SOAP Fundamentals

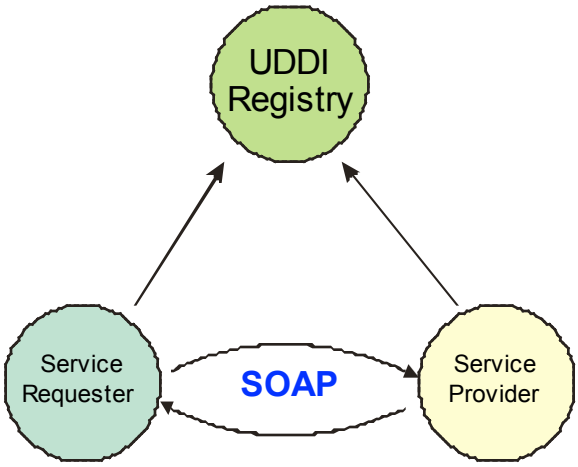
- SOAP Messages
- Encoding Styles

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What is SOAP?

- SOAP is an XML-based protocol for distributed computing
- Major goal: interoperability
- SOAP messages can be transported using standard Internet protocols such as HTTP or SMTP or proprietary protocols like the Java Message Service (JMS)



SOAP is the basic binding protocol for the Web service-oriented architecture. It provides a lightweight wire protocol using XML syntax for passing messages from a service requester to a service provider.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

History of SOAP

- SOAP 1.0 and a related protocol, XML-RPC were originally developed by UserLand, DevelopMentor and Microsoft in 1998
- IBM joined the effort and contributed to SOAP 1.1, which was developed in May 2000

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

SOAP and the W3C

- From the W3C: "SOAP Version 1.2 is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment"
- SOAP 1.2 became a W3C Recommendation on June 24, 2003

The SOAP 1.2 recommendation is available at http://www.w3.org/TR/2003/REC-soap12-part1-20030624/.

There also is a SOAP 1.2 Part 0: Primer document available at http://www.w3.org/TR/2003/REC-soap12-part0-20030624/. It acts as a useful tutorial introduction to SOAP.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

SOAP 1.1 vs SOAP 1.2

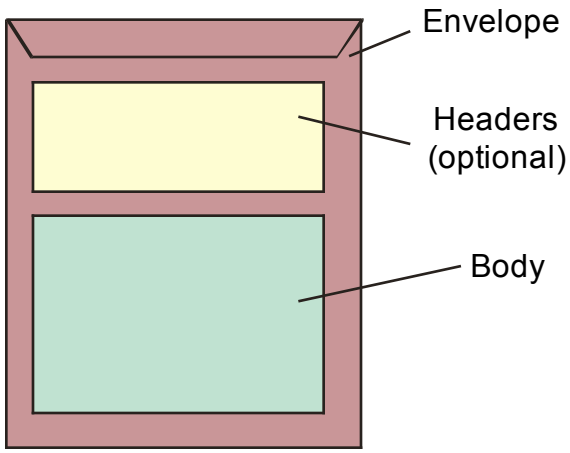
- SOAP 1.1 was a W3C *technical note*, not a *Recommendation* -- it's available at http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
- As of this writing, many SOAP toolkits have better support for SOAP 1.1. than SOAP 1.2
- We will cover both versions in this chapter

As is normal, it takes a while for toolkits to upgrade to the latest specification, so in the short term, students learning SOAP will need to study both versions.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Structure of a SOAP Message

- A SOAP message consists of a required *envelope*, optional *headers* and a required *body*
- The body contains the actual SOAP message
- The optional headers can carry routing information, authentication and encryption information and so forth



A SOAP message is an XML-formatted chunk of content that is organized into an envelope, header and body. The envelope is the root element and the optional header and required body elements are direct children of the envelope.

We will cover headers in more detail later in the chapter.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

SOAP Messages and Namespaces

- The *Envelope* and *Body* must be namespace qualified using a URI that varies according to the version of SOAP being used
- The *payload* must be namespace qualified with an application-defined namespace

```
1    <soap:Envelope
2      xmlns:soap="xxxxxxxxxxxxx">
3      <soap:Body>
4        <m:MyMessage
5            xmlns:m="yyyyyyyyy">
6           . . .
7        </m:MyMessage>
8      </soap:Body>
9    </soap:Envelope>
```

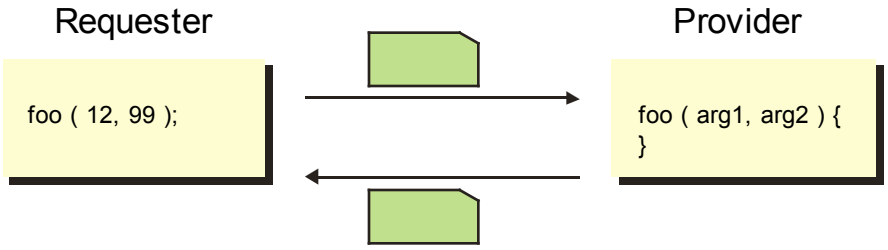

SOAP uses namespaces to help SOAP processors identify the version number of SOAP being used and to discriminate between services.

