Liberty Reverse HTTP Binding 1.0

Version: 1.0-05

Editors:
Robert Aarts, Nokia

Abstract:

SOAP is a lightweight protocol for 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".

Notice

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

Revision History

Revision: 0.2 **Date:** 10 February 2003

Draft based upon feedback in Phoenix meeting to draft-arch-nokia-client-protocols-1.06.doc. Also includes results from Miami meeting. To be released as stand-alone document in a form suitable for eventual submission to W3C.

Revision: 0.3 **Date:** 11 February 2003

Fixed a number of minor errors. Added section for SOAP Faults.

Revision: 0.4 **Date:** 16 March 2003

Proposes solutions for issues: 36, 37, 38, 49,.51 and partly 52. So many minor corrections and imporvements.

Revision: 0.5 **Date:** 10 April 2003

Included figure. Minor corrections.

Contents

1. Overview

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

SOAP is a lightweight protocol for 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. This document specifies a binding that enables HTTP clients (user agents) 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". The (informal) abbreviation for this binding specification is "PAOS".

Note:

Although this specification normatively refers to the SOAP 1.1 specification, including the SOAP 1.1 HTTP binding, every attempt has been made to be as compatible with the SOAP 1.2 Protocol Binding Framework as possible. To this end the terminology in this specification is primarily derived from the SOAP 1.2 specification.

This specification, however, does refer to the SOAP *1.2* MIME content type.

2. Optionality

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

3. Use of HTTP

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

4. HTTP Media Type

Conforming applications of this binding:

1. MUST be capable of sending and receiving messages serialized using media type "application/soap+xml" whose proper use and parameters are described in "The application/soap+xml Media Type".
2. MAY, when sending requests, provide an HTTP Accept header field. This header field

    a. SHOULD indicate an ability to accept at minimum "application/soap+xml"

    b. MAY additionally indicate willingness to accept other media types

5. Binding Name

This binding is specified by the URN:

"urn:soap:bindings:paos:1.0"

Note:

TODO: establish the actual URN, the above URN is to be treated as *tentative*.

6. Indication of binding support

HTTP User Agents that are ready to receive SOAP messages in HTTP responses SHOULD add a "PAOS" HTTP header to the HTTP request. The value of this header informs the HTTP server about the SOAP service(s) available at the User Agent. The header MUST be named `PAOS` and is defined using Augmented BNF as specified in section 2 of RFC 2616 as:

```
PAOS = "PAOS" ":" PAOS_Version ["," 1#Extension] *(";"  Service  ["," #Option])
PAOS_Version = "ver" "=" 1#quotedURI
Extension = ExtName "=" ExtValue
ExtName = (["." host] | <any field-value but ".", "," or "="> <any field-value but "=" or ",">
ExtValue = <any field-value but ",">)
Service = quotedURI
Option = quotedURI
quotedURI = <">anyURI<">
```

The comment, field-value, and product productions are defined in RFC 2616. PAOS_Version identifies the versions of the PAOS specification that are supported by this User-Agent. The versions are identified by a URI. HTTP Servers receiving a PAOS header MUST ignore any URIs listed in the PAOS_Version production that they do not recognize. All User-Agents compliant with this specification MUST send out, at minimum, the URI `urn:soap:paos:1.0` as a value in the PAOS_Version production. The ordering of the URIs in the PAOS_Version header is meaningful; therefore, HTTP servers are encouraged to use the first version in the list that they support. Supported versions are not negotiated between the User-Agent and HTTP server. The User-Agent simply advertises what version it does support.

Optional extensions MAY be added to the PAOS header to indicate new information. The value of the ExtName production MUST use the "host" ";" prefixed form, unless the new extension name has been standardized and registered with Liberty or its designated registration authorities. The value of the host production MUST be an IP or DNS address that is owned by the issuer of the new name.

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

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

Note:

A `Service` field corresponds to a `<ResourceOfferering>` as defined in the ID-WSF Discovery Service specification. This correspondence is as follows:

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

- The `<ServiceType>` element of the `<ServiceInstance>` element corresponds to the primary value (URI) of the `Service` field. This URI refers to *abstract WSDL*; for PAOS this is sufficient as there is no need to inform about the binding or endpoint ot the service. Hence there is no need in the PAOS header for an equivalent for the `<Description>` element.

