3 Web Service Description Language (WSDL)

Objectives

After reading this chapter, you should:

1. Possess a basic understanding of a WSDL
2. Be able to decipher a WSDL

WSDL assists service clients that need to know how to bind a service automatically. A service contract must be established between the service consumer and provider. A published WSDL describes in detail the contract, which may include messages, operations, bindings and locations of the service.

When a Web Service is ready for use, its location and access are made known to external systems. WSDL is based on the Interface Description Language (IDL), which describes the interface of a software component for other components to use. In RPC, a developer defines an interface of a component to be exposed to external applications that do not share the same language.

Once an interface is described, in most cases, a tool is used to generate client and server stubs for the client side and the server side, respectively, to use. The server application uses the server stub for its implementation of the service, while the client application uses the client stub for its service invocation.

3.1 WSDL structure

WSDL consists of two parts: abstract interface and concrete implementation. While the abstract interface describes the operations and messages of a service, the concrete implementation part binds the abstract interface with a concrete network address. Thus, the two together comprise a service.

A WSDL is an XML document with a root element named *definitions*. The definition includes two parts: abstract and concrete. The abstract descriptions consist of *types, message,* and *portType*. The concrete part consists of *binding*, and *service*. Abstract parts describe the operations that are independent of way in which the service will be implemented, while the concrete part specifies the protocol and location of the service where it can be invoked.

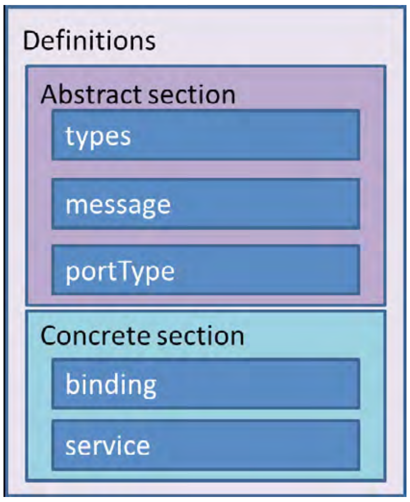
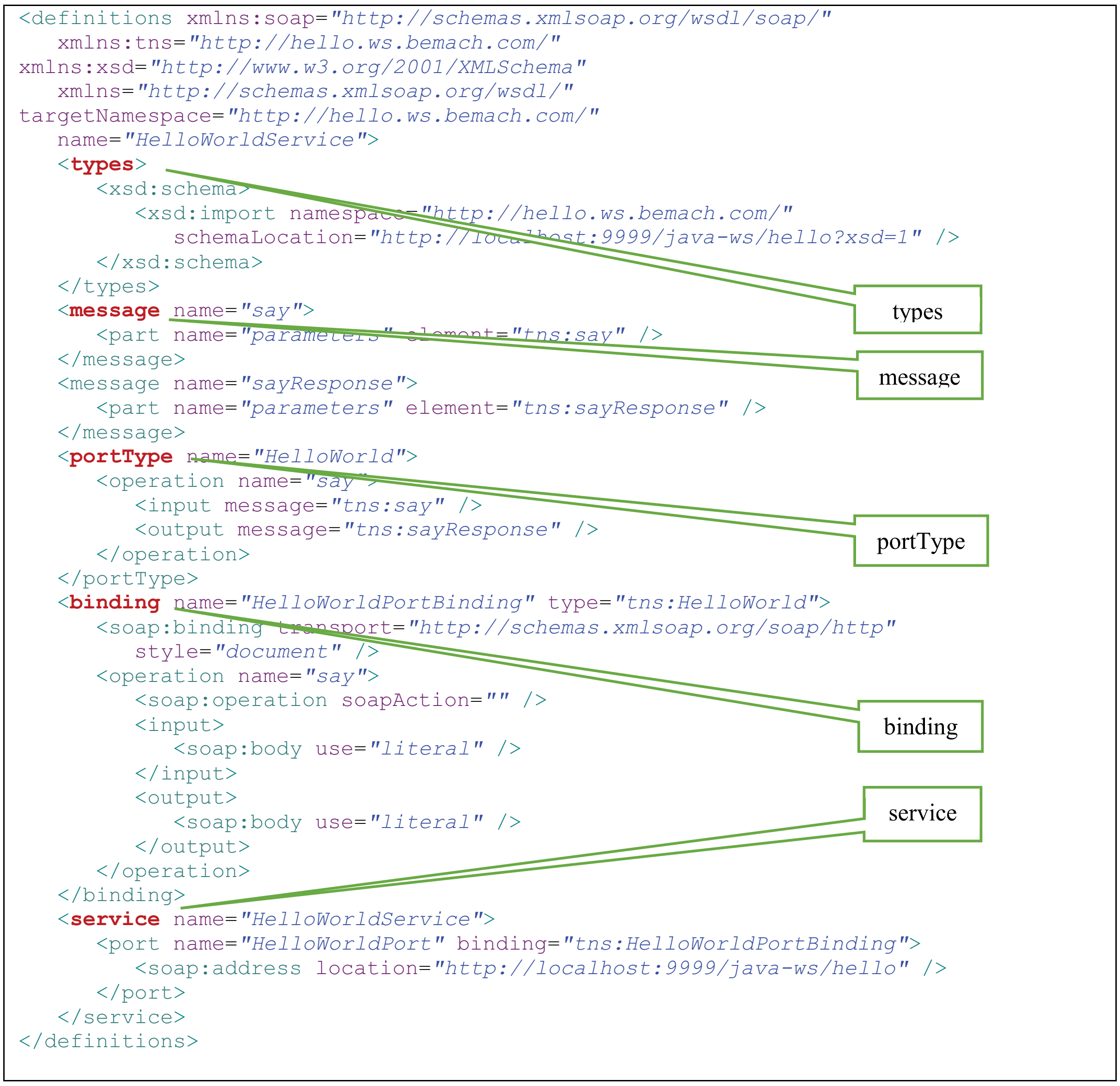


