond to the message within the number of seconds specified in the maxProcessingTime
1535 attribute.

1536 If the receiver is unable to complete processing within the number of seconds specified in the maxProcessingTime
1537 attribute of the Timeout header, then they SHOULD respond with a SOAP Fault with a code of
1538 sb-ext:ProcessingTimeout.

1539 **Note:**

1540 If the sender of a message does not include a Timeout header, but the receiver wishes to indicate to the sender
1541 that server processing failed due to a timeout, then the receiver MAY respond with a SOAP Fault with a
1542 code of sb-ext:ProcessingTimeout.

1543 **6.6. The <UsageDirective> Header Block**

1544 This section defines the ID-* usage directive facilities.

1545 **6.6.1. Overview**

1546 Participants in the ID-WSF framework may need to indicate the privacy policy associated with a message. To facilitate
1547 this, senders, acting as either a client or a server, may add one or more <UsageDirective> header blocks to the
1548 SOAP Header of the message being sent. A <UsageDirective> appearing in a SOAP-based ID-* request message
1549 expresses *intended usage*. A <UsageDirective> appearing in a response expresses *how* the receiver of the response
1550 is to use the response data. A <UsageDirective> in a response message containing no ID-WSF response message
1551 data, a fault response for example, may be used to express policies acceptable to the responder.

1552 **6.6.2. UsageDirective Header Type and Element**

1553 Senders MAY add a <UsageDirective> element to the SOAP header. This element is based upon the
1554 UsageDirectiveType which is defined as:

- 1555 • `ref` [Required] – An attribute referring to an element of the SOAP-based ID-* message to which the usage directive
1556 applies.
- 1557 • `id` [Optional] – An attribute facilitating references to elements of this type.
- 1558 • `S:mustUnderstand` [Optional] – The SOAP mustUnderstand attribute [SOAPv1.1].
- 1559 • `S:actor` [Optional] – The SOAP actor attribute [SOAPv1.1].
- 1560 • `<element>(s)` [Optional] – Elements, comprising an instance of some policy expression language, whose
1561 purpose is to express the actual policy the usage directive is conveying. The `ref` attribute above points at the
1562 element in the overall SOAP-based ID-* message to which the usage directive applies.

1563 The schema fragments in [Figure 15](#) and [Figure 16](#) defines the <UsageDirective> header type and element.

```
1564
1565
1566     <element name="UsageDirective" type="UsageDirectiveType" />
1567
1568
```

1569 **Figure 15. The <UsageDirective> Header Block Element Schema**

```
1570
1571
1572     <complexType name="UsageDirectiveType">
1573         <sequence>
1574             <xs:any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
1575         </sequence>
1576         <attribute name="ref" type="reference" use="required"/>
1577         <attribute name="id" type="id" use="optional"/>
1578         <attribute ref="S:mustUnderstand" use="optional"/>
1579         <attribute ref="S:actor" use="optional"/>
1580     </complexType>
1581
1582
```

1583 **Figure 16. The UsageDirective Header Block Type Schema**

1584 **6.6.3. Usage Directive Examples**

1585 **Example 15** illustrates a SOAP-based ID-* message, containing a <UsageDirective> header block, and
1586 conveying a Personal Profile (ID-PP) Modify message [LibertyIDPP]. The <UsageDirective> header block
1587 contains a usage directive expressed in a policy language identified by the cot: namespace and the URI
1588 <http://circle-of-trust.com/policies/eu-compliant>, and applying to the ID-PP Query message
1589 identified by the id of datarequest001.

```
1590
1591
1592     <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
1593       xmlns:sb="urn:liberty:sb:2003-08"
1594       xmlns:pp="urn:liberty:id-sis-pp:2003-08">
1595
1596     <S:Header>
1597
1598       <!-- other header blocks, eg wsse:security, may go here -->
1599
1600       <UsageDirective
1601         id="directive1000"
1602         ref="#datarequest001"
1603         S:mustUnderstand="1">
1604
1605         <cot:PrivacyPolicyReference
1606           xmlns:cot="http://circle-of-trust.com/isf">
1607           http://circle-of-trust.com/policies/eu-compliant
1608         </cot:PrivacyPolicyReference>
1609
1610       </UsageDirective>
1611
1612       <Correlation S:mustUnderstand="1"
1613         id="A13454...245"
1614         S:actor="http://schemas.../next"
1615         messageID="uuid:efefefef-eeee-ffff-cccc-fffffffbbbb"
1616         timestamp="2112-03-15T11:12:14Z"/>
1617
1618       <!-- other header blocks, eg wsse:security, may go here -->
1619
1620     </S:Header>
1621
1622     <S:Body>
1623
1624       <pp:Query id="datarequest001" xmlns:pp="urn:liberty:id-sis-pp:2003-08">
1625         <pp:ResourceID>data:d8ddw6dd7m28v628</pp:ResourceID>
1626         <pp:QueryItem>
1627           <pp:Select>/pp:IDPP/pp:IDPPAddressCard</pp:Select>
1628         </pp:QueryItem>
1629       </pp:Query>
1630
1631     </S:Body>
1632
1633   </S:Envelope>
1634
1635
```

