IDL and WSDL - A Comparison

**By: Gopi Kumar Bulusu, CEO and Managing Director,
Sankhya Technologies Private Limited
gopi@sankhya.com**

Object oriented programmers think of an object as an entity that can receive and respond to messages. The set of messages an object can receive defines an object's public interface. IDL can be used to describe the interface, or the set of messages that can be sent to an object. In fact, IDL stands for Interface Definition Language. IDL™ is a standard from the Object Management Group and is an essential part of the CORBA® specification.

WSDL stands for the Web Services Description Language and is part of the WS-1 specification from W3C. Even though WSDL is an interface description language it differs from IDL in many important ways. Understanding the similarities and differences between these two standards is important for distributed systems engineers.

Let us now take a simple "Hello World" IDL sample, containing an interface hello that contains the method greet, which has a single string parameter message.

```
// hello Interface with the greet method

interface hello {
        string greet(in string message);
};
```

Let's now take a look at how a WSDL description of a service that provides a greet method looks like. This WSDL has been generated using an IDL to WSDL translator and removing parts of the output file to simplify.

First, consider the header, which consists of a version and a comment from the WSDL generator.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
  File generated using SANKHYA Varadhi IDL Compiler
  idlc - 1.2 Beta
  Language Option: IDL to WSDL Mapping
  Please DO NOT edit
-->
```

The header is followed by a set of definitions starting with the tag:

```
<definitions name="hello"
```

This is followed by XML namespace declarations used in this description. The references to the CORBA name spaces are there because, this WSDL is created using an IDL to WSDL translator and are not required otherwise.

```
        xmlns:tns="http://www.omg.org/IDL-MAPPED/"
        targetNamespace="http://www.omg.org/IDL-MAPPED/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema/"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:CORBA="http://www.omg.org/IDL-WSDL/1.0/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">

        <wsdl:documentation>
            <CORBA:SourceIDL>
              <CORBA:source>hello.idl</CORBA:source>
              <CORBA:version>1.0</CORBA:version>
                </CORBA:SourceIDL>
        </wsdl:documentation>
```

Type definitions are embedded between the <type> and </type> tags. Here the standard CORBA exceptions are defined by the WSDL generator, as these can be returned in a fault response message. Other services may define different types like structures that may be required for passing data between the web services client (request) and server (response).

The CORBA.completion_status is being defined as a simple type, that is a string, taking three 3 possible values COMPLETED_YES, COMPLETED_NO, COMPLETED_MAYBE.

```
<types>
  <xsd:schema targetNamespace="http://www.omg.org/IDL-WSDL/1.0"
          xmlns="http://www.w3.org/2001/XMLSchema">

  <xsd:simpleType name="CORBA.completion_status">
      <xsd:restriction base="xsd:string">
          <xsd:enumeration value="COMPLETED_YES"/>
          <xsd:enumeration value="COMPLETED_NO"/>
          <xsd:enumeration value="COMPLETED_MAYBE"/>
      </xsd:restriction>
  </xsd:simpleType>
```

The CORBA.SystemException is being defined as a complexType and includes completion status and the minor number of the exception.

```
  <xsd:complexType name="CORBA.SystemException">
      <xsd:sequence>
          <xsd:element
              name="minor" type="xsd:unsignedInt"
              maxOccurs="1" minOccurs="1"/>
          <xsd:element
              name="completion_status" type="CORBA.completion_status"
              maxOccurs="1" minOccurs="1"/>
      </xsd:sequence>
  </xsd:complexType>

  </xsd:schema>
</types>
```

After defining the types, the different messages that can be exchanged between the client and server are defined. The message CORBA.SystemExceptionMessage can be returned by the server to the client in case of a fault.

```
<!-- Message related to CORBA System Exception-->
<message name="CORBA.SystemExceptionMessage">
  <part name="_return" type="CORBA.SystemException"/>
</message>
```

The message hello.greet defines the message that can be sent by the client to the server to invoke the greet method. The greetResponse message can be returned by the server in response to the greet message. These are combined to form a single operation in the next part.

