XEP-0072: SOAP Over XMPP

Fabio Forno
mailto:fabio.forno@gmail.com
xmpp:ff@jabber.bluendo.com

Peter Saint-Andre
mailto:stpeter@stpeter.im
xmpp:stpeter@jabber.org
https://stpeter.im/

2022-03-22
Version 1.0.1

Status	Type	Short Name
Draft	Standards Track	soap

This specification defines methods for transporting SOAP messages over XMPP. Although the protocol supports only the request-response message exchange pattern, the protocol is formally defined as a SOAP Protocol Binding in accordance with version 1.2 of the W3C SOAP specification. In addition, a WSDL definition is defined for the purpose of advertising the availability of this protocol binding.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2024 by the XMPP Standards Foundation (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the ”Specification”), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

NOTE WELL: This Specification is provided on an ”AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF’s Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

Contents

1 Introduction 1

2 Architectural Assumptions 1

3 Use Cases 2
- 3.1 Discovering Support . . . 2
- 3.2 Exchanging SOAP Messages . . . 3
  - 3.2.1 Exchanging SOAP Messages Using XMPP IQ Stanzas . . . 3
  - 3.2.2 Exchanging SOAP Messages Using XMPP Message Stanzas . . . 7
- 3.3 Sending Associated Data . . . 7
  - 3.3.1 File Transfer . . . 10
  - 3.3.2 Including Links . . . 12
- 3.4 Specifying a WSDL Definition . . . 15

4 SOAP XMPP Binding 17
- 4.1 Binding Name . . . 17
- 4.2 Supported Features . . . 17
- 4.3 Supported Message Exchange Patterns . . . 18
- 4.4 Operation of Request-Response Message Exchange Pattern . . . 18
  - 4.4.1 Behavior of Requesting SOAP Node . . . 19
  - 4.4.2 Behavior of Responding SOAP Node . . . 21

5 W3C Considerations 24
- 5.1 W3C Review . . . 24
- 5.2 SOAP Versioning . . . 25
- 5.3 XML Versioning . . . 25

6 Error Handling 25

7 Business Rules 26
- 7.1 Encoding . . . 26

8 Security Considerations 26

9 IANA Considerations 27

10 XMPP Registrar Considerations 27
- 10.1 Protocol Namespaces . . . 27
- 10.2 Service Discovery Identity . . . 27

11 XML Schema 27
- 11.1 SOAP Envelope . . . 27
- 11.2 Application-Specific XMPP Errors . . . 28

12 Implementation Notes 28

13 Acknowledgements 30

1 Introduction

SOAP [1] is a lightweight protocol that defines a method for the exchange of messages independently from the programming language and platform. For interoperability, the SOAP specification is also agnostic about possible transport protocols, though almost all existing implementations use mainly HTTP.
The primary limitation of HTTP consists in the fact that HTTP-based message exchanges allow only synchronous request-response semantics. To overcome this limitation, SMTP is often used to carry asynchronous messages, but it is a complex protocol and inefficient for passing short and frequent messages that should be delivered in close to real time.
Thus XMPP (see XMPP Core [2]) can be the ideal transport protocol for many of the application fields of web services, since it can carry efficiently and reliably both types of messages, synchronous and asynchronous. Moreover, XMPP-based web services will not need complex support protocols, such as WS-Routing and WS-Referral, in order to deliver messages to entities that cannot be identified by static public IP addresses. Therefore, this document defines a binding of SOAP to XMPP as an alternative to the existing HTTP and SMTP bindings. (Note: The main body of this document provides descriptive text suitable for use by XMPP developers. A formal description of the SOAP XMPP Binding itself is provided in the section of this document entitled SOAP XMPP Binding.)

2 Architectural Assumptions

The usual architecture of XMPP is described in RFC 6120. In essence, XMPP is most commonly deployed using a client-server (or logical peer-to-peer) architecture quite similar to that of the email system, except that XMPP does not have multiple hops between servers, enforces domain names to prevent address spoofing, and enables channel encryption (via TLS) and authentication (via SASL) between client and server as well as among servers.
The binding of SOAP to XMPP assumes that most SOAP-enabled XMPP entities will be implemented as XMPP clients that communicate with other entities as logical peers. However, in order to deploy more scalable services, such entities could also be implemented as server-side components (see Jabber Component Protocol (XEP-0114) [3]) or even as special-purpose XMPP servers.
The SOAP specification defines the concepts of ”SOAP intermediary” and ”ultimate SOAP receiver” (see Section 1.5.3 of SOAP Version 1.2 Part 1). In general, this specification assumes that XMPP entities that support the SOAP XMPP Binding will be ultimate SOAP receivers, since SOAP intermediaries tend to be artifacts of the existing SOAP bindings (HTTP and SMTP) rather than applicable to all possible bindings. SOAP intermediaries are usually deployed in order to (1) cross trust boundaries in protocols that do not enforce domain names or authenticate end-points, (2) ensure scalability, (3) secure messages sent over unencrypted

[1] SOAP <http://www.w3.org/TR/SOAP/>.
[2] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[3] XEP-0114: Jabber Component Protocol <https://xmpp.org/extensions/xep-0114.html>.

channels, and (4) provide message tracing. However, these issues are addressed natively in XMPP (e.g., channel encryption is defined in RFC 6120), in XMPP extensions (e.g., message tracing is defined in Advanced Message Processing (XEP-0079) [4]), or in deployment decisions such as business level agreements between XMPP domains. One final justification for SOAP intermediaries is to act as gateways between different transport mechanisms (e.g., between HTTP and SMTP), and XMPP entities may well be SOAP intermediaries for that reason. For further details about gateways between XMPP and other SOAP bindings, refer to the Implementation Notes section of this document.