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

7. Supported Message Exchange Patterns

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

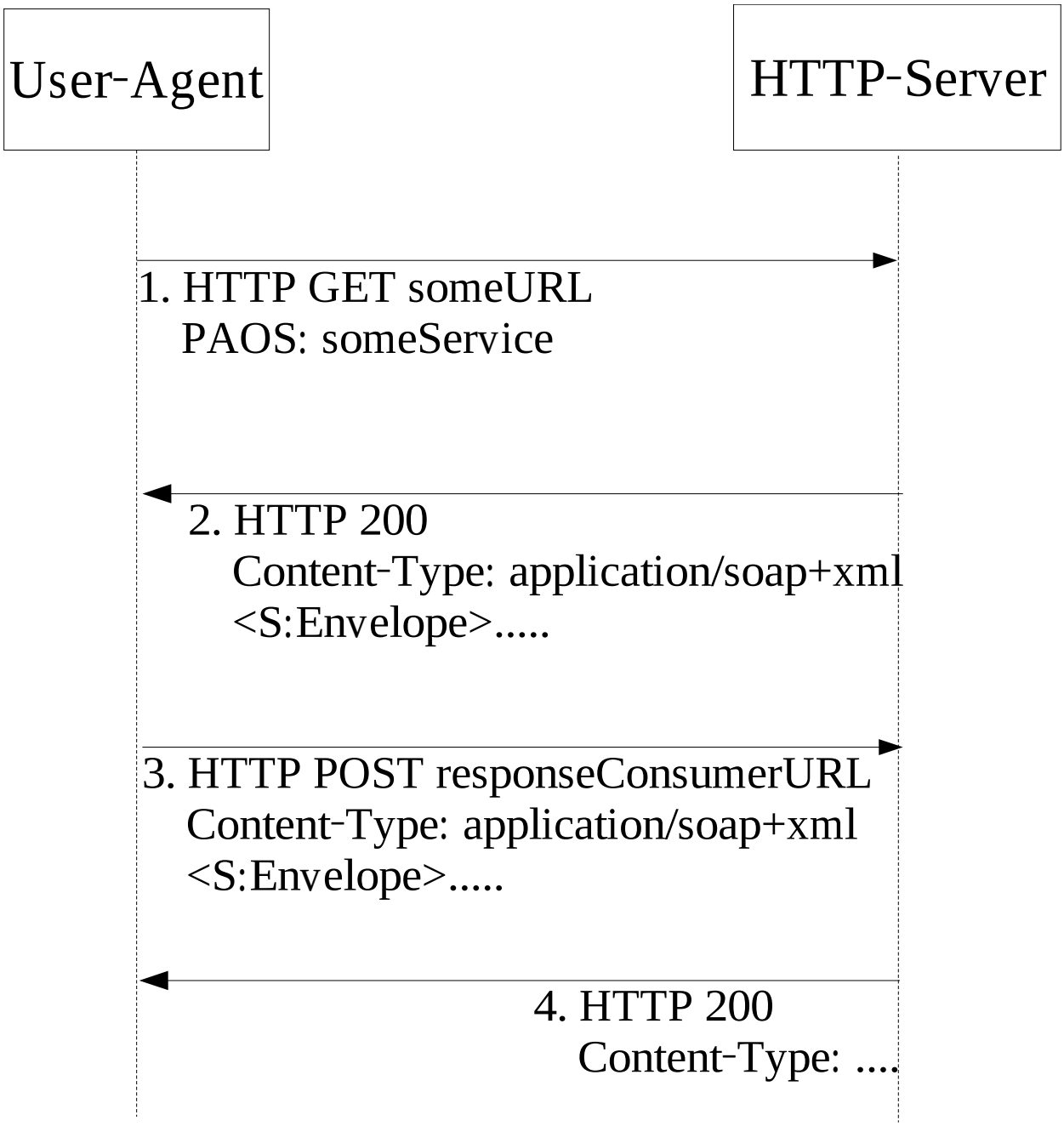
Note:

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

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

8. Operation of the Request-Response Message Exchange Pattern

Figure 1. PAOS Request-Response MEP



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

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 SOAP 1.2 MIME type in the value of the HTTP Accept header.

2. The HTTP server responds with a SOAP message in the body of the HTTP response. A SOAP processor associated with the resource of the HTTP request constructs a SOAP request message that (provided that the SOAP processor wishes to use the PAOS binding) MUST contain a `<paos:Request>`SOAP header block. The `<paos:Request>` element contains the URL where the User Agent should POST the SOAP response message. The SOAP (request) message MUST be the body of the HTTP response. The HTTP response MUST have its HTTP Content-Type header set to the SOAP 1.2. MIME type: `application/soap+xml`.