1636 **Example 15. A Usage Directive on a Request for the Address of a Principal.**

1637 6.6.4. Processing Rules

1638 The sender of a SOAP-based ID-* message with a <UsageDirective> header block MUST ensure that the value
1639 of the ref attribute is set to the value of the id of the appropriate element in the message. The sender SHOULD
1640 ensure that the <UsageDirective> is integrity-protected. The protection mechanism, if utilized, SHOULD be in
1641 accordance with those defined in [LibertySecMech].

- 1642 A receiver of a SOAP-based ID-* message with an attached `<UsageDirective>` header block MUST check the
1643 actor attribute and determine if it, the receiver, is the actor the header block is targeted at. If so, the receiver MUST
1644 check the `mustUnderstand` attribute. If set to `TRUE` the receiver MUST process the contents. If the attribute is absent
1645 or set to `FALSE` the receiver SHOULD attempt to process the content of the `<UsageDirective>` header block.
- 1646 A receiver that processes the contents of a `<UsageDirective>` header block SHOULD verify the integrity of the
1647 header block – that is, it should verify any digital signatures that list the header block in its manifest [XMLDsig]. The
1648 receiver MUST verify that the `ref` attribute refers to an element in the message. That receiver MUST further process
1649 the message according to the policy expressed by the children elements of the `<UsageDirective>` header block.
1650 Those children elements will be imported from a foreign namespace, and MUST be parsed and interpreted according
1651 to the applicable schema and processing rules of that foreign namespace.
- 1652 A receiver that cannot process a `<UsageDirective>` with `mustUnderstand="TRUE"` MUST respond with a
1653 `<S:Fault>`. The `s:Fault` MUST contain a `<S:Detail>` element which in turn MUST contain a `<Status>` element
1654 with its `<StatusCode>` set to `sb:CannotHonourUsageDirective`. The `<Status>` element SHOULD possess a
1655 `ref` attribute with its value set to the value of the `id` attribute of the offending `<UsageDirective>` header block in
1656 the request message.
- 1657 A receiver that cannot honour a non-mandatory (with `mustUnderstand="FALSE"`) `<UsageDirective>` must
1658 respond according to the contained policy. In addition, in this case the receiver MAY respond with a SOAP-based ID-
1659 * message that includes a `<Status>` element with its `<StatusCode>` set to `sb:CannotHonourUsageDirective`.
1660 This `<Status>` element instance SHOULD include a `ref` attribute with its value set to the value of the `id` attribute of
1661 the `<UsageDirective>` header block in the request message that could not be honoured.
- 1662 In this case, the receiver MAY include one or more new `<UsageDirective>` header blocks in its response message,
1663 each expressing a policy that the receiver would have been able to honour. The `ref` attribute of these headers SHOULD
1664 be set to the `messageID` of the `<Correlation>` element.

1665 7. Security Considerations

- 1666 • The header blocks specified in this document should be integrity-protected using the mechanisms detailed in
1667 [\[LibertySecMech\]](#).
- 1668 • Header blocks should be signed in accordance with [\[LibertySecMech\]](#). The receiver of a message containing a
1669 signature that covers specific header blocks should verify the signature as part of verifying the integrity of the
1670 header block.
- 1671 • Metadata [\[LibertyMetadata\]](#) should be used to the greatest extent possible to verify message sender identity claims.
- 1672 • Message senders and receivers should be authenticated to one another via the mechanisms discussed in [\[Liberty-](#)
1673 [SecMech\]](#).
- 1674 • To prevent message replay, receivers should maintain a message cache, and check received `messageID` values
1675 against the cache.

