Tutorial on Fast Web Services

This document provides tutorial material on Fast Web Services (**it is equivalent to Annex C of X.892 | ISO/IEC 24824-2**). Some of the advantages of using Fast Web Services are described. The differences between the conceptual and optimized processing of SOAP messages are highlighted, followed by an example. The example is based on a simple exchange in which a client sends a request message and receives a response message. The use of service descriptions is discussed, followed by an example service description (in WSDL 1.1 – see [2]) that describes the service provided by the messaging example.

C.1 Advantages of Fast Web Services

The Fast Web Services specification is based on the use of an ASN.1 definition of SOAP messages and their contents, and on the use of binary encodings of those messages. This provides the main advantage (fast computer processing and low message bandwidth) of Fast Web Services, but a number of further optimizations of XML SOAP are discussed below.

C.1.1 ASN.1 tools

ASN.1 tools can be used in the development of ASN.1 SOAP processors, whereas XML SOAP processors are, for the most part, written by hand, with the W3C XML Schema for SOAP used only as a guide, since XML binding tools are unlikely to aid in the development of optimal XML SOAP processors. The ASN.1 approach allows for a choice of either tools or hand crafting to develop SOAP processors, without any serious performance penalties, and with potential gains in time-to-market.

C.1.2 Optimized features

ASN.1 SOAP provides a number of optimization features (beyond compaction and efficient processing offered by the use of ASN.1 and PER – see ITU-T Rec. X.691 | ISO/IEC 8825-2) for SOAP nodes:

a) The body of an ASN.1 SOAP message is explicitly separated from the encoding of the fault of an ASN.1 SOAP message. This makes faults easier to identify and manage.

b) Recursive fault sub-code values for W3C SOAP messages are flattened to a sequence of fault sub-code values for ASN.1 SOAP messages. This enables a decoder to know how many fault sub-codes there are before decoding.

c) ASN.1 relative object identifiers can be used instead of qualified names. Messages for service descriptions can be annotated with relative object identifiers and such identifiers, when encoded, are generally much more compact than qualified names, resulting in smaller message sizes.

d) Default values for all attribute-related ASN.1 SOAP header block components are specified.

e) Enumerated values are used for W3C SOAP-specified fault codes instead of qualified names.

C.1.3 Compact messages and efficient processing

ASN.1 SOAP messages encoded using the ASN.1 Packed Encoding Rules generally provides Web services that require less processing power (and hence provide a higher transaction processing rate) and that require less network bandwidth than use of the character encoding of XML data. This can be advantageous in a number of domains:

a) Constrained devices, such as mobile phones, smart cards or even Radio-Frequency Identification (RFID) devices, which have limited processing power, memory and battery life.

NOTE – There is no equivalent Moore's law for battery technology (battery life is not doubling every 18 months).

b) Bandwidth-restricted systems, such as wireless networks.

NOTE – Radio frequencies for wireless networks, such as the mobile phone GSM (Global System for Mobile Communications) network, can be fixed for 10 years. There is no equivalent Moore's law for radio frequencies (bandwidth is not doubling every 18 months).

c) High throughput transaction systems, such as systems required to process a required number of SOAP messages per second from many clients.

C.1.4 Efficient processing for SOAP intermediaries

SOAP intermediaries have the potential to process many more SOAP messages than initial SOAP senders and ultimate SOAP receivers. SOAP intermediaries processing ASN.1 SOAP messages may easily identify ASN.1 SOAP header blocks for processing (including decoding) while skipping (and copying) other SOAP header blocks (destined for other SOAP intermediaries or the SOAP ultimate receiver) and the SOAP body. (This is because the SOAP header blocks and the SOAP body are encoded as a length prefixed sequence of octets).

NOTE – ASN.1 SOAP intermediaries can also manage **fault**s efficiently, since a **fault** will always occur at the end of a message (after the SOAP header blocks) and will be guaranteed to start at a byte boundary if header blocks are present. Thus it is not necessary for an intermediary to decode the **fault** unless the intermediary performs processes not specified by the SOAP processing model.

C.2 Conceptual and optimized processing of ASN.1 SOAP messages

C.2.1 General

C.2.1.1 The conceptual mapping from ASN.1 SOAP messages to W3C SOAP messages and vice versa ensures that the W3C SOAP processing model can be applied to ASN.1 SOAP messages. The six following subclauses highlight the conceptual steps required by an initial SOAP sender, a SOAP intermediary and an ultimate SOAP receiver to process messages, and the optimized steps required for a SOAP intermediary.

