Chapter 5
Describing Web services

Mike P. Papazoglou
mikep@uvt.nl
Topics

- Why is a service description needed?
- Web Service Description Language
Why is a Service description needed?

- Web services must be defined in a consistent manner to be discovered and used by other services and applications. Web service consumers must determine the precise XML interface of a web service:
  - XML Schema alone cannot describe important additional details involved in communicating with a Web service.
- **Service description** reduces the amount of required common understanding and custom programming and integration:
  - It is a machine understandable standard describing the operations of a Web service.
  - It specifies the wire format and transport protocol that the Web service uses to expose this functionality.
  - It can also describe the payload data using a type system.
- **Service description + SOAP infrastructure** isolates all technical details, e.g., machine- and implementation language-specific elements, away from the service requestor’s application and the service provider’s Web service.
Topics

- Why is a service description needed?
- Web Service Description Language
Web Services Description Language

The web services description language (WSDL) is the XML-based service representation language used to describe the details of the complete interfaces exposed by Web services and thus is the means to accessing a Web service.

- For instance, neither the service requestor nor the provider should be aware of each other’s technical infrastructure, programming language, or distributed object framework (if any).
WSDL as a contract

- A Web service description in WSDL is an XML document that describes the mechanics of interacting with a particular Web service.
- It is inherently intended to constrain both the service provider and the service requestor that make use of that service. This implies that WSDL represents a “contract” between the service requestor and the service provider.
- WSDL is platform and language independent and is used primarily (but not exclusively) to describe SOAP-enabled services. Essentially, WSDL is used to describe precisely:
  - **what** a service does, i.e., the operations the service provides,
  - **where** it resides, i.e., details of the protocol-specific address, e.g., a URL, and
  - **how** to invoke it, i.e., details of the data formats and protocols necessary to access the service’s operations.
Characteristics of WSDL

- Operations and messages are described abstractly.
- Defines bindings to message formats and protocols:
  - Endpoints defined by binding concrete network protocol and message format to abstract operations and messages.
  - Can describe any endpoint regardless of the underlying network protocol or message format.
  - Defines how to locate the endpoint for the service:
    - Example: URLs for HTTP.
- Defines extensible SOAP and HTTP extensions.
Structure of WSDL documents

- WSDL documents can be separated into distinct sections:
  - The **service-interface definition** describes the general Web service interface structure. This contains all the operations supported by the service, the operation parameters, and abstract data types.
  - The **service implementation part** binds the abstract interface to a concrete network address, to a specific protocol, and to concrete data structures.
    - A web service client may bind to such an implementation and invoke the service in question.
- This enables each part to be defined separately and independently, and **reused** by other parts.
- The combination of these two parts contains **sufficient information** to describe to the service requestor how to invoke and interact with the Web service at a provider’s site.
  - Using WSDL, a requestor can locate a web service and invoke any of the publicly available operations.
WSDL document content

- Abstract (interface) definitions
  - `<types>` data type definitions
  - `<message>` operation parameters
  - `<operation>` abstract description of service actions
  - `<portType>` set of operation definitions

- Concrete (implementation) definitions
  - `<binding>` operation bindings
  - `<port>` association of an endpoint with a binding
  - `<service>` location/address for each binding

- Also:
  - `<import>` used to reference other XML documents
Web Service Interface Definition

- WSDL specifies a grammar and syntax that describes Web services as a collection of communicating endpoints.
  - A complete WSDL definition contains all of the information necessary to invoke a web service.
- The data being exchanged between the endpoints are specified as part of messages and every kind of processing activity allowed at an endpoint is considered as an operation.
- WSDL is layered top of the XML schema and provides the means to group messages into operations and operations into interfaces.
  - Collections of permissible operations at an endpoint are grouped together into **port types**.
  - WSDL also provides a way to define **bindings** for each interface and protocol combination along with the endpoint address for each one.
Listing 1: Example of WSDL interface definition

Port type with one operation
An operation with request (input) & response (output) message

Abstract data type definitions

Data that is sent
Data that is returned
<types> element

- The WSDL <types> element serves as a container that contains all abstract data types that define a Web service interface.
- A <type> element in WSDL is comparable to a data type in Java or C++.
  - WSDL uses a few primitive data types that XML schema definition (XSD) defines, e.g., int, float, long, short, string, boolean, and allows developers to either use them directly or build complex data types based on those primitive ones before using them in messages.
  - The data types and elements defined in the <types> element are used by message definitions when declaring the parts (payloads) of messages.
  - Any complex data type that the service uses must be defined using a <types> element.
- Listing 1 illustrates two complex types: POType and InvoiceType.
<message> element

