XML Protocol SOAP Envelope and Encoding

Bin LI, Jieyu ZHAO
Institute of Information Science and Technology, Ningbo University

Abstract: As an important standard language, XML has been paid more and more attention to. In this paper, an XML protocol SOAP (Simple Object Access Protocol) Envelope, SOAP Encoding, SOAP RPC and the relationship between SOAP, XML and HTTP are introduced by referring to the materials on W3C.

Key Words: XML; SOAP; E-commerce

1. XML

In the field of E-commerce, a great amount of data should be processed. XML is a set of rules, guidelines, and conventions for designing text formats for such data. It makes files easy to be generated and read by a computer, and there are no ambiguities. Also it is an international standard with plenty of flexibility and platform-independency. XML is generated from SGML, which is standardized by ISO in 1986. Development of XML started in 1996 and it became a W3C standard in February 1998. The designers of XML simply took the best parts of SGML, and produced something that is not less powerful than SGML, but more regular and simpler to use.

XML separates representations and data, improves the user interface and has integrating functions, scalability, simplicity and open standard. Actually, XML is a standard that fits the Internet well, which has interoperability, multi-system supporting and browser characteristics.

2. XML Protocol

2.1 XML protocol

XML is a language based on identifiers and it allows the users to define their own marks. This makes it possible for various organizations to define their own rules on it. XML Protocol can be used to develop a rule that allows two or more peers to communicate in a distributed environment, using XML as its encapsulation language. It provides robustness, simplicity, reusability and interoperability on a layered architecture.

2.2 XML protocol at present

As a result of the increased interest in XML protocols, W3C and others have recently organized several activities on various XML protocol. Some important activities are as follows:
Sept 2000: Creation of the XML Protocol Working Group
May 15-19, 2000: W3C organizes a panel on XML Protocols at WWW9 in Amsterdam
May 2000: W3C provides an initial comparison of the various initiatives already in the marketplace
Mar 2000: Michael Condry, Sun, organizes a B2BXML BOF at IETF 47 in Adelaide, Australia, 26-31
Feb/Mar 2000: W3C organizes an XML protocols BOF at XTech 2000
Feb 2000: W3C provides an interim plan for XML protocol work

3. SOAP Envelope and Encoding

3.1 SOAP constitution

SOAP is a lightweight XML protocol for the exchange of information in a decentralized, distributed environment. It attracts much attention of many famous companies and is highly recommended. SOAP consists of three parts: an envelope, a set of encoding rules and a convention for representing remote procedure calls and responses.

Although these parts are described together as part of SOAP, they are functionally independent. In particular, the envelope and the encoding rules are defined in different namespaces in order to promote simplicity through modularity.

3.2 The SOAP Message Exchange Model

SOAP messages are fundamentally one-way transmissions from a sender to a receiver, but SOAP messages are often combined to implement patterns such as request/response. SOAP implementations can be optimized to exploit the unique characteristics of particular network systems. For example, the HTTP binding, it uses the same connection as the inbound request.

Regardless of the protocol to which SOAP is bound, messages are routed along a so-called "message path", which includes one or more intermediate nodes in addition to the ultimate destination.

A SOAP application receiving a SOAP message process that message by performing the following actions in the order listed below:

1. Identify all parts of the SOAP message intended for the application.
2. Verify that all mandatory parts identified in step 1 are supported by the application for this message and process them accordingly. If this is not the case then discard the
message. The processor MAY ignore optional parts identified in step 1 without affecting the outcome of the processing.
3. If the SOAP application is not the ultimate destination of the message then remove all parts identified in step 1 before forwarding the message.

Processing a message or a part of a message requires that the SOAP processor understands the exchange pattern being used (one way, request/response, multicast, etc.), the role of the recipient in that pattern, the employment (if any) of RPC mechanisms, the representation or encoding of data, as well as other semantics necessary for correct processing.

3.3 Relation to XML

All SOAP messages are encoded using XML. A SOAP application should include the proper SOAP namespace on all elements and attributes defined by SOAP in messages that it generates. A SOAP application must be able to process SOAP namespaces in messages that it receives. SOAP defines two namespaces:
The SOAP envelope has the namespace identifier
"http://schemas.xmlsoap.org/soap/envelope/
"The SOAP serialization has the namespace identifier
"http://schemas.xmlsoap.org/soap/encoding/

A SOAP message must not contain a Document Type Declaration and Processing Instructions. SOAP uses the local, unqualified "id" attribute of type "ID" to specify the unique identifier of an encoded element. SOAP uses the local, unqualified attribute "href" of type "uri-reference" to specify a reference to that value, in a manner conforming to the XML Specification.