C.2.1.2 An initial SOAP sender (see W3C SOAP Part 1, 1.5.3) implementing the ASN.1 SOAP HTTP Binding generates ASN.1 SOAP messages in the following steps:

a) create a new W3C SOAP message and insert new embedded ASN.1 abstract values into the W3C SOAP message; and

e) map the W3C SOAP message to an ASN.1 SOAP message; and

f) encode the ASN.1 SOAP message, using Basic Aligned PER, to a sequence of octets that is the content of an HTTP request.

C.2.1.3 If the initial SOAP sender uses the SOAP Request-Response Message Exchange Pattern (see W3C SOAP Part 2, 6.2), then the SOAP sender (see W3C SOAP Part 1, 1.5.3) will wait for a response and change roles to become an ultimate SOAP receiver (see W3C SOAP Part 1, 1.5.3).

C.2.1.4 A SOAP intermediary (see W3C SOAP Part 1, 1.5.3) implementing the ASN.1 SOAP HTTP Binding processes ASN.1 SOAP messages in the following steps:

a) decode the sequence of octets, obtained from the content of an HTTP request or response, using Basic Aligned PER, to obtain an inbound ASN.1 SOAP message; and

b) map the inbound ASN.1 SOAP message to obtain an inbound W3C SOAP message; and

c) identify and process embedded ASN.1 abstract values in the inbound W3C SOAP message; and

d) modify the inbound W3C SOAP message to become an outbound W3C SOAP message and insert new embedded ASN.1 abstract values into the outbound W3C SOAP message; and

e) map the outbound W3C SOAP message to an outbound ASN.1 SOAP message; and

f) encode the outbound ASN.1 SOAP message, using Basic Aligned PER, to a sequence of octets that is the content of an HTTP response or request.

C.2.1.5 An ultimate SOAP receiver implementing the ASN.1 SOAP HTTP Binding processes ASN.1 SOAP messages in the following steps:

a) decode the sequence of octets, obtained from the content of an HTTP request, using Basic Aligned PER, to obtain an ASN.1 SOAP message; and

b) map the ASN.1 SOAP message to obtain a W3C SOAP message; and

c) identify and process embedded ASN.1 abstract values in the W3C SOAP message.

C.2.1.6 If the ultimate SOAP receiver uses the SOAP Request-Response Message Exchange Pattern, then the SOAP node will change roles to become an initial SOAP sender and will send an ASN.1 SOAP message in reply.

C.2.1.7 The conceptual steps to map to and from W3C SOAP messages and process embedded ASN.1 values (identify and process in a W3C SOAP message and insert into a W3C SOAP message) are specified in clauses 6 to 9 of ITU-T Rec. X.891 | ISO/IEC 24824-2. A SOAP node may, however, choose to optimize the process by skipping the conceptual steps as long as the results are the same as if the conceptual steps were performed. For example steps b) to e) in C.2.1.4 are conceptual steps and the SOAP intermediary may choose to optimize by processing ASN.1 SOAP messages in the following steps:

a) decode the sequence of octets, obtained from content of the HTTP request, using Basic Aligned PER, to obtain an inbound ASN.1 SOAP message; and

b) identify and process embedded ASN.1 abstract values in the inbound ASN.1 SOAP message; and

c) modify the inbound ASN.1 SOAP message to become an outbound ASN.1 SOAP message (or create a new outbound ASN.1 SOAP message) and insert new embedded ASN.1 abstract values into the outbound ASN.1 SOAP message; and

d) encode the outbound ASN.1 SOAP message, using Basic Aligned PER, to a sequence of octets that is the content of the HTTP response.

C.2.2 Example

An example is given in the following subclauses from the perspective of an application sending an ASN.1 SOAP message request and receiving a response. The Fast Web Service is specified in C.3.2 using WSDL 1.1, and is based on the sample W3C SOAP message in W3C SOAP Part 1, 1.4. The service is essentially one in which an application may request the latest alert concerning some information that is important to the application (or user of the application). The requesting application will send an empty ASN.1 SOAP message (with no application-defined content) and receive, in response, an ASN.1 SOAP message with two pieces application-defined content (specified in C.3.2 using WSDL 1.1) for the alert that corresponds to:

a) a SOAP header block for properties of the alert, namely the priority of the alert and the time it expires; and

b) a SOAP body content for the alert itself, which is a textual description of the alert.

