WSDL 1.1 Overview

Marlon Pierce and Geoffrey Fox
Community Grids Lab
Indiana University
mpierce@cs.indiana.edu
What Is WSDL?

- **Web Service Description Language**
  - W3C specification
  - See [http://www.w3.org/TR/wsdl](http://www.w3.org/TR/wsdl) for the official “note” for WSDL 1.1.
  - WSDL 1.1 never became a full “recommendation”.
  - WSDL 2.0 working draft just completed its public call for comments.

- **This slide set will review WSDL 1.1, which is still the “standard”**.
  - WSDL 2.0 should replace this soon.

- **We review 2.0 briefly**.
Why Use WSDL?

- WSDL uses XML to describe interfaces
  - Programming language independent way to do this.
  - So you can use (for example) C++ programs to remotely invoke Java programs and vice versa.

- Consider Web browsers and Web servers:
  - All web browsers work pretty well with all web sites.
  - You don’t care what kind of web server Amazon.com uses.
  - Amazon doesn’t care if you use IE, Mozilla, Konqueror, Safari, etc.
  - You all speak HTTP.

- WSDL (and SOAP) are a generalization of this.

- Note I will describe WSDL from an Remote Procedure Call/Remote Method Invocation point of view.
  - But WSDL and SOAP also support more a more message-centric point of view.
  - C.f. Java Messaging System.
  - This is probably the way of the future for Web Services.
public class echoService implements echoServiceInterface{
    public String echo(String msg) {
        return msg;
    }
    public static void main(String[] args) {
        new echoService().echo("hello");
    }
}
The Echo Interface

/**
 * All implementers of this interface must implement the echo() method.
 */

public interface echoServiceInterface {
    public String echo(String toEcho);
}
Now Use Echo As A Remote Service

- We can take the previous Java program and deploy it in Tomcat as a service.
- Clients can then invoke the echo service.
  - WSDL tells them how to do it.
  - Clients don’t need to know anything about the service implementation or even language.
- WSDL is the latest IDL
  - DCE and CORBA IDL were two older examples.
What Does echoServiceInterface Look Like In WSDL?

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions
    targetNamespace="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"
    xmlns="http://schemas.xmlsoap.org/wsdl/">

  <wsdl:types />

  <wsdl:message name="echoResponse">
    <wsdl:part name="echoReturn" type="xsd:string" />
  </wsdl:message>

  <wsdl:message name="echoRequest">
    <wsdl:part name="in0" type="xsd:string" />
  </wsdl:message>

  <wsdl:portType name="Echo">
    <wsdl:operation name="echo" parameterOrder="in0">
      <wsdl:input message="impl:echoRequest" name="echoRequest" />
      <wsdl:output message="impl:echoResponse" name="echoResponse" />
    </wsdl:operation>
  </wsdl:portType>

</wsdl:definitions>
```

There’s more...
What Does This Look Like In WSDL, Continued?