1676 **8. Acknowledgements**

1677 The members of the Liberty Technical Expert group, especially Greg Whitehead, Jonathan Sergent, Xavier Serret,
1678 and Conor Cahill, provided valuable input to this specification. The docbook source code for this specification was
1679 hand set to the tunes of The Sugarcubes, King Crimson, Juliana Hatfield, Smashing Pumpkins, Evanescence, Mad at
1680 Gravity, Elisa Korenne, The Breeders, fIREHOSE, Polly Jean Harvey, Jimi Hendrix, and various others.

Bibliography

1681

Normative

1682

1683 [LibertyDST] "Liberty ID-WSF Data Services Template Specification," Version 1.1, Liberty Alliance Project (14
1684 December 2004). <http://www.projectliberty.org/specs> Kainulainen, Jukka, Ranganathan, Aravindan, eds.

1685 [LibertyIDFFOverview] Wason, Thomas, eds. "Liberty ID-FF Architecture Overview," Version 1.2-errata-v1.0,
1686 Liberty Alliance Project (12 September 2004). <http://www.projectliberty.org/specs>

1687 [LibertyInteract] Aarts, Robert, eds. "Liberty ID-WSF Interaction Service Specification," Version 1.1, Liberty Alliance
1688 Project (14 December 2004). <http://www.projectliberty.org/specs>

1689 [LibertyMetadata] Davis, Peter, eds. "Liberty Metadata Description and Discovery Specification," Version 1.1, Liberty
1690 Alliance Project (14 December 2004). <http://www.projectliberty.org/specs>

1691 [LibertyAuthnContext] Madsen, Paul, eds. "Liberty ID-FF Authentication Context Specification," Version 1.3, Liberty
1692 Alliance Project (14 December 2004). <http://www.projectliberty.org/specs>

1693 [LibertySecMech] Ellison, Gary, eds. "Liberty ID-WSF Security Mechanisms," Version 1.2, Liberty Alliance Project
1694 (14 December 2004). <http://www.projectliberty.org/specs>

1695 [RFC2045] Freed, N., Borenstein, N., eds. (November 1996). "Multipurpose Internet Mail Extensions
1696 (MIME) Part One: Format of Internet Message Bodies ," RFC 2045., Internet Engineering Task
1697 Force <http://www.ietf.org/rfc/rfc2045.txt> [November 1996].

1698 [RFC2119] Bradner, S., eds. "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet
1699 Engineering Task Force (March 1997). <http://www.ietf.org/rfc/rfc2119.txt> [March 1997].

1700 [RFC2396] Berners-Lee, T., Fielding, R., Masinter, L., eds. (August 1998). "Uniform Resource Identifiers (URI):
1701 Generic Syntax," RFC 2396, The Internet Engineering Task Force <http://www.ietf.org/rfc/rfc2396.txt>
1702 [August 1998].

1703 [RFC2828] Shirey, R., eds. (May 2000). "Internet Security Glossary," RFC 2828., Internet Engineering Task Force
1704 <http://www.ietf.org/rfc/rfc2828.txt> [May 2000].

1705 [SAMLGloss] Hodges, Jeff, Maler, Eve, eds. (05 November 2002). "Glossary for the OASIS Security Assertion
1706 Markup Language (SAML)," Version 1.0, OASIS Standard, Organization for the Advancement of Structured
1707 Information Standards <http://www.oasis-open.org/committees/security/#documents>

1708 [Schema1] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (May
1709 2002). "XML Schema Part 1: Structures," Recommendation, World Wide Web Consortium
1710 <http://www.w3.org/TR/xmlschema-1/>

1711 [Schema2] Biron, Paul V., Malhotra, Ashok, eds. (May 2002). "XML Schema Part 2: Datatypes," Recommendation,
1712 World Wide Web Consortium <http://www.w3.org/TR/xmlschema-2/>