With the exception of the SOAP mustUnderstand attribute and the SOAP actor attribute, it is generally permissible to have attributes and their values appear in XML instances or alternatively in schemas, with equal effect.

3.4 SOAP Envelope

A SOAP message is an XML document that consists of a mandatory SOAP envelope, an optional SOAP header, and a mandatory SOAP body as the following:
The Envelope is the top element of the XML document representing the message.
The Header is a generic mechanism for adding features to a SOAP message without prior agreement between the communicating parties. SOAP defines a few attributes that can be used to indicate some extra information.
The Body is a container for mandatory information intended for the ultimate recipient of the message. SOAP defines one element for the body, which is the Fault element used for reporting errors.

The grammar rules are as follows:
1. Envelope
The element name is "Envelope". The element MUST be present in a SOAP message
The element MAY contain namespace declarations as well as additional attributes. If
present, such additional attributes MUST be namespace-qualified. Similarly, the
element MAY contain additional sub elements. If present these elements MUST be
namespace-qualified and MUST follow the SOAP Body element.

2. Header
The element name is "Header". The element MAY be present in a SOAP message. If
present, the element MUST be the first immediate child element of a SOAP Envelope
element. The element MAY contain a set of header entries each being an immediate
child element of the SOAP Header element. All immediate child elements of the
SOAP Header element MUST be namespace-qualified.

3. Body
The element name is "Body". The element MUST be present in a SOAP message and
MUST be an immediate child element of a SOAP Envelope element. It MUST
directly follow the SOAP Header element if present. Otherwise it MUST be the first
immediate child element of the SOAP Envelope element. The element MAY contain a
set of body entries each being an immediate child element of the SOAP Body element.
Immediate child elements of the SOAP Body element MAY be namespace-qualified.

SOAP defines the SOAP Fault element, which is used to indicate error messages.
The SOAP encodingStyle global attribute can be used to indicate the serialization
rules used in a SOAP message. This attribute MAY appear on any element, and is
scoped to that element's contents and all child elements not themselves containing
such an attribute, much as an XML namespace declaration is scoped.

SOAP provides a flexible mechanism for extending a message in a decentralized and
modular way without prior knowledge between the communicating parties. Typical
examples of extensions that can be implemented as header entries are authentication,
transaction management, payment etc.

The encoding rules for header entries are as follows:

1. A header entry is identified by its fully qualified element name, which consists of
the namespace URI and the local name. All immediate child elements of the SOAP
Header element MUST be namespace-qualified.

2. The SOAP encodingStyle attribute MAY be used to indicate the encoding style
used for the header entries.

3. The SOAP mustUnderstand attribute and SOAP actor attribute MAY be used to
indicate how to process the entry and by whom.

A SOAP message travels from the originator to the ultimate destination, potentially by
passing through a set of SOAP intermediaries along the message path. A SOAP
intermediary is an application that is capable of both receiving and forwarding SOAP
messages. Both intermediaries as well as the ultimate destination are identified by a
URI.

The SOAP actor global attribute can be used to indicate the recipient of a header
element. The value of the SOAP actor attribute is a URI.
The SOAP mustUnderstand global attribute can be used to indicate whether a header
entry is mandatory or optional for the recipient to process. The value of the
mustUnderstand attribute is either "1" or "0". The absence of the SOAP
mustUnderstand attribute is semantically equivalent to its presence with the value "0".

4
If a header element is tagged with a SOAP mustUnderstand attribute with a value of "1", the recipient of that header entry either MUST obey the semantics (as conveyed by the fully qualified name of the element) and process correctly to those semantics.

The SOAP Body element provides a simple mechanism for exchanging mandatory information intended for the ultimate recipient of the message. Typical uses of the Body element include marshalling RPC calls and error reporting.

All immediate child elements of the Body element are called body entries and each body entry is encoded as an independent element within the SOAP Body element.

The encoding rules for body entries are as follows:

1. A body entry is identified by its fully qualified element name, which consists of the namespace URI and the local name. Immediate child elements of the SOAP Body element MAY be namespace-qualified.

2. The SOAP encodingStyle attribute MAY be used to indicate the encoding style used for the body entries.

SOAP defines one body entry, which is the Fault entry used for reporting errors. The SOAP Fault element is used to carry error and/or status information within a SOAP message. If present, the SOAP Fault element MUST appear as a body entry and MUST NOT appear more than once within a Body element. The SOAP Fault element defines four sub-elements.