```xml
<wSDL:binding name="EchoSoapBinding" type="impl:Echo">
  <wSDL:soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wSDL:operation name="echo">
    <wSDL:soap:operation soapAction=""/>
    <wSDL:input name="echoRequest">
      <wSDL:soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"
        use="encoded"/>
    </wSDL:input>
    <wSDL:output name="echoResponse">
      <wSDL:soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"
        use="encoded"/>
    </wSDL:output>
  </wSDL:operation>
</wSDL:binding>
<wSDL:service name="EchoService">
  <wSDL:port binding="impl:EchoSoapBinding" name="Echo">
    <wSDL:soap:address location="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"/>
  </wSDL:port>
</wSDL:service>
</wSDL:definitions>
```

Don’t strain your eyes. We will break this down.
I’m sure you are impressed with the previous two slides. One could write WSDL by hand, but this is not the usual way. It was automatically generated by Apache Axis. Most other Web service tools will do the same. We will go through the construction, though, for understanding. You should not think of WSDL (and SOAP) as programming languages.

- They are just assertions, or descriptions.
WSDL Parts

- **Types**
  - Used to define custom message types

- **Messages**
  - Abstraction of request and response messages that my client and service need to communicate.

- **PortTypes**
  - Contains a set of operations.
  - Operations organize WSDL messages.
  - Operation->method name, PortType->java interface

- **Bindings**
  - Binds the PortType to a specific protocol (typically SOAP over http).
  - You can bind one PortType to several different protocols by using more than one port.

- **Services**
  - Gives you one or more URLs for the service.
  - Go here to execute “echo”.
Echo Service WSDL,
Section by Section
The WSDL Schema

- See [http://www.w3.org/TR/wsdl](http://www.w3.org/TR/wsdl) for the official recommendation.

- The full WSDL schema is given here.
  - Bryan Carpenter’s XML lectures cover everything you need to understand this schema.
  - But we will instead focus on a specific example.
Namespaces

- The WSDL document begins with several XML namespace definitions.
- Namespaces allow you to compose a single XML document from several XML schemas.
- Namespaces allow you to identify which schema an XML tag comes from.
  - Avoids name conflicts.
- See earlier XML lectures
- As we will see, the Axis namespace generator went overboard.
  - Not all of these are used.
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"
xmlns="http://schemas.xmlsoap.org/wSDL/"
xmns:apachesoap="http://xml.apache.org/xml-soap"
xmns:impl="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"
xmns:intf="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"
xmns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmns:wSDL="http://schemas.xmlsoap.org/wSDL/"
xmns:wSDLsoap="http://schemas.xmlsoap.org/wSDL/soap/"
xmns:xsd="http://www.w3.org/2001/XMLSchema">

...

</wsdl:definitions>
Namespace Quiz

- What is the default namespace of our XML doc?
- What does \textless wsdl:definitions ... \textgreater{} mean?
- What does \texttt{xmlns:xsd=\texttt{http://www.w3.org/2001/XMLSchema}} mean?
- Is \texttt{http://www.w3c.org/2001/XMLSchema} a URI or a URL?
- What is the target namespace of this document?
- What is the target namespace used for?
Quiz Answers

- [http://schemas.xmlsoap.org/wsd1/](http://schemas.xmlsoap.org/wsd1/)
- This means `<definitions>` belongs to the schema named [http://schemas.xmlsoap.org/wsd1/](http://schemas.xmlsoap.org/wsd1/), labeled wsdl: in this doc.
- It means that elements from the XML schema that appear in this WSDL document will be labeled by `<xsd:...>`
- Technically, it is used here as a URI; that is, it is a structured name. The URL does exist, however.
  - Recall URLs are special cases of URIs. URIs are names/identifiers. URLs also have a specific location on the web.
- [http://grids.ucs.indiana.edu:8045/GCWS/services/Echo](http://grids.ucs.indiana.edu:8045/GCWS/services/Echo)
- The target namespace is the namespace that will be used when we validate the document.
  - Note this isn’t used in our document since we define no complex types of our own.
  - See next section.
WSDL Types

Use `<types/>` to declare local message structures.
What Does echoServiceInterface Look Like In WSDL?

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions ...>
  <wsdl:types />
  <wsdl:message name="echoResponse">
    <wsdl:part name="echoReturn" type="xsd:string" />
  </wsdl:message>
  <wsdl:message name="echoRequest">
    <wsdl:part name="in0" type="xsd:string" />
  </wsdl:message>
...
</wsdl:definitions>
```
WSDL Types

- WSDL messages don’t need to declare types when just sending XML Schema primitive objects.

- EchoService just has string messages.
  - So no special types definitions are needed in our WSDL.

- Strings are an XML schema built-in type.
  - See earlier XML lectures.
  - Or next slide...
Schema Built In Types

- anyType
- anySimpleType
- all complex types
- boolean
- base64Binary
- hexBinary
- float
- double
- anyURI
- QName
- NOTATION
- string
- decimal
- integer
- token
- nonPositiveInteger
- long
- nonNegativeInteger
- language
- Name
- MM_TOKEN
- negativeInteger
- int
- unsignedLong
- positiveInteger
- short
- unsignedInt
- byte
- unsignedShort
- unsignedByte
When Would I Need A Type?

- Any time your Web Service needs to send data formatted by anything other than XML Schema built-in types, you must define the type in WSDL.

- Example: Arrays are not built-in types!
  - Arrays of strings, ints, etc., must be defined in the WSDL `<type></type>` structure.
How Does WSDL Encode String Arrays?