1713 [SOAPv1.1] "Simple Object Access Protocol (SOAP) 1.1," Box, Don, Ehnebuske, David , Kakivaya, Gopal, Layman,
1714 Andrew, Mendelsohn, Noah, Nielsen, Henrik Frystyk, Winer, Dave, eds. World Wide Web Consortium W3C
1715 Note (08 May 2000). <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

1716 [SOAPv1.1-Schema] "SOAP 1.1 Envelope schema," W3C W3C Note <http://schemas.xmlsoap.org/soap/envelope/>

1717 [WSDLv1.1] "Web Services Description Language (WSDL) 1.1," Christensen, Erik, Curbera, Francisco, Meredith,
1718 Greg, Weerawarana, Sanjiva, eds. World Wide Web Consortium W3C Note (15 March 2001).
1719 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>

- 1720 [XML] Bray, Tim, Paoli, Jean, Sperberg-McQueen, C.M., Maler, Eve, eds. (Oct 2000). "Extensible
1721 Markup Language (XML) 1.0 (Second Edition)," Recommendation, World Wide Web Consortium
1722 <http://www.w3.org/TR/2000/REC-xml-20001006>
- 1723 [XMLDsig] Eastlake, Donald, Reagle, Joseph, Solo, David, eds. (12 Feb 2002). "XML-Signature Syntax and
1724 Processing," Recommendation, World Wide Web Consortium <http://www.w3.org/TR/xmlsig-core>
- 1725 **Informational**
- 1726 [LibertyDisco] Sergeant, Jonathan, eds. "Liberty ID-WSF Discovery Service Specification," Version 1.2, Liberty
1727 Alliance Project (12 December 2004). <http://www.projectliberty.org/specs>
- 1728 [LibertyGlossary] "Liberty Technical Glossary," Version 1.4, Liberty Alliance Project (14 Dec 2004).
1729 <http://www.projectliberty.org/specs> Hodges, Jeff, eds.
- 1730 [LibertyIDWSFOverview] Tourzan, Jonathan, Koga, Yuzo, eds. "Liberty ID-WSF Web Services Framework
1731 Overview," Version 1.1, Liberty Alliance Project (14 December 2004). <http://www.projectliberty.org/specs>
- 1732 [LibertyIDPPP] Kellomaki, Sampo, eds. "Liberty Identity Personal Profile Service Specification," Version 1.0, Liberty
1733 Alliance Project (12 November 2003). <http://www.projectliberty.org/specs>
- 1734 [Merriam-Webster] "Merriam-Webster Dictionary," <http://www.merriam-webster.com/>
- 1735 [RFC1750] Eastlake , D., Crocker, S., Schiller, J., eds. (December 1994). "Randomness Recommendations for
1736 Security ," RFC 1750, Internet Engineering Task Force <http://www.ietf.org/rfc/rfc1750.txt> [August 2003].
- 1737 [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., eds. (June
1738 1999). "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, The Internet Engineering Task Force
1739 <http://www.ietf.org/rfc/rfc2616.txt> [June 1999].
- 1740 [SOAPv1.2] "SOAP Version 1.2 Part 1: Messaging Framework," Gudgin, Martin, Hadley, Marc, Mendelsohn,
1741 Noah, Moreau, Jean-Jacques, Nielsen, Henrik Frystyk, eds. World Wide Web Consortium W3C Proposed
1742 Recommendation (07 May 2003). <http://www.w3.org/TR/2003/PR-soap12-part1-20030507/>

