



Liberty Reverse HTTP Binding for SOAP Specification

Version: v1.1

Editors:

Robert Aarts, Nokia Corporation

Abstract:

SOAP is a lightweight protocol for the exchange of information in a decentralized, distributed environment. SOAP enables exchange of SOAP messages using a variety of underlying protocols. The formal set of rules for carrying a SOAP message within or on top of another protocol (underlying protocol) for the purpose of exchange is called a binding. Here a binding is specified that enables **HTTP** clients to expose services using the SOAP protocol. The primary difference from the normal **HTTP binding for SOAP** is that here a SOAP *request* is bound to a HTTP *response* and vice versa. Hence the name "Reversed HTTP binding for SOAP".

Filename: liberty-paos-v1.1.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. Overview	4
33	2. Optionality	5
34	3. Use of HTTP	6
35	4. HTTP Media Type	7
36	5. Binding Name	8
37	6. Indication of Binding Support	9
38	7. Supported Message Exchange Patterns	10
39	8. Operation of the Request-Response Message Exchange Pattern	11
40	9. Operation of the Response Message Exchange Pattern	14
41	10. PAOS Request Header Type	15
42	11. PAOS Response Header Type	17
43	12. Security Considerations	18
44	13. XML Schema for PAOS	19
45	References	20
46	A. Request for MIME Media Type Application/Vendor Tree - vnd.paos+xml	21

47 1. Overview

48 A large and growing number of devices, such as mobile terminals, personal computers, and appliances are nowadays
49 equipped with HTTP clients. At the same time most of these devices do not operate a HTTP server because of memory
50 limitations, power limitations, or because these devices are not generally addressable or reachable from the Internet.
51 Yet in many cases these devices could offer valuable services to other parties. For example, a mobile terminal could
52 host a [profile service](#), or a personal computer could host a calendar service. Such services could be especially valuable
53 when such devices interact with an HTTP-based server (or service). When a user of a mobile terminal visits a web
54 site, that web site could use some of the data from a personal profile service to personalize the offered content.

55 [SOAP](#) is a lightweight protocol for the exchange of information in a decentralized, distributed environment. SOAP
56 enables exchange of SOAP messages using a variety of underlying protocols. The formal set of rules for carrying
57 a SOAP message within or on top of another protocol (underlying protocol) for the purpose of exchange is called a
58 binding. This document specifies a binding that enables [HTTP](#) clients (user agents) to expose services using the SOAP
59 protocol.

60 The primary difference from the normal [HTTP binding for SOAP](#) is that here a SOAP *request* is bound to a HTTP
61 *response* and vice versa. Hence the name "Reversed HTTP binding for SOAP". The (informal) abbreviation for this
62 binding specification is "PAOS".

63 **Note:**

64 Although this specification normatively refers to [\[SOAPv1.1\]](#), including the SOAP 1.1 HTTP binding, every
65 attempt has been made to be as compatible with [\[SOAPv1.2part2\]](#) as possible. To this end the terminology in
66 this specification is primarily derived from the [SOAP 1.2 specification](#).

67 **2. Optionality**

68 PAOS is optional and SOAP nodes are NOT required to implement it. A SOAP node that correctly and completely
69 implements PAOS may be said to "conform to the Reversed HTTP Binding for SOAP." A SOAP node that conforms
70 to PAOS MAY support other bindings, but is not required to.

71 **3. Use of HTTP**

72 This binding of SOAP to HTTP is intended to make appropriate use of HTTP as an application protocol. This
73 binding is not intended to fully exploit the features of HTTP, but rather to use HTTP specifically for the purpose
74 of communicating with other SOAP nodes implementing the same binding. Therefore, this HTTP binding for SOAP
75 does not specify the use and/or meaning of all possible HTTP methods, header fields and status responses. It specifies
76 only those which are pertinent to this binding, or those which are likely to be introduced by HTTP mechanisms (such
77 as proxies) acting between the SOAP nodes.