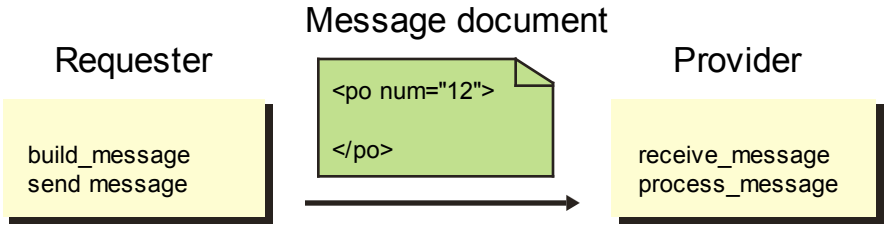
Also note that the "Header" should be namespace qualified as well as the individual child elements within the Header.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

SOAP Message Exchange

- SOAP messaging usually takes one of two forms: *remote procedure call* (RPC) or generic XML *document exchange*





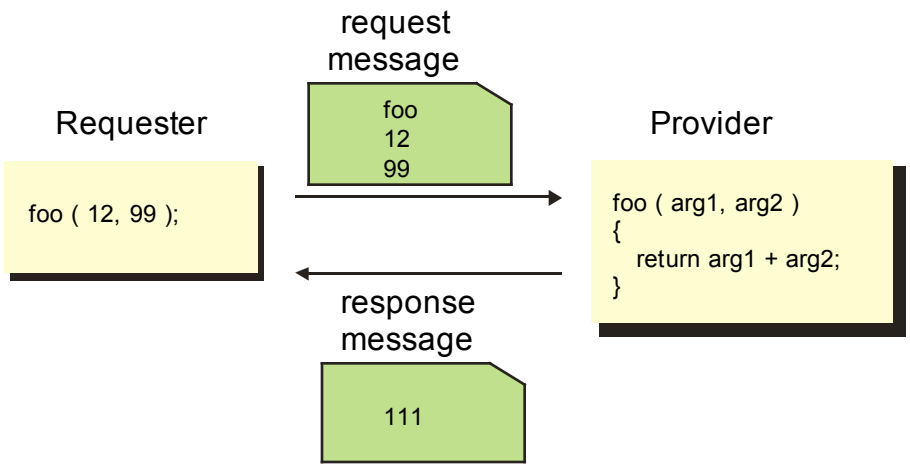
SOAP provides a simple mechanism to pass XML-formatted messages from sender to receiver. The XML content can be any well-formed XML document.

However, SOAP also, from the very beginning, has supported the notion of Remote Procedure Calls (RPC), where there are special rules for the content of the XML.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Introduction to SOAP RPC

- An *remote procedure call* is a form of distributed computing where a client invokes a remote function
- In SOAP, an RPC typically consists of two messages: a request and a response



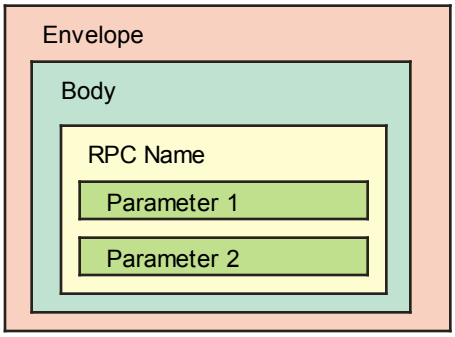
SOAP RPCs model earlier distributed computing technologies such as CORBA, Java RMI and Microsoft DCOM. However, they have the advantage of being platform and language neutral and the ability to use standard Internet protocols such as HTTP.

An RPC is normally synchronous with respect to the caller – that is, the caller sends the request message and waits for the response message before continuing.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

SOAP RPC Fundamentals

- The first, and only child element of a SOAP RPC request's **Body** specifies the *name* of the remote procedure
- Within that element, child elements contain any parameters for the RPC
- Within the SOAP response, the first and only child element of the **Body** contains any return values as child elements



RPC is essentially a special case of generic document exchange, where there are conventions and rules for the Body content and for how messages are exchanged. The goal is to simulate a local procedure call.

Most programming toolkits provide client-side "stubs" to make it easier for client programmers to call RPCs using the language of their choice.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

SOAP RPC Fundamentals, cont'd

- Most SOAP toolkits map an RPC to a language function or method
- The toolkits must map the child elements of the RPC element onto language types

SOAP RPC

```
<soap:Envelope
  xmlns:soap="xxxxxxxxxxxxx">
  <soap:Body>
    <m:MyAdd
       xmlns:m="yyyyyyyyy">
      <op1>12</op1>
      <op2>42</op2>
    </m:MyAdd>
  </soap:Body>
</soap:Envelope>
```