3. At some point the User-Agent sends a HTTP POST message to the HTTP server. The resource field (URL) of this HTTP request MUST be set to the value of the `responseConsumerURL` attribute of the `<paos:Request>`. The HTTP Content-Type header MUST be set to the SOAP MIME type. The User-Agent also SHOULD include the SOAP MIME type in the value of the HTTP Accept header. The body of the HTTP request contains a SOAP response message. In case the `<paos:Request>` contained a `messageID` attribute the SOAP message MUST contain a `<paos:Response>` a SOAP header block with its `inResponseTo` attribute set to the value of the `messageID`.

4. The HTTP server responds with some content that is acceptable to the User-Agent.

Here is an example exchange between a User-Agent that exposes some personal information profile service and a HTTP server hosting a horoscope service. The example is loosely, and not necessarily correctly, based upon a ID-WSF Profile Service hsoted at the user agent.

```
1. User-Agent requests a page...

GET /index HTTP/1.1
Host: horoscope.example.com
Accept: text/html; application/soap+xml
PAOS: ver="urn:liberty:paos:1.0"; "urn:liberty:idpp:1.0", "urn:liberty:idpp:demographics"

2. Server responds by asking date of birth...

HTTP 200
Content-Type: application/soap+xml
Content-Length: 1234

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
        <paos:Request xmlns:paos="urn:soap:bindings:paos:1.0"
            responseConsumerURL="/soap"
            service="urn:liberty:idpp:1.0"
            mustUnderstand="1"
            actor= "http://schemas.xmlsoap.org/soap/actor/next" />
        <sb:Correlation xmlns:sb="urn:liberty:sb:1.0"
            messageID="6c3a4f8b9c2d"/>
     </soap:Header>
     <soap:Body>
         <idpp:Query xmlns:idpp="urn:liberty:idpp:1.0">
            <Resource>urn:liberty:dsc:resource-implied</Resource>
            <QueryItem>
               <Select>/IDPP/Demographics/Birthday</Select>
            </QueryItem>
        </idpp:Query>
    </soap:Body>
</soap:Envelope>

3. Service at User Agent responds to SOAP request with a SOAP response inside a HTTP request...

POST /soap HTTP/1.1
Host: horoscope.example.com
Accept: text/html; application/soap+xml
PAOS: ver="urn:liberty:paos:1.0"; "urn:liberty:idpp:1.0", "urn:liberty:idpp:demographics"
Content-Type: application/soap+xml
Content-Length: 2345

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
       xmlns:sb="urn:liberty:sb:1.0">
   <soap:Header>
        <sb:Correlation xmlns:sb="urn:liberty:sb:1.0"
            messageID="4d3eae2e3f5g"
```

```
            refToMessageID="6c3a4f8b9c2d"/>
   </soap:Header>
   <soap:Body>
      <idpp:QueryResponse xmlns:idpp="urn:liberty:idpp:1.0">
         <Data>
            <Demographics xmlns="urn:librty:idpp:1.0">
               <Birthday>-05-09</Birthday>
            </Demographics>
         </Data>
      </idpp:QueryResponse>
   </soap:Body>
</soap:Envelope>

4. Finally the server responds with a page containing a personalized horoscope...

HTTP 200
Content-Type: text/html
Content-Length: 1234

<html>
   <head>
      <title>Your Horoscope from horoscope.example.com</title>
   </head>
   <body>
      <p>Dear Virgo, <br/>
      In May 2003 you will have to sit through many boring meetings.
      But this ordeal will be worth it and you will make new friends.</p>
   </body>
</html>
```

9. Operation of the Response Message Exchange Pattern

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

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 SOAP 1.2 MIME type in the value of the HTTP Accept header.

2. The HTTP server responds with a SOAP message in the body of the HTTP response. A SOAP processor associated with the resource of the HTTP request constructs a SOAP response message, i.e. a SOAP message to which the sender does not expect a response. The SOAP (response) message MUST be the body of the HTTP response. The HTTP response MUST have its HTTP Content-Type header set to the SOAP 1.2. MIME type: `application/soap+xml`.
   Note: In this response pattern there is no need for a SOAP header block of the `PAOSRequestType`.
   This MEP is the "reverse" of the standard SOAP 1.2 Response MEP; here a normal HTTP request is followed by a HTTP response containing SOAP, wheras in the standard SOAP 1.2 binding a HTTP POST containing SOAP is followed by a normal HTTP response.