78 4. HTTP Media Type

79 Conforming applications of this binding:

- 80 1. MUST be capable of sending and receiving messages serialized using media type "application/vnd.paos+xml"
81 whose proper use and parameters are described in [Appendix A](#)
- 82 2. MAY, when sending requests, provide an [HTTP](#) Accept header field. This header field:
- 83 a. SHOULD indicate an ability to accept at minimum "application/vnd.paos+xml"
- 84 b. MAY additionally indicate willingness to accept other media types

85 **5. Binding Name**

86 This binding is specified by the URN:

87 "urn:liberty:paos:2003-08"

88 6. Indication of Binding Support

89 HTTP user agents that are ready to receive SOAP messages in HTTP responses SHOULD add a "PAOS" HTTP header
90 to the HTTP request. The value of this header informs the HTTP server about the SOAP service(s) available at the
91 user agent. The header MUST be named PAOS and is defined, using Augmented BNF as specified in section 2 of
92 [\[RFC2616\]](#), as:

```
93  
94  
95     PAOS = "PAOS" ":" PAOS_Version [ "," Extension ] * ( ";" Service [ "," #Option ] )  
96     PAOS_Version = "ver" "=" 1#quotedURI  
97     Extension = "ext" "=" 1#quotedURI  
98     Service = quotedURI  
99     Option = quotedURI  
100     quotedURI = <">anyURI<">  
101  
102
```

103 The comment, field-value, and product productions are defined in [\[RFC2616\]](#). PAOS_Version identifies the versions
104 of the PAOS specification that are supported by this User-Agent. The versions are identified by a URI ([\[RFC2396\]](#)).
105 HTTP Servers receiving a PAOS header MUST ignore any URIs listed in the PAOS_Version production that they
106 do not recognize. All User-Agents compliant with this specification MUST send out, at a minimum, the URI
107 `urn:liberty:paos:2003-08` as a value in the PAOS_Version production. The ordering of the URIs in the
108 PAOS_Version header is meaningful; therefore, HTTP servers are encouraged to use the first version in the list that
109 they support. Supported versions are not negotiated between the User-Agent and HTTP server. The User-Agent simply
110 advertises what version it does support.

111 Optional extensions MAY be added to the PAOS header to indicate new information.

112 Each optional Service field is a URI that refers to a service description in (abstract) WSDL. The URI may be
113 a registered URN associated with a standard service, or an absolute URI that can be resolved to a particular
114 `<wsdl:Service>` element in a WSDL document (this may require the use of id attributes on such `<wsdl:Service>`
115 elements). A User-Agent that supports PAOS SHOULD add at least one Service to the PAOS header.

116 For each Service one or more Option fields can be added; these are intended to further describe the capabilities of the
117 advertised service.

118 **Note:**

119 A Service field corresponds to a `<ResourceOffering>` as defined in the [ID-WSF Discovery Service](#)
120 specification. This correspondence is as follows:

121 • The `<Resource>` element would always be `urn:liberty:isf:implied-resource` and hence is not
122 needed in the PAOS HTTP header.

123 • The `<ServiceType>` element of the `<ServiceInstance>` element corresponds to the primary value
124 (URI) of the Service field. This URI refers to *abstract WSDL*; for PAOS this is sufficient, as there is no
125 need to provide further information to the service about the binding or endpoint. Hence there is no need in the
126 PAOS header for an equivalent for the `<Description>` element.

127 • The `<Option>` elements of the `<ResourceOffering>` element corresponds to the values (URIs) of the
128 Option fields in the PAOS HTTP header.

129 • The URI values of `<SecurityMechID>` elements of the `<ResourceOffering>` element can be given
130 as PAOS Extensions (if the security mechanism(s) can be used for all PAOS exposed services), or as
131 Option(s) for a particular Service.