3 Use Cases

3.1 Discovering Support

In order to determine whether a potential responding entity supports the SOAP XMPP Binding, a requesting entity SHOULD send a Service Discovery (XEP-0030) [5] information request to the potential responding entity:

Listing 1: Requester queries responder regarding protocol support

```
<iq from='requester@example.com/soap-client'
    to='responder@example.com/soap-server'
    id='disco1'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

If the responding entity supports the SOAP XMPP Binding and the requesting entity is not blocked from communicating with the responding entity, the responding entity MUST include a feature of "http://jabber.org/protocol/soap" in its reply and SHOULD specify a service discovery identity of "automation/soap".

Listing 2: Responder replies regarding protocol support

```
<iq from='responder@example.com/soap-server'
    to='requester@example.com/soap-client'
    id='disco1'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='automation' type='soap'/>
    <feature var='http://jabber.org/protocol/soap'/>
  </query>
</iq>
```

[4] XEP-0079: Advanced Message Processing <https://xmpp.org/extensions/xep-0079.html>.
[5] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

3.2 Exchanging SOAP Messages

When a requesting entity wants to interact with a responding entity via the SOAP XMPP Binding, it faces a fundamental choice: to use <iq/> stanzas or to use <message/> stanzas. The following guidelines may prove useful:

1. <iq/> stanzas SHOULD be used when more formal request-response semantics are needed or when an immediate answer is required.
2. <message/> stanzas SHOULD be used when less formal request-response semantics are acceptable or when store-and-forward ("offline message") delivery is needed (e.g., because the intended recipient may be temporarily unavailable).

Examples of both approaches are provided below, encapsulating the SOAP message examples (a travel reservation flow) to be found in SOAP Version 1.2 Part 0 [6].

3.2.1 Exchanging SOAP Messages Using XMPP IQ Stanzas

The transport with <iq/> stanzas is performed in a way similar to that described for XML-RPC in Jabber-RPC (XEP-0009) [7]. Request envelopes are carried by <iq/> stanzas of type "set", and answer envelopes by <iq/> stanzas of type "result". SOAP errors are encoded with standard SOAP envelopes, and returned in stanzas of type "error" with appropriate codes in order to distinguish them from errors specific to the XMPP transport layer (see Error Handling for details).
Each <iq/> stanza of type "set" MUST contain a SOAP envelope as the first-level child element, since it already represents a properly namespaced XML subtree qualified by the 'http://www.w3.org/2003/05/soap-envelope' namespace.

Listing 3: Requesting entity sends IQ-set

```
<iq from='requester@example.com/soap-client'
    id='soap1'
    to='responder@example.com/soap-server'
    type='set'>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <m:reservation
          xmlns:m="http://travelcompany.example.org/reservation"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
          env:mustUnderstand="true">
        <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</
            m:reference>
        <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
```

[6] SOAP Version 1.2 Part 0: Primer <http://www.w3.org/TR/soap12-part0>.
[7] XEP-0009: Jabber-RPC <https://xmpp.org/extensions/xep-0009.html>.

```
      </m:reservation>
      <n:passenger
         xmlns:n="http://mycompany.example.com/employees"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
        <n:name>Ake Jogvan Ovind</n:name>
      </n:passenger>
    </env:Header>
    <env:Body>
      <p:itinerary xmlns:p="http://travelcompany.example.org/
          reservation/travel">
        <p:departure>
          <p:departing>New York</p:departing>
          <p:arriving>Los Angeles</p:arriving>
          <p:departureDate>2001-12-14</p:departureDate>
          <p:departureTime>late afternoon</p:departureTime>
          <p:seatPreference>aisle</p:seatPreference>
        </p:departure>
        <p:return>
          <p:departing>Los Angeles</p:departing>
          <p:arriving>New York</p:arriving>
          <p:departureDate>2001-12-20</p:departureDate>
          <p:departureTime>mid-morning</p:departureTime>
          <p:seatPreference/>
        </p:return>
      </p:itinerary>
      <q:lodging xmlns:q="http://travelcompany.example.org/reservation
          /hotels">
        <q:preference>none</q:preference>
      </q:lodging>
    </env:Body>
  </env:Envelope>
</iq>
```

If the responding entity does not support the SOAP XMPP Binding, it SHOULD return a <service-unavailable/> error:

Listing 4: Responding entity reports that it cannot handle SOAP messages

```
<iq type='result' to='requester@example.com/soap-client' id='soap1'>
  <error code='503' type='cancel'>
    <service-unavailable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If a SOAP-related fault occurs, the mappings in Error Handling SHOULD be used.

Listing 5: Responding entity indicates SOAP fault

```
<iq type='error' to='requester@example.com/soap-client' id='soap1'>
  <env:Envelope
       xmlns:env='http://www.w3.org/2003/05/soap-envelope'
       xmlns:rpc='http://www.w3.org/2003/05/soap-rpc'>
    <env:Body>
      <env:Fault>
        <env:Code>
          <env:Value>env:Sender</env:Value>
          <env:Subcode>
            <env:Value>rpc:BadArguments</env:Value>
          </env:Subcode>
        </env:Code>
        <env:Reason>
          <env:Text xml:lang='en-US'>Processing error</env:Text>
        </env:Reason>
      </env:Fault>
    </env:Body>
  </env:Envelope>
  <error code='500' type='modify'>
    <undefined-condition xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <Sender xmlns='http://jabber.org/protocol/soap#fault'/>
  </error>
</iq>
```

If the responding entity does not return an error, it MUST respond with an IQ of type "result":

Listing 6: Responding entity returns IQ-result

```
<iq from='responder@example.com/soap-server'
    id='soap1'
    to='requester@example.com/soap-client'
    type='result'>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <m:reservation xmlns:m="http://travelcompany.example.org/
          reservation"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
        <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</
            m:reference>
        <m:dateAndTime>2001-11-29T13:35:00.000-05:00</m:dateAndTime>
      </m:reservation>
      <n:passenger xmlns:n="http://mycompany.example.com/employees"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
        <n:name>Ake Jogvan Ovind</n:name>
      </n:passenger>
    </env:Header>
    <env:Body>
```

```
      <p:itineraryClarification xmlns:p="http://travelcompany.example.
          org/reservation/travel">
        <p:departure>
          <p:departing>
            <p:airportChoices>JFK LGA EWR</p:airportChoices>
          </p:departing>
        </p:departure>
        <p:return>
          <p:arriving>
            <p:airportChoices>JFK LGA EWR</p:airportChoices>
          </p:arriving>
        </p:return>
      </p:itineraryClarification>
    </env:Body>
  </env:Envelope>
</iq>
```

At this point the requesting entity could send another IQ-set:

Listing 7: Requesting entity sends another IQ-set

```
<iq from='requester@example.com/soap-client'
    id='soap2'
    to='responder@example.com/soap-server'
    type='set'>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <m:reservation
          xmlns:m="http://travelcompany.example.org/reservation"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
          env:mustUnderstand="true">
        <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</
            m:reference>
        <m:dateAndTime>2001-11-29T13:36:50.000-05:00</m:dateAndTime>
      </m:reservation>
      <n:passenger xmlns:n="http://mycompany.example.com/employees"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
          env:mustUnderstand="true">
        <n:name>Ake Jogvan Ovind</n:name>
      </n:passenger>
    </env:Header>
    <env:Body>
      <p:itinerary xmlns:p="http://travelcompany.example.org/
          reservation/travel">
        <p:departure>
          <p:departing>LGA</p:departing>
        </p:departure>
        <p:return>
          <p:arriving>EWR</p:arriving>
```

```
        </p:return>
      </p:itinerary>
    </env:Body>
  </env:Envelope>
</iq>
```

3.2.2 Exchanging SOAP Messages Using XMPP Message Stanzas

The process for exchanging SOAP messages using the XMPP <message/> stanza type is effectively no different from the use with <iq/> stanzas, except that message stanzas may be sent to bare JIDs (user@host) rather than full JIDs (user@host/resource), message stanzas may be stored for later delivery, etc. The following business rules apply:

1. The message stanza containing a request MUST carry one SOAP envelope as a first-level child element.
2. The 'id' attribute MUST be used to track the XMPP messages and eventually associate errors or answers with the related requests (this is for tracking at the XMPP level, not the SOAP level).

