Liberty Reverse HTTP Binding for SOAP Specification

Version: v2.0

Editors:
Robert Aarts, (formerly of) Nokia Corporation
John Kemp, 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. A SOAP header block and a binding of that header block to HTTP are specified such that a client software application may 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 "Reverse HTTP binding for SOAP".

Filename: liberty-paos-v2.0.pdf

Notice

This document has been prepared by Sponsors of the Liberty Alliance. Permission is hereby granted to use the document solely for the purpose of implementing the Specification. No rights are granted to prepare derivative works of this Specification. Entities seeking permission to reproduce portions of this document for other uses must contact the Liberty Alliance to determine whether an appropriate license for such use is available.

Implementation of certain elements of this document may require licenses under third party intellectual property rights, including without limitation, patent rights. The Sponsors of and any other contributors to the Specification are not and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights. **This Specification is provided "AS IS", and no participant in the Liberty Alliance makes any warranty of any kind, express or implied, including any implied warranties of merchantability, non-infringement of third party intellectual property rights, and fitness for a particular purpose.** Implementers of this Specification are advised to review the Liberty Alliance Project's website (http://www.projectliberty.org/) for information concerning any Necessary Claims Disclosure Notices that have been received by the Liberty Alliance Management Board.

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

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 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. 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 [SOAPv1.1], including the SOAP 1.1 HTTP binding, every attempt has been made to be as compatible with [SOAPv1.2part2] as possible. To this end the terminology in this specification is primarily derived from the SOAP 1.2 specification.

This document specifies:

1. A description of PAOS message exchanges and related example.
2. A SOAP header block that may be used to indicate the use of PAOS, and services exposed using PAOS.
3. A binding of the afore-mentioned SOAP header block to HTTP, including specification of an associated HTTP header, and an HTTP MIME media type, used to indicate a PAOS message payload, or the ability of the client to accept such a payload.

Note:

This specification defines a transport-independent SOAP header block. A single, specific transport binding to HTTP is defined for that SOAP header block. Other bindings MAY be created, but no further mention is made of any other binding in this document.

1.1. Glossary of Terms

PAOS Requester.

PAOS Responder.

1.2. Namespace Definitions

The following XML namespaces are referred to in this document:

- The prefix *P:* represents the PAOS namespace. This namespace is the default for instance fragments, type names, and element names in this document. In schema listings, and in examples, this is the default namespace *when* no prefix is shown:

  urn:liberty:paos:2006-08

- The prefix *A:* stands for the W3C Web Services Addressing (WSA) namespace [WSAv1.0]:

  http://www.w3.org/2005/03/addressing

- The prefix *xs:* stands for the W3C XML schema namespace [Schema1-2]:

  http://www.w3.org/2001/XMLSchema

- The prefix *S:* stands for the SOAP v1.1 namespace:

  http://schemas.xmlsoap.org/soap/envelope

2. Optionality

PAOS support is optional. 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. 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 request, indicating that it supports PAOS. The recipient of this message (the *PAOS requester*) 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 request. The response to this second request typically is normal content. The response pattern consists of a request to which the PAOS requester responds with a SOAP (response) message. Note that "request-response" and "response" refer to the SOAP interactions, not to any underlying interactions. In the normal 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 Patterns (MEP) in terms of state transitions in some detail. This specification does not normatively refer to SOAP 1.2 and 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.