132 7. Supported Message Exchange Patterns

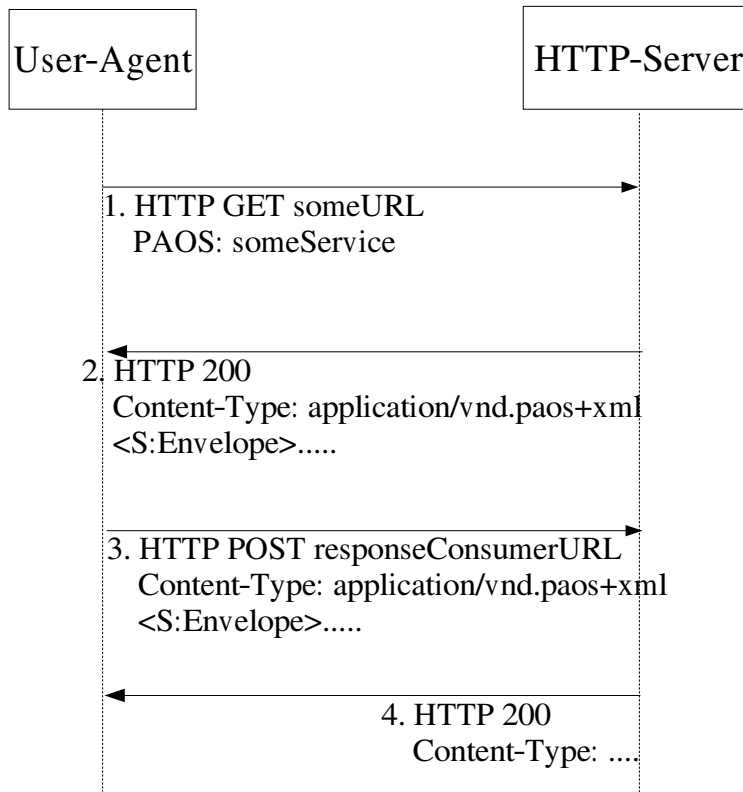
133 This binding supports two *message exchange patterns*, a request-response pattern, and a response pattern. In the
134 request-response pattern the PAOS enabled user agent makes an HTTP request to which the HTTP server responds
135 with a SOAP request message. The SOAP processor at the User-Agent then constructs a SOAP response message
136 which is sent as the body of a second HTTP request. The response to this second HTTP request typically is normal
137 content. The response pattern consists of an HTTP request to which the HTTP server responds with a SOAP (response)
138 message. Note that "request-response" and "response" refer to the SOAP interactions, not to the HTTP interactions.
139 In the normal HTTP binding for SOAP a SOAP request-response message exchange involves a single HTTP request
140 followed by a single HTTP response. The very same SOAP message exchange over PAOS involves two HTTP request-
141 response pairs.

142 **Note:**

143 [SOAP 1.2](#) specifies Message Exchange Patterns (MEP) in terms of state transitions in some detail. This
144 specification does not normatively refer to SOAP 1.2, but also does not attempt to define the Message
145 Exchange Patterns with rigor. It is expected that a future version of this binding specification will explicitly
146 refer to SOAP 1.2 and hence will refer to the MEPs of SOAP 1.2.

147 The request-response pattern described here has the same function and characteristics as the SOAP 1.2
148 request-response *MEP*, but the HTTP binding is different! Likewise the response pattern is specified as a
149 MEP in SOAP 1.2.

150 8. Operation of the Request-Response Message Exchange Pattern



151

152

Figure 1. PAOS Request-Response MEP

153 The request-response message exchange pattern bounded to PAOS, consists of four steps:

154

1. The user agent contacts a HTTP Server and sends a normal HTTP request. To inform the HTTP server that the user agent exposes one or more services over PAOS it SHOULD add a [PAOS header](#) to the HTTP request. The User-Agent also SHOULD include the PAOS MIME type in the value of the HTTP Accept header.

155

156

- 157 2. The HTTP server responds with a SOAP message in the body of the HTTP response. A SOAP processor
158 associated with the resource of the HTTP request constructs a SOAP request message that (provided that the
159 SOAP processor wishes to use the PAOS binding) MUST contain a `<paos:Request>` SOAP header block. The
160 `<paos:Request>` element contains the URL where the user agent should POST the SOAP response message.
161 The SOAP (request) message MUST be the body of the HTTP response. The HTTP response MUST have its
162 HTTP Content-Type header set to the PAOS MIME type: `application/vnd.paos+xml`.
- 163 3. At some point the User-Agent sends a HTTP POST message to the HTTP server. The resource field (URL) of
164 this HTTP request MUST be set to the value of the `responseConsumerURL` attribute of the `<paos:Request>`.
165 The HTTP Content-Type header MUST be set to the PAOS MIME type. The User-Agent also SHOULD include
166 the PAOS MIME type in the value of the HTTP Accept header. The body of the HTTP request contains a SOAP
167 response message. In case the `<paos:Request>` contained a `messageID` attribute the SOAP message MUST
168 contain a `<paos:Response>` a SOAP header block with its `inResponseTo` attribute set to the value of the
169 `messageID`.
- 170 4. The HTTP server responds with some content that is acceptable to the User-Agent.