Figure 3-1. *WSDL structure*

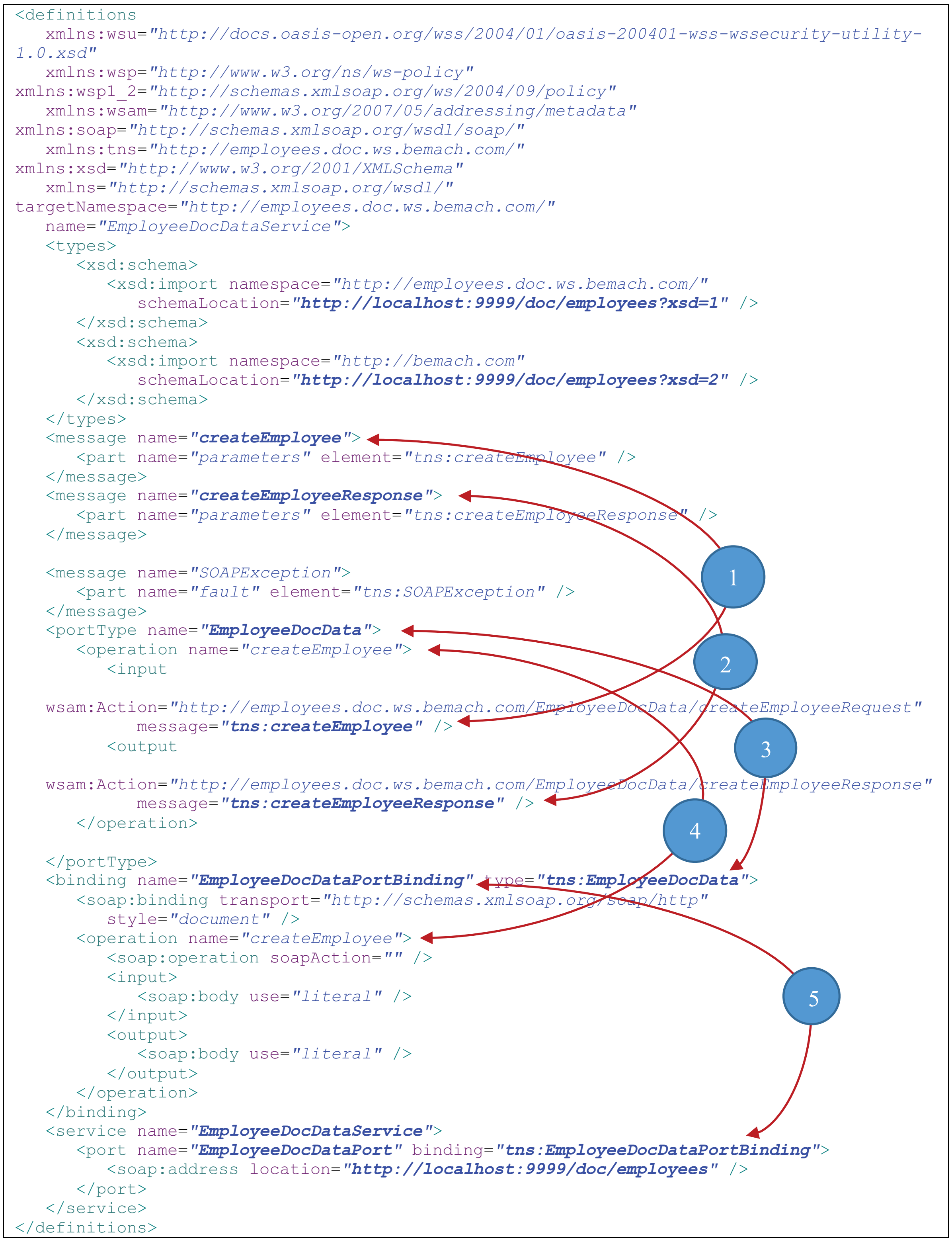
In the following section, we will examine the structure of a WSDL more closely to understand the relationships between the abstract interface and the concrete implementation.