C.2.2.1 W3C SOAP message request

The application requests the latest alert by executing (using some appropriate programming language, such as Java) a method call with no input parameters that will return the alert. The initial SOAP sender will create a W3C SOAP message, with no content, represented in XML as:

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
        <env:Body>
        </env:Body>
</env:Envelope>
```

C.2.2.2 ASN.1 SOAP message request

This W3C SOAP message is mapped to an ASN.1 SOAP message request consisting of:

```
envelope Envelope ::= {
      header {}
      body-or-fault : body {} }
```

where the **Envelope** type is defined in Annex A and subclause 6.1 of ITU-T Rec. X.892 | ISO/IEC 24824-2.

C.2.2.3 HTTP request

The ASN.1 SOAP message is then encoded, using Basic Aligned PER, to a sequence of octets that is the content of an HTTP request. The HTTP header field **Content-Type** is "**application/fastsoap**" and the **action** parameter is set to "**urn:alert**". The initial SOAP node declares, using the HTTP header field **Accept,** that both ASN.1 SOAP messages and XML SOAP messages (in this case SOAP 1.1 messages [1]) are supported.

```
POST /AlertPort HTTP/1.1
 Content-Type: application/fastsoap; action="urn:alert"
 Accepts: application/fastsoap, application/text+xml
 Content-Length: ....

 ... sequence of octets …
```

C.2.2.4 HTTP response

The initial SOAP sender will then change roles and become an ultimate SOAP receiver and waits until it receives a response to the request. The HTTP header field **Content-Type** on the response is "**application/fastsoap**".

```
HTTP/1.1 200 OK
 Content-Type: application/fastsoap
 Content-Length: ....

 ... sequence of octets …
```

C.2.2.5 ASN.1 SOAP message response

The ASN.1 SOAP message is decoded using Basic Aligned PER, to produce the ASN.1 value:

```
envelope Envelope ::= {
  header { {
    role "http://example.org/alertrole",
    content : encoded-value {
      id : qName {
        uri "http://example.org/alertcontrol",
        name "altertcontrol"},
      encoding  {......}}}},
  body-or-fault : body {
    content : encoded-value {
      id : qnName {
        uri "http://example.org/alert",
        name "alert"},
      encoding {......}}}}
```

C.2.2.6 W3C SOAP message response

C.2.2.6.1 The ASN.1 SOAP message is mapped to a W3C SOAP message. The W3C SOAP message contains an **alertcontrol** W3C SOAP header block and an **alert** element (information item) as the child of a **Body** EII:

```
<env:Envelope
        xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
        <n:altertcontrol
                xmlns:n="http://example.org/alertcontrol"
                env:role="http://example.org/alertrole"
                env:encodingStyle="urn:fastws:encoding:APER">
        … Base64 content …
        </n:altertcontrol>
    </env:Header>
    <env:Body>
        <m:alert
                xmlns:m=" http://example.org/alert"
                env:encodingStyle="urn:fastws:encoding:APER">
        … Base64 content …
        </m:alert>
    </env:Body>
</env:Envelope>
```

C.2.2.6.2 The embedded ASN.1 abstract value for the **alertcontrol** W3C SOAP header block is identified and processed since the requesting SOAP node operates in the "**http://example.org/alertrole**" role. The **Identifier** value for the **alertcontrol** W3C SOAP header block and the embedded ASN.1 value, decoded using Basic Aligned PER from the Base64 content using the ASN.1 type, **AlertControl**, associated with the **Identifier**, are as follows:

```
alertControlIdentifier Identifier ::= qName : {
            uri "http://example.org/alertcontrol",
            name "altertcontrol" }

alertcontrol Alertcontrol ::= {
            role "http://example.org/alertrole",
            priority 1,
            expires "2001-06-22T14:00:00-05:00" }
```

C.2.2.6.3 The embedded ASN.1 abstract value for the **alert** element (information item) is identified and processed as the SOAP node is an ultimate SOAP receiver. The **Identifier** value for the **alert** and the embedded ASN.1 value, decoded using Basic Aligned PER from the Base64 content using the ASN.1 type **Alert** associated with the **Identifier** are as follows:

```
alertIdentifier Identifier ::= qName : {
            uri "http://example.org/alert",
            name "alert" }