171 Here is an example exchange between a User-Agent that exposes some personal information profile service and a
172 HTTP server hosting a horoscope service. The example is loosely, and not necessarily correctly, based upon a [ID-](#)
173 [WSF Profile Service](#) hosted at the user agent.

```
174
175
176     1. User-Agent requests a page...
177
178     GET /index HTTP/1.1
179     Host: horoscope.example.com
180     Accept: text/html; application/vnd.paos+xml
181     PAOS: ver="urn:liberty:paos:2003-08"; "urn:liberty:id-sis-pp:2003-08", ↵
182     "urn:liberty:id-sis-pp:demographics"
183
184     2. Server responds by asking for a date of birth...
185
186     HTTP 200
187     Content-Type: application/vnd.paos+xml
188     Content-Length: 1234
189
190     <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
191     <soap:Header>
192
193         <paos:Request xmlns:paos="urn:liberty:paos:2003-08"
194             responseConsumerURL="/soap"
195             service="urn:liberty:id-sis-pp:2003-08"
196             mustUnderstand="1"
197             actor="http://schemas.xmlsoap.org/soap/actor/next" />
198
199         <sb:Correlation xmlns:sb="urn:liberty:sb:2003-08"
200             messageID="6c3a4f8b9c2d" />
201
202     </soap:Header>
203
204     <soap:Body>
205
206         <pp:Query xmlns:pp="urn:liberty:id-sis-pp:2003-08">
207
208             <QueryItem>
209                 <Select>pp:PP/pp:Demographics/pp:BirthDay</Select>
210             </QueryItem>
211
212         </pp:Query>
213
214     </soap:Body>
215
```

```
216     </soap:Envelope>
217
218     3. Service at user agent responds to the SOAP request with a SOAP response inside an HTTP_
219 request...
220
221     POST /soap HTTP/1.1
222     Host: horoscope.example.com
223     Accept: text/html; application/vnd.paos+xml
224     PAOS: ver="urn:liberty:paos:2003-08"; "urn:liberty:id-sis-pp:2003-08
225 ", "urn:liberty:id-sis-pp:demographics"
226     Content-Type: application/vnd.paos+xml
227     Content-Length: 2345
228
229     <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
230
231     <soap:Header>
232     <sb:Correlation xmlns:sb="urn:liberty:sb:2003-08"
233     messageID="4d3eae2e3f5g"
234     refToMessageID="6c3a4f8b9c2d" />
235 </soap:Header>
236
237     <soap:Body>
238
239     <pp:QueryResponse xmlns:pp="urn:liberty:id-sis-pp:2003-08">
240
241     <Data>
242     <Birthday>--05-09</Birthday>
243     </Data>
244
245     </pp:QueryResponse>
246
247     </soap:Body>
248
249 </soap:Envelope>
250
251 4. Finally the server responds with a page containing a personalized horoscope...
252
253 HTTP 200
254 Content-Type: text/html
255 Content-Length: 1234
256
257 <html>
258 <head>
259 <title>Your Horoscope from horoscope.example.com</title>
260 </head>
261 <body>
262 <p>Dear Virgo, <br/>
263     In July 2003 you will have to sit through many boring meetings.
264     But this ordeal will be worth it and you will make new friends.</p>
265 </body>
266 </html>
267
```

268 9. Operation of the Response Message Exchange Pattern

269 The response message exchange pattern bounded to PAOS consists of the following two steps:

270 1. The user agent contacts a HTTP Server and sends a normal HTTP request. To inform the HTTP server that the
271 user agent exposes one or more services over PAOS it SHOULD add a [PAOS header](#) to the HTTP request. The
272 User-Agent SHOULD include the PAOS MIME type in the value of the HTTP Accept header.

273 2. The HTTP server responds with a SOAP message in the body of the HTTP response. A SOAP processor
274 associated with the resource of the HTTP request constructs a SOAP response message, i.e. a SOAP message
275 to which the sender does not expect a response. The SOAP (response) message MUST be the body of the
276 HTTP response. The HTTP response MUST have its HTTP Content-Type header set to the PAOS MIME type:
277 `application/vnd.paos+xml`.

