Web Service Foundations: WSDL

What Is WSDL?

- Web Service Description Language
  - WSDL 2.0 is a W3C recommendation since 2007
    - Not yet fully supported by most vendors
  - WSDL 1.1 never became a full recommendation.
- We will review WSDL 1.1, which is the one compatible with Jolie and WS-BPEL.
  - Still very used.

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.
- The slides describe WSDL from a Remote Procedure Call/Remote Method Invocation point of view.
  - But WSDL (and SOAP) also support a more message-centric point of view.
  - We will see an example in Jolie

A Very Simple Example: Echo

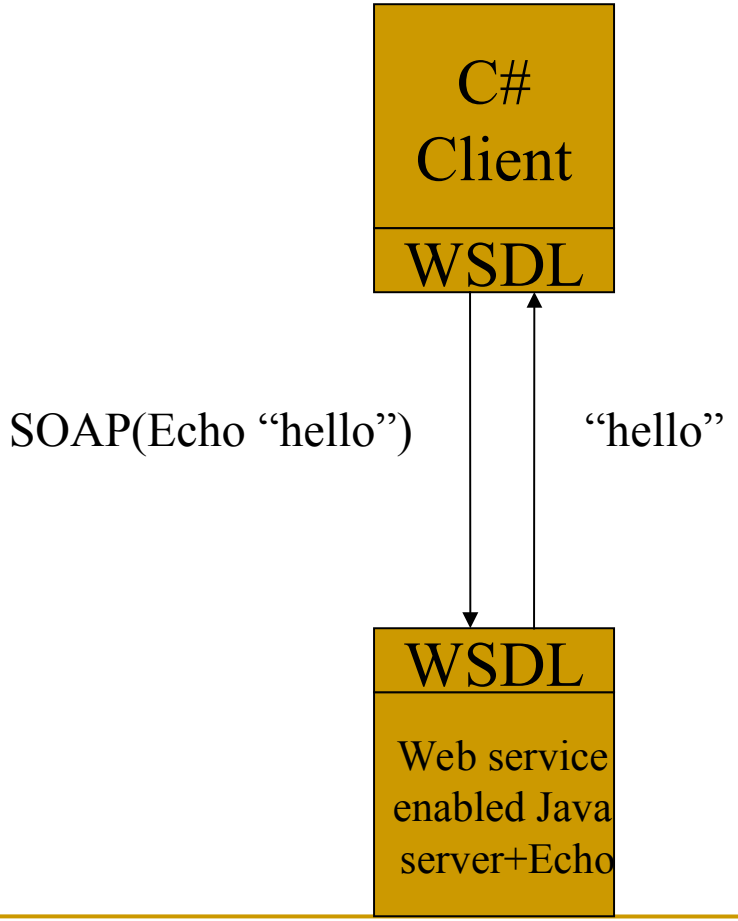
```
public class echoService implements echoServiceInterface{
    public String echo(String msg) {
        return msg;
    }
    public static void main(String[] args) {
        new echoService().echo(“hello”);
    }
}
```

The Echo Interface

```
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 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.



The echoServiceInterface In WSDL?

```
<?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/"
    xmlns:apachesoap="http://xml.apache.org/xml-soap"
    xmlns:impl="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"
    xmlns:intf="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <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>
```

There’s more…

What Does This Look Like In WSDL, Continued?

```
<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 break this down

Writing WSDL

- 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 (e.g., Jolie) will do the same from your service code.
- 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
  - Contain a set of operations.
  - Operations organize WSDL messages.
  - Operation->method name, portType->java interface
- Bindings
  - Bind 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
  - Give you one or more URLs for the port.
  - Connect to here to execute the service

Echo Service WSDL, Section by Section

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.
- The Axis namespace generator went overboard.
  - Not all of these are used.
- targetNamespace is the namespace of the service we are describing

Front Matters

```
<?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/"
    xmlns:apachesoap="http://xml.apache.org/xml-soap"
    xmlns:impl="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"
    xmlns:intf="http://grids.ucs.indiana.edu:8045/GCWS/services/Echo"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">


…
</wsdl:definitions>
```

WSDL Types

Use <types/> to declare local message structures.