1743 A. Liberty ID-WSF SOAP Binding Schema

```
1744 <?xml version="1.0" encoding="UTF-8"?>
1745 <xs:schema targetNamespace="urn:liberty:sb:2003-08"
1746   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
1747   xmlns:sb="urn:liberty:sb:2003-08"
1748   xmlns:xs="http://www.w3.org/2001/XMLSchema"
1749   xmlns="urn:liberty:sb:2003-08"
1750   elementFormDefault="qualified"
1751   attributeFormDefault="unqualified"
1752   version="1.0-02">
1753
1754   <!-- Filename: lib-arch-soap-binding.xsd -->
1755   <!-- $Id: lib-arch-soap-binding.xsd,v 1.8.4.1 2005/01/25 18:39:11 dchampagne Exp $ -->
1756   <!-- Author: Jeff Hodges -->
1757   <!-- Last editor: $Author: dchampagne $ -->
1758   <!-- $Date: 2005/01/25 18:39:11 $ -->
1759   <!-- $Revision: 1.8.4.1 $ -->
1760
1761   <xs:import
1762     namespace="http://schemas.xmlsoap.org/soap/envelope/"
1763     schemaLocation="http://schemas.xmlsoap.org/soap/envelope/" />
1764
1765   <xs:include schemaLocation="liberty-idwsf-utility-v1.1.xsd" />
1766
1767   <xs:annotation>
1768     <xs:documentation>
1769       Liberty ID-WSF SOAP Binding Specification XSD
1770     </xs:documentation>
1771     <xs:documentation>
1772
1773 The source code in this XSD file was excerpted verbatim from:
1774
1775 Liberty ID-WSF SOAP Binding Specification
1776 Version 1.0
1777 12th November 2003
1778
1779 Copyright (c) 2003-2005 Liberty Alliance participants, see
1780 http://www.projectliberty.org/specs/idwsf\_copyrights.html
1781
1782     </xs:documentation>
1783   </xs:annotation>
1784
1785   <!-- message correlation header block -->
1786
1787   <xs:complexType name="CorrelationType">
1788     <xs:attribute name="messageID" type="IDType" use="required"/>
1789     <xs:attribute name="refToMessageID" type="IDType" use="optional"/>
1790     <xs:attribute name="timestamp" type="xs:dateTime" use="required"/>
1791     <xs:attribute name="id" type="xs:ID" use="optional"/>
1792     <xs:attribute ref="S:mustUnderstand" use="optional"/>
1793     <xs:attribute ref="S:actor" use="optional"/>
1794   </xs:complexType>
1795
1796   <xs:element name="Correlation" type="CorrelationType"/>
1797
1798   <!-- provider= header block -->
1799
1800   <xs:complexType name="ProviderType">
1801     <xs:attribute name="providerID" type="xs:anyURI" use="required"/>
1802     <xs:attribute name="affiliationID" type="xs:anyURI" use="optional"/>
1803     <xs:attribute name="id" type="xs:ID" use="optional"/>
1804     <xs:attribute ref="S:mustUnderstand" use="optional"/>
1805     <xs:attribute ref="S:actor" use="optional"/>
1806   </xs:complexType>
1807
1808   <xs:element name="Provider" type="ProviderType"/>
```

```
1809
1810 <!-- processing context header block -->
1811
1812 <xs:complexType name="ProcessingContextType">
1813   <xs:simpleContent>
1814     <xs:extension base="xs:anyURI">
1815       <xs:attribute name="id" type="xs:ID" use="optional"/>
1816       <xs:attribute ref="S:mustUnderstand" use="optional"/>
1817       <xs:attribute ref="S:actor" use="optional"/>
1818     </xs:extension>
1819   </xs:simpleContent>
1820 </xs:complexType>
1821
1822 <xs:element name="ProcessingContext" type="ProcessingContextType"/>
1823
1824 <!-- consent claim header block -->
1825
1826 <xs:complexType name="ConsentType">
1827   <xs:attribute name="uri" type="xs:anyURI" use="required"/>
1828   <xs:attribute name="timestamp" type="xs:dateTime" use="optional"/>
1829   <xs:attribute name="id" type="xs:ID" use="optional"/>
1830   <xs:attribute ref="S:mustUnderstand" use="optional"/>
1831   <xs:attribute ref="S:actor" use="optional"/>
1832 </xs:complexType>
1833
1834 <xs:element name="Consent" type="ConsentType"/>
1835
1836 <!-- usage directive header block -->
1837
1838 <xs:complexType name="UsageDirectiveType">
1839   <xs:sequence>
1840     <xs:any namespace="##other" processContents="lax"
1841 maxOccurs="unbounded"/>
1842   </xs:sequence>
1843   <xs:attribute name="id" type="xs:ID" use="optional"/>
1844   <xs:attribute name="ref" type="xs:IDREF" use="required"/>
1845   <xs:attribute ref="S:mustUnderstand" use="optional"/>
1846   <xs:attribute ref="S:actor" use="optional"/>
1847 </xs:complexType>
1848
1849 <xs:element name="UsageDirective" type="UsageDirectiveType"/>
1850
1851 </xs:schema>
1852
1853
1854
```