3.3 Sending Associated Data

SOAP messages may contain associated (usually binary) data, and XMPP stanzas that encapsulate such SOAP messages could invoke bandwidth restriction settings (commonly called "karma" in XMPP) tuned for normal text chats. The problem could be bypassed by servers having special karma settings for larger messages, or by SOAP-enabled entities being implemented as components rather than XMPP nodes; however, server-to-server communications risk becoming a serious bottleneck, especially in terms of latency and responsiveness when too many large messages are sent. Therefore, it is desirable to support the sending of attachments or files in order to exchange large amounts of binary data associated with SOAP requests and responses. As summarized in the following table, here are four possible methods:

Method	Description	Recommendation	Reasoning
File Transfer	Negotiate file transfer using SI File Transfer (XEP-0096) XEP-0096: SI File Transfer <https://xmpp.org/extensions/xep-0096.html>. and Publishing Stream Initiation Requests (XEP-0137) XEP-0137: Publishing Stream Initiation Requests <https://xmpp.org/extensions/xep-0137.html>..	SHOULD	Recommended approach for file transfer over XMPP (e.g., see Intermediate IM Protocol Suite (XEP-0117) XEP-0117: Intermediate IM Protocol Suite <https://xmpp.org/extensions/xep-0117.html>.).
Include Link	Represent the binary data as a file, publish it to an accessible file server (e.g., HTTP or FTP URL), and insert a link to the file directly into the XMPP message stanza (via Out-of-Band Data (XEP-0066) XEP-0066: Out of Band Data <https://xmpp.org/extensions/xep-0066.html>.) or into the SOAP envelope (via Resource Representation SOAP Header Block Resource Representation SOAP Header Block <http://www.w3.org/TR/soap12-rep>.).	MAY	Fallback if file transfer is not possible (not all clients can publish to file servers).

Method	Description	Recommendation	Reasoning
Alternate Transports	Send SOAP XML plus binary data over alternate transports such as WS-Attachments WS-Attachments <http://www.watersprings.org/pub/id/draft-nielsen-dime-soap-01.txt> (work in progress). or SOAP-over-BEEP as defined in RFC 4227 RFC 4227: Using the Simple Object Access Protocol (SOAP) in Blocks Extensible Exchange Protocol (BEEP) <http://tools.ietf.org/html/rfc4227>..	SHOULD NOT	These methods are just other transport protocols and would needlessly complicate implementations of SOAP over XMPP.
MIME	Encode SOAP envelopes and attachments as MIME multipart messages using SOAP 1.2 Attachment Feature SOAP 1.2 Attachment Feature <http://www.w3.org/TR/soap12-af/>. (or, more recently, SOAP Message Transmission Optimization Mechanism SOAP Message Transmission Optimization Mechanism <http://www.w3.org/TR/soap12-mtom>. and XML-binary Optimized Packaging XML-binary Optimized Packaging <http://www.w3.org/TR/xop10/>.).	MUST NOT	XML streams are pure XML and are not MIME-aware.

The recommended approaches (file transfer and including a link) are described more fully below.

3.3.1 File Transfer

The recommended method for sending associated data is to use the file transfer protocol described in XEP-0096. Because this is the common and standardized method for XMPP entities to transfer large or binary files outside the XMPP band, it SHOULD be used.
In particular, the entity that has the file SHOULD advertise the availability of the associated stream using XEP-0137 by including the SI-pub data extension along with the XMPP <message/> stanza with which the data is associated: [8]

Listing 8: Sender sends message with SI-pub

```
<message from='requester@example.com/soap-client'
         id='soap2'
         to='responder@example.com/soap-server'>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <m:reservation
         xmlns:m="http://travelcompany.example.org/reservation"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
        <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</
           m:reference>
        <m:dateAndTime>2001-11-29T13:36:50.000-05:00</m:dateAndTime>
      </m:reservation>
      <n:passenger xmlns:n="http://mycompany.example.com/employees"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
        <n:name>Ake Jogvan Ovind</n:name>
      </n:passenger>
    </env:Header>
    <env:Body>
      <p:itinerary xmlns:p="http://travelcompany.example.org/
          reservation/travel">
        <p:departure>
         <p:departing>LGA</p:departing>
        </p:departure>
        <p:return>
          <p:arriving>EWR</p:arriving>
        </p:return>
      </p:itinerary>
```

[8] In accordance with RFC 6120, an <iq/> stanza MUST NOT include multiple payload child elements; therefore, a <message/> stanza must be used when sending associated data.

```
    </env:Body>
  </env:Envelope>
  <sipub xmlns='http://jabber.org/protocol/sipub'
         id='publish-2345'
         mime-type='image/png'
         profile='http://jabber.org/protocol/si/profile/file-transfer'
            >
    <file xmlns='http://jabber.org/protocol/si/profile/file-transfer'
          name='me.png'
          size='4238'
          date='2005-11-01T23:11Z'/>
  </sipub>
</message>
```

The entity that is to receive the file SHOULD initiate the file transfer process sending an IQ-get to the sender, using the <start xmlns='http://jabber.org/protocol/sipub'/> element. This element contains the 'id' attribute to specify which published stream to retrieve:

Listing 9: Receiver requests start of stream

```
<iq type='get'
    id='sipub-request-0'
    from='responder@example.com/soap-server'
    to='requester@example.com/soap-client'>
  <start xmlns='http://jabber.org/protocol/sipub'
         id='publish-2345'/>
</iq>
```

If the sender accepts the request, it responds with an IQ-result containing a <starting/> element. This element indicates the stream initiation identifier to be used:

Listing 10: Sender accepts request to start stream

```
<iq type='result'
    id='sipub-request-0'
    from='requester@example.com/soap-client'
    to='responder@example.com/soap-server'>
  <starting xmlns='http://jabber.org/protocol/sipub'
            sid='session-87651234'/>
</iq>
```

Then the sender begins the stream initiation negotiation:

Listing 11: Sender starts negotiation

```
<iq type='set'
    id='sipub-set-1'
    from='requester@example.com/soap-client'
    to='responder@example.com/soap-server'>
```

```
  <si xmlns='http://jabber.org/protocol/si'
      id='session-87651234'
        mime-type='image/png'
      profile='http://jabber.org/protocol/si/profile/file-transfer'>
    <file xmlns='http://jabber.org/protocol/si/profile/file-transfer'
          name='me.png'
          size='4238'
          date='2005-11-01T23:11Z'/>
  </si>
</iq>
```

For details regarding file transfer and advertising of file transfer stream initiation requests, refer to XEP-0096 and XEP-0137.