- Imagine that my echo service actually echoes back an array of strings.
- Arrays are not part of the built-in types, so I will have to define them myself.
- Luckily for us, SOAP defines arrays, so we can import this definition.
- Next slide shows what this looks like.
String Array Example

```xml
<wsl:types>
  <schema
    targetNamespace="http://grids.ucs.indiana.edu:8045/GCWS/services/EchoArray"
    xmlns="http://www.w3.org/2001/XMLSchema">
    <import
      namespace="http://schemas.xmlsoap.org/soap/encoding/" />
    <complexType name="ArrayOf_xsd_string">
      <complexContent>
        <restriction base="soapenc:Array">
          <attribute ref="soapenc:arrayItemType"
            wsdl:arrayType="xsd:string[]" />
        </restriction>
      </complexContent>
    </complexType>
    <element name="ArrayOf_xsd_string" nillable="true"
      type="impl:ArrayOf_xsd_string" />
  </schema>
</wsl:types>
```
WSDL String Array Types

- WSDL `<type/>` is nothing more than an extensibility placeholder in WSDL.
- Technically, the WSDL schema specifies that `<type>  </type>` can contain a `<sequence>` of 0 or more `<any>` tags.
  - Look at the WSDL schema.
- And note that the `<any/>` tag acts like wildcard.
  - You can insert any sort of xml here.
- See slides from XML lectures.
Inserting a Type

- Between `<type>` </type>`, we insert a `<schema>`.  

- Since arrays are defined in SOAP encoding rules, I next `import` the appropriate schema.
  - I import the definition of the SOAP Array and extend it to a String array.
  - Typically imports also have “location” attributes
    - “This namespace is located here for download.”

- Next, insert our own local definition of a type called “ArrayOf_xsd_string”.

- This is a restricted extension of the SOAP Array complex type.
  - We only allow 1 dimensional string arrays
  - It is also nullable—I am allowed to return a “null” value for the string.
Handling Other XML Types

- You can also express other message arguments as XML.
  - Examples: a purchase order, an SVG description of an image, a GML description of a map.
- In practice, these are handled by automatic Bean serializers/deserializers.
  - Castor is an example: [http://www.castor.org/](http://www.castor.org/)
  - XMLBeans is another [http://xml.apache.org/xmlbeans/](http://xml.apache.org/xmlbeans/)
- These are tools that make it easy to convert between XML and JavaBeans.
- By “JavaBeans” I mean objects that associate simple get/set methods with all data.
- Implementation dependent.
WSDL Messages
The echoServiceInterface messages