1855 B. Liberty ID-WSF SOAP Binding Extension Schema (April 2004)

```
1856 <?xml version="1.0" encoding="UTF-8"?>
1857 <xs:schema targetNamespace="urn:liberty:sb:2004-04"
1858   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
1859   xmlns:sb-ext="urn:liberty:sb:2004-04"
1860   xmlns:lib="urn:liberty:iff:2003-08"
1861   xmlns:xs="http://www.w3.org/2001/XMLSchema"
1862   xmlns="urn:liberty:sb:2004-04"
1863   elementFormDefault="qualified"
1864   attributeFormDefault="unqualified">
1865
1866   <!-- Author: John Kemp -->
1867   <!-- Last editor: $Author: dchampagne $ -->
1868   <!-- $Date: 2005/01/25 18:39:11 $ -->
1869   <!-- $Revision: 1.3.2.1 $ -->
1870
1871   <xs:import
1872     namespace="http://schemas.xmlsoap.org/soap/envelope/"
1873     schemaLocation="http://schemas.xmlsoap.org/soap/envelope/" />
1874
1875   <xs:import
1876     namespace="urn:liberty:iff:2003-08"
1877     schemaLocation="liberty-idff-protocols-schema-1.2-errata-v3.0.xsd" />
1878
1879   <xs:include schemaLocation="liberty-idwsf-utility-v1.1.xsd" />
1880
1881   <xs:annotation>
1882     <xs:documentation>
1883       Liberty ID-WSF SOAP Binding Specification Extension XSD
1884     </xs:documentation>
1885     <xs:documentation>
1886       The source code in this XSD file was excerpted verbatim from:
1887
1888       Liberty ID-WSF SOAP Binding Specification
1889       Version 1.2
1890       December 2004
1891
1892       Copyright (c) 2004-2005 Liberty Alliance participants, see
1893       http://www.projectliberty.org/specs/idwsf_1_1_copyrights.php
1894     </xs:documentation>
1895   </xs:annotation>
1896
1897   <xs:complexType name="CredentialsContextType">
1898     <xs:sequence>
1899       <xs:element ref="lib:RequestAuthnContext" minOccurs="0" />
1900       <xs:element name="SecurityMechID" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded" />
1901     </xs:sequence>
1902     <xs:attribute name="id" type="xs:ID" use="optional" />
1903     <xs:attribute ref="S:mustUnderstand" use="optional" />
1904     <xs:attribute ref="S:actor" use="optional" />
1905   </xs:complexType>
1906
1907   <xs:element name="CredentialsContext" type="CredentialsContextType" />
1908
1909   <xs:complexType name="ServiceInstanceUpdateType">
1910     <xs:sequence>
1911       <xs:element name="SecurityMechID" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded" />
1912       <xs:element name="Credential" minOccurs="0" maxOccurs="unbounded">
1913         <xs:complexType>
1914           <xs:sequence>
1915             <xs:any namespace="##any" processContents="lax" />
1916           </xs:sequence>
1917           <xs:attribute name="notOnOrAfter" type="xs:dateTime" use="optional" />
1918         </xs:complexType>
1919       </xs:element>
1920     </xs:sequence>
```

```
1921     <xs:element name="Endpoint" type="xs:anyURI" minOccurs="0" />
1922   </xs:sequence>
1923   <xs:attribute name="id" type="xs:ID" use="optional" />
1924   <xs:attribute ref="S:mustUnderstand" use="optional" />
1925   <xs:attribute ref="S:actor" use="optional" />
1926 </xs:complexType>
1927
1928 <xs:element name="ServiceInstanceUpdate" type="ServiceInstanceUpdateType" />
1929
1930 <xs:complexType name="TimeoutType">
1931   <xs:attribute name="maxProcessingTime" type="xs:integer" use="required" />
1932   <xs:attribute name="id" type="xs:ID" use="optional" />
1933   <xs:attribute ref="S:mustUnderstand" use="optional" />
1934   <xs:attribute ref="S:actor" use="optional" />
1935 </xs:complexType>
1936
1937 <xs:element name="Timeout" type="TimeoutType" />
1938
1939 </xs:schema>
1940
1941
```