3.3.2 Including Links

If the file transfer method is not possible (e.g., because file transfer is not implemented or transfer attempts fails), the entity that is sending the associated data MAY as a fallback publish the associated data as a file (e.g., at an HTTP or FTP URL) and include a link to the file as out-of-band content by including the out-of-band data extension along with the XMPP <message/> stanza with which the data is associated: [9]

Listing 12: Sender sends message with out-of-band data

```
<message from='requester@example.com/soap-client'
         id='soap2'
         to='responder@example.com/soap-server'>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <m:reservation
         xmlns:m="http://travelcompany.example.org/reservation"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
        <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</
           m:reference>
        <m:dateAndTime>2001-11-29T13:36:50.000-05:00</m:dateAndTime>
      </m:reservation>
      <n:passenger xmlns:n="http://mycompany.example.com/employees"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
        <n:name>Ake Jogvan Ovind</n:name>
      </n:passenger>
    </env:Header>
    <env:Body>
```

[9] As above, in accordance with RFC 6120, an <iq/> stanza MUST NOT include multiple payload child elements; therefore, a <message/> stanza must be used when sending associated data.

```
      <p:itinerary xmlns:p="http://travelcompany.example.org/
          reservation/travel">
        <p:departure>
          <p:departing>LGA</p:departing>
        </p:departure>
        <p:return>
          <p:arriving>EWR</p:arriving>
        </p:return>
      </p:itinerary>
    </env:Body>
  </env:Envelope>
  <x xmlns='jabber:x:oob'>
    <url>http://example.org/me.png</url>
    <desc>John Q. Public</desc>
  </x>
</message>
```

Alternatively, if all else fails, the file may be included as a SOAP representation header:

Listing 13: IQ-set with SOAP representation header

```
<iq from='requester@example.com/soap-client'
    id='soap2'
    to='responder@example.com/soap-server'
    type='set'>
  <env:Envelope xmlns:env='http://www.w3.org/2003/05/soap-envelope'
                xmlns:rep='http://www.w3.org/2004/08/representation'
                xmlns:xmlmime='http://www.w3.org/2004/11/xmlmime'>
    <env:Header>
      <rep:Representation resource='http://example.org/me.png'>
        <rep:Data xmlmime:contentType='image/png'>/aWKKapGGyQ=</
            rep:Data>
      </rep:Representation>
      <m:reservation
          xmlns:m="http://travelcompany.example.org/reservation"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
          env:mustUnderstand="true">
        <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</
            m:reference>
        <m:dateAndTime>2001-11-29T13:36:50.000-05:00</m:dateAndTime>
      </m:reservation>
      <n:passenger xmlns:n="http://mycompany.example.com/employees"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
          env:mustUnderstand="true">
        <n:name>Ake Jogvan Ovind</n:name>
      </n:passenger>
    </env:Header>
    <env:Body>
```

```
      <p:itinerary xmlns:p="http://travelcompany.example.org/
          reservation/travel">
        <p:departure>
          <p:departing>LGA</p:departing>
        </p:departure>
        <p:return>
          <p:arriving>EWR</p:arriving>
        </p:return>
      </p:itinerary>
      <x:MyData xmlns:x='http://example.org/mystuff'>
        <x:name>John Q. Public</x:name>
        <x:img src='http://example.org/me.png'/>
      </x:MyData>
    </env:Body>
  </env:Envelope>
</iq>
```

Naturally, in order to maximize the likelihood that the receiver will be able to retrieve the file, the sender MAY include the SI-pub extension, out-of-band-data extension, and SOAP representation header in the message stanza:

Listing 14: Sender sends message with SI-pub, OOB, and representation header

```
<message from='requester@example.com/soap-client'
         id='soap2'
         to='responder@example.com/soap-server'>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
                xmlns:rep="http://www.w3.org/2004/08/representation"
                xmlns:xmlmime="http://www.w3.org/2004/11/xmlmime">
    <env:Header>
      <rep:Representation resource='http://example.org/me.png'>
        <rep:Data xmlmime:contentType='image/png'>/aWKKapGGyQ=</
            rep:Data>
      </rep:Representation>
      <m:reservation
          xmlns:m="http://travelcompany.example.org/reservation"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
          env:mustUnderstand="true">
        <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</
            m:reference>
        <m:dateAndTime>2001-11-29T13:36:50.000-05:00</m:dateAndTime>
      </m:reservation>
      <n:passenger xmlns:n="http://mycompany.example.com/employees"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
          env:mustUnderstand="true">
        <n:name>Ake Jogvan Ovind</n:name>
      </n:passenger>
    </env:Header>
    <env:Body>
```

```
      <p:itinerary xmlns:p="http://travelcompany.example.org/
          reservation/travel">
        <p:departure>
         <p:departing>LGA</p:departing>
        </p:departure>
        <p:return>
          <p:arriving>EWR</p:arriving>
        </p:return>
      </p:itinerary>
      <x:MyData xmlns:x='http://example.org/mystuff'>
        <x:name>John Q. Public</x:name>
        <x:img src='http://example.org/me.png'/>
      </x:MyData>
    </env:Body>
  </env:Envelope>
  <sipub xmlns='http://jabber.org/protocol/sipub'
         id='publish-2345'
         mime-type='image/png'
         profile='http://jabber.org/protocol/si/profile/file-transfer'
            >
    <file xmlns='http://jabber.org/protocol/si/profile/file-transfer'
          name='me.png'
          size='4238'
          date='2005-11-01T23:11Z'/>
  </sipub>
  <x xmlns='jabber:x:oob'>
    <url>http://example.org/me.png</url>
    <desc>John Q. Public</desc>
  </x>
</message>
```

3.4 Specifying a WSDL Definition

WSDL [10] provides a machine-readable, formal description of web services operations, protocol bindings, and end points (i.e., network addresses). WSDL definitions attempt to specify a loose coupling of SOAP envelopes and their transports in order to maintain their independence and flexibility.
The definition of an XMPP SOAP transport in WSDL is straightforward. The following rules apply:

1. The 'transport' attribute of the <soap:binding> element MUST be set to "http://jabber.org/protocol/soap".
2. The 'style' attribute of the <soap:binding> element SHOULD be set to "document".