278 **Note:**

279 In this response pattern there is no need for a SOAP header block of the [PAOSRequestType](#) .

280 Here is an example exchange between a User-Agent that polls a messaging service for a delivery confirmation.

```
281
282
283 1. User-Agent requests a page...
284
285 GET /confirmation HTTP/1.1
286 Host: message.example.com
287 Accept: text/html; application/vnd.paos+xml
288 PAOS: ver="urn:liberty:paos:2003-08"; "urn:example:message"
289
290 2. Server responds by sending a status report about an earlier requested message delivery
291
292 HTTP 200
293 Content-Type: application/vnd.paos+xml
294 Content-Length: 1234
295
296 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
297
298   <soap:Body>
299
300     <msg:StatusReport xmlns:msg="urn:example:message"
301       message="987654321" status="msg:delivered" />
302
303   </soap:Body>
304
305 </soap:Envelope>
306
307
308
```

309 10. PAOS Request Header Type

310 A SOAP processor that initiates a Request-Response MEP using this PAOS binding MUST add a child element of type
311 `PAOSRequestType` to the `soap:Header` element of the SOAP request message. The `PAOSRequestType` contains:

- 312 • A `responseConsumerURL` attribute, with a URL as its value. This URL SHOULD be relative to the URL that
313 was requested by the user agent (in the HTTP request that resulted in the creation of the SOAP message). If the
314 URL nevertheless is absolute it MUST have `http` or `https` as the protocol and SHOULD have a domain that is
315 owned by the same party as the owner of the domain in the URL of the HTTP request.
- 316 • A `service` attribute that MUST have one of the service URIs that were present in the [PAOS HTTP header](#) as its
317 value.
- 318 • An optional `messageID` attribute. Presence of this attribute will cause the User-Agent to respond with a
319 SOAP message that has a header element with an `refToMessageID` attribute with the same value. It is
320 RECOMMENDED that the value of this attribute have nonce characteristics.
- 321 • The SOAP `mustUnderstand` and `actor` attributes. Both are required. The `mustUnderstand` attribute MUST
322 be set to 1. The `actor` attribute MUST be set to `http://schemas.xmlsoap.org/soap/actor/next`.

323 **Note:**

324 The particular service that is exposed using PAOS may already require message id tokens; the `messageID`
325 attribute is intended to be used in case the service does not specify such tokens. For example, the [ID-WSF](#)
326 SOAP Binding 1.0 requires that compliant services add a SOAP header element with such message id tokens.

327 In general, the HTTP Server that acts as a SOAP requestor will have a need to link the future SOAP response
328 message to the SOAP request message that it makes. Naturally, the well-known HTTP techniques for session
329 management could be used for this purpose. For example the HTTP server could add a session or message
330 identifier as parameter to the `responseConsumerURL`, or set a cookie. However, in a layered architecture it
331 is expected to be beneficial to have a message identifier at the SOAP level, hence the `messageID` attribute.

332 The following schema fragment defines the `paos:Request` element:

```
333 <element name="Request" type="paos:RequestType" />
334 <complexType name="RequestType">
335   <attribute name="responseConsumerURL" type="anyURI" use="required"/>
336   <attribute name="service" type="anyURI" use="required" />
337   <attribute name="messageID" type="IDType" use="optional"/>
338   <attribute ref="S:mustUnderstand" use="required"/>
339   <attribute ref="S:actor" use="required"/>
340 </complexType>
```

344 An example is:

```
345 <Request
346   responseConsumerURL="/soap"
347   service="urn:liberty:idpp:2003-08"
348   messageID="6c3a4f8b9c2d"
349   mustUnderstand="1"
350   actor="http://schemas.xmlsoap.org/soap/actor/next" />
```

354 **10.1. Processing Rules**

355 The recipient of a `paos:Request` element MUST send any SOAP response in the body of a HTTP request. That
356 HTTP request:

357 1. MUST have the `responseConsumerURL` as the requested resource.

358 2. MUST be submitted with the POST method.

359 3. SHOULD be submitted to the same host from which the SOAP request was received.

360 4. SHOULD have an HTTP Content-Type header with its value set to the PAOS MIME type (`applica-`
361 `tion/vnd.paos+xml`).