Java

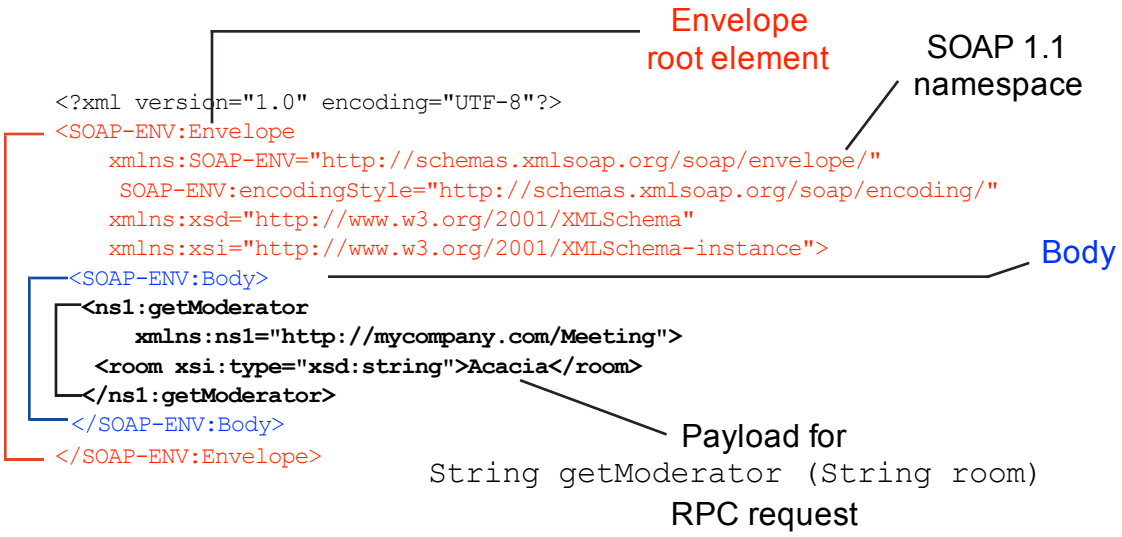
```
public int MyAdd(int op1, int op2)
{

}
```


The goal of RPC is for language programmers to easily invoke remote behavior. To satisfy this goal, most toolkits map the RPC onto native language facilities. The WSDL aids this process.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Sample SOAP 1.1 RPC Request



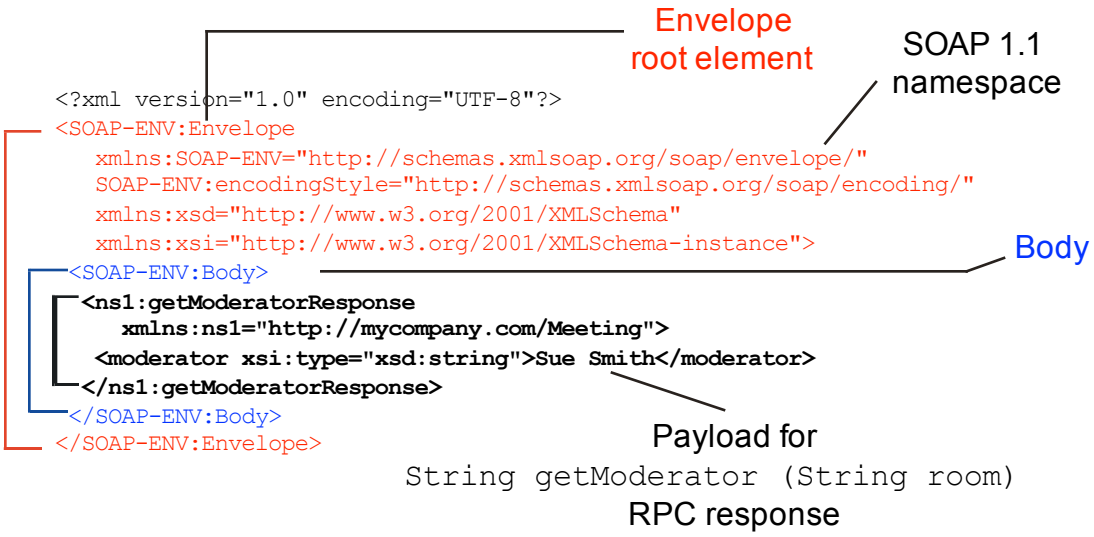
This RPC request message is for a service that provides information about meetings at a conference facility. The getModerator() message returns a string with the meeting moderator's name, given the meeting room (in this case, the room name is "Acacia").

Note the Envelope is namespace-qualified with the SOAP 1.1 namespace URI. It also defines the SOAP 1.1 encoding style (covered later) as well as prefixes for XML Schema datatypes.

The only child element of the SOAP-ENV:Body is the "getModerator" element, which is the name of the remote procedure. This element, like all message payloads, is qualified by an application-defined namespace. Since this RPC requires a parameter, the getModerator element contains a child element whose value is the meeting-room-name parameter. Note that this parameter's type is defined as the XML Schema xsd:string type - SOAP does not require parameters to be typed, but typing aids in interoperability.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Sample SOAP 1.1 RPC Response