[10] WSDL 1.1 Specification <http://www.w3.org/TR/wsdl>.

3. The 'soapAction' attribute of the <soap:operation> element MAY be used; if so, it SHOULD be transported in an appropriate env:Header element for compatibility with the HTTP transport.

4. A valid XMPP URI/IRI (see RFC 5122 [11]) MUST be used for the 'location' attribute in the <soap:address> element.

The following is an example of a WSDL definition for an endpoint that supports the SOAP XMPP binding: a mythical service that translates Shakespearean English into selected modern languages and dialects.

Listing 15: Example of WSDL definition for a translation service that supports SOAP over XMPP

```
<definitions
    name='ShakespeareTranslation'
    targetNamespace='http://www.example.org/services/BabelFishService.
        wsdl'>
    xmlns='http://schemas.xmlsoap.org/wsdl/'
    xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
    xmlns:tns='http://shakespeare.lit/translation.wsdl'>

  <binding name='ShakespeareTranslationSoap' type='
      tns:TranslationPortType'>
    <soap:binding style='document' transport='http://jabber.org/
        protocol/soap'/>
    <operation name='Translate'>
      <input>...</input>
      <output>...</output>
    </operation>
  </binding>

  <service name='ShakespeareTranslationService'>
    <documentation>Translates Shakespearean text.</documentation>
    <port name='TranslationPort' binding='
        tns:ShakespeareTranslationSoap'>
      <soap:address location='xmpp:translation@shakespeare.lit'/>
    </port>
  </service>

</definitions>
```

Although there is no standard procedure for publishing WSDL documents, usually they are made available through HTTP at some URL discoverable with public registries such as UDDI servers. WSDL descriptions for XMPP bindings MAY follow the same publishing process, or MAY be discoverable through Jabber/XMPP specific mechanisms such as Service Discovery

[11] RFC 5122: Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) for the Extensible Messaging and Presence Protocol (XMPP) <http://tools.ietf.org/html/rfc5122>.

(XEP-0030) [12] or Publish-Subscribe (XEP-0060) [13].

4 SOAP XMPP Binding

Section 4 of SOAP Version 1.2 Part 1 [14] defines a SOAP Protocol Binding Framework; two instantiations of that framework are the SOAP HTTP Binding (specified in Section 7 of SOAP Version 1.2 Part 2 [15]) and the SOAP Email Binding [16]. (Additionally, a binding to BEEP is described in RFC 4227.) As an alternative to the HTTP and Email bindings, this section formally defines the SOAP XMPP Binding in accordance with the SOAP Protocol Binding Framework.
Note: The SOAP XMPP Binding is optional, and SOAP nodes are not required to implement it. A SOAP node that correctly and completely implements the SOAP XMPP Binding as described herein may be said to "conform to the SOAP 1.2 XMPP Binding".

4.1 Binding Name

The SOAP XMPP Binding is identified by the following URI:

- http://jabber.org/protocol/soap

4.2 Supported Features

XMPP is a pure XML streaming protocol used to exchange snippets of structured data called "XML stanzas" (see RFC 6120) between any two network endpoints.
Because XMPP is a direct messaging protocol, it does not possess the equivalent of web methods such as the HTTP GET, PUT, POST, and DELETE methods. Therefore, it is NOT RECOMMENDED for a SOAP node that supports only the SOAP XMPP Binding to provide the "SOAP Web Method Feature" described in Section 6.4 of SOAP Version 1.2 Part 2. (A SOAP gateway between XMPP and HTTP should support the SOAP Web Method Feature in order to ensure interoperability; however, description of such gateways is outside the scope of this document.)
Because XMPP is a pure XML protocol, it does not use MIME types (RFC 2045 [17]) or XML media types (RFC 3023 [18]), but rather sends XML directly over the wire. Therefore, it is NOT RECOMMENDED for a SOAP node that supports only the SOAP XMPP Binding to provide the "SOAP Action Feature" described in Section 6.5 of SOAP Version 1.2 Part 2. (A SOAP

[12] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[13] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[14] SOAP Version 1.2 Part 1: Messaging <http://www.w3.org/TR/soap12-part1>.
[15] SOAP Version 1.2 Part 2: Adjuncts <http://www.w3.org/TR/soap12-part2>.
[16] SOAP Version 1.2 Email Binding <http://www.w3.org/TR/soap12-email>.
[17] RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies <http://tools.ietf.org/html/rfc2045>.
[18] RFC 3023: XML Media Types <http://tools.ietf.org/html/rfc3023>.

gateway between XMPP and HTTP should support the SOAP Action Feature in order to ensure interoperability; however, description of such gateways is outside the scope of this document.)

4.3 Supported Message Exchange Patterns

XMPP inherently provides request-response semantics via the <iq/> stanza type and <message/> stanza type, where the <iq/> stanza type requires more formality regarding preservation of request-response semantics in the context of synchronous communications, whereas the <message/> stanza provides a looser mapping to request-response semantics as well as the ability to ensure store-and-forward capabilities similar to those provided by email (see the Implementation Notes section of this document). Because both stanza types support request-response semantics, an implementation of the SOAP XMPP Binding MUST support only the following message exchange pattern (MEP) defined in the core SOAP 1.2 specification:

- http://www.w3.org/2003/05/soap/mep/request-response/

4.4 Operation of Request-Response Message Exchange Pattern

The request-response message exchange pattern is described in Section 6.2 of SOAP Version 1.1 Part 2. For binding instances conforming to the specification of the SOAP XMPP Binding:

- A SOAP node instantiated at an XMPP entity may assume the role (i.e., the http://www.w3.org/2003/05/soap/bindingFramework/ExchangeContext/Role property) of "RequestingSOAPNode".
- A SOAP node instantiated at an XMPP entity may assume the role (i.e., the http://www.w3.org/2003/05/soap/bindingFramework/ExchangeContext/Role property) of "RespondingSOAPNode".

The remainder of this section describes the message exchange pattern (MEP) state machine and its relation to XMPP as described in RFC 6120. For the sake of brevity, relative URIs are used (the base URI being http://www.w3.org/2003/05/soap/bindingFramework/ExchangeContext/Role), the string "fail:" is used as a conventional prefix for the namespace http://www.example.org/2001/12/soap/mep/FailureReasons/, and the string "reqresp:" is used as a conventional prefix for the namespace http://www.example.org/2001/12/soap/mep/request-response/. In the state tables below, the states are defined as values of the http://www.w3.org/2003/05/soap/bindingFramework/ExchangeContext/State property (see Section 6.2 of SOAP Version 1.2 Part 2) and are of type xs:anyURI.