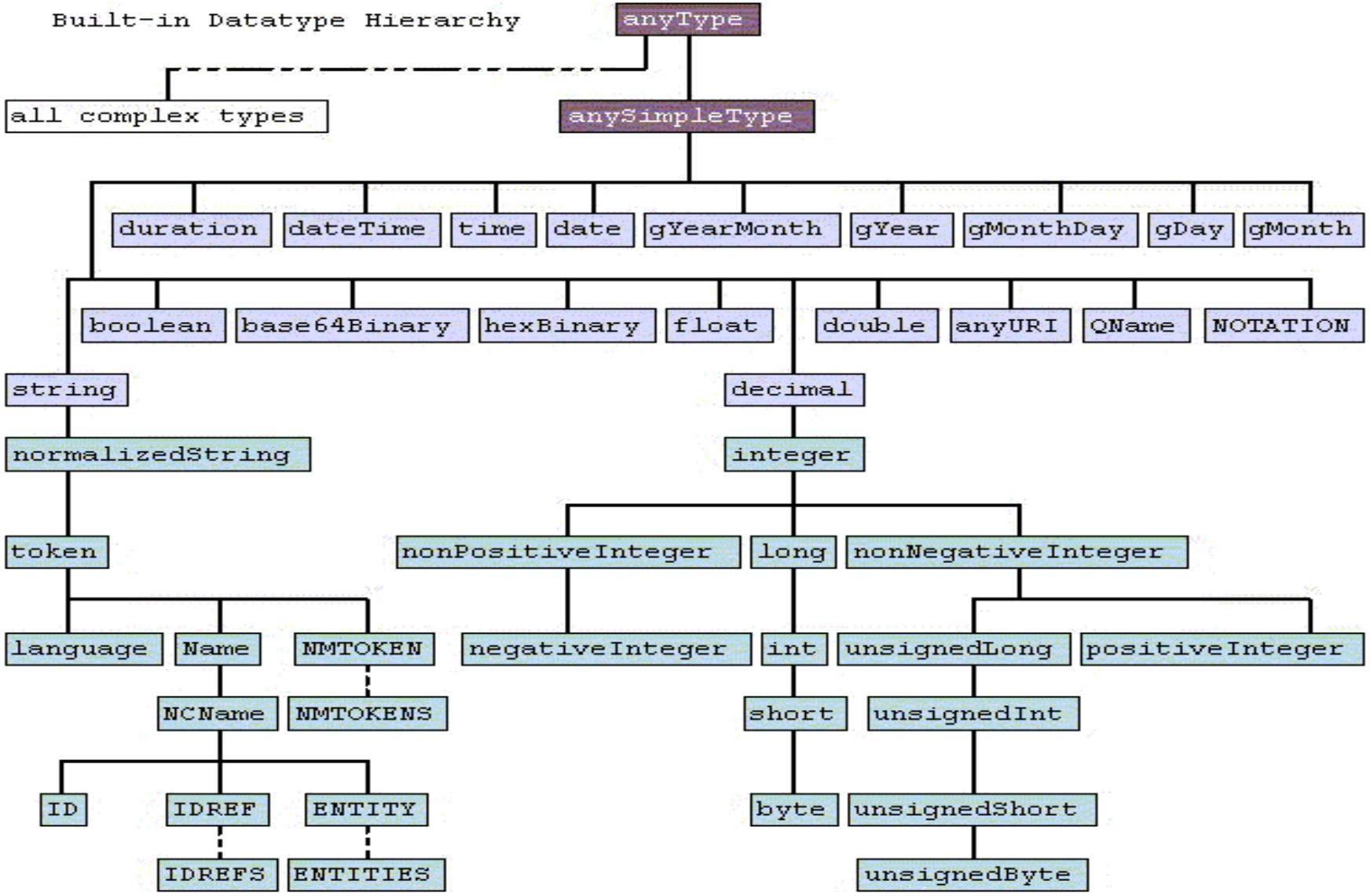
Types in the echo service

```
<?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 type definitions are needed in our WSDL.
- Strings are an XML schema built-in type.

Schema Built In Types



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.
- Another example: C structs or (some) Java classes

WSDL Type Tag

- 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.
- And note that the <any/> tag acts like wildcard.
  - You can insert any sort of xml here.
  - In practice you insert XML Schema type definitions.

WSDL Messages

WSDL Messages

- The “message” section specifies communications that will go on between endpoints.
  - Gives each message a name (to be used later for reference).
  - Specifies the type of message
    - Can be primitive types, like strings
    - Can be defined types

The echoServiceInterface messages

```
<?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>
```

Echo Service Messages

- 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 which message is the request and which is the response.
  - That is the job of the portType, 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 (e.g, parameter) 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.

PortTypes and Operations

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
    portTypes -> java interfaces.

The echoServiceInterface portType