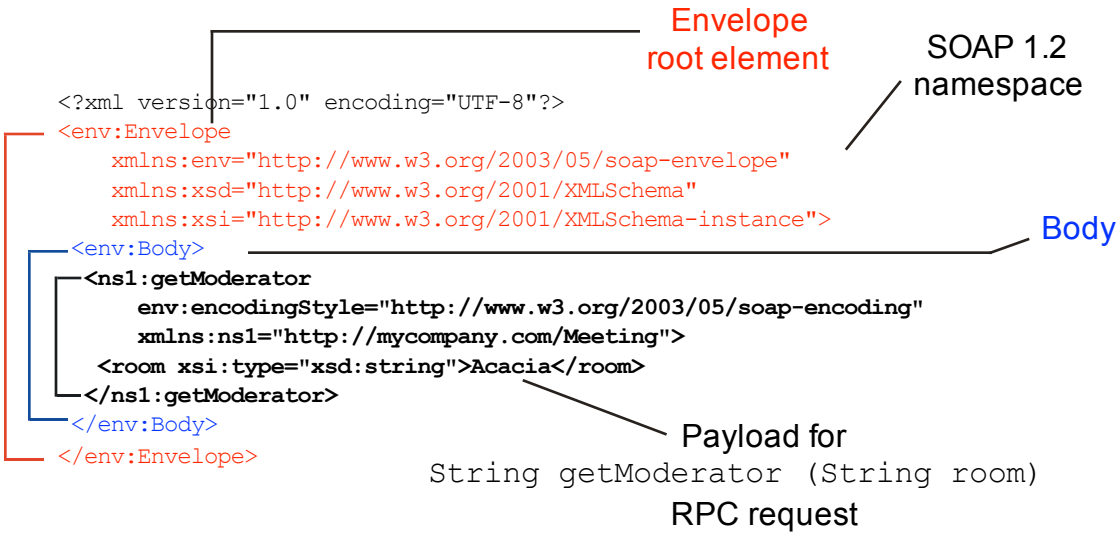


Here we show the response for the "getModerator()" RPC. Evidently, Sue Smith is the moderator for the Acacia room.

Note that the response message has the same basic Envelope-Body-Payload structure as the request message and is namespace qualified in the same fashion.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

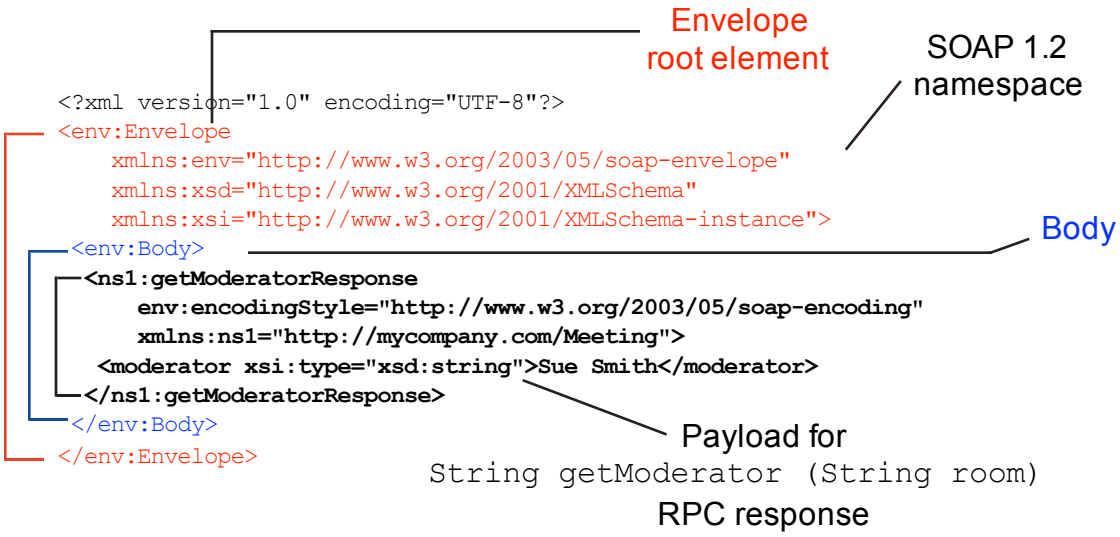
Sample SOAP 1.2 RPC Request



In SOAP 1.2, the message format changes a bit. First of all, there's a different namespace prefix that reflects the fact that SOAP is now administered by the W3C. Second, the encoding definition moves from the Envelope to the payload, and its URI changes. As mentioned, we will cover encoding in more detail later.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Sample SOAP 1.2 RPC Response

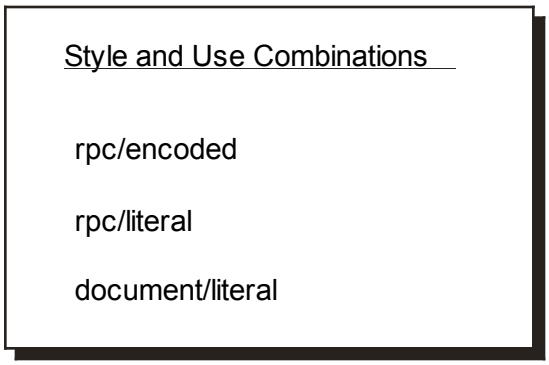


Here we show the response for the SOAP 1.2 request on the last page. Again, note the SOAP 1.2 namespace.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Introduction to SOAP RPC Encoding

- Encoding refers to how typed data (e.g. integer or float) is represented in the XML SOAP message
- The SOAP specification suggests an encoding scheme (SOAP encoding) that uses the XML Schema language, but other encoding schemes are possible
- **Important:** Encoded messages are forbidden by the WS-I (interoperability) guidelines



SOAP encoding applies only to RPC messages, since document-exchange messages contain actual XML documents. And by and large, SOAP messages are encoded using the XML Schema standard. Primitive (simple) types are encoded using standard XML Schema simple types, while structures, which correspond to JavaBean objects, are encoded as complex types.