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions>
  <wsdl:types />
  <wsdl:message name="echoResponse">
    <wsdl:part name="echoReturn" type="xsd:string" />
  </wsdl:message>
  <wsdl:message name="echoRequest">
    <wsdl:part name="in0" type="xsd:string" />
  </wsdl:message>
  <wsdl:portType name="Echo">
    <wsdl:operation name="echo" parameterOrder="in0">
      <wsdl:input message="impl:echoRequest" name="echoRequest" />
      <wsdl:output message="impl:echoResponse" name="echoResponse" />
    </wsdl:operation>
  </wsdl:portType>
  ...
</wsdl:definitions>
```
Our Echo Messages

<wsdl:message name="echoResponse">
  <wsdl:part name="echoReturn" type="xsd:string" />
</wsdl:message>

<wsdl:message name="echoRequest">
  <wsdl:part name="in0" type="xsd:string" />
</wsdl:message>
Remember the Echo Service implementation?

- Our echo service takes a string argument and returns a string answer.
- In WSDL, I first abstract these as messages.
  - Echo needs two messages.
- Note we have not yet said message is the request and which is the response.
  - That is the job of the portType operations, coming up.
Structure of a Message

- WSDL `<message>` elements have name attributes and one or more `parts`.
  - The message name should be unique for the document.
  - `<operation>` elements will refer to messages by name.

- I need one `<part>` for each piece of data I need to send in that message.

- Each `<part>` is given a name and specifies its type.
  - `<part>` types can point to `<wsdl:type>` definitions if necessary.
  - Our service just needs `xsd:strings`, so no problem.
More Messages

- Our simple service only has one method.
  - What if it had echoEnglish(), echoSpanish(), and echoFrench()?
  - Each takes a string in English and echoes back the string in another language.
- Then we would need 6 messages, each with one part.
portTypes and operations
The echoServiceInterface portType

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions>
  <wsdl:types />
  <wsdl:message name="echoResponse">
    <wsdl:part name="echoReturn" type="xsd:string" />
  </wsdl:message>
  <wsdl:message name="echoRequest">
    <wsdl:part name="in0" type="xsd:string" />
  </wsdl:message>
  <wsdl:portType name="Echo">
    <wsdl:operation name="echo" parameterOrder="in0">
      <wsdl:input message="impl:echoRequest" name="echoRequest" />
      <wsdl:output message="impl:echoResponse" name="echoResponse" />
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```
EchoService portType

<wSDL:portType name="Echo">
  <wSDL:operation name="echo" parameterOrder="in0">
    <wSDL:input message="impl:echoRequest" name="echoRequest" />
    <wSDL:output message="impl:echoResponse" name="echoResponse" />
  </wSDL:operation>
</wSDL:portType>
WSDL portTypes

- WSDL messages are only abstract messages.
  - We bind them to operations within the portType.

- The structure of the portType specifies (still abstractly) how the messages are to be used.
  - Think of operations->java methods and portTypes->java interfaces.
PortTypes support four types of messaging:

- **One way**: Client sends a message to the service and doesn’t want a response.
  - `<input>` only.
- **Request-Response**: Client sends a message and waits for a response.
  - `<input>`, then `<output>`
- **Solicit-Response**: Service sends a message to the client first, then the client responds.
  - `<output>`, then `<input>`
- **Notification**: `<output>` only.

These still are abstract. We must implement them using some message protocol.

- HTTP units of transmission are request and response, so mapping Solicit-Response to HTTP will take some work.
The echo service has one method, echo.

It takes one string argument and returns one string.

In WSDL, the portType is “Echo”, the operation is “echo”.

The messages are organized into input and output.
  • Messages are placed here as appropriate.
  • That is, <input> takes the <echoRequest> message.
Parameter Order

- This attribute of operation is used to specify zero or more space-separated values.
- The values give the order that the input messages must be sent.
- Echo is a bad example, since it only has one input parameter, named *in0*. 
WSDL Self-Referencing

- The WSDL `<input>` and `<output>` tags need to point back to the `<message>` definitions above:

  `<wsdl:message name="echoResponse">`
  `<wsdl:part name="echoReturn" type="xsd:string" />
  `</wsdl:message>`

  ...

  `<wsdl:portType name="Echo">`
  `<wsdl:operation name="echo" parameterOrder="in0">`
  ...
  `<wsdl:output message="impl:echoResponse" name="echoResponse" />
  `</wsdl:operation>`
  `</wsdl:portType>`
The Picture So Far…

Input Message

- Part

Output Message

- Part

portType

- Operation
  - Input
  - Output

hasInput

hasOutput
Bindings
Binding Section of WSDL

```xml
<wsdl:definitions>
...
<wsdl:binding name="EchoSoapBinding" type="impl:Echo">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="echo">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="echoRequest">
      <wsdlsoap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"
        use="encoded"/>
    </wsdl:input>
    <wsdl:output name="echoResponse">
      <wsdlsoap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"
        use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service name="EchoService">
  <wsdl:port binding="impl:EchoSoapBinding" name="Echo">
    <wsdlsoap:address location="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Don’t strain your eyes--we will zoom in.
So Far...

- We have defined abstract messages, which have XML values.
  - Simple or custom-defined types.

- We have grouped messages into operations and operations into portTypes.