```
<?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:definition>
```

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>
```

portType Message Patterns

- PortTypes support four types of messaging:
  - One way: Client send 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
- From the server point of view
- If the name of a message is not specified a default is used

portType for EchoService

- 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

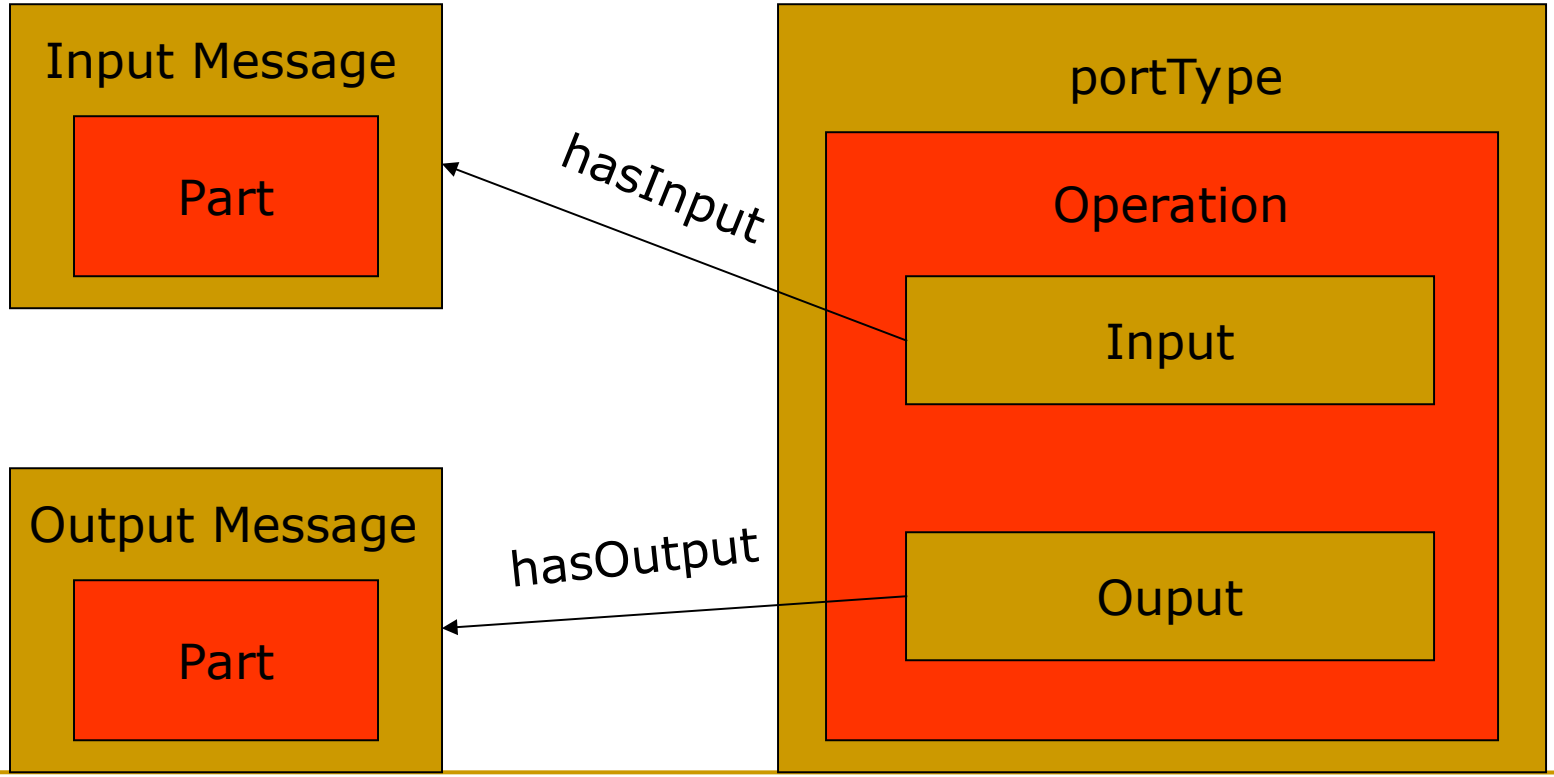
- This (optional) attribute parameterOrder of operation is used to specify zero or more space-separated values.
- The values give the order that the input parameters must be sent.
- Echo is a bad example, since it only has one input parameter, named ***in0***.

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...



Bindings

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.

Binding Section of WSDL

```
<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.

The Binding for Echo

```
<wsdl:binding name="EchoSoapBinding" type="impl:Echo">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="echo">
     <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.