362 If the processed `paos:Request` element contains a `messageID` attribute then the SOAP response MUST contain a
363 `paos:Response` element in the `S:Header`, and that `paos:Response` element MUST have its `refToMessageID`
364 attribute set to the value of the `messageID`.

365 If processing of the `paos:Request` fails, the processor has no opportunity to send a SOAP Fault or any other message
366 back to the SOAP requestor. In this case it is RECOMMENDED that the user agent resubmits the HTTP request of
367 step 1 (see [Section 9](#)), but omit the PAOS HTTP header.

368 11. PAOS Response Header Type

369 A SOAP processor that responds to a SOAP message that contained a `paos:Request` header block SHOULD add
370 a `paos:Response` element to the `soap:Header` element of the SOAP response message. It contains the optional
371 `refToMessageID` attribute.

372 If the SOAP message is a response to a PAOS request that contained a `paos:Request` element with a `messageID`
373 attribute then the `paos:Request` header MUST be included and its `refToMessageID` attribute MUST be present
374 and set to the value of the of the `messageID`.

375 Both the SOAP `mustUnderstand` and `actor` attributes are required. The `mustUnderstand` attribute MUST be set
376 to 1. The `actor` attribute MUST be set to `http://schemas.xmlsoap.org/soap/actor/next`.

377 The following schema fragment defines the `paos:Response` element:

```
378  
379  
380 <element name="Response" type="paos:ResponseType" />  
381 <complexType name="ResponseType">  
382   <attribute name="refToMessageID" type="IDType" use="optional"/>  
383   <attribute ref="S:mustUnderstand" use="required" />  
384   <attribute ref="S:actor" use="required" />  
385 </complexType>  
386
```

387 An example is:

```
388  
389  
390 <paos:Response  
391   refToMessageID="6c3a4f8b9c2d"  
392   mustUnderstand="1"  
393   actor="http://schemas.xmlsoap.org/soap/actor/next" />  
394
```

395 11.1. Processing Rules

396 There are no processing rules for the recipient of a `paos:Response` element.

397 11.2. SOAP Faults

398 In this binding the SOAP responder, i.e. the user agent, has no possibility to use HTTP status codes to indicate the
399 status of processing the SOAP request. It is recommended that implementations of PAOS on the HTTP server do not
400 rely on HTTP status codes. The SOAP Responder should use `soap:Fault` elements as specified for [SOAP](#).

401 12. Security Considerations

402 The use of PAOS enables a simple exchange of information between user agent hosted services and remote servers.
403 As PAOS is likely to be used for the exchange of personal information, security issues should be carefully considered
404 by implementers. The following paragraphs introduce an incomplete list of potential issues.

405 12.1. Message Integrity and Confidentiality

406 For the typical PAOS deployment it will be important to ensure the integrity of the SOAP messages. Often it may also
407 be important to have reasonable assurance that the parties are authentic. One option is to set up the server such that
408 the SOAP messages will be transported over SSL/TLS, thus ensuring that the relevant URLs use the `https` protocol
409 scheme. This seems especially appropriate for the `responseConsumerURL` in the `paos:Request` header. Another
410 option would be to sign messages but it is not very likely that PAOS enabled user agents will have the software and/or
411 certificates to generate or validate signatures, whereas most user agents support SSL/TLS.

412 12.2. Authentication

413 It is in the interest of the service at the PAOS-exposed user agent to have some assurance about the HTTP server, as
414 typically the HTTP server will ask the user agent for some information or to access some service. This is a similar
415 situation to that of form-filling at a browser, and similar solutions apply. In particular, the use of SSL/TLS combined
416 with server side certificate verification is RECOMMENDED.

417 The HTTP server may require assurance that the information it obtains is reliable. To this end the server may want
418 to authenticate the User-Agent. All methods for authentication may be applied, including but not limited to HTTP
419 Basic and Digest Authentication, as well as single-sign-on technologies such as the [ID-Federation Framework](#). Such
420 HTTP authentication could well happen before any PAOS message exchange pattern, and the HTTP server may want
421 to employ technologies for HTTP session management such as the use of cookies or URL-rewriting.

422 12.3. Protection of Information

423 A PAOS enabled user agent should make reasonable efforts to ensure that a SOAP response is sent to the correct server.
424 It should check that the `responseConsumerURL` in the `paos:Request` header points to the server that made the
425 SOAP request. It is RECOMMENDED that the response is posted using TLS and that the client verifies the server
426 certificate.