```
<!-- Messages related to portType : hello -->
<message name="hello.greet">
  <part name="message" type="xsd:string"/>
</message>

<message name="hello.greetResponse">
  <part name="_return" type="xsd:string"/>
</message>
```

The next important part of a WSDL file is a list of port types, these are collections of operations, where an operation establishes a valid request message and a valid response message. A WSDL portType corresponds to an IDL interface. The declaration below, shows how the hello.greet, hello.greetResponse and CORBA.SystemException messages are combined into a single operation, which is the only member of a portType hello.

```
<!-- portType for interface "hello" -->
<portType name="hello">

  <operation name="greet">
    <input message ="tns:hello.greet"/>
    <output message ="tns:hello.greetResponse"/>
    <fault message ="tns:CORBA.SystemException"/>
  </operation>

</portType>
```

Here the similarity between the two languages ends. A WSDL file, in addition to the interface description, also provides information on the binding of a portType (interface) to a particular transport as follows:

```
<!-- binding for interface "hello" -->
<binding name ="helloBinding" type="tns:hello">
  <soap:binding
    style="rpc"
    transport = "http://schemas.xmlsoap.org/soap/http"/>

  <operation name="greet">
```

```
    <soap:operation soapAction="hello#greet"/>
    <input>
      <soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://www.omg.org/IDL-MAPPED/hello"
        use="encoded"/>
    </input>
    <fault message ="CORBA.SystemException"/>
  </operation>

</binding>
```

The binding tag provides information required for encoding the messages using SOAP (Simple Object Access Protocol) over HTTP transport. This information is not part of the CORBA IDL interface. CORBA applications can exchange messages using GIOP (General Inter ORB Protocol) over TCP/IP, referred to as IIOP (Internet Inter ORB Protocol).

A WSDL service tag is used to define an instance of a service that supports one or more port types. It basically provides the address where such a service can be contacted, and can include additional information. The service tag can be used to declare a service (helloService), specify a set of portTypes supported by the service (helloPort) and provide the location for the service as shown below.

```
<service name ="helloService">
  <port name ="helloPort" binding="tns:helloBinding">
    <soap:address location="http://www.dummyurl.com/hello"/>
  </port>
</service>
</definitions>
```

In the case of CORBA IDL, interfaces are supported by Objects (instead of a service) and a CORBA server object can be referenced using an IOR, a corbaloc or a name (using a naming service). This information is not part of the interface description.

Conclusion:

We have compared IDL and WSDL using a simple example. Both these languages are in reality more powerful and can be used to express more complex intefaces using interface inheritance, modules and complex types including arrays, sequences and structures.

However, based on this simple comparison, it can be seen that an IDL type corresponds to a WSDL type, an IDL method corresponds to a WSDL operation and an IDL interface corresponds to a WSDL portType. A WSDL service corresponds to a CORBA object which can be referenced with a corbaloc, corbaname or an IOR.

We can conclude that IDL is a general purpose high level interface description language, easy to create, read and understand for humans. Along with other specifications from OMG® IDL forms a solid foundation for developing and deploying a wide range of distributed systems.

WSDL is a lower level interface description language which is more difficult to create and understand for humans. Web services clients can get all of the information required to find and access a service in a single WSDL document. The WSDL description of an operation is also simple to convert into XML or HTML to present as a form to a user making it suitable for developing distributed applications that take advantage of the World Wide Web.

For Details, contact:

Sankhya Technologies Private Limited,
#30-15-58, Third Floor, Silver Willow,
Dabagardens, Visakhapatnam 530 020 India.

Tel: +91 891 5542 666
Fax: +91 891 5542 665
Toll Free: 1 600 12 4477

Email: sales@sankhya.com

Website: www.sankhya.com