alert Alert ::= {
            msg "Pick up Mary at school at 2pm" }
```

C.3 Service descriptions

C.3.1 General

C.3.1.1 Service descriptions expressed in WSDL 1.1 [2] can be used, without modification, for the description of ASN.1 SOAP endpoints. This increases the scope and usage of Fast Web Services, since the impact on the Web services developers is minimized.

C.3.1.2 A WSDL 1.1 binding interface for SOAP 1.1 [1] can be reused for an ASN.1 SOAP binding interface provided the WSDL document is a SOAP-oriented service description and WSDL 1.1 binding conforms to the clarifications and amendments specified by the WS-I Basic Profile 1.0 [3].

C.3.2 Example

C.3.2.1 The service description (expressed in WSDL 1.1) shown in C.3.3 specifies an ASN.1 SOAP interface binding for the example in C.2.2.

C.3.2.2 The WSDL document has the two **xsd:schema** definitions contained in the **wsdl:type**s (specifying the child content of a **Body** EII, and W3C SOAP header block for the only response). The equivalent ASN.1 schema is obtained by applying ITU-T Rec. X.694 | ISO/IEC 8825-5 to the two schemas.

C.3.2.3 The ASN.1 SOAP HTTP Binding will be used because the value of the **transport** attribute on the **soapbind:binding** element (the ASN.1 SOAP interface binding) is equal to "**http://schemas.xmlsoap.org/soap/http/**".

C.3.2.4 Fast Web Services support is explicitly specified for the ASN.1 SOAP binding interface by use of the ASN.1 SOAP interface binding annotation (a **fast-service:binding** element) in the **wsdl:binding** element and after the **soapbind:binding** element.

C.3.2.5 The style of the ASN.1 SOAP binding interface is the document-style, since the interface binding conforms to document-literal binding as specified by the WS-I Basic Profile 1.0 [3].

C.3.2.6 The input message definition is empty (no top-level **element declaration**, since the **soapbind:body** in the **wsdl:input** in the **AlertOperation** operation binding references, implicitly, no **wsdl:parts**). However, a SOAP action URI exists, since the **AlertOperation** operation binding has a **soapAction** attribute. The URI "**urn:alert**" will be placed in the **action** parameter of the "**application/fastsoap**" MIME type for the HTTP header field **Content-Type** of the HTTP request (that contains the empty ASN.1 SOAP message).

C.3.2.6 The output message definition has one top-level **element declaration**, **alert:alert** (since the **soapbind:body** in the **wsdl:output** in the **AlertOperation** operation binding references, implicitly, one **wsdl:part**).

C.3.2.7 A SOAP header block definition (the **alertcontrol** W3C SOAP header block) is specified for output of the **AlertOperation** operation binding with a top-level **element declaration alertcontrol:alertcontrol**.

C.3.3 Service description expressed in WSDL 1.1

```
<definitions name="Alert"
     xmlns="http://schemas.xmlsoap.org/wsdl/"
     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/ "
     xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/"
     xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:fast-service=" urn:fastws:description "
     xmlns:tns="http://example.org/alert/service"
     targetNamespace="http://example.org/alert/service"
     xmlns:alert="http://example.org/alert"
     xmlns:alertcontrol="http://example.org/alertcontrol">

   <types>
     <schema
          targetNamespace="http://example.org/alertcontrol"
          xmlns="http://www.w3.org/2001/XMLSchema"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          elementFormDefault="qualified">
       <import namespace=" http://schemas.xmlsoap.org/wsdl/soap/"/>
       <element name="alertcontrol">
         <complexType>
           <sequence>
             <element name="priority" type="xsd:integer"/>
             <element name="expires" type="xsd:dateTime"/>
           </sequence>
           <xsd:attribute ref="soap:role"/>
         </complexType>
       </element>
     </schema>
     <schema
          targetNamespace="http://example.org/alert"
          xmlns="http://www.w3.org/2001/XMLSchema"
          elementFormDefault="qualified">
       <element name="alert">
         <complexType>
           <sequence>
             <element name="msg" type="xsd:string"/>
           </sequence>
         </complexType>
       </element>
     </schema>
   </types>

   <message name="AlertRequest">
   </message>
```

```
  <message name="AlertResponse">
    <part name="header" element="alertcontrol:alertcontrol"/>
    <part name="body" element="alert:alert"/>
  </message>

  <portType name="AlertPortType">
    <operation name="AlertOperation">
      <input message="tns:AlertRequest"/>
      <output message="tns:AlertResponse"/>
    </operation>
  </portType>

  <binding name="AlertBinding" type="tns:AlertPortType">
    <soapbind:binding
      transport="http://schemas.xmlsoap.org/soap/http"
      style="document"/>
    <fast-service:binding/>

    <operation name="AlertOperation" soapAction="urn:alert">
      <input message="tns:AlertRequest">
        <soapbind:body use="literal"/>
      </input>
      <output message="tns:AlertResponse">
        <soapbind:body use="literal" parts="body"/>
        <soapbind:header
            use="literal"
            message="tns:AlertResponse"
            part="header"/>
      </output>
    </operation>
  </binding>

  <service name="AlertService">
    <port name="AlertPort" binding="tns:AlertBinding">
      <soapbind:address location="http://example.org/AlertPort"/>
    </port>
  </service>
</definitions>
```

Bibliography

[1] W3C Note, *Simple Object Access Protocol (SOAP) 1.1*, Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Nielsen, Satish Thatte, Dave Winer, *8 May 2000*.

[2] W3C Note, *Web Services Description Language (WSDL) 1.1*, Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, *15 March 2001*.

[3] WS-I, *Basic Profile Version 1.0, Final Material*, *16 April 2004*.