427 12.4. PAOS Intermediaries

428 A PAOS enabled user agent could encapsulate a normal SOAP service, and simply forward some incoming SOAP
429 message (over PAOS) inside a new HTTP request to a HTTP server (using the normal SOAP over HTTP binding).
430 If this responding service wishes to ensure that the contents of the SOAP response has not been compromised by the
431 User-Agent, it should protect the SOAP message by signing the relevant parts, e.g. the SOAP Body and possible SOAP
432 header blocks. There is no guarantee that the PAOS User-Agent will send the SOAP response to the correct party, so
433 the responding service must establish some trust in its immediate client (the User-Agent), perhaps by employing some
434 form of authentication.

435 13. XML Schema for PAOS

```
436 <?xml version="1.0" encoding="UTF-8"?>
437 <xs:schema targetNamespace="urn:liberty:paos:2003-08"
438   xmlns:xs="http://www.w3.org/2001/XMLSchema"
439   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
440   xmlns="urn:liberty:paos:2003-08"
441   elementFormDefault="qualified"
442   attributeFormDefault="unqualified">
443 <xs:annotation>
444 <xs:documentation>
445
446   The source code in this XSD file was excerpted verbatim from:
447
448   Liberty Reverse HTTP Binding
449   Version 1.1
450   14th December 2004
451
452   Copyright (c) 2003-2005 Liberty Alliance participants, see
453   http://www.projectliberty.org/specs/idwsf_1_1_copyrights.php
454
455 </xs:documentation>
456 </xs:annotation>
457 <xs:import namespace="http://schemas.xmlsoap.org/soap/envelope/"
458   schemaLocation="http://schemas.xmlsoap.org/soap/envelope/" />
459 <xs:include schemaLocation="liberty-utility-v1.1.xsd" />
460 <xs:element name="Request" type="RequestType" />
461 <xs:complexType name="RequestType">
462   <xs:attribute name="responseConsumerURL" type="xs:anyURI" use="required" />
463   <xs:attribute name="service" type="xs:anyURI" use="required" />
464   <xs:attribute name="messageID" type="IDType" use="optional" />
465   <xs:attribute ref="S:mustUnderstand" use="required" />
466   <xs:attribute ref="S:actor" use="required" />
467 </xs:complexType>
468 <xs:element name="Response" type="ResponseType" />
469 <xs:complexType name="ResponseType">
470   <xs:attribute name="refToMessageID" type="IDType" use="optional" />
471   <xs:attribute ref="S:mustUnderstand" use="required" />
472   <xs:attribute ref="S:actor" use="required" />
473 </xs:complexType>
474 </xs:schema>
475
476
```

477 **References**

478 **Normative**

- 479 [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., eds. (June
480 1999). "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, The Internet Engineering Task Force
481 <http://www.ietf.org/rfc/rfc2616.txt> [June 1999].
- 482 [SOAPv1.1] "Simple Object Access Protocol (SOAP) 1.1," Box, Don, Ehnebuske, David , Kakivaya, Gopal, Layman,
483 Andrew, Mendelsohn, Noah, Nielsen, Henrik Frystyk, Winer, Dave, eds. World Wide Web Consortium W3C
484 Note (08 May 2000). <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- 485 [RFC2396] Berners-Lee, T., Fielding, R., Masinter, L., eds. (August 1998). "Uniform Resource Identifiers (URI):
486 Generic Syntax," RFC 2396, The Internet Engineering Task Force <http://www.ietf.org/rfc/rfc2396.txt>
487 [August 1998].