The WS-I guidelines forbid encoding because it impedes interoperability. The main issue is that because of the way SOAP encoding serializes "graphs" of objects, it's impossible to write a schema that can validate the message (SOAP encoding uses "references", which are like HTML anchor tags to allow parts of the message to "point" to other parts). In fact , that is the primary benefit of using encoded messages -- the object graphs can contain cyclical references -- if you use RPC/literal, there's no standard way to handle cyclical references, and the SOAP processor might get into a infinite loop.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Encoding Issues

- The SOAP encoding scheme specification includes:
  - How to encode simple types (e.g. float)
  - How to encode arrays
  - How to encode **object graphs**
- Simple types are encoded using XML Schema types
- Arrays are encoded using a special **soapenc:Array** type
- Object graphs are encoded using *references*

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Encoded RPC Request

- Encoded RPC SOAP messages specify a **type** for each parameter

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <SOAP-ENV:Body>
  <ns1:getModerator
      xmlns:ns1="http://mycompany.com/Meeting">
   <room  xsi:type="xsd:string">Acacia</room>
  </ns1:getModerator>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```



Encoded parameter

Encoded RPC messages specify the type of each parameter in the message itself, while "literal" messages just have the parameter names and values.

You can tell whether or not a particular service expects encoded or literal by reading the WSDL for the service and examining the "binding" section.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Literal RPC Request

- Literal RPC messages do not specify types -- instead, the sender and recipient must agree on the types, usually via WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <SOAP-ENV:Body>
  <ns1:getModerator
      xmlns:ns1="http://mycompany.com/Meeting">
   <room>Acacia</room>
  </ns1:getModerator>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

SOAP and Complex Types

```
public class Student
{
  private int studentID;
  private String name;
  private double gpa;

  public int getStudentID()
  {
    return this.studentID;
  }
  . . .
}
```

```
<complexType name="Student">
 <sequence>
   <element name="gpa" type="xsd:double"/>
   <element name="name" type="xsd:string"/>
   <element name="studentID" type="xsd:int"/>
 </sequence>
</complexType>
```

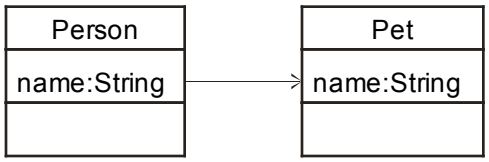

In most SOAP implementations, SOAP structures map to language elements such as C++ structures or Java object classes. Here we show part of the Java class definition for the student struct from the last page. Note that most Java SOAP toolkits require objects passed as SOAP structs to be written in the form of a JavaBean with standard get/set methods for the data that will be encoded into the SOAP message.

Also note that such structure types can be defined using XML Schema complexType definitions. The complexType definition is not included in the actual SOAP message, but is normally included in the WSDL for the service. That makes it possible for clients to understand the format of the structure.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Object Graphs

- Language objects (e.g. Java objects) can have references to other objects
- When such an object is converted into XML, the SOAP processor must build an **object graph** by following the references




```
<schema . . .>
 <complexType name="Person">
  <sequence>
   <element name="name" type="xsd:string"/>
   <element name="pet" type="impl:Pet"/>
  </sequence>
 </complexType>
 <complexType name="Pet">
  <sequence>
   <element name="name" type="xsd:string"/>
  </sequence>
 </complexType>
</schema>
```


Here we show an object class named "Person" that references another class, Pet. If an RPC accepts or returns a Person object, the SOAP processor must also serialize the referenced Pet object.

Note how the schema illustrates the reference -- the Person element has a child element named "pet" that assumes the type "Pet".

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Encoded Object Graphs

- In SOAP RPC using **SOAP encoding**, the SOAP processor keeps track of each object as it is converted to XML and uses **references** if the same object needs to be saved more than once
- This referencing schema allows for arbitrary object graphs, even those containing *cyclical references*
- In SOAP RPC using a **literal** style, the SOAP processor may not be able handle cyclical references

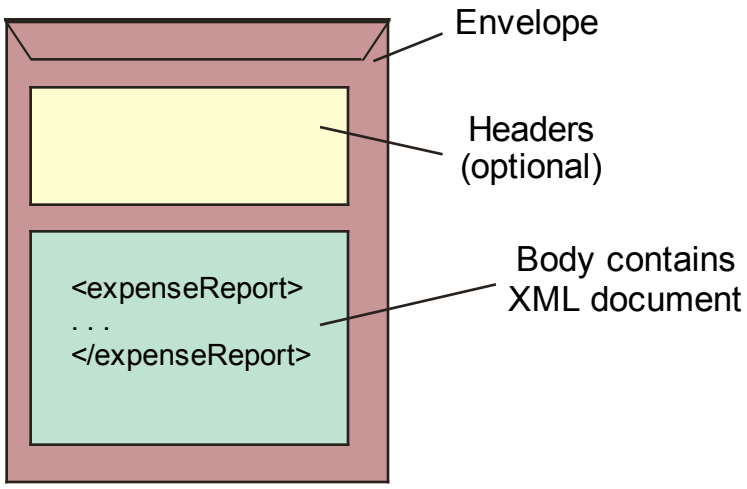
The SOAP encoding style uses assigns a unique ID to each object and uses a syntax much like HTML anchor references to build object graphs.

In literal RPC, there's no standard way to reference an object multiple times -- the SOAP processor is essentially mapping an object graph into a basic XML tree. This can lead to problems like infinite recursion.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Intro to SOAP Document Exchange

- SOAP *document exchange* lets a client pass an XML document to a web service, which can then parse and process the document
- Unlike with RPC, there is no typical message exchange pattern, but request/response is quite common



In many ways, SOAP document exchange is simpler than SOAP RPC. Encoding is not required, for example, since it's expected that the sender and recipient understand the structure of the XML payload. In addition, there's no requirement for synchronous processing as is the rule for RPC.