### 3.5 SOAP Encoding

The SOAP encoding style is based on a simple type system that is a generalization of the common features found in type systems in programming languages, databases and semi-structured data. A type either is a simple type or is a compound type constructed as a composite of several parts, each with a type.

The namespace identifier defined is: "http://schemas.xmlsoap.org/soap/encoding/". XML allows very flexible encoding of data. SOAP defines a narrower set of rules for encoding. The types of encoding includes: Value, Simple Value, Compound Value, Array, Struct, Simple Type, Compound Type, etc.

The above rules are much alike some programming languages. An example of a compound type is the class of purchase order values sharing the same accessors (shipTo, totalCost, etc.). This makes the type closer to the applications like e-commerce.

The following is an example of a struct of type "Book":

```xml
<e:Book>
   <author>Henry Ford</author>
   <preface>Prefatory text</preface>
   <intro>This is a book.</intro>
</e:Book>
```

And this is a schema fragment describing the above structure:

```xml
<element name="Book">
   <complexType>
      <element name="author" type="xsd:string"/>
   </complexType>
</element>
```
3.6 Using SOAP in HTTP

Binding SOAP to HTTP provides the advantage of being able to use the formalism and decentralized flexibility of SOAP with the rich feature set of HTTP. Carrying SOAP in HTTP does not mean that SOAP overrides existing semantics of HTTP but rather that the semantics of SOAP over HTTP maps naturally to HTTP semantics. SOAP naturally follows the HTTP request/response message model providing SOAP request parameters in a HTTP request and SOAP response parameters in a HTTP response. Note, however, that SOAP intermediaries are NOT the same as HTTP intermediaries. That is, an HTTP intermediary addressed with the HTTP Connection header field cannot be expected to inspect or process the SOAP entity body carried in the HTTP request.

HTTP applications MUST use the media type "text/xml" according to RFC 2376 [3] when including SOAP entity bodies in HTTP messages.

SOAP HTTP binding only defines SOAP within HTTP POST requests. The SOAPAction HTTP can be used to indicate the intent of the SOAP HTTP request. The value is a URI identifying the intent. SOAP places no restrictions on the format or specificity of the URI or that it is resolvable. An HTTP client MUST use this header field when issuing a SOAP HTTP Request.

The following is an example:

```
soapaction = "SOAPAction" "::" [ "" URI-reference ""]
URI-reference = <as defined in RFC 2396 [4]>
```

SOAP HTTP follows the semantics of the HTTP Status codes for communicating status information in HTTP. For example, a 2xx status code indicates that the client's request including the SOAP component was successfully received, understood, and accepted etc. In case of a SOAP error while processing the request, the SOAP HTTP server MUST issue an HTTP 500 "Internal Server Error" response and include a SOAP message in the response containing a SOAP Fault element indicating the SOAP processing error.

A SOAP message MAY be used together with the HTTP Extension Framework in order to identify the presence and intent of a SOAP HTTP request.

3.7 Using SOAP for RPC

One of the design goals of SOAP is to encapsulate and exchange RPC calls using the extensibility and flexibility of XML.

Using SOAP for RPC is orthogonal to the SOAP protocol binding. In the case of using HTTP as the protocol binding, an RPC call maps naturally to an HTTP request and an RPC response maps to an HTTP response. However, using SOAP for RPC is
not limited to the HTTP protocol binding. RPC method calls and responses are both carried in the SOAP Body element. A method invocation and response is modeled as a struct, A method fault is encoded using the SOAP Fault element. Applications MAY process requests with missing parameters but also MAY return a fault. Additional information relevant to the encoding of a method request but not part of the formal method signature MAY be expressed in the RPC encoding. If so, it MUST be expressed as a supplement of the SOAP Header element. An example of the use of the header element is the passing of a transaction ID along with a message.

4. SOAP Example

In this example, a GetLastTradePrice SOAP request is sent to a StockQuote service. The request takes a string parameter, ticker symbol, and returns a float in the SOAP response. The SOAP Envelope element is the top element of the XML document representing the SOAP message. XML namespaces are used to disambiguate SOAP identifiers from application specific identifiers. The example illustrates the HTTP bindings. XML payload format in SOAP are entirely independent of the fact that the payload is carried in HTTP.

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The following is the response:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePriceResponse xmlns:m="Some-URI">
      <Price>34.5</Price>
    </m:GetLastTradePriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
References
[8] 《微软电子商务解决方案》人民邮电出版社 1999 年 12 月