488 **Informative**

- 489 [SOAPv1.2] "SOAP Version 1.2 Part 1: Messaging Framework," Gudgin, Martin, Hadley, Marc, Mendelsohn,
490 Noah, Moreau, Jean-Jacques, Nielsen, Henrik Frystyk, eds. World Wide Web Consortium W3C Proposed
491 Recommendation (07 May 2003). <http://www.w3.org/TR/2003/PR-soap12-part1-20030507/>
- 492 [SOAPv1.2part2] "SOAP Version 1.2 Part 2: Adjuncts," Gudgin, Martin, Hadley, Marc, Mendelsohn, Noah, Moreau,
493 Jean-Jacques, Nielsen, Henrik Frystyk, eds. World Wide Web Consortium W3C Recommendation (24 June
494 2003). <http://www.w3.org/TR/soap12-part2/>
- 495 [LibertyIDFFOverview] Wason, Thomas, eds. "Liberty ID-FF Architecture Overview," Version 1.2-errata-v1.0,
496 Liberty Alliance Project (12 September 2004). <http://www.projectliberty.org/specs>
- 497 [LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, eds. "Liberty ID-WSF SOAP Binding Specification
498 ," Version 1.2, Liberty Alliance Project (14 December 2004). <http://www.projectliberty.org/specs>
- 499 [LibertyDisco] Sergeant, Jonathan, eds. "Liberty ID-WSF Discovery Service Specification," Version 1.2, Liberty
500 Alliance Project (12 December 2004). <http://www.projectliberty.org/specs>
- 501 [LibertyIDPPP] Kellomaki, Sampo, eds. "Liberty Identity Personal Profile Service Specification," Version 1.0, Liberty
502 Alliance Project (12 November 2003). <http://www.projectliberty.org/specs>

503 **A. Request for MIME Media Type Application/Vendor Tree -** 504 **vnd.paos+xml**

505 Title: Request for MIME media type Application/Vendor Tree - vnd.paos+xml
506
507 Name : John Kemp
508
509 Email : john.kemp@earthlink.net
510
511 MIME media type name : Application
512
513 MIME subtype name : Vendor Tree - vnd.paos+xml
514
515 Required parameters : None
516
517 Optional parameters : None
518
519 Encoding considerations : 8bit
520 This media type may require encoding on transports not capable of handling 8
521 bit text.
522
523 Security considerations :
524 To paraphrase section 3 of RFC 1874, XML MIME entities contain
525 information to be parsed and processed by the recipient's XML system.
526 These entities may contain and such systems may permit explicit system
527 level commands to be executed while processing the data. To
528 the extent that an XML system will execute arbitrary command strings,
529 recipients of XML MIME entities may be at risk.
530
531 In addition to this general concern, the paos+xml typed
532 documents will contain data that may identify or pertain to an individual.
533
534 To counter potential issues, paos+xml typed documents
535 contain data that must be signed appropriately by the sender. Any such
536 signature must be verified by the recipient of the data - both as a
537 valid signature, and as being the signature of the sender.
538
539 There is no executable content passed via this MIME type. To counter any
540 privacy concerns, opaque handles are assigned to individuals, which may only
541 identify an individual when used by either the sender or the recipient of
542 the data. Transport-level security is ensured by Liberty
543 transactions occurring over secured channels.
544
545 For a more detailed discussion of general security considerations of
546 the Liberty protocol & profiles, please reference:
547
548 1) Section 4 of: Liberty ID-FF Bindings & Profiles Specification, Version
549 1.2, Liberty
550 Alliance Project, <"http://www.projectliberty.org/specs">
551 2) Liberty ID-WSF Security Profiles, Version 1.0, Liberty Alliance Project,
552 <"http://www.projectliberty.org/specs">
553 3) Liberty ID-WSF Security & Privacy Guidelines, Version 1.0, Liberty
554 Alliance Project,
555 <"http://www.projectliberty.org/specs">
556
557
558 Interoperability considerations :
559 There are no known interoperability concerns regarding this media type
560
561 Published specification :
562 The media type is used for the Liberty Reverse HTTP Binding for SOAP (PAOS)
563
564 The relevant specification is:
565
566 Liberty Reverse HTTP Binding for SOAP, Version 1.0
567 <http://www.projectliberty.org/specs/>

568
569
570
571 Applications which use this media :
572 Any implementation of the Liberty Reverse HTTP Binding for SOAP
573 (none are known yet)
574
575 Additional information :
576
577 1. Magic number(s) : n/a
578 2. File extension(s) : n/a
579 3. Macintosh file type code : n/a
580 4. Object Identifiers: n/a
581
582
583
584 Person to contact for further information :
585
586 1. Name : John Kemp
587 2. Email : john.kemp@earthlink.net
588
589 Intended usage : Limited Use
590
591
592 Author/Change controller : John Kemp of IEEE-ISTO
593 (john.kemp@ieee-isto.org) has change control for any future
594 updates.
595
596
597