- The <message> element describes the payload of a message used by a web service. A message consists of <part> elements, which are linked to <types> elements.
- In Listing 1, the PurchaseOrder service defines two <message> elements to describe the parameters and return values of that service.
  - POMessage (also below) describes the input parameters of the service, while
  - InvMessage represents the return (output) parameters.

```xml
<!-- message elements that describe input and output parameters for the PurchaseOrderService -->
<!--input message -->
<wsdl:message name="POMessage">
  <wsdl:part name="PurchaseOrder" type="tns:POType"/>
  <wsdl:part name="CustomerInfo" type="tns:CustomerInfoType"/>
</wsdl:message>
<!--output message -->
<wsdl:message name="InvMessage">
  <wsdl:part name="Invoice" type="tns:InvoiceType"/>
</wsdl:message>
```

```xml
<!-- message element that describes input and output parameters -->
<wsdl:message name="POMessage">
  <wsdl:part name="PurchaseOrder" element="tns:PurchaseOrder"/>
</wsdl:message>
```

RPC-style message

Document-style message
<portType>, <operation> elements

• A <portType> element defines an abstract type and its operations but not an implementation. A <portType> element is a logical grouping of <operations> in a Web service.
  – It describes the kinds of operations that a Web service supports – the messaging mode and payloads – without specifying the Internet protocol or physical address used.
  – The <portType> element is central to a WSDL description; the rest of the elements in the definition are essentially details that the <portType> element depends upon.

• Operations in WSDL represent the methods exposed by the service: they include the name of the method and the input and output parameters.
  – A typical <operation> element is composed of at most one <input> or <output> element and any number of <fault> elements.

• The WSDL example in Listing 1 contains a <portType> named PurchaseOrderPortType that supports a single <operation> called SendPurchase.
WSDL Implementation

- The purpose of WSDL is to specify a Web service abstractly and then to define how the WSDL developer will reach the implementation of these services.
- The service implementation part of WSDL contains the elements `<binding>`, `<port>`, and `<service>` and describes how a particular service interface is implemented by a given service provider.
- The service implementation describes
  - where the service is located, or more precisely;
  - which network address the message must be sent to in order to invoke the web service;
  - a WSDL service element.
- A service implementation document can contain references to more than one service interface document by means of `<import>` elements.
WSDL Elements Hierarchy

- Part
- Message
- Operation
- PortType
- Binding
- Port
- Service

Abstract endpoint type (no transport or encoding)
Concrete endpoint type (with transport or encoding)
Endpoint instance (with network address)
(related endpoint instances)
Listing 2: Example of WSDL implementation

Bind an abstract operation to this implementation and map the abstract input and output messages to these concrete messages.

Service name: PurchaseOrderService
Network address of service: http://supply.com:8080/PurchaseOrderService
<binding>, <port>, <service> elements

- The central element of the implementation description is the <binding> element. This element specifies how the client and Web service should exchange messages. The client uses this information to access the Web service.
- A <binding> element contains information of how the elements in an abstract service interface (<portType> element) are converted into concrete representation in a particular combination of
  - concrete protocols, e.g., SOAP or HTTP,
  - messaging styles, e.g., RPC or documents styles, and
  - formatting (encoding) styles, e.g., literal or SOAP encoding.
- A <port> element defines the location of a web service and we can think of it as the URL where the service can be found. A <port> associates an endpoint, for instance, a network address location or URL, with a specific WSDL <binding> element.
  - It is possible for two or more <port> elements to assign different URLs to the same <binding> element. This might be, for instance, useful for load balancing or fail-over purposes.
- A <service> element contains a collection (usually one) of WSDL <port> elements. Each <service> element is named, and each name must be unique among all services in a WSDL document.
Mapping the SendPurchase operation to an RPC-style SOAP message

```xml
<?xml version= "1.0" encoding= "UTF-8" ?>
<soap:Envelope
xmlns:soapbind= "http://schemas.xmlsoap.org/soap/envelope"
xmlns:tns= "http://supply.com/ PurchaseService/wsdl ">
<soap:Body>
<tns:SendPurchase>
<POType>
<PONumber> 223451 </PONumber>
<PODate> 10/28/2004 </PODate>
</POType>
......
<tns:SendPurchase>
</soap:Body>
</soap:Envelope>
```

Connecting the service interface with the service implementation
Elements of WSDL as part of requestor–service interaction

Each message part is defined by some type, either custom defined or XSD provided.