- We are now ready to bind the portTypes to specific protocols.
The Binding for Echo

```xml
<wSDL:binding name="EchoSoapBinding" type="impl:Echo">
  <wsdlsoap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="echo">
    <wsdlsoap:operation soapAction="" />
    <wsdl:input name="echoRequest">
      <wsdlsoap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="[echo service namespace URI]"
        use="encoded" />
    </wsdl:input>
    <wsdl:output name="echoResponse">
      <wsdlsoap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="[echo service namespace URI]"
        use="encoded" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

The highlighted “wsdlsoap:” tags are extensions for SOAP message binding and not part of the WSDL schema.
WSDL SOAP Bindings

- In the previous slide, we specify several things:
  - We will use SOAP/HTTP
  - We will use RPC encoding style
    - Other choice is literal “document” style.
  - We specify the namespace associated with the Echo service input and output messages.
- All of this corresponds to SOAP message parts.
  - We will expand this in the next lecture.
Binding tags

- Binding tags are meant to bind the parts of portTypes to sections of specific protocols.
  - SOAP, HTTP GET/POST, and MIME are provided in the WSDL specification.
- Bindings refer back to portTypes by name, just as operations point to messages.
WSDL Internal References

- portType
  - Operation
    - Input
    - Output
  - binding
  - Operation
    - Input
    - Output
Structure of the Binding

- `<binding>` tags are really just placeholders.
- They are meant to be extended at specific places by `wsdl` protocol bindings.
  - These protocol binding rules are defined in supplemental schemas.
- The following box figure summarizes these things
  - Green boxes are part of WSDL
    - From the `wsdl` namespace, that is.
  - Red boxes are parts of the document from other schemas
    - From `wsdlsoap` namespace in the echo example.
Binding Structure
SOAP Bindings

- The WSDL bindings are meant to prescribe how the parts of the portType get transported.
- All the given bindings are to parts of SOAP messaging formats.
  - WSDL’s SOAP bindings define mappings.
  - We will look at these in upcoming lectures.
For now, note the following

- We specify SOAP encoding
- SOAP is a message format and needs a transport protocol, so we specify HTTP.
- Operation styles may be either “RPC” or “Document”.
  - We use RPC.
- SOAP Body elements will be used to actually convey message payloads.
  - RPC requires “encoded” payloads.
    - Each value (echo strings) is wrapped in an element named after the operation.
    - Useful RPC processing on the server side.
  - Documents are literal (unencoded)
    - Use to just send a payload of XML inside SOAP.
Binding Associations to SOAP

WSDL

- Binding
  - Operation
    - Input
    - Output

SOAP

- SOAP RPC
  - SOAP Action
    - SOAP Body
    - SOAP Body
Binding Restrictions

- Binding elements point by name to portTypes.
- WSDL allows more than one binding element to point to the same port type.

Why?
- Because a service may support multiple, alternative protocol bindings.
What Does It Mean?

- WSDL is not a programming language.
- A service that exposes an WSDL interface is just telling a client what it needs to do to communicate with the service.
  - Send me strings and I will return strings.
  - I expect SOAP messages that include the strings in the body.
  - I expect this body to be RPC encoded with the operation name so that I will know which operation the body contents belong to.
  - I will return SOAP messages that include Strings in the body.
  - These will also be encoded so that you know what to do with them.
Ports and Services
What Does This Look Like In WSDL, Continued?

```xml
<wSDL:definitions>
  ...
  <wSDL:binding>
    ...
  </wSDL:binding>
  <wSDL:service name="EchoService">
    <wSDL:port binding="impl:EchoSoapBinding" name="Echo">
      <wSDL:soap:address location="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo" />
    </wSDL:port>
  </wSDL:service>
</wSDL:definitions>
```
Ports and Services

<wSDL:service name="EchoService">
  <wSDL:port
    binding="impl:EchoSoapBinding"
    name="Echo">
    <wSDL:soap:address
      location="http://...../">
  </wSDL:port>
</wSDL:service>
Port and Service Tags

- The service element is a collection of ports.
  - That’s all it is for.
- Ports are intended to point to actual Web service locations
  - The location depends on the binding.
  - For SOAP bindings, this is a URL.
Ports and Services

- A service can have more than one port.
- Two ports can point back to the same binding element.
  - Ports refer to bindings by name
  - This allows you to provide alternative service locations.
- The figure on next slide conceptually depicts associating two ports to a single binding.
  - The ports differ only in the URLs of their services.
Port Associations to Bindings

Binding

Operation

Input

Output

Service

Port #1

URL #1

Port #2

URL #2
Summary of WSDL

- WSDL decouples remote service operations.
  - Types=custom message definitions.
    - Any data types not in the XML schema.
  - Message=name the messages that must be exchanged and their data types, possibly defined by <type>.
  - PortTypes=service interfaces
    - Operations=remote method signatures.
  - Bindings=mappings of portType operations to real message formats
  - Ports=locations (URLs) of real services.