WSDL – Web Service Description Language

Preface

WSDL is an XML-based language for describing network services (XML Web Services). WSDL describes the service (its methods), and also the way the communication between a client and service will be performed. As per WSDL 1.1 specification: "WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information."

This document has been produced for Self-Study purposes which accounts for two credits in module Service Oriented Architecture (SOA), WAT 5101, at the University of Technology Mauritius. The focus is on WSDL 1.1 specification, which was submitted as a W3C Note by Ariba, IBM and Microsoft for describing services for the W3C XML Activity on XML Protocols in March 2001. (Note that WSDL 2.0 is a W3C recommendation as from 26 June 2007).

Introduction

WSDL represents a contract between the service requestor and the service provider, in much the same way that a Java interface represents a contract between client code and the actual Java object. The crucial difference is that WSDL is platform and language independent, and is used primarily (although not exclusively) to describe SOAP services.

Using WSDL, a client can locate a web service and invoke any of its publicly available functions. With WSDL-aware tools, you can also automate this process, enabling applications to easily integrate new services with little or no manual code. WSDL therefore represents a cornerstone of the web service architecture, because it provides a common language for describing services and a platform for automatically integrating those services.

WSDL Document Definition

A WSDL document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: messages, which are abstract descriptions of the data being exchanged, and port types which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitute a reusable binding. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL document uses the following elements: <definitions>, <types>, <portType>, <message>, <binding>, <service>. Table 1.0 below gives a description of each element.

Element	Description
definitions	The definitions element must be the root element of all WSDL documents. It defines the name of the web service, declares multiple namespaces used throughout the remainder of the document, and contains all the service elements.
types	The types element describes all the data types used between the client and server. WSDL is not tied exclusively to a specific typing system, but it uses the W3C XML Schema specification as its default choice. If the service uses only XML Schema built-in simple types, such as strings and integers, the types element is not required. A full discussion of the types element and XML Schema is deferred to the end of the chapter.
message	The message element describes a one-way message, whether it is a single message request or a single message response. It defines the name of the message and contains zero or more message part elements, which can refer to message parameters or message return values.
portType	The portType element combines multiple message elements to form a complete one-way or round-trip operation. For example, a portType can combine one request and one response message into a single request/response operation, most commonly used in SOAP services. Note that a portType can (and frequently does) define multiple operations.
binding	The binding element describes the concrete specifics of how the service will be implemented on the wire. WSDL includes built-in extensions for defining SOAP services, and SOAP-specific information therefore goes here.
service	The service element defines the address for invoking the specified service. Most commonly, this includes a URL for invoking the SOAP service.

Structure of a WSDL Document

The main structure of a WSDL document is presented below. Note that a WSDL document can also contain other elements, like extension elements and a service element that makes it possible to group together the definitions of several web services in one single WSDL document.

```
<definitions>
<types>
  definition of types........
</types>

<message>
  definition of a message....
</message>

<portType>
  definition of a port.......
</portType>

<binding>
  definition of a binding....
</binding>

</definitions>
```

A complete WSDL document structure from the w3C

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri"?>

  <import namespace="uri" location="uri"/>*

  <wsdl:documentation .... /> ?

  <wsdl:types> ?
    <wsdl:documentation .... />?
    <xsd:schema .... />*
    <-- extensibility element --> *
  </wsdl:types>

  <wsdl:message name="nmtoken"> *
    <wsdl:documentation .... />?
    <part name="nmtoken" element="qname"? type="qname"?/> *
  </wsdl:message>

  <wsdl:portType name="nmtoken">*
    <wsdl:documentation .... />?
    <wsdl:operation name="nmtoken">*
      <wsdl:documentation .... /> ?
      <wsdl:input name="nmtoken"? message="qname">?
        <wsdl:documentation .... /> ?
      </wsdl:input>
      <wsdl:output name="nmtoken"? message="qname">?
        <wsdl:documentation .... /> ?
```

```
27          </wsdl:output>
28          <wsdl:fault name="nmtoken" message="qname"> *
29              <wsdl:documentation .... /> ?
30          </wsdl:fault>
31       </wsdl:operation>
32    </wsdl:portType>
33
34    <wsdl:binding name="nmtoken" type="qname">*
35       <wsdl:documentation .... />?
36       <-- extensibility element --> *
37       <wsdl:operation name="nmtoken">*
38          <wsdl:documentation .... /> ?
39          <-- extensibility element --> *
40          <wsdl:input> ?
41              <wsdl:documentation .... /> ?
42              <-- extensibility element -->
43          </wsdl:input>
44          <wsdl:output> ?
45              <wsdl:documentation .... /> ?
46              <-- extensibility element --> *
47          </wsdl:output>
48          <wsdl:fault name="nmtoken"> *
49              <wsdl:documentation .... /> ?
50              <-- extensibility element --> *
51          </wsdl:fault>
52       </wsdl:operation>
53    </wsdl:binding>
54
55    <wsdl:service name="nmtoken"> *
56       <wsdl:documentation .... />?
57       <wsdl:port name="nmtoken" binding="qname"> *
58          <wsdl:documentation .... /> ?
59          <-- extensibility element -->
60       </wsdl:port>
61       <-- extensibility element -->
62    </wsdl:service>
63
64    <-- extensibility element --> *
65
66 </wsdl:definitions>
```

Types (7 line to 11): provides datatypes that will be used for communication between the XML Web Service and its clients.

Messages (13 line to 16): provides a message name, associated with a type, that will be used for communication

PortTypes (18 line to 32): Associates specific messages with port types, such as HTTPPost.

Bindings (34 line to 53): Binds the ports and XML Web Service methods to Internet protocols, such as SOAP.

Services (55 line to 62): supplies the address information for a service’s different ports of communication.

Use the following table as a guide:

<types>	What data types will be transmitted?
<messages>	What messages will be transmitted?
<portType>	What operations (function) will be supported?
<binding>	How will the messages be transmitted on the wire? What SOAP-specific details are there?
<service>	Where is the service located?

Operations in WSDL

Operations represent the various methods being exposed by the service. A typical operation contains two elements, *input* and *output*, but may also contain *documentation* and *fault*.

The input and output elements of an operation basically link the service's methods to SOAP messages that will provide the transport for input parameters and output results. Faults define the message that will be used to transport error messages should errors be encountered. The documentation element provides a method for attaching comments to a service's methods.

portType

WSDL supports four basic patterns of operation:

1) **One-way**
   The service receives a message. The operation therefore has a single input element.

2) **Request-response**
   The service receives a message and sends a response. The operation therefore has one input element, followed by one output element. To encapsulate errors, an optional fault element can also be specified.

3) **Solicit-response**
   The service sends a message and receives a response. The operation therefore has one output element, followed by one input element. To encapsulate errors, an optional fault element can also be specified.

4) **Notification**
   The service sends a message. The operation therefore has a single `output` element.

A simplified WSDL example:

```
<message name="getTermRequest">
   <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
   <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
  <operation name="getTerm">
      <input message="getTermRequest"/>
      <output message="getTermResponse"/>
  </operation>
</portType>
```

In the simplified example above, the <portType> element defines "glossaryTerms" as the name of a port, and "getTerm" as the name of an operation. The "getTerm" operation has an input message called "getTermRequest" and an output message called "getTermResponse". The <message> elements define the parts of each message and the associated data types.

In traditional programming, glossaryTerms is a function library, "getTerm" is a function with "getTermRequest" as the input parameter and getTermResponse as the return parameter.

A Complete WSDL document example

```
<?xml version="1.0" encoding="utf-8" ?>
  <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://tempuri.org/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" targetNamespace="http://tempuri.org/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
  <s:element name="generateFibo">
  <s:complexType>
  <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="iMaxRange" type="s:int" />
  </s:sequence>
  </s:complexType>
  </s:element>
  <s:element name="generateFiboResponse">
  <s:complexType>
  <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="generateFiboResult"
type="tns:ArrayOfAnyType" />
  </s:sequence>
  </s:complexType>
  </s:element>
  <s:complexType name="ArrayOfAnyType">
  <s:sequence>
  <s:element minOccurs="0" maxOccurs="unbounded" name="anyType" nillable="true" />
  </s:sequence>
  </s:complexType>
  </s:schema>
  </wsdl:types>
  <wsdl:message name="generateFiboSoapIn">
  <wsdl:part name="parameters" element="tns:generateFibo" />
  </wsdl:message>
  <wsdl:message name="generateFiboSoapOut">
  <wsdl:part name="parameters" element="tns:generateFiboResponse" />
  </wsdl:message>
  <wsdl:portType name="FibonacciServiceSoap">
  <wsdl:operation name="generateFibo">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">returns fibonacci from 0
to max range</wsdl:documentation>
  <wsdl:input message="tns:generateFiboSoapIn" />
  <wsdl:output message="tns:generateFiboSoapOut" />
  </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="FibonacciServiceSoap" type="tns:FibonacciServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="generateFibo">
  <soap:operation soapAction="http://tempuri.org/generateFibo" style="document" />
  <wsdl:input>
  <soap:body use="literal" />
```

```
</wsdl:input>
<wsdl:output>
<soap:body use="literal" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:binding name="FibonacciServiceSoap12" type="tns:FibonacciServiceSoap">
<soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="generateFibo">
<soap12:operation soapAction="http://tempuri.org/generateFibo" style="document" />
<wsdl:input>
<soap12:body use="literal" />
</wsdl:input>
<wsdl:output>
<soap12:body use="literal" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="FibonacciService">
<wsdl:port name="FibonacciServiceSoap" binding="tns:FibonacciServiceSoap">
<soap:address location="http://localhost:50612/FibonacciService.asmx" />
</wsdl:port>
<wsdl:port name="FibonacciServiceSoap12" binding="tns:FibonacciServiceSoap12">
<soap12:address location="http://localhost:50612/FibonacciService.asmx" />
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

This WSDL document has been generated for a service named FibonacciService – which generates the Fibonacci numbers, starting from 0 up to the Fibonacci number nearest to a user input argument. It consists of only one method, the *generateFibo* method, which is defined by a class named *FibonacciService*. The method *generateFibo* takes one parameter, *iMaxRange* of type int (integer), and returns *generateFiboResult* of type array (array of anytype means an arraylist with type not defined, in this case).

The following lines provide a guideline to identify the main elements in this example: Line 10 to 33: Types definition, Line 34 to 39: Message definition, Line 40 to 47: Port definition, Line 48 to 71: Binding definition, Line 72 to 79: Services definition.

Message:

generateFiboSoapIn : iMaxrange as input parameter

generateFiboSoapOut: generateFiboResult as return type

PortType:

generateFibo operation that consists of a resquest/response (generateFiboSoapIn/ generateFiboSoapOut)

Binding:

SOAP over HTTP

Service:

Service is available at: http://localhost:50612/FibonacciService.asmx