Listing 3-1 Sample WSDL



```
<definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://hello.ws.bemach.com/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://hello.ws.bemach.com/"
   name="HelloWorldService">
   <types>
      <xsd:schema>
         <xsd:import namespace="http://hello.ws.bemach.com/"
            schemaLocation="http://localhost:9999/java-ws/hello?xsd=1" />
      </xsd:schema>
   </types>
   <message name="say">
      <part name="parameters" element="tns:say" />
   </message>
   <message name="sayResponse">
      <part name="parameters" element="tns:sayResponse" />
   </message>
   <portType name="HelloWorld">
      <operation name="say">
         <input message="tns:say" />
         <output message="tns:sayResponse" />
      </operation>
   </portType>
   <binding name="HelloWorldPortBinding" type="tns:HelloWorld">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
         style="document" />
      <operation name="say">
         <soap:operation soapAction="" />
         <input>
            <soap:body use="literal" />
         </input>
         <output>
            <soap:body use="literal" />
         </output>
      </operation>
   </binding>
   <service name="HelloWorldService">
      <port name="HelloWorldPort" binding="tns:HelloWorldPortBinding">
         <soap:address location="http://localhost:9999/java-ws/hello" />
      </port>
   </service>
</definitions>
```

Listing 3-2. Another Sample WSDL



```
<definitions
   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd"
   xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://employees.doc.ws.bemach.com/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://employees.doc.ws.bemach.com/"
   name="EmployeeDocDataService">
   <types>
      <xsd:schema>
         <xsd:import namespace="http://employees.doc.ws.bemach.com/"
            schemaLocation="http://localhost:9999/doc/employees?xsd=1" />
      </xsd:schema>
      <xsd:schema>
         <xsd:import namespace="http://bemach.com"
            schemaLocation="http://localhost:9999/doc/employees?xsd=2" />
      </xsd:schema>
   </types>
   <message name="createEmployee">
      <part name="parameters" element="tns:createEmployee" />
   </message>
   <message name="createEmployeeResponse">
      <part name="parameters" element="tns:createEmployeeResponse" />
   </message>

   <message name="SOAPException">
      <part name="fault" element="tns:SOAPException" />
   </message>
   <portType name="EmployeeDocData">
      <operation name="createEmployee">
         <input

   wsam:Action="http://employees.doc.ws.bemach.com/EmployeeDocData/createEmployeeRequest"
            message="tns:createEmployee" />
         <output

   wsam:Action="http://employees.doc.ws.bemach.com/EmployeeDocData/createEmployeeResponse"
            message="tns:createEmployeeResponse" />
      </operation>

   </portType>
   <binding name="EmployeeDocDataPortBinding" type="tns:EmployeeDocData">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
         style="document" />
      <operation name="createEmployee">
         <soap:operation soapAction="" />
         <input>
            <soap:body use="literal" />
         </input>
         <output>
            <soap:body use="literal" />
         </output>
      </operation>
   </binding>
   <service name="EmployeeDocDataService">
      <port name="EmployeeDocDataPort" binding="tns:EmployeeDocDataPortBinding">
         <soap:address location="http://localhost:9999/doc/employees" />
      </port>
   </service>
</definitions>
```

3.2 WSDL Interface

The service interface is defined using the portType element. This element contains a list of operation elements that make up a Web Service. Each operation consists of one in and one out message, and optionally one fault message. All operations exchange messages between a client and a server. A message is an abstract data type that contains the data.