To process the payload, the recipient extracts it from the SOAP envelope and body and then typically uses standard XML parsing techniques (e.g. DOM or SAX) to process the message.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Sample SOAP 1.2 Document Exchange

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <expense-report xmlns="http://mycompany.com/Expenses">
      <submitter-name>Paul Westerberg</submitter-name>
      <submission-date>2003-July-08</submission-date>
      <expenses>
        <expense>
          <type>Airfare</type>
          <amount>434.55</amount>
          <date>2003-June-23</date>
        </expense>
      </expenses>
    </expense-report>
  </soapenv:Body>
</soapenv:Envelope>
```


Here we show a sample document-exchange message that contains an expense report as its payload. Note the payload is namespace-qualifed (using a default namespace). This is a SOAP 1.2 version message since it uses the SOAP 1.2 namespace URI on the envelope.

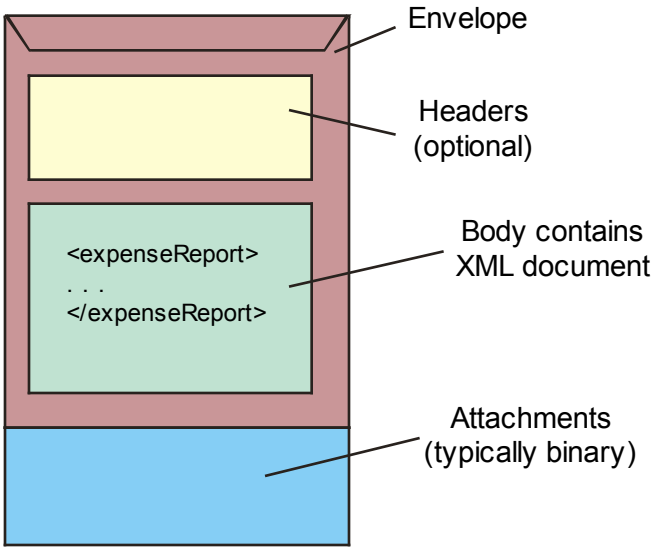
The type of the payload is typically defined as an XML Schema in the WSDL for the service, but is not carried in the actual SOAP message.

Note that there is no encoding information specified - that's because we only need encoding to specify the types of arguments and return from an RPC. With document exchange, all that's required is that both the sender and receiver are able to parse and understand the payload.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

SOAP Attachments

- The SOAP with Attachments specification allows SOAP messages to transmit MIME-structured data
- Unfortunately, not all SOAP implementations support attachments in the same fashion



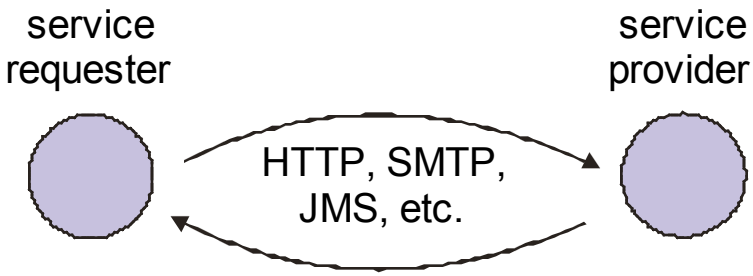
In many cases, especially with document exchange, it is useful to include binary data along with the SOAP message. For example, using the expense-report example, a submitter could attach an image file (e.g. a JPEG) of the expense receipts. SOAP allows for base64-encoded binary data within the message body itself, but base64 is tedious and inefficient, especially for large chunks of data.

So to pass binary data, SOAP allows for attachments appended to the message itself, outside of the SOAP envelope. Unfortunately, at this time, there are two competing specifications for these attachments: SOAP with Attachments, which is supported by most Java SOAP toolkits, and the DIME specification for attachments, which is Microsoft's choice. Until there is agreement or synthesis on this issue, using attachments will likely cause interoperability issues.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Introduction to SOAP Protocol Binding

- SOAP is protocol agnostic, but most services currently use HTTP
- This maps well to SOAP RPC, since HTTP requests have both a request and a response



The SOAP 1.2 recommendation (part 2, adjuncts) defines a binding for SOAP messages over HTML. It specifies, among other things, that successful responses are sent with HTTP status code 200 and failures indicated by 4XX or 5XX codes and that the standard SOAP Request-Response message exchange pattern be used.

But it is important to note that HTTP is not a requirement for a SOAP processor even though it is currently the most widely used.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Sample SOAP 1.1 HTTP Request

```
POST /axis/services/Meeting HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: text/xml
User-Agent: Axis/1.0
Host: localhost
SOAPAction: ""
Content-Length: 475

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Body>
     . . .
 </soapenv:Body>
</soapenv:Envelope>
```



Here we show a SOAP 1.1 request sent as an HTTP POST. Especially notice the "Content-Type", which tells the web service what type of data is being sent. Note that there's nothing SOAP-specific in the type.

The SOAPAction header is required in SOAP 1.1, but can be empty as it is here. This header allows the requester to specify the "intent" of the message. If provided, it's normally a unique URI and allows firewalls to filter traffic.

Note that the message body contains the SOAP envelope as we've already discussed.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Sample SOAP 1.1. HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Date: Wed, 23 Jul 2003 01:41:35 GMT
Server: Apache Coyote/1.0
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
   . .
  </soapenv:Body>
</soapenv:Envelope>
```