4.4.1 Behavior of Requesting SOAP Node

The overall flow of the behavior of a Requesting SOAP Node follows the outline state machine description contained in Section 6.2 of SOAP Version 1.2 Part 2. The following subsections describe each state in more detail, where ”Requesting SOAP Node” is to be understood as a logical entity made up of the binding and the local SOAP node associated with the XMPP entity that generates a SOAP request.
The following table formally describes the ”Init” state of the Requesting SOAP Node in the SOAP XMPP Binding:

Feature	Value / Description
State Name	Init
Description	Formulate and send request message
Pre-Conditions	Control of the outbound transport message exchange context is transferred from the local SOAP node to the binding
Actions	Formulate and send XMPP <iq/> or <message/> request stanza (see table ”Init: XMPP Fields (Requesting)”) that encappsulates SOAP envelope transferred from local SOAP node to binding
Post-Conditions	None
Transitions	See table ”Init: Transitions (Requesting)”

In the ”Init” state, an XMPP stanza (either <iq/> or <message/>) is formulated by the Requesting SOAP Node according to the following table:

Field	Value / Description
XMPP Method	For XMPP <iq/> stanzas, the value of the XMPP ’type’ attribute MUST be ”set”; does not apply to XMPP <message/> stanzas
Originator	The XMPP address (JID) carried in the reqresp:ImmediateSender property of the message exchange context is encapsulated as the value of the XMPP ’from’ attribute; normally this is set by the XMPP server to which the originator connects
Destination	The XMPP address (JID) carried in the reqresp:ImmediateDestination property of the message exchange context is encapsulated as the value of the XMPP ’to’ attribute
Correlation Request Message ID	As required for XMPP <iq/> stanzas in general and required for XMPP <message/> stanzas sent in the context of the SOAP XMPP Binding, a correlation request message ID is generated by the sender and encapsulated as the value of the XMPP ’id’ attribute

Field	Value / Description
XMPP Stanza Contents	The XML of the SOAP envelope carried in the reqresp:OutboundMessage property of the transport message exchange context is encapsulated as a direct child element of the XMPP <iq/> or <message/> stanza

The following table summarizes the transitions from the ”Init” state of the Requesting SOAP Node:

Event / Condition	Next State	Failure Reason
Request Successfully Sent	Requesting	N/A
Failure to Send Request	Fail	fail:TransmissionFailure

The following table formally describes the ”Requesting” state of the Requesting SOAP Node in the SOAP XMPP Binding:

Feature	Value / Description
State Name	Requesting
Description	Waiting for correlated XMPP response (Request Message completely sent on exit from Init state)
Pre-Conditions	Completion of Init state
Actions	Wait for a receive XMPP response stanza
Post-Conditions	Instantiate or replace the reqresp:ImmediateSender property with an XMPP address (JID) that denotes the sender of the XMPP response stanza
Transitions	See table ”Requesting: Transitions”

The following table summarizes the transitions from the ”Requesting” state of the Requesting SOAP Node:

Event / Condition	Next State	Failure Reason
Received Correlated XMPP Response	Sending+Receiving	N/A
Reception Failure (various XMPP errors)	Fail	fail:ReceptionFailure

For a listing of relevant XMPP error conditions, refer to RFC 6120.
The following table formally describes the ”Sending+Receiving” state of the Requesting SOAP Node in the SOAP XMPP Binding:

Feature	Value / Description
State Name	Sending+Receiving
Description	Receive correlated XMPP response including SOAP envelope
Pre-Conditions	Completion of Receiving state
Actions	Process XMPP <iq/> or <message/> response stanza and included SOAP envelope, instantiating or replacing the reqresp:InboundMessage property with an infoset representation of the SOAP envelope contained in the XMPP response stanza
Post-Conditions	Control of the inbound transport message exchange context is transferred from the binding to the local SOAP node
Transitions	See table ”Sending+Receiving: Transitions”

The following table summarizes the transitions from the ”Sending+Receiving” state of the Requesting SOAP Node:

Event / Condition	Next State	Failure Reason
Received Well-Formed Response Message	Success	N/A
Reception Failure (various XMPP errors)	Fail	fail:ReceptionFailure
Malformed Response Message (invalid SOAP envelope)	Fail	fail:BadRequestMessage

For a listing of relevant XMPP error conditions, refer to RFC 6120.
A given instance of a request-response transport message exchange terminates when the state ”Success” or ”Fail” is reached; control over the transport message exchange context returns to the Requesting SOAP Node.

4.4.2 Behavior of Responding SOAP Node

The overall flow of the behavior of a Responding SOAP Node follows the outline state machine description contained in Section 6.2 of SOAP Version 1.2 Part 2. The following subsections describe each state in more detail, where ”Responding SOAP Node” is to be understood as a logical entity made up of the binding and the local SOAP node associated with the XMPP entity that responds to a SOAP request.
The following table formally describes the ”Init” state of the Responding SOAP Node in the

SOAP XMPP Binding:

Feature	Value / Description
State Name	Init
Description	Receive request message
Pre-Conditions	None
Actions	Receive and validate inbound XMPP <iq/> or <message/> request stanza; instantiate or replace the reqresp:ImmediateSender property with an XMPP address (JID) that denotes the sender of the XMPP request; instantiate or replace the reqresp:InboundMessage property with an infoset representation of the included SOAP envelope
Post-Conditions	Control of the inbound transport message exchange context is transferred from the binding to the local SOAP node
Transitions	See table ”Init: Transitions (Responding)”

The following table summarizes the transitions from the ”Init” state of the Responding SOAP Node:

Event / Condition	Next State	Failure Reason
Received Well-Formed Request Message	Receiving	N/A
Reception Failure (various XMPP errors)	Fail	fail:ReceptionFailure
Malformed Response Message (invalid SOAP envelope)	Fail	fail:BadRequestMessage

For a listing of relevant XMPP error conditions, refer to RFC 6120.
The following table formally describes the ”Receiving” state of the Responding SOAP Node in the SOAP XMPP Binding:

Feature	Value / Description
State Name	Receiving
Description	Waiting for local SOAP node to return response message
Pre-Conditions	Completion of Init state
Actions	None
Post-Conditions	Control of the outbound transport message exchange context is transferred from the local SOAP node to the binding
Transitions	See table ”Receiving: Transitions”

The following table summarizes the transitions from the ”Receiving” state of the Responding SOAP Node:

Event / Condition	Next State	Failure Reason
Response Message Becomes Available	Receiving+Sending	N/A

The following table formally describes the ”Receiving+Sending” state of the Responding SOAP Node in the SOAP XMPP Binding:

Feature	Value / Description
State Name	Receiving+Sending
Description	Waiting for local SOAP node to return response message
Pre-Conditions	Completion of Receiving state
Actions	Formulate and send XMPP <iq/> or <message/> response stanza (see table ”Receiving+Sending: XMPP Fields”)
Post-Conditions	None
Transitions	See table ”Receiving+Sending: Transitions”

In the ”Receiving+Sending” state, an XMPP stanza (either <iq/> or <message/>) is formulated by the Responding SOAP Node according to the following table:

Field	Value / Description
XMPP Method	For XMPP <iq/> stanzas, the value of the XMPP ’type’ attribute MUST be ”result”; does not apply to XMPP <message/> stanzas
Originator	The XMPP address (JID) carried in the reqresp:ImmediateSender property of the message exchange context is encapsulated as the value of the XMPP ’from’ attribute; normally set by the XMPP server to which the originator connects
Destination	The XMPP address (JID) carried in the reqresp:ImmediateDestination property of the message exchange context is encapsulated as the value of the XMPP ’to’ attribute

Field	Value / Description
Correlation Request Message ID	As required for XMPP <iq/> stanzas in general and required for XMPP <message/> stanzas sent in the context of the SOAP XMPP Binding, the correlation request message ID is copied from the ID of the request and encapsulated as the value of the XMPP 'id' attribute
XMPP Stanza Contents	The XML of the SOAP envelope carried in the reqresp:OutboundMessage property of the transport message exchange context is encapsulated as a direct child element of the XMPP <iq/> or <message/> stanza

The following table summarizes the transitions from the "Receiving+Sending" state of the Responding SOAP Node:

Event / Condition	Next State	Failure Reason
Response Message Successfully Sent	Success	N/A
Failure to Send Response Message	Fail	fail:TransmissionFailure

A given instance of a request-response transport message exchange terminates when the state "Success" or "Fail" is reached; from the perspective of the Responding SOAP Node, the transport message exchange has completed.

5 W3C Considerations

The main body of text that addresses the requirements of the W3C with regard to SOAP bindings is provided in the SOAP XMPP Binding section of this document. The current section addresses only the topic of organizational interaction between the W3C and the XMPP Standards Foundation (XSF) [19] regarding the SOAP XMPP Binding.

5.1 W3C Review

As was done with XHTML-IM (XEP-0071) [20], the SOAP XMPP Binding defined herein has been reviewed informally by one or more appropriate experts from the W3C before the

[19] The XMPP Standards Foundation (XSF) is an independent, non-profit membership organization that develops open extensions to the IETF's Extensible Messaging and Presence Protocol (XMPP). For further information, see <https://xmpp.org/about/xmpp-standards-foundation>.

[20] XEP-0071: XHTML-IM <https://xmpp.org/extensions/xep-0071.html>.

XMPP Council [21] advanced it to a status of Draft within the XSF's standards process. Before this specification proceeds to a status of Final within the XSF's standards process, it should undergo a formal review through communication with the W3C's XML Protocol Working Group. To that end, revised versions of this specification will be announced on the W3C's public xml-dist-app@w3.org mailing list.

5.2 SOAP Versioning

This specification addresses SOAP 1.2 only. This specification may be superseded or supplemented in the future by a XMPP Extension Protocol specification that defines methods for encapsulating content defined by future versions of SOAP as published by the W3C.

5.3 XML Versioning

Per RFC 6120, XMPP supports XML 1.0 only. If future versions of XMPP support XML 1.1 or subsequent versions, this specification may be modified to address handling of SOAP messages that are encoded in versions other than XML 1.0.

6 Error Handling

SOAP provides its own encoding scheme for errors due to message processing or application execution, and it uses SOAP envelopes for reporting. In the SOAP HTTP Binding, these errors are mapped to corresponding HTTP status codes. In the SOAP XMPP Binding, they are mapped to the catch-all XMPP error of <undefined-condition/> along with application-specific error condition elements qualified by the 'http://jabber.org/protocol/soap#fault' namespace (this is consistent with RFC 6120, see also Error Condition Mappings (XEP-0086) [22]). The element names of these application-specific error conditions map directly to the SOAP fault codes specified in Section 5.4.6 of SOAP Version 1.2 Part 1.
The following table provides a mapping between SOAP, HTTP, and application-specific XMPP errors.

SOAP Fault	HTTP Status Code	XMPP Application Error
env:DataEncodingUnknown	500	<DataEncodingUnknown/>
env:MustUnderstand	500	<MustUnderstand/>
env:Receiver	500	<Receiver/>
env:Sender	400	<Sender/>

[21] The XMPP Council is a technical steering committee, authorized by the XSF Board of Directors and elected by XSF members, that approves of new XMPP Extensions Protocols and oversees the XSF's standards process. For further information, see <https://xmpp.org/about/xmpp-standards-foundation#council>.
[22] XEP-0086: Error Condition Mappings <https://xmpp.org/extensions/xep-0086.html>.

SOAP Fault	HTTP Status Code	XMPP Application Error
env:VersionMismatch	500	<VersionMismatch/>

Note: When errors are due to the XMPP transport protocol alone and not to the application layer defined by SOAP, errors MUST be reported with standard XMPP error codes only instead of the XMPP <undefined-condition/> condition plus application-specific condition.

7 Business Rules

7.1 Encoding

Because XMPP does not require the parsing of arbitrary and complete XML documents and does not require implementations to support the full XML specification, transported SOAP envelopes MUST comply with the XML restrictions specified in RFC 6120. In particular, all envelope elements MUST be properly namespaced (SOAP allows elements within the default namespace, but they are deprecated since SOAP 1.2).
SOAP envelopes may contain arbitrary data encoded in valid XML as well as byte arrays encoded with SOAP-specific elements. The SOAP specification recommends to encode byte arrays in Base 64 (see RFC 3548 [23]), with the result that envelopes with binary data can be transported within regular XMPP stanzas. All the remaining PCDATA MUST be encoded as UTF-8 in order to match the XML stream encoding.

8 Security Considerations

SOAP has been supplemented by several support protocols that help ensure message integrity and confidentiality (WS-Security [24]) as well as transaction management for failing message exchanges (see WS-Transaction). These protocols are all based on SOAP messages and take into account that the underlying protocols can be unreliable and not trusted, thus there are no arguments against their application with XMPP. Alternatively, implementations MAY use native XMPP security such as XMPP E2E [25].

[23] RFC 3548: The Base16, Base32, and Base64 Data Encodings <http://tools.ietf.org/html/rfc3548>.
[24] WS-Security <http://msdn.microsoft.com/ws/2002/04/Security/>.
[25] RFC 3923: End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP) <http://tools.ietf.org/html/rfc3923>.

9 IANA Considerations

No interaction with the Internet Assigned Numbers Authority (IANA) [26] is required by this document.

10 XMPP Registrar Considerations

10.1 Protocol Namespaces

The XMPP Registrar [27] includes 'http://jabber.org/protocol/soap' and 'http://jabber.org/protocol/soap#fault' in its registry of protocol namespaces.

10.2 Service Discovery Identity

The XMPP Registrar includes a Service Discovery type of "soap" within the "automation" category.
The registry submission is as follows:

```
<category>
  <name>automation</name>
  <type>
    <name>soap</name>
    <desc>A SOAP receiver (either intermediate or ultimate).</desc>
    <doc>XEP-0072</doc>
  </type>
</category>
```

11 XML Schema

11.1 SOAP Envelope

Because the SOAP envelope is included as a first-level child element of an <iq/> or <message/> stanza via standard XMPP extension mechanisms, an XML schema is not required for this document. An XML schema for the SOAP envelope element is provided at <http://www.w3.org/2003/05/soap-envelope/>.

[26] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.

[27] The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <https://xmpp.org/registrar/>.

11.2 Application-Specific XMPP Errors

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='http://jabber.org/protocol/soap#fault'
    xmlns='http://jabber.org/protocol/soap#fault'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0072: http://www.xmpp.org/extensions/xep-0072.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='DataEncodingUnknown' type='empty'/>
  <xs:element name='MustUnderstand' type='empty'/>
  <xs:element name='Receiver' type='empty'/>
  <xs:element name='Sender' type='empty'/>
  <xs:element name='VersionMismatch' type='empty'/>

  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

12 Implementation Notes

This section is non-normative.

An XMPP entity that supports the SOAP XMPP binding could function as a "SOAP intermediary" that hands a SOAP message off to some other deployment for subsequent processing (HTTP, email, a specialized enterprise messaging platform, etc.) rather than functioning as the "ultimate SOAP receiver" for the message (as these terms are defined in Section 1.5.3 of SOAP Version 1.2 Part 1). If the intended recipient functions as a SOAP intermediary, implementations should be aware that subsequent processing may alter the representation of SOAP messages.

As an example, consider a component that functions as a gateway between XMPP-based and HTTP-based web services. Its purpose might be to mix HTTP and XMPP for web services and to invoke any web services already accessible through HTTP from XMPP clients.

WS-Routing, whose aim is to dynamically compose SOAP message paths and processing sequences, can be used in order to reference web services outside of an XMPP network from

within it. WS-Routing extends SOAP Envelope Headers with the <path/> element, which specifies the following for the message: the sender's URL (<from/>), the final destination's URL (<to/>), a forward (<forward/>) path with an arbitrary number of intermediaries (<via/>), and an optional return path (<reverse/>). Each intermediary MUST process the <path/> header and update it accordingly to the already performed path; moreover it MAY process the Body of the message.
A SOAP message originated by an XMPP entity ('xmpp:orig@A.example.com/soap'), and directed to an end point accessible through HTTP ('http://C.example.net/some/endpoint'), could be built using a <path/> header having:

1. the <to/> element set to 'http://C.example.net/some/endpoint'
2. one <via/> element set to an HTTP<->XMPP gateway, such as 'xmpp:soapgw@B.example.org/soap', in the forward path
3. an appropriate SOAP action in the <action> element of the <path/> header (this may be required by the HTTP end point)
4. a blank return path

Then the SOAP message can be sent within an <iq/> stanza to the gateway's JID. The gateway processes the SOAP headers, and looking through the headers it discovers that it must act only as intermediary. From the <to/> element it reads the URL of the final end point, extracts the SOAP action, changes the path removing the step already performed, and issues an HTTP request with the modified envelope and appropriate HTTP headers. Once it has received a response, it prepares a new <iq/> stanza of type "result" or "error" and sends its reply to the original requester. The following example shows the possible SOAP headers of the described process.

Listing 16: Gateway-generated SOAP headers

```
<S:Envelope xmlns:S='http://www.w3.org/2003/05/soap-envelope'>
   <S:Header>
      <m:path xmlns:m='http://www.soap.org/path'>
         <m:action>http://im.example.org/chat</m:action>
         <m:to>http://C.example.net/some/endpoint</m:to>
         <m:forward>
            <m:via>xmpp:soapgw@B.example.org/soap</m:via>
         </m:forward>
         <m:reverse>
            <m:via/>
         </m:reverse>
         <m:from>xmpp:orig@A.example.com/soap</m:from>
         <m:id>uuid:84b9f5d0-33fb-4a81-b02b-5b760641c1d6</m:id>
      </m:path>
   </S:Header>
   <S:Body>
```

```
      ...
   </S:Body>
</S:Envelope>
```

Generic XMPP routers that conform to RFC 6120 may also ”store and forward” Jabber messages. This feature is usually called ”offline message handling”: the router makes a decision as to whether to deliver the message to the local intended recipient based on the recipient's presence, and if the recipient is offline when the router processes the message then it may store the message for delivery when the recipient next comes online (rather than returning an error to the sender). Although it is possible to write an XMPP router that directly supports the SOAP XMPP binding and implements the SOAP processing model, generic XMPP routers do not contain such support. Accordingly, generic XMPP routers will not forward an XMPP message to an alternate SOAP transport such as HTTP or SMTP, or provide other functions of a SOAP intermediary or ultimate receiver. When a generic XMPP router delivers a message to the intended recipient (whether immediately or as delayed in ”offline storage”) and the intended recipient supports the SOAP XMPP binding, SOAP processing is performed; such an intended recipient MAY act either as a SOAP intermediary or as an ultimate SOAP receiver.
With regarding to exchange of associated data, an XMPP entity that functions as a gateway to other SOAP bindings it SHOULD use W3C-recommended protocols for transporting SOAP attachments over non-XMPP SOAP bindings (e.g., HTTP and SMTP) when communicating with non-XMPP entities.

13 Acknowledgements

Many thanks to Noah Mendelsohn for his assistance regarding SOAP binding definitions and conformance issues. Thanks also to Michael Mahan and Rich Salz for their comments.
Some text in the SOAP XMPP Binding section of this document is closely modelled on Section 7 of SOAP Version 1.2 Part 2 and on SOAP Version 1.2 Email Binding.