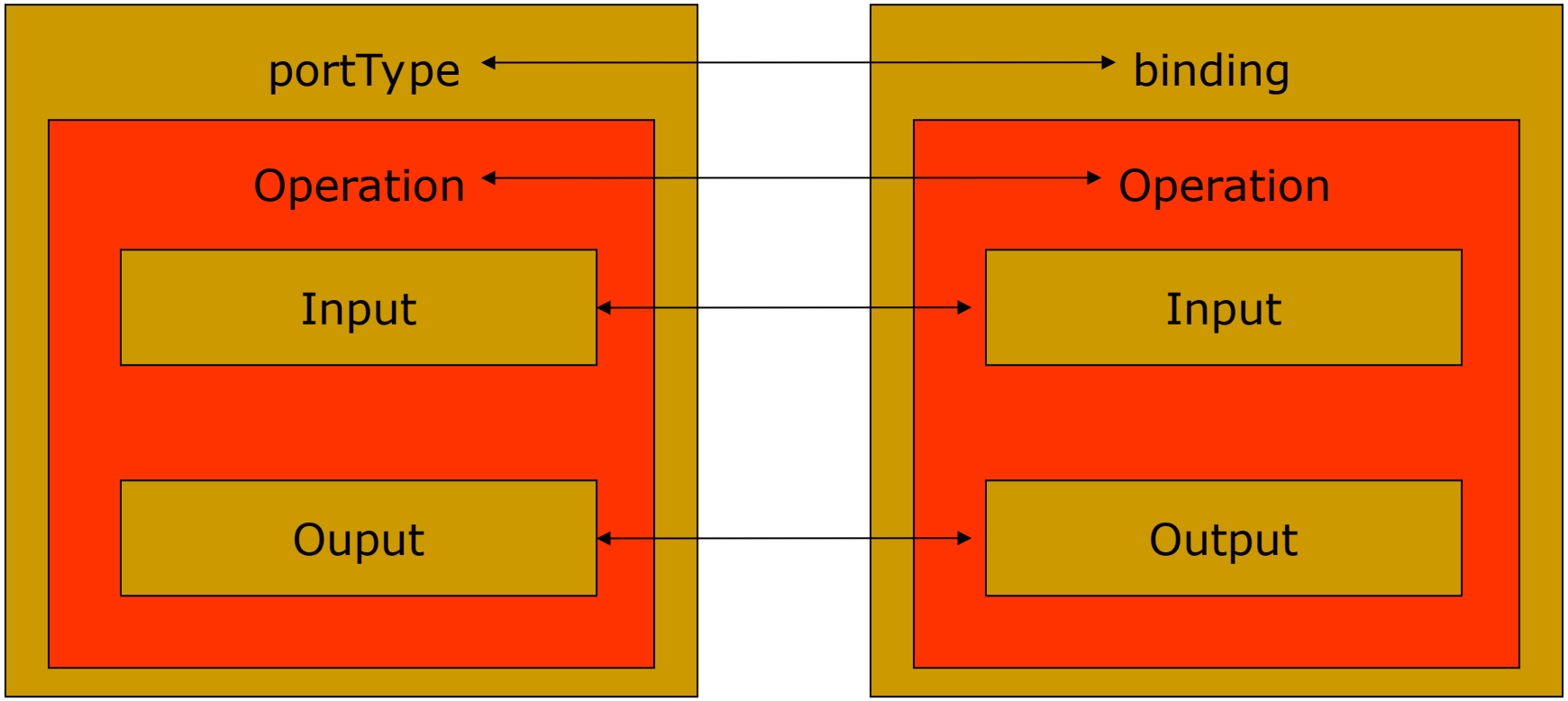
WSDL SOAP Bindings

- In the previous slide, we specify several things:
  - We will use SOAP/HTTP
  - We will use RPC encoding style
  - We specify the namespace associated with the Echo service input and output messages.
- All of this corresponds to SOAP message parts.

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.
  - They are mirror images of the portTypes.
  - Each part is extended by schema elements for a particular binding protocol (i.e. SOAP).
- In our WSDL bindings, we will have two messages (input and output).
  - Each corresponds to SOAP body sections.
  - Additionally, we specify that the body should be encoded.
    - That is, RPC encoded.
      - Alternatively, could be “document” (no RPC structure) instead of RPC and “literal” (no type information) instead of “encoded”.