1942 C. Liberty ID-WSF Utility Schema Listing

```
1943 <?xml version="1.0" encoding="UTF-8"?>
1944 <!-- filename: liberty-idwsf-utility-v1.1.xsd -->
1945
1946 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1947   elementFormDefault="qualified" attributeFormDefault="unqualified">
1948   <xs:annotation>
1949     <xs:documentation>
1950       Liberty Alliance Project utility schema. A collection of common
1951       IDentity Web Services Framework (ID-WSF) elements and types.
1952       This schema is intended for use in ID-WSF schemas.
1953
1954       Copyright 2003-2005 Liberty Alliance Project, see
1955       http://www.projectliberty.org/specs/idwsf_1_1_copyrights.php
1956
1957       This file intended for inclusion, rather than importation,
1958       into other schemas.
1959
1960       This version: 2004-12-14
1961
1962     </xs:documentation>
1963   </xs:annotation>
1964   <xs:simpleType name="IDType">
1965     <xs:annotation>
1966       <xs:documentation>This type should be used to provided IDs to components that have IDs that
1967 may not be scoped within the local xml instance document. </xs:documentation>
1968     </xs:annotation>
1969     <xs:restriction base="xs:string"/>
1970   </xs:simpleType>
1971   <xs:simpleType name="IDReferenceType">
1972     <xs:annotation>
1973       <xs:documentation> This type can be used when referring to elements that are
1974       identified using an IDType </xs:documentation>
1975     </xs:annotation>
1976     <xs:restriction base="xs:string"/>
1977   </xs:simpleType>
1978   <xs:element name="Status" type="StatusType">
1979     <xs:annotation>
1980       <xs:documentation> A standard Status type</xs:documentation>
1981     </xs:annotation>
1982   </xs:element>
1983   <xs:complexType name="StatusType">
1984     <xs:annotation>
1985       <xs:documentation> A type that may be used for status codes. </xs:documentation>
1986     </xs:annotation>
1987     <xs:sequence>
1988       <xs:element ref="Status" minOccurs="0" maxOccurs="unbounded"/>
1989     </xs:sequence>
1990     <xs:attribute name="code" type="xs:QName" use="required"/>
1991     <xs:attribute name="ref" type="IDReferenceType" use="optional"/>
1992     <xs:attribute name="comment" type="xs:string" use="optional"/>
1993   </xs:complexType>
1994   <xs:complexType name="EmptyType">
1995     <xs:annotation>
1996       <xs:documentation> This type may be used to create an empty element </xs:documentation>
1997     </xs:annotation>
1998     <xs:complexContent>
1999       <xs:restriction base="xs:anyType"/>
2000     </xs:complexContent>
2001   </xs:complexType>
2002   <xs:element name="Extension" type="extensionType">
2003     <xs:annotation>
2004       <xs:documentation>An element that contains arbitrary content extensions from other
2005 namespaces</xs:documentation>
2006     </xs:annotation>
2007   </xs:element>
```



```
2008     <xs:complexType name="extensionType">
2009         <xs:annotation>
2010             <xs:documentation>A type for arbitrary content extensions from other_
2011 namespaces</xs:documentation>
2012         </xs:annotation>
2013         <xs:sequence>
2014             <xs:any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
2015         </xs:sequence>
2016     </xs:complexType>
2017 </xs:schema>
2018
2019
```

2020 D. SOAP Envelope Schema Listing

```
2021 <?xml version='1.0' encoding='UTF-8' ?>
2022
2023 <!-- Schema for the SOAP/1.1 envelope
2024
2025     This schema has been produced using W3C's SOAP Version 1.2 schema
2026     found at:
2027
2028     http://www.w3.org/2001/06/soap-envelope
2029
2030     Copyright 2001 Martin Gudgin, Developmentor.
2031
2032     Changes made are the following:
2033     - reverted namespace to http://schemas.xmlsoap.org/soap/envelope/
2034     - reverted mustUnderstand to only allow 0 and 1 as lexical values
2035     - made encodingStyle a global attribute 20020825
2036
2037     Further changes:
2038
2039     - removed default value from mustUnderstand attribute declaration - 20030314
2040
2041     Original copyright:
2042
2043     Copyright 2001 W3C (Massachusetts Institute of Technology,
2044     Institut National de Recherche en Informatique et en Automatique,
2045     Keio University). All Rights Reserved.
2046     http://www.w3.org/Consortium/Legal/
2047
2048     This document is governed by the W3C Software License [1] as
2049     described in the FAQ [2].
2050
2051     [1] http://www.w3.org/Consortium/Legal/copyright-software-19980720
2052     [2] http://www.w3.org/Consortium/Legal/IPR-FAQ-20000620.html#DTD
2053 -->
2054
2055
2056 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
2057     xmlns:tns="http://schemas.xmlsoap.org/soap/envelope/"
2058     targetNamespace="http://schemas.xmlsoap.org/soap/envelope/" >
2059
2060     <!-- Envelope, header and body -->
2061
2062     <xs:element name="Envelope" type="tns:Envelope" />
2063
2064     <xs:complexType name="Envelope" >
2065         <xs:sequence>
2066             <xs:element ref="tns:Header" minOccurs="0" />
2067             <xs:element ref="tns:Body" minOccurs="1" />
2068             <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax" />
2069         </xs:sequence>
2070         <xs:anyAttribute namespace="##other" processContents="lax" />
2071     </xs:complexType>
2072
2073
2074     <xs:element name="Header" type="tns:Header" />
2075
2076     <xs:complexType name="Header" >
2077         <xs:sequence>
2078             <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" processContents="lax" />
2079         </xs:sequence>
2080         <xs:anyAttribute namespace="##other" processContents="lax" />
2081     </xs:complexType>
2082
2083
2084     <xs:element name="Body" type="tns:Body" />
2085
```

```

2086
2087 <xs:complexType name="Body" >
2088 <xs:sequence>
2089 <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded" processContents="lax" />
2090 </xs:sequence>
2091 <xs:anyAttribute namespace="##any" processContents="lax" >
2092 <xs:annotation>
2093 <xs:documentation>
2094     Prose in the spec does not specify that attributes are allowed on the Body element
2095 </xs:documentation>
2096 </xs:annotation>
2097 </xs:anyAttribute>
2098 </xs:complexType>
2099
2100
2101 <!-- Global Attributes. The following attributes are intended to be usable via qualified_
2102 attribute names on any complex type referencing them. -->
2103
2104 <xs:attribute name="mustUnderstand" >
2105 <xs:simpleType>
2106 <xs:restriction base='xs:boolean'>
2107 <xs:pattern value='0|1' />
2108 </xs:restriction>
2109 </xs:simpleType>
2110 </xs:attribute>
2111
2112 <xs:attribute name="actor" type="xs:anyURI" />
2113
2114 <xs:simpleType name="encodingStyle" >
2115 <xs:annotation>
2116 <xs:documentation>
2117     'encodingStyle' indicates any canonicalization conventions followed in the contents of f_
2118 the containing element. For example, the value 'http://schemas.xmlsoap.org/soap/encoding/'_
2119 indicates the pattern described in SOAP specification
2120 </xs:documentation>
2121 </xs:annotation>
2122 <xs:list itemType="xs:anyURI" />
2123 </xs:simpleType>
2124
2125 <xs:attribute name="encodingStyle" type="tns:encodingStyle" />
2126
2127
2128 <xs:attributeGroup name="encodingStyle" >
2129 <xs:attribute ref="tns:encodingStyle" />
2130 </xs:attributeGroup>
2131
2132 <xs:element name="Fault" type="tns:Fault" />
2133
2134 <xs:complexType name="Fault" final="extension" >
2135 <xs:annotation>
2136 <xs:documentation>
2137     Fault reporting structure
2138 </xs:documentation>
2139 </xs:annotation>
2140 <xs:sequence>
2141 <xs:element name="faultcode" type="xs:QName" />
2142 <xs:element name="faultstring" type="xs:string" />
2143 <xs:element name="faultactor" type="xs:anyURI" minOccurs="0" />
2144 <xs:element name="detail" type="tns:detail" minOccurs="0" />
2145 </xs:sequence>
2146 </xs:complexType>
2147
2148 <xs:complexType name="detail">
2149 <xs:sequence>
2150 <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded" processContents="lax" />
2151 </xs:sequence>
2152

```

```
2153         <xs:anyAttribute namespace="##any" processContents="lax" />
2154     </xs:complexType>
2155
2156 </xs:schema>
2157
2158
2159
2160
2161
2162
2163
2164
```