4. Operation of the Request-Response Message Exchange Pattern

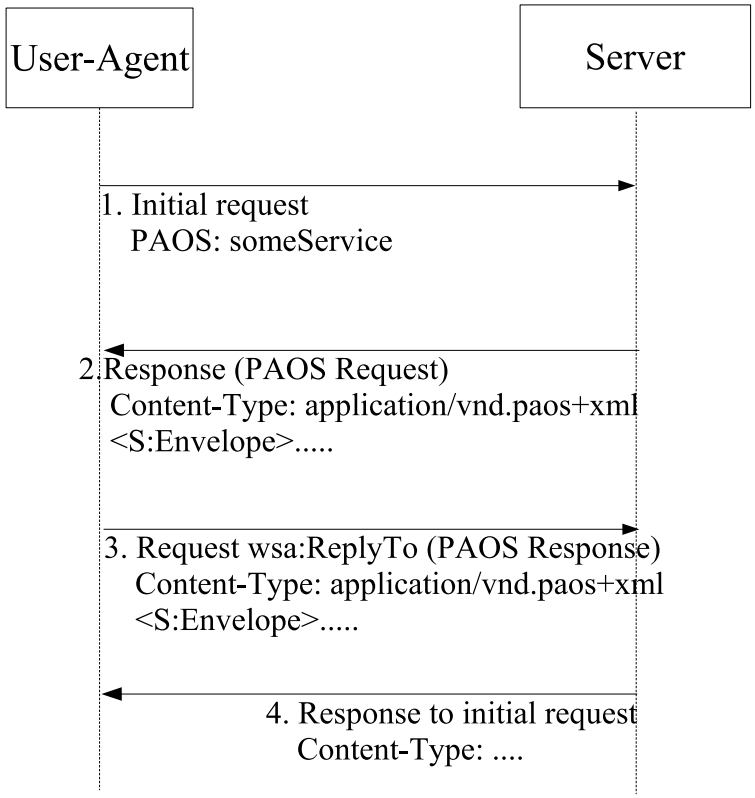

Figure 1. PAOS Request-Response MEP

The request-response message exchange pattern bound to PAOS consists of four steps.

1. The PAOS responder contacts a PAOS requester and sends a request (typically a request for content or service made available by the PAOS requester). To inform the PAOS requester that the PAOS responder exposes one or more services over PAOS it SHOULD add an indication of PAOS support to the request.

2. The PAOS requester responds with a SOAP message. The PAOS requester constructs a SOAP request message that (provided that the SOAP processor wishes to use the PAOS binding) contains SOAP header blocks indicating that the message is a request for a service exposed over PAOS. The `<A:ReplyTo>` element contains the URL where the PAOS responder should POST the SOAP response message..

3. At some point the PAOS responder sends a response message to the PAOS requester..

4. The PAOS requester responds with some content that is acceptable to the PAOS responder, which may be in reponse to the original request for service, prior to the PAOS interactions.

Here are two example exchanges between a PAOS requester that exposes some personal information profile service and a horoscope service. The example is loosely, and not necessarily correctly, based upon a ID-WSF Profile Service hosted at the user agent.