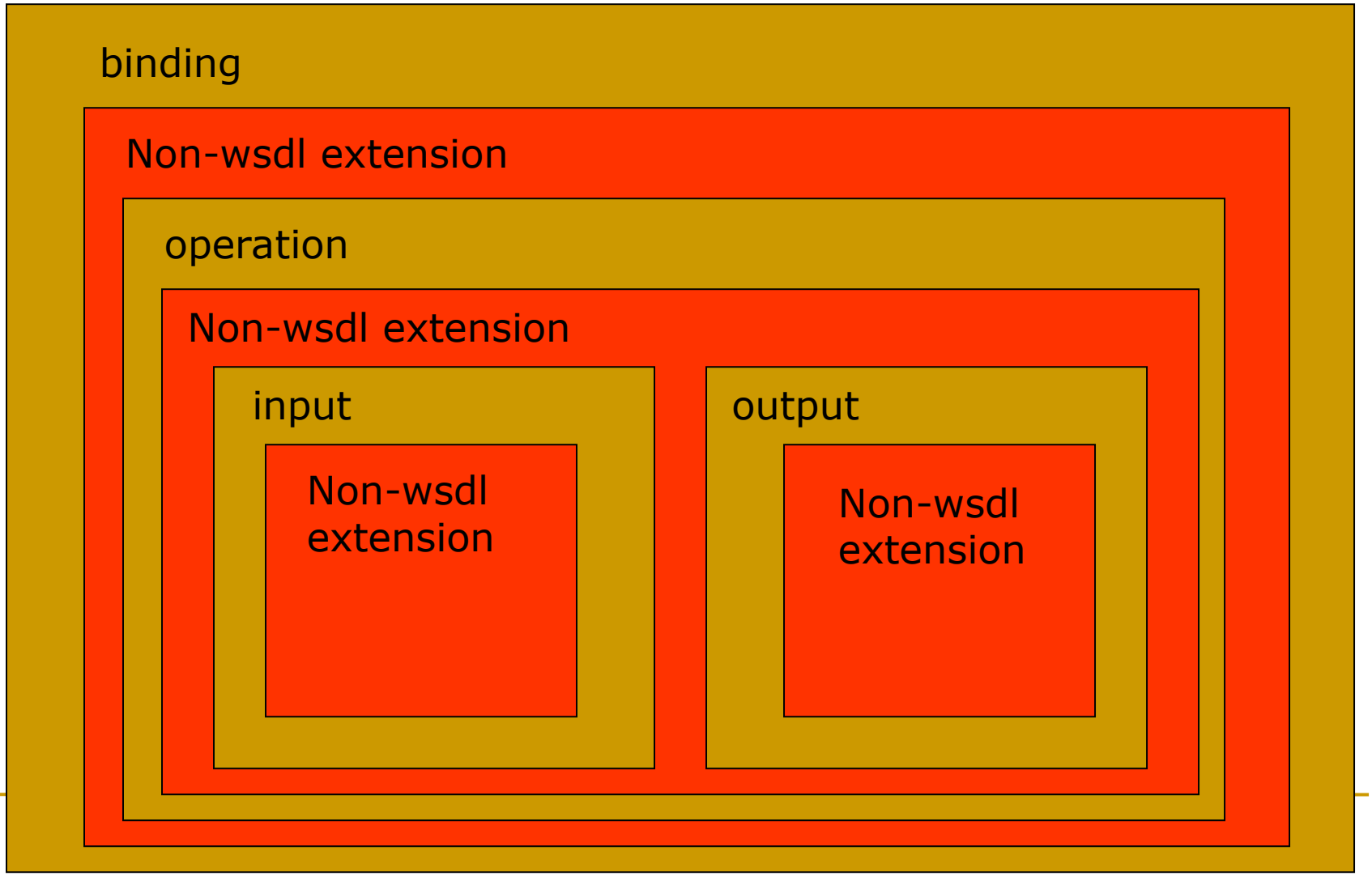
WSDL Internal References



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
  - Brown 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



A little more on encoding…

- 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 vs document

- RPC is used for procedure call and the like
  - The topmost element of the SOAP body is the name of the invoked operation
  - The internal elements are the parameters
  - RPC normally used with “encoded” payload.
    - Types are specified directly, and are XSD datatypes
- Document leaves more freedom
  - The SOAP body can contain any XML
  - Normally used with “literal” payload
    - Payload described by an arbitrary Schema