Input and output messages form an operation. A set of operations forms a port type.

A port exposes the service using a specific binding.

A binding specifies how the operations are invoked using a specific protocol, e.g., SOAP.

A service is a collection of related endpoints (ports) that the client wishes to invoke.

Java implementation
Connecting the abstract and concrete levels of a Web service.

<table>
<thead>
<tr>
<th>Concrete level</th>
<th>Abstract level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service name = S</td>
<td>portType name = PT</td>
</tr>
<tr>
<td>Port name = Po</td>
<td>operation name = Op</td>
</tr>
<tr>
<td>binding = B</td>
<td>input name = in</td>
</tr>
<tr>
<td>soap:address</td>
<td>message name = M</td>
</tr>
<tr>
<td>Binding name = B</td>
<td>(or output, fault)</td>
</tr>
<tr>
<td>port = PT</td>
<td>complexType name = T</td>
</tr>
<tr>
<td>operation name = Op</td>
<td>(or simpleType name = T)</td>
</tr>
<tr>
<td>input</td>
<td>message name = M</td>
</tr>
<tr>
<td>(or output, fault)</td>
<td>part name = Pa</td>
</tr>
<tr>
<td></td>
<td>type = T</td>
</tr>
</tbody>
</table>
WSDL Message Exchange Patterns

- WSDL interfaces support four common types of operations that represent possible combinations of input and output messages.
- The WSDL operations correspond to the incoming and outgoing versions of two basic operation types:
  - an incoming single message passing operation and its outgoing counterpart ("one-way" and "notification" operations),
  - the incoming and outgoing versions of a synchronous two-way message exchange ("request/response" and "solicit response").
- Any combination of incoming and outgoing operations can be included in a single WSDL interface:
  - these four types of operations provide support for both push and pull interaction models at the interface level.
One-way operation

- A one-way operation is an operation in which the service endpoint receives a message, but does not send a response.
  - An example of a one-way operation might be an operation representing the submission of an order to a purchasing system. Once the order is sent, no immediate response is expected.
  - This message exchange pattern is typically thought of as asynchronous messaging. In an RPC environment, a one-way operation represents a procedure call to which no return value is assigned.
  - A one-way message defines only an input message. It requires no output message and no fault. Next to the request/response message exchange pattern, this is the most popular message exchange pattern employed today.

```xml
<!-- portTyepe element describes the abstract interface of a Web service -->
<wsdl:portType name="SubmitPurchaseOrder_PortType">
  <wsdl:operation name="SubmitPurchaseOrder">
    <wsdl:input name="order" message="tns:SubmitPurchaseOrder_Message"/>
  </wsdl:operation>
</wsdl:portType>
```
Request/response operation

- A request/response operation is an operation in which the service end point receives a message and returns a message in response.
- If an <operation> element is declared with a single <input> element followed by a single <output> element, it defines a request/response operation. By listing the <input> tag first, the <operation> indicates that the Web service receives a message that is sent by the client. Listing the <output> tag second indicates that the Web service should respond to the message.

```xml
<!-- portType element describes the abstract interface of a Web service -->
<wsdl:portType name="PurchaseOrder_PortType">
  <wsdl:operation name="SendPurchase">
    <wsdl:input message="tns:POMessage"/>
    <wsdl:output message="tns:InvMessage"/>
  </wsdl:operation>
</wsdl:portType>
```
Notification operation

• A notification operation is an operation in which the service endpoint sends a message to a client, but it does not expect to receive a response.
• This type of messaging is used by services that need to notify clients of events.
  – Notification is when a <portType> element contains an <output> tag, but no <input> message definitions.
• Here the client (subscriber) has registered with the Web service to receive messages (notifications) about an event.
  – An example of this could be a service model in which events are reported to the service and where the endpoint periodically reports its status.
  • No response is required in this case, as most likely the status data is assembled and logged and not acted upon immediately.
Solicit/response operation

- A solicit/response operation is an operation in which the service endpoint sends a message and expects to receive an answering message in response.
- This is the opposite of the request/response operation since the service endpoint is initiating the operation (soliciting the client), rather than responding to a request.
- Solicit/response is similar to notification messaging, except that the client is expected to respond to the Web service.
- With this type of messaging the `<portType>` element first declares an `<output>` tag and then an `<input>` message definition – exactly the reverse of a request/response operation.
  - An example of this operation might be a service that sends out order status to a client and receives back a receipt.