Refer to listings 3-1 and 3-2 above, which highlight the elements of a basic WSDL document. The five major components of the WSDL and their relationships are visible.

3.2.1 <types> element

All data types are defined inside this element types. Usually, the element types points to an external URL that contains an XML schema which may contain other schemas. WSDL specification does not prohibit the use of other type definition systems; however, it prefers the XML schema.

3.2.2 <message> element

A message element is an abstract data type describing the input and output of an operation. It consists of parts that are linked to types. Each operation (@WebMethod) may contain three message types: input, output, and fault. Each of these message types is unique throughout the entire WSDL document. Each has a unique name. Inside each message, the parts define the actual types that are properly defined inside the <types> element specifically (RPC style) or via a schema (Document style).

3.2.3 <portType> element

The <portType> element consists of an abstract set of operations. These operations may be supported by one or more endpoints. Operations are used to exchange messages between a service consumer and the provider. Message exchange protocol can take one of the following patterns:

- One-way – the service endpoint receives a message
- Request-response – the service endpoint receives a message and sends a correlated message
- Solicit-response – the service endpoint sends a message and receives a correlated message
- Notification – the service endpoint sends a message

In the example above (Listing 3-2), the WS consists of one operation (i.e., createEmployee) that has two messages – one input and one output message. The input message is named 'tns:createEmployee' and the output message 'tns:createEmployeeResponse'. These two messages can be traced back to the message elements. Think of portType as being like a Java class where methods and attributes are defined.

3.2.3.1 <operation> element

A service may have one or more operations (or methods). Each operation is independently defined with one input message, one output message, and an optional fault message. These messages are defined using the message element. Similar to a Java class definition, each operation defines a behavior of a service.

3.3 WSDL Implementation

The <binding> element of the WSDL connects the WSDL abstract interface to concrete implementation. In this section, we will pay specific attention to the <binding> element where the connection takes place. Thus, in some way, <binding> is the central element of the entire WSDL specification. This is where the two worlds meet.

3.3.1 <binding> element

Binding maps <portType> to a implementation specified in the <service> element. From listing 3-1, <portType> HelloWorld is bound to <port> HelloWorldPort inside the <service> element. The implementation of HelloWorld is SOAP over HTTP (<soap:binding>). SOAP specifies data formats and HTTP is a specific protocol to be used for the service offering.

Binding does not specify any particular language in its implementation. The way in which the service is implemented is beyond the scope of the WSDL.

3.3.2 <service> element

A service is a collection of network-specific addresses (<port>) where the service may be rendered. <port> and <portType> are linked via a <binding> element. The connection is critical for runtime dynamic binding between the service consumers and providers.

The <port> element describes the network address that enables a service consumer to interact with the service being offered. A service with multiple ports is possible; thus, the choice is left to the consumer. In the sample WSDL, only one port is provided and its location is specified as http://localhost:9999/java-ws/hello. Note that the host name and port number can be modified at runtime to point to whichever server is hosting the service.

The following describes the linkages between the major elements of a WSDL.

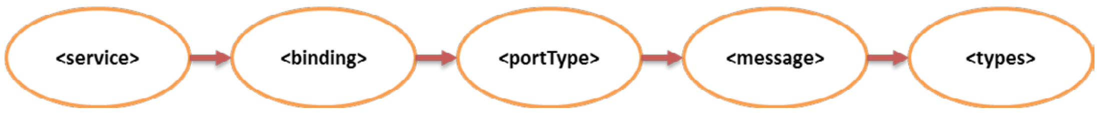


Figure 3-2 *Linkages inside WSDL*

3.4 References

Flaherty, B. (2004). “WSDL: Defining Web Services.” Intercom 51(8): 26–28.

Lessen, T., J. Nitzsche, et al. (2009). “Conversational Web Services: Leveraging BPEL (light) for expressing WSDL 2.0 message exchange patterns.” Enterprise Information Systems 3(3): 347–367.

Web Services Description Language (WSDL) 1.1, Retrieved August 27, 2007 from: http://www.w3.org/TR/wsdl

WSDL Tutorial, Retrieved November 7, 2007 from: http://www.w3schools.com/wsdl/default.asp