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 a 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?

```
<wsdl:definitions>
   …
  <wsdl:binding>
   …
 </wsdl:binding>
<wsdl:service name="EchoService">
   <wsdl:port binding="impl:EchoSoapBinding"
   name="Echo">
        <wsdlsoap:address
   location="http://grids.ucs.indiana.edu:8045/GCWS/ser
   vices/Echo" />
   </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

Ports and Services

```
<wsdl:service name="EchoService">
   <wsdl:port
   binding="impl:EchoSoapBinding"
   name="Echo">
      <wsdlsoap:address
        location=“http://..../"/>
  </wsdl:port>
</wsdl:service>
```

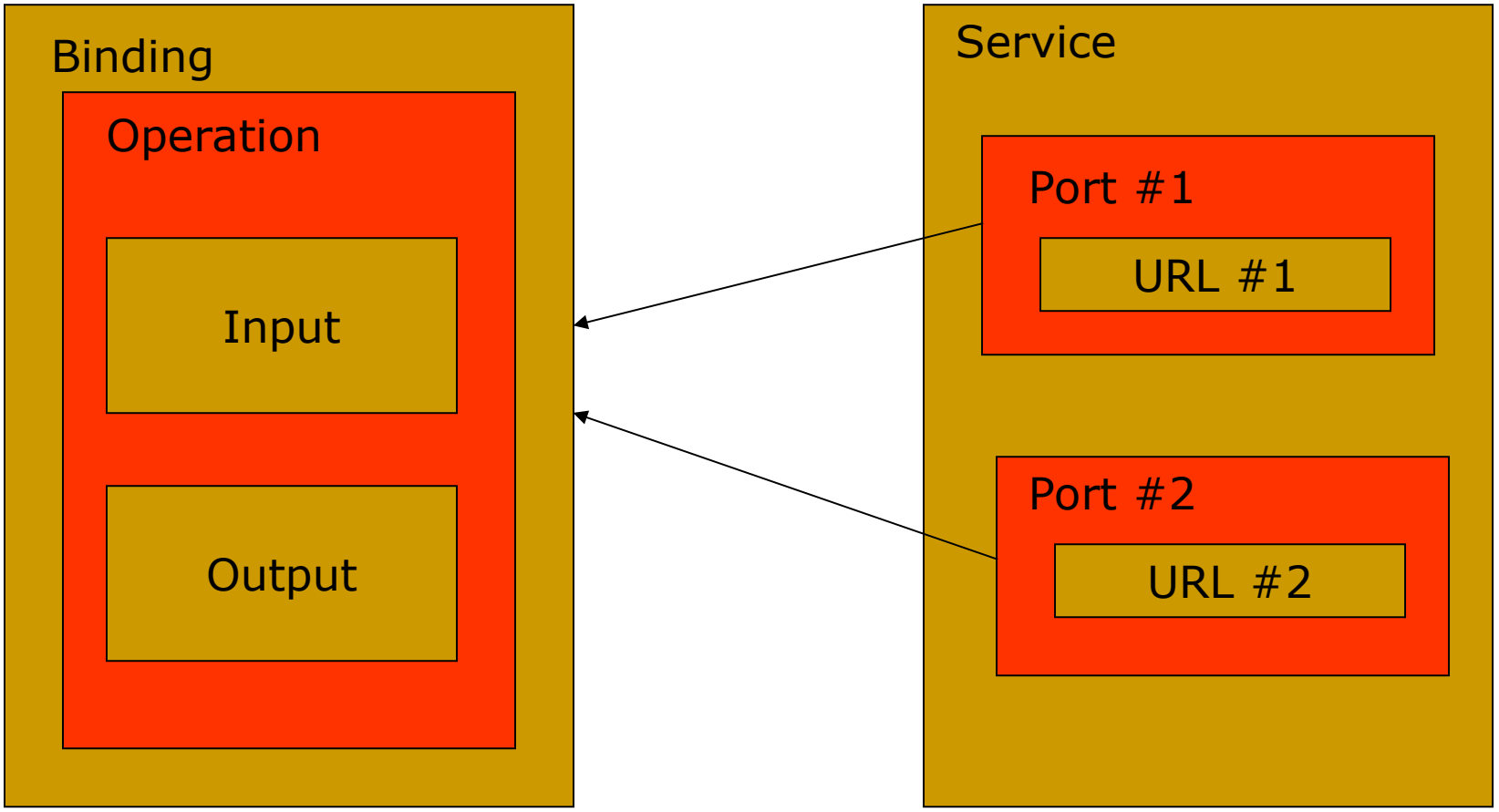
Port and Service Tags

- 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 form of 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



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.