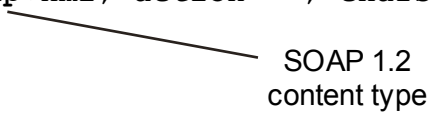

The response from the message sent on the last page is shown here. Note the HTTP status code to indicate success and also not the Content-Type. And as before, the SOAP envelope immediately follows the HTTP headers.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Sample SOAP 1.2 HTTP Request

```
POST /axis/services/Meeting HTTP/1.0
Content-Type: application/soap+xml; action=""; charset=utf-8
Accept: application/soap+xml
User-Agent: Axis/1.0
Host: localhost
Content-Length: 475

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
   xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
   . . .
  </soapenv:Body>
</soapenv:Envelope>
```



In SOAP 1.2, the W3C applied for a special "media type" for SOAP applications: application/soap+xml. By using this type, the Web server can quickly determine that this is SOAP request instead of a normal Web request.

Note also that the SOAPAction header has been deprecated and is now an optional "action" attribute on the Content-Type header. It performs the same function as the explicit SOAPAction header did in SOAP 1.1.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Sample SOAP 1.2 HTTP Response

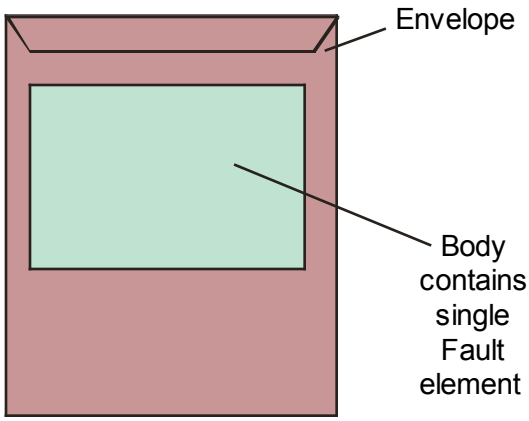
```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Date: Wed, 23 Jul 2003 01:41:35 GMT
Server: Apache Coyote/1.0
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
   xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
   <soapenv:Body>
      . . .
   </soapenv:Body>
</soapenv:Envelope>
```


Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

SOAP Faults

- If the message cannot be processed, a **SOAP Fault** may be generated
- The format of a SOAP fault message is defined by the SOAP specification
- SOAP faults are often mapped to language-specific error handling facilities (e.g. Java exceptions)

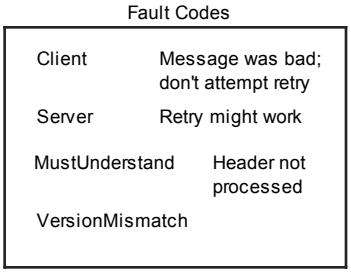


The format of SOAP Faults is different in SOAP 1.1 and SOAP 1.2.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Example SOAP 1.1 Fault

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
 <soap:Body>
  <soap:Fault>
   <faultcode>soap:Client</faultcode>
   <faultstring>Processing Error</faultstring>
   <faultactor></faultactor>
   <detail>
    <e:myFaultDetails xmlns:e='http://mycompany.com/errors'>
     <e:message>Unknown Policy Holder ID</e:request>
     <e:errorcode>1234<e:errorcode>
    </e:myFaultDeatils>
   </detail>
  </soap:Fault>
 </soap:Body>
</soap:Envelope>
```



In SOAP 1.1, a Fault contains a required "faultcode", of which there are four pre-defined values (it is possible to create new values), a required, human-readable "faultstring", an optional "faultactor" which specifies the intermediary (if any) that generated the fault, and an optional "detail" element that contains application-defined information about the error.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Example SOAP 1.2 Fault