The first example shows how a SOAP client might request service from a SOAP horoscope service exposed over HTTP, indicating PAOS support via the PAOS SOAP header block. Although this example shows use of the HTTP binding for PAOS (and thus duplicates the PAOS indications), the PAOS header block might equally be bound to some other transport.

```
      1. Client requests a horoscope...

      POST /soap/horoscope HTTP/1.1
      Host: horoscope.example.com
      Accept: text/html; application/vnd.paos+xml
      PAOS: ver="urn:liberty:paos:2006-08"
, "urn:liberty:paos:2003-08"; "urn:liberty:id-sis-pp:2003-08", "urn:liberty:id-sis-pp:demographics"


      <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">

        <S:Header xmlns:A="http://www.w3.org/2005/03/addressing">

          <PAOS xmlns="urn:liberty:paos:2006-08">

            <Version>urn:liberty:2006-08</Version>
            <Version>urn:liberty:2003-08</Version>

            <A:EndpointReference>

              <!-- shown for completeness only. this is the default if ommitted! -->

              <A:Address>
                http://www.projectliberty.org/2006/01/role/paos
              </A:Address>

              <A:Metadata>

                <ServiceType>urn:liberty:id-sis-pp:2003-08</ServiceType>

                <Options>
                  <Option>urn:liberty:id-sis-pp:demographics</paos:Option>
                </Options>

              </A:Metadata>

            </A:EndpointReference>

          </PAOS>

          <A:MessageID>urn:uuid-a43bde-2900-f70c-f0ba-a5ee61bde</A:MessageID>
          <A:Action>http://horoscope.example.com/soap/horoscope</A:Action>
```

```
        <A:ReplyTo>
          <A:Address>http://www.projectliberty.org/2006/02/role/paos</A:Address>
        </A:ReplyTo>

      </S:Header>

      <S:Body>
        <horo:GetHoroscope xmlns:horo="http://horoscope.example.com/soap/horoscope/2005/12"/>
      </S:Body>

    </S:Envelope>

    2. PAOS requester asks for a date of birth...

    HTTP 200
    Content-Type: application/vnd.paos+xml
    Content-Length: 1234

    <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">

      <S:Header xmlns:A="http://www.w3.org/2005/03/addressing">

        <A:MessageID>urn:uuid-C8797D0D-9020-07FC-AF0A-5622C01F4A61</wsA:MessageID>
        <A:Action>urn:liberty:id-sis:pp:2003-08</A:Action>

        <A:ReplyTo>

          <A:Address>http://horoscope.example.com/soap/horoscope</A:Address>

        </A:ReplyTo>

      </S:Header>

      <S:Body>
        <pp:Query xmlns:pp="urn:liberty:id-sis-pp:2003-08">
          <pp:QueryItem>
            <pp:Select>/pp:PP/pp:Demographics/pp:Birthday</pp:Select>
          </pp:QueryItem>
        </pp:Query>
      </S:Body>

    </S:Envelope>

    3. PAOS responder returns a SOAP response inside a new HTTP request...

    POST /soap/horoscope HTTP/1.1
    Host: horoscope.example.com
    Accept: text/html; application/vnd.paos+xml
    PAOS: ver="urn:liberty:paos:2006-08", "urn:liberty:paos:2003-08"; "urn:liberty:id-sis-pp:2003-08", "urn:libe

    Content-Type: application/vnd.paos+xml
    Content-Length: 2345

    <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">

      <S:Header xmlns:A="http://www.w3.org/2005/03/addressing">

        <A:MessageID>uuid-ab342ed-635ffee-142311ab-bedff67</A:MessageID>
        <A:RelatesTo>urn:uuid-C8797D0D-9020-07FC-AF0A-5622C01F4A61</A:RelatesTo>

        <A:Action>...</A:Action>

      </S:Header>

      <S:Body>

        <pp:QueryResponse xmlns:pp="urn:liberty:id-sis-pp:2003-08">
```

```
          <Data>
            <Birthday>--10-11</Birthday>
          </Data>
        </pp:QueryResponse>

      </S:Body>

    </S:Envelope>

    4. Finally the server responds with a SOAP message containing a personalized horoscope...

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

    <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">

      <S:Header xmlns:A="http://www.w3.org/2005/03/addressing">

        <PAOS xmlns="urn:liberty:paos:2006-08">

          <Version>urn:liberty:2006-08</Version>
          <Version>urn:liberty:2003-08</Version>

          <A:EndpointReference>

            <A:Address>
              http://www.projectliberty.org/2006/01/role/paos
            </A:Address>

            <A:Metadata>

              <ServiceType>urn:liberty:id-sis-pp:2003-08</ServiceType>

              <Options>
                <Option>urn:liberty:id-sis-pp:demographics</paos:Option>
              </Options>

            </A:Metadata>

          </A:EndpointReference>

        </PAOS>

        <A:MessageID>urn:uuid-a43bde-2900-f70c-f0ba-a5ee61bde</A:MessageID>
        <A:Action>http://horoscope.example.com/soap/horoscope</A:Action>

        <A:ReplyTo>
          <A:Address>http://www.projectliberty.org/2006/02/role/paos</A:Address>
        </A:ReplyTo>

      </S:Header>

      <S:Body>
        <h:Horoscope xmlns:h="http://horoscope.example.com/soap/horoscope/2005/12">
          <h:Sign>Virgo</h:Sign>
          <h:Horoscope>
            In July 2010 you will still have to sit through many boring
            meetings. But this ordeal may still be worth it as you will renew
            acquaintances with old friends. Or, you might be flying...
          </h:Horoscope>
      </S:Body>

    </S:Envelope>
```

The following example shows how PAOS may be initiated to perform the same action as shown above, with an initial HTTP GET request (rather than a SOAP request).

```
1. UA requests a page

GET /index HTTP/1.1
Host: horoscope.example.com
Accept: text/html; application/vnd.paos+xml
PAOS: ver="urn:liberty:paos:2006-08", "urn:liberty:paos:2003-08"; "urn:liberty:id-sis-pp:2003-08", "urn:libe


2. Server responds, asking for DOB

HTTP 200
Content-Type: application/vnd.paos+xml
Content-Length: 1234

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">

  <S:Header xmlns:A="http://www.w3.org/2005/03/addressing">

    <A:MessageID>uuid:C8797D0D-9020-07FC-AF0A-5622C01F4A61</A:MessageID>
    <A:Action>urn:liberty:id-sis-pp:2003-08:Query</A:Action>

    <A:ReplyTo>

      <A:Address>http://horoscope.example.com/soap/horoscope</A:Address>

    </A:ReplyTo>

  </S:Header>

  <S:Body>
    <pp:Query xmlns:pp="urn:liberty:id-sis-pp:2003-08">
      <QueryItem>
        <Select>/pp:PP/pp:Demographics/pp:Birthday</Select>
      </QueryItem>
    </pp:Query>
  </S:Body>

</S:Envelope>

3. UA Service responds to request

POST /soap/horoscope HTTP/1.1
Host: horoscope.example.com
Accept: text/html; application/vnd.paos+xml
PAOS: ver="urn:liberty:paos:2006-08", "urn:liberty:paos:2003-08";
"urn:liberty:id-sis-pp:2003-08", "urn:liberty:id-sis-pp:demographics"
Content-Type: application/vnd.paos+xml
Content-Length: 2345

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">

  <S:Header xmlns:A="http://www.w3.org/2005/03/addressing">

    <A:MessageID>uuid:ab342ed-635ffee-142311ab-bedff67</A:MessageID>

    <A:RelatesTo>uuid:C8797D0D-9020-07FC-AF0A-5622C01F4A61</A:RelatesTo>

    <A:Action/>

  </S:Header>

  <S:Body>
```

```
      <pp:QueryResponse xmlns:pp="urn:liberty:id-sis-pp:20 03-08">
       <Data>
        <Birthday>--10-11</Birthday>
       </Data>
      </pp:QueryResponse>

    </S:Body>

  </S:Envelope>

  4. Server responds with a web page

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

  <html>
    <head>
      <title>Your Horoscope from horoscope.example.com</title>
    </head>
    <body>
      <p>Dear Libra,<br/>
        In July 2010 you will still have to sit through many boring
        meetings.
        But this ordeal may still be worth it as you will renew
        acquaintances with old friends.</p>
    </body>
  </html>
```

5. Operation of the Response Message Exchange Pattern

The response message exchange pattern bound to PAOS consists of the following two steps.

1. The PAOS responder contacts a PAOS requester and sends a request. To inform the PAOS requester that the PAOS responder exposes one or more services over PAOS it SHOULD add an indication of PAOS support to the request.

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

```
1. PAOS responder polling

POST /confirmation HTTP/1.1
Host: message.example.com
Accept: text/html; application/vnd.paos+xml
PAOS: ver="urn:liberty:paos:2006-08", "urn:liberty:paos:2003-08"; "urn:example:message"

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

HTTP 200
Content-Type: application/vnd.paos+xml
Content-Length: 1234

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">

  <S:Header>
    <A:MessageID>uuid:C8797D0D-9020-07FC-AF0A-5622C01F4A61</A:MessageID>
    <A:Action>...</A:Action>
  </S:Header>

  <S:Body>

    <msg:StatusReport xmlns:msg="urn:example:message"
      message="987654321" status="msg:delivered" />

  </S:Body>

</S:Envelope>
```

6. Initiating a PAOS Request using WS-Addressing

A SOAP processor that initiates a SOAP Request-Response MEP using the PAOS HTTP binding MUST add the following child elements, defined in the WS-Addressing specification [REF] to the `S:Header` element of the SOAP request message.

- A `A:ReplyTo` element, with a URL as the value of the contained `A:Address` element. This specification defines the special URI `http://www.projectliberty.org/2006/02/role/paos`. This indicates that the PAOS-capable requester is indicating that it is not addressable other than by using the current transport channel, but that the responder may hold its response to this initial request while making a PAOS request.
  Note

  TODO: This URI value is to be considered tentative.

- An `A:Action` element that MUST have as its value one of the action URIs that were present in the Indication of PAOS support as its value. If no action URI was indicated in the indication of PAOS support, the PAOS requester MAY use instead the value of a service type specified in the request.

Given that a PAOS request may be initiated as the response to a SOAP *request*, it is possible that the original request contained a `A:MessageID`. A SOAP node initiating a PAOS request MUST NOT add an `A:RelatesTo` header block to its request.

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

The initiator of a PAOS request must follow the rules defined in [WS-ADDRESSING] for constructing the above SOAP headers. In case of conflict, this specification has precedence.

Notes:

If the PAOS requester wishes to receive a correlated SOAP response, a `A:MessageID` SOAP header block should be added to the request.

In general, the SOAP node acting as a PAOS requester will have a need to link the future SOAP response message to the SOAP request message that it makes. Naturally, the well-known HTTP techniques for session management could be used for this purpose, if the SOAP node utilizes HTTP as a transport. For example an HTTP server could 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 presence of the `A:MessageID` element.

An example is shown below.

```
      <A:MessageID mustUnderstand="1" actor= "http://schemas.xmlsoap.org/soap/actor/next">
        uuid-C8797D0D-9020-07FC-AF0A-5622C01F4A61
      </A:MessageID>

      <A:Action mustUnderstand="1" actor= "http://schemas.xmlsoap.org/soap/actor/next
">urn:liberty:id-sis:pp:2003-08</A:Action>

      <A:ReplyTo mustUnderstand="1" actor= "http://schemas.xmlsoap.org/soap/actor/next">
        <A:Address>http://horoscope.example.com/soap/horoscope</A:Address>
      </A:ReplyTo>
```

7. The PAOS Response

A SOAP processor that responds to a SOAP message that contained a PAOS request containing an `A:MessageID` MUST add a `A:RelatesTo` element to the `soap:Header` element of the SOAP response message, referencing the value of the `A:MessageID` element from the PAOS request.

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`.

An example is.

```
<A:RelatesTo mustUnderstand="1" actor= "http://schemas.xmlsoap.org/soap/actor/next">
  uuid-C8797D0D-9020-07FC-AF0A-5622C01F4A61
</A:RelatesTo>
```

8. The PAOS SOAP header

PAOS services may be advertised during a SOAP message exchange. Given that SOAP messages may be sent over more than one transport, this specification defines a SOAP header block that may be used to indicate support for PAOS at the SOAP layer, independently of the particular transport used to convey the SOAP message. This SOAP header contains an indication of the PAOS version support by the requesting SOAP node, and endpoint references for any services that the SOAP node is exposed via the PAOS binding.

The schema for the PAOS SOAP header block is shown below:

```
<xs:element name="PAOS" type="PaosType"/>

<xs:complexType name="PaosType">
<xs:complexContent>
  <xs:attribute ref="S:mustUnderstand" use="required"/>
  <xs:attribute ref="S:actor" use="required"/>
  <xs:sequence>

    <xs:element name="Version"
      type="xs:anyURI"
      minOccurs="1"
      maxOccurs="unbounded"/>

    <xs:element ref="a:EndpointReference"
      minOccurs="0"
      maxOccurs="unbounded"/>

    <xs:any namespace="##other"
      processContents="lax"
      minOccurs="0"
      maxOccurs="unbounded"/>

  </xs:sequence>
</xs:simpleContent>
</xs:complexType>
```

1. The `PAOS` SOAP header block is a container for one or more WS-Addressing [REF] *endpoint references*. Each endpoint reference MAY contain a `A:Address`, and if this element is present, its value SHOULD be *http://www.projectliberty.org/2006/01/role/paos*.

2. A PAOS responder may advertise any service describable with a WS-Addressing endpoint reference, which includes endpoint references of the type defined in the Liberty ID-WSF Discovery Service specification (section 2.5) [REF].

3. The `Version` elements identify the versions of the PAOS specification that are supported by this SOAP requester. The versions are identified by a URI ([RFC3986]). SOAP responders receiving a PAOS header MUST ignore any URIs listed that they do not recognize. All implementations compliant with this specification MUST send out, at a minimum, the URI `urn:liberty:paos:2006-08` as a Version value. The ordering of the URIs in the PAOS header is meaningful; therefore, recipients of the header are encouraged to use the first version in the list that they support. Supported versions are not negotiated between the SOAP requester and responder. The requester simply advertises what version it does support.

9. PAOS HTTP Binding

This document contains a binding of SOAP to HTTP. That binding is intended to make appropriate use of HTTP as an application protocol. The 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.

9.1. PAOS HTTP Media Type

Conforming applications of the PAOS HTTP binding:

1. MUST be capable of sending and receiving messages serialized using media type "application/vnd.paos+xml" whose proper use and parameters are described in Appendix A .

2. MAY, when sending requests, provide an HTTP Accept header field. This header field:

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

    b. MAY additionally indicate willingness to accept other media type.

9.2. Binding Name

This binding is specified by the URN:

"`urn:liberty:paos:2006-08`".

Note

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

9.3. HTTP Indication of Binding Support

9.3.1. HTTP header

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 [RFC2616], as:

```
PAOS = "PAOS" ":" PAOS_Version ["," Extension] *(";"  Service  ["," #Option] ["," #Action])
PAOS_Version = "ver" "=" 1#quotedURI
Extension = "ext" "=" 1#quotedURI
Service = quotedURI
Option = quotedURI
Action = "action" "=" 1#quotedURI
quotedURI = <">anyURI<">
```

Note

As an HTTP user agent may be associated with a SOAP requester, and thus may indicate in a SOAP message that it supports PAOS (by means of the PAOS SOAP header defined above), it may not be necessary for the HTTP user agent to add an HTTP indication of PAOS support.

If both the PAOS SOAP header, and the PAOS HTTP header are added to a request message, the values and cardinality of all of the PAOS header block child elements MUST match those sub-elements of the PAOS HTTP header production, as described below.

The comment, field-value, and product productions are defined in [RFC2616]. PAOS_Version identifies the versions of the PAOS specification that are supported by this User-Agent. The versions are identified by a URI ([RFC3986]). 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 a minimum, the URI `urn:liberty:paos:2006-08` 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.

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 an `<EndpointReference>` of the XML type as defined in the ID-WSF Discovery Service specification. This correspondence is as follows.

- The `/ServiceInstanceEPR/ServiceInstanceEPRMetadata/ServiceType` 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 provide further information to the service about the binding or endpoint. Hence there is no need in the PAOS header for an equivalent for the `<Description>` element.

  As the value of this URI is used to create a corresponding `A:Action` in the PAOS request, this field may refer to a specific action described in abstract WSDL, rather than the service itself.

- The `/ServiceInstanceEPR/ServiceInstanceEPRMetadata/Options/Option` elements of the endpoint reference correspond to the values (URIs) of the `Option` fields in the PAOS HTTP header.

9.4. Processing Rules

The following processing rules are stipulated for a PAOS requester utilizing the PAOS HTTP binding:

1. The PAOS request MUST form the body of the HTTP response message.
2. The HTTP response MUST have its HTTP `Content-type` header set to the PAOS HTTP Media Type: `application/vnd.paos+xml`.
3. An HTTP response containing a PAOS request MUST have the HTTP status code `202 Accepted`, indicating that the initial message was accepted for processing.

The *PAOS responder* MUST send any SOAP response in the body of an HTTP request. That HTTP request:

1. SHOULD have the value of the supplied endpoint reference `Address` element as the requested resource (URL).
2. MUST be submitted via the HTTP POST method.
3. SHOULD be submitted to the same host from which the SOAP request was received
4. SHOULD have an HTTP Content-Type header with its value set to the PAOS HTTP Media Type: `application/vnd.paos+xml`.

If processing of the PAOS request message fails, the processor has no opportunity to send a SOAP Fault or any other message back to the PAOS requester. In this case it is RECOMMENDED that the user agent resubmits the HTTP request of step 1 (see Section 5), omitting any indication of PAOS support (such as the PAOS SOAP header block, or the PAOS HTTP header).

10. 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 implementers. The following paragraphs introduce an incomplete list of potential issues.

10.1. 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 any relevant URLs use the `https` protocol scheme. 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 SSL/TLS.

10.2. Authentication

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

A PAOS requester may require assurance that the information it obtains is reliable. To this end the requester may want to authenticate the PAOS responder, prior to sending a PAOS request. 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 authentication could well happen before any PAOS message exchange pattern, and if the PAOS requester functions as an HTTP server, it may employ technologies for HTTP session management such as the use of cookies or URL-rewriting.

In the event that a PAOS message exchange is initiated by a SOAP request from the PAOS responder, OASIS WS-Security [REF] SOAP message security may be employed for authenticating the PAOS responder.

10.3. Protection of Information

A PAOS enabled user agent should make reasonable efforts to ensure that a SOAP response is sent to the correct server. This implies that it may be necessary for the PAOS responder to validate that any URI present in the `A:ReplyTo` matches indications about the PAOS requester that are present in any underlying transport. It is RECOMMENDED that the response is posted using TLS and that the client verifies the server certificate.

10.4. 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 request to some other SOAP node (using the normal SOAP over HTTP binding). If this ultimate SOAP node wishes to ensure that the contents of the SOAP response has not been compromised by the intermediary, 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 responder will send the SOAP response to the correct party, so the ultimate SOAP node must establish some trust in its immediate requester, perhaps by employing some form of authentication.

11. XML Schema for PAOS

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:liberty:paos:2006-08"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns="urn:liberty:paos:2006-08"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified">
  <xs:annotation>
    <xs:documentation>

The source code in this XSD file was excerpted verbatim from:

Liberty Reverse HTTP Binding
Version 2.0
30 July, 2006

Copyright (c) 2006 Liberty Alliance participants, see
https://www.projectliberty.org/specs/idwsf_2_0_final_copyrights.html

      </xs:documentation>
  </xs:annotation>

  <xs:import namespace="http://schemas.xmlsoap.org/soap/envelope/"
    schemaLocation="http://schemas.xmlsoap.org/soap/envelope/"/>
  <xs:include schemaLocation="liberty-utility-v2.0.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>
```

References

Normative

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

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

[RFC3986] Berners-Lee, T., Fielding, R., Masinter, L., eds. (January 2005). "Uniform Resource Identifier (URI): Generic Syntax," RFC 3986 (Obsoletes RFC2732, RFC2396, RFC1808) (Updates RFC1738) (Also STD0066) (Status: STANDARD), The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc3986.txt*

Informative

[SOAPv1.2] "SOAP Version 1.2 Part 1: Messaging Framework," Gudgin, Martin, Hadley, Marc, Mendelsohn, Noah, Moreau, Jean-Jacques, Nielsen, Henrik Frystyk, eds. World Wide Web Consortium W3C Recommendation (07 May 2003). *http://www.w3.org/TR/2003/PR-soap12-part1-20030507/*

[SOAPv1.2part2] "SOAP Version 1.2 Part 2: Adjuncts," Gudgin, Martin, Hadley, Marc, Mendelsohn, Noah, Moreau, Jean-Jacques, Nielsen, Henrik Frystyk, eds. World Wide Web Consortium W3C Recommendation (24 June 2003). *http://www.w3.org/TR/soap12-part2/*

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

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

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

[LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, Whitehead, Greg, Madsen, Paul, eds. "Liberty ID-WSF SOAP Binding Specification," Version 2.0, Liberty Alliance Project (30 July, 2006). *http://www.projectliberty.org/specs*

[LibertyDisco] Hodges, Jeff, Cahill, Conor, eds. "Liberty ID-WSF Discovery Service Specification," Version 2.0, Liberty Alliance Project (30 July, 2006). *http://www.projectliberty.org/specs*

[LibertyIDPP] Kellomäki, Sampo, Lockhart, Rob, eds. "Liberty ID-SIS Personal Profile Service Specification," Version 1.1, Liberty Alliance Project (29 September, 2005). *http://www.projectliberty.org/specs*

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

```
Title: Request for MIME media type Application/Vendor Tree - vnd.paos+xml

Name : John Kemp

Email : john.kemp@earthlink.net

MIME media type name : Application

MIME subtype name : Vendor Tree - vnd.paos+xml

Required parameters : None

Optional parameters : None

Encoding considerations : 8bit
This media type may require encoding on transports not capable of handling 8
bit text.

Security considerations :
To paraphrase section 3 of RFC 1874, XML MIME entities contain
information to be parsed and processed by the recipient's XML system.
These entities may contain and such systems may permit explicit system
level commands to be executed while processing the data.  To
the extent that an XML system will execute arbitrary command strings,
recipients of XML MIME entities may be at risk.

In addition to this general concern, the paos+xml typed
documents will contain data that may identify or pertain to an individual.

To counter potential issues, paos+xml typed documents
contain data that must be signed appropriately by the sender. Any such
signature must be verified by the recipient of the data - both as a
valid signature, and as being the signature of the sender.

There is no executable content passed via this MIME type. To counter any
privacy concerns, opaque handles are assigned to individuals, which may only
identify an individual when used by either the sender or the recipient of
the data. Transport-level security is ensured by Liberty
transactions occurring over secured channels.

For a more detailed discussion of general security considerations of
the Liberty protocol & profiles, please reference:

1) Section 4 of: Liberty ID-FF Bindings & Profiles Specification, Version
1.2, Liberty
Alliance Project, <"http://www.projectliberty.org/specs">
2) Liberty ID-WSF Security Profiles, Version 1.0, Liberty Alliance Project,
<"http://www.projectliberty.org/specs">
3) Liberty ID-WSF Security & Privacy Guidelines, Version 1.0, Liberty
Alliance Project,
<"http://www.projectliberty.org/specs">


Interoperability considerations :
There are no known interoperability concerns regarding this media type

Published specification :
The media type is used for the Liberty Reverse HTTP Binding for SOAP (PAOS)

The relevant specification is:

Liberty Reverse HTTP Binding for SOAP, Version 1.0
<http://www.projectliberty.org/specs/>
```

```
Applications which use this media :
Any implementation of the Liberty Reverse HTTP Binding for SOAP
(none are known yet)

Additional information :

1. Magic number(s) : n/a
2. File extension(s) : n/a
3. Macintosh file type code : n/a
4. Object Identifiers: n/a



Person to contact for further information :

1. Name : John Kemp
2. Email : john.kemp@earthlink.net

Intended usage : Limited Use


Author/Change controller : John Kemp of IEEE-ISTO
(john.kemp@ieee-isto.org) has change control for any future
updates.
```