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

```
1. User-Agent requests a page...

GET /confirmation HTTP/1.1
Host: message.example.com
Accept: text/html; application/soap+xml
PAOS: ver="urn:liberty:paos:1.0"; "urn:example:message"

2. Server responds by sending a status report about an earlier requested message delivery

HTTP 200
Content-Type: application/soap+xml
Content-Length: 1234

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <msg:StatusReport xmlns:msg="urn:example:message"
              message="987654321" status="msg:delivered" />
   </soap:Body>
</soap:Envelope>
```

10. PAOS Request header type

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

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

Note:

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

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

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

```
<element name="Request" type="paos:RequestType" />
<complexType name="RequestType">
   <attribute name="responseConsumerURL" type="anyURI" use="required"/>
   <attribute name="service" type="anyURI" use="required"
   <attribute name="messageID" type="IDType" use="optional"/>
   <attribute ref="S:mustUnderstand" use="required"/>
   <attribute ref="S:actor" use="required"/>
</complexType>
```

An example is:

```
<Request
      responseConsumerURL="/soap"
      service="urn:liberty:idpp:1.0"
      messageID="6c3a4f8b9c2d"
      mustUnderstand="1"
      actor= "http://schemas.xmlsoap.org/soap/actor/next" />
```

10.1. Processing rules

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

1. MUST have the responseConsumerURL as the requested resource.
2. MUST be submitted with the POST method.
3. SHOULD be submitted to the same host from which the SOAP request was received.
4. SHOULD have a HTTP Content-Type header with it's value set to the SOAP 1.2 MIME type (application/soap+xml).

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

In case processing of the `paos:Request` fails the processor has no opportunity to send a SOAP Fault or any other message back to the SOAP requestor. In this case it is RECOMMENDED that the User Agent resubmits the HTTP request of step 1, but omit the PAOS HTTP header.

11. PAOS Response header type

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

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

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

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

```
<element name="Response" type="paos:ResponseType" />
<complexType name="ResponseType">
   <attribute name="refToMessageID" type="IDType" use="optional"/>
   <attribute ref="S:mustUnderstand" use="required" />
   <attribute ref="S:actor" use="required"/>
</complexType>
```

An example is:

```
<paos:Response
      refToMessageID="6c3a4f8b9c2d"
      mustUnderstand="1"
      actor= "http://schemas.xmlsoap.org/soap/actor/next" />
```

11.1. Processing rules

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

11.2. SOAP Faults

In this binding the SOAP responder, i.e. the User Agent, has no possibility to use HTTP status codes to indicate status of processing the SOAP request. It is recommended that implementations of PAOS on the HTTP server do not rely on HTTP status codes. The SOAP Responder should use `soap:Fault` elements as specified for SOAP.

12. Security Considerations

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

12.1. Message integrity and confidentiality

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

12.2. Authentication

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

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

12.3. Protection of information

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

12.4. PAOS intermediaries

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

13. XML Schema for PAOS

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:liberty:1.0"
 xmlns="urn:liberty:1.0"
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:annotation>
      <xs:documentation>Editor: Robert Aarts, Nokia</xs:documentation>
      <xs:documentation>Copyright 2003 Liberty Alliance Project</xs:documentation>
   </xs:annotation>
   <xs:import namespace="http://schemas.xmlsoap.org/soap/envelope/"
 schemaLocation="http://schemas.xmlsoap.org/soap/envelope/"/>
   <xs:include schemaLocation="lib-arch-utility.xsd"/>
   <xs:element name="Request" type="RequestType"/>
   <xs:complexType name="RequestType">
      <xs:attribute name="responseConsumerURL" type="xs:anyURI" use="required"/>
      <xs:attribute name="service" type="xs:anyURI" use="required"/>
      <xs:attribute name="messageID" type="IDType" use="optional"/>
      <xs:attribute ref="S:mustUnderstand" use="required"/>
      <xs:attribute ref="S:actor" use="required"/>
   </xs:complexType>
   <xs:element name="Response" type="ResponseType"/>
   <xs:complexType name="ResponseType">
      <xs:attribute name="refToMessageID" type="IDType" use="optional"/>
      <xs:attribute ref="S:mustUnderstand" use="required"/>
      <xs:attribute ref="S:actor" use="required"/>
   </xs:complexType>
</xs:schema>
```

Bibliography

Normative

[ref_HTTP] "HTTP1.1 spec," Version 1.0-05.

[ref_SOAP] "SOAP 1.1 specification,"

[ref_HTTPBindingForSOAP] "SOAP 1.1 specification, HTTP binding section,"

[ref_SOAPMIMEType] "SOAP 1.2 specification part 2, MIME type Appendix,"

Informative

[ref_SOAP12] "SOAP 1.2 specification,"

[ref_SOAP12BindingFramework] "SOAP 1.2 specification part 2,"

[ref_ID-WSF-DS] "ID-WSF Discovery Service 1.0,"

[ref_ID-SIS-PP] "ID-Personal Profile 1.0,"

[ref_ID-WSF-SOAP-Binding] "ID-WSF SOAP Binding 1.0,"

[ref_ID-FF] "ID-FF Architectural Overview 1.2,"