```
<?xml version='1.0' ?>
<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:Detail>
      <e:myFaultDetails xmlns:e="http://mycompany.com/errors">
        <e:message>Uknown policy holder ID</e:message>
        <e:errorcode>1234</e:errorcode>
      </e:myFaultDetails>
     </env:Detail>
   </env:Fault>
 </env:Body>
</env:Envelope
```

Fault Codes

Sender	Message was bad; don't attempt retry
Receiver	Retry might work
MustUnderstand	Header not processed
VersionMismatch	
DataEncodingUnknown	

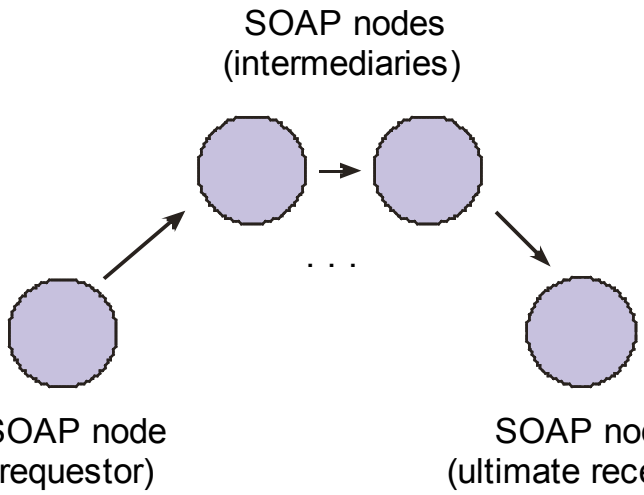
In SOAP 1.2, a Fault message must contain a Code and a Reason, in that order. Within Code, the Value is required, but the SubCode is optional (and can contain nested Value and SubCode elements). The Reason element is intended to provide a human-readable explanation of the fault.

The Detail element is optional and allows you to include application-defined error information.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

SOAP Message Path

- On its journey from requestor to the ultimate receiver, a SOAP message may be processed by **intermediaries**
- Intermediaries perform tasks such as logging, routing, load-balancing, security and so forth



Each participant in the message path is considered to be a "node". Intermediary nodes act as both senders and receivers.

Intermediaries can stop message processing by generating a SOAP fault.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

SOAP Intermediaries and Headers

- Intermediaries often examine and process SOAP **headers**
- Like the payload, header elements must be namespace qualified with an application-defined namespace

SOAP 1.1 Example

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
  <soap:Header>
    <m:GenTime xmlns:m="http://www.example.com/ClientCreate">
       10:10 GMT June 13, 2005
    </m:GenTime>
    <x:Logger xmlns:x="http://mylogger.com/names"
           soap:actor="http://mylogger.com/logUtil"
           soap:mustUnderstand="1">
       <logMode>secure</logMode>
    </x:Logger>
  </soap:Header>
 <soap:Body>
   <!-- message goes here -->
 </soap:Body>
</soap:Envelope>
```


You can use headers to add "metadata" to the message, i.e. information about the message that's not part of the message itself. Examples include the time that the message was generated, information about the sender's credentials, the algorithm used to encrypt the message and so forth.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The actor Attribute

- Headers can define an attribute named **actor** (**role** in SOAP 1.2) that allows you to target the header to a particular intermediary
- The value of this attribute is a pseudo-URL that identifies the intermediary, or is one of the predefined values defined in the SOAP specification

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
  <soap:Header>
    <m:GenTime xmlns:m="http://www.example.com/ClientCreate">
       10:10 GMT June 13, 2005
    </m:GenTime>
    <x:Logger xmlns:x="http://mylogger.com/names"
           soap:actor="http://mylogger.com/logUtil"
           soap:mustUnderstand="1">
       <logMode>secure</logMode>
    </x:Logger>
  </soap:Header>
 <soap:Body>
   <!-- message goes here -->
 </soap:Body>
</soap:Envelope>
```


Headers can optionally have an "actor" (SOAP 1.1) or "role" (SOAP 1.2) attribute that identifies the intermediary that should process the header. If the role or actor is omitted, then the header should be processed by the ultimate recipient node.

In most cases, an intermediary will remove any header targeted toward it before forwarding the message to the next SOAP node.

The two predefined actor/roles in SOAP v1.1 are:

```
http://schemas.xmlsoap.org/soap/actor/ultimateReceiver
http://schemas.xmlsoap.org/soap/actor/next
```

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The mustUnderstand Attribute

- Headers can define an attribute named **mustUnderstand** that lets you mark a header as mandatory
- The value of this attribute is "1" or "0" -- "1" indicates that the header is required and the the processing node should generate a SOAP fault if the node cannot process the header for any reason

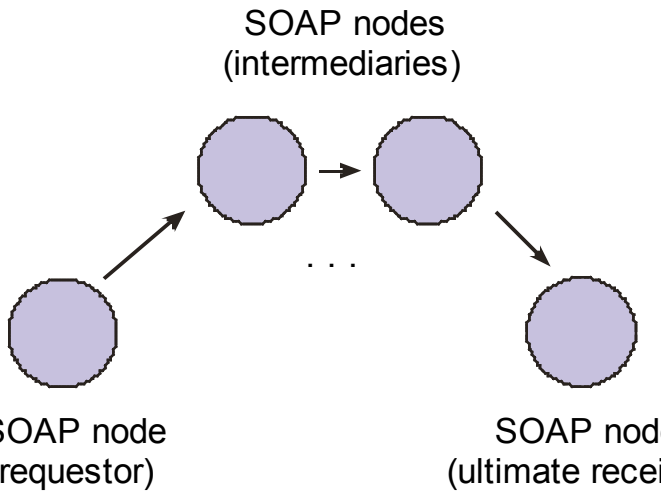
```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
  <soap:Header>
    <m:GenTime xmlns:m="http://www.example.com/ClientCreate">
       10:10 GMT June 13, 2005
    </m:GenTime>
    <x:Logger xmlns:x="http://mylogger.com/names"
           soap:actor="http://mylogger.com/logUtil"
           soap:mustUnderstand="1">
       <logMode>secure</logMode>
    </x:Logger>
  </soap:Header>
 <soap:Body>
   <!-- message goes here -->
 </soap:Body>
</soap:Envelope>
```


If the processing node cannot process a mandatory header, it must generate a SOAP fault message and not forward the message to the next node in the message path.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Processing Headers

- Headers are often inserted by intermediaries running on the SOAP client or further down the message path
- According to the SOAP specification, a node that processes a header must remove the header from the message (but it's OK to add it back in)



Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Chapter Summary

In this chapter, you learned:

- SOAP Fundamentals
- SOAP 1.1 vs SOAP 1.2

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum