Woden WSDL 2.0 Procesor

(Session WE12, 28 June 2006)

John Kaputin   Jeremy Hughes

ApacheCon
Europe 06

Agenda

- Background to WSDL 2.0 and Apache Woden
- WSDL 2.0 Overview
- WSDL 1.1 to WSDL 2.0 Comparison
- Woden API Overview
- WSDL 2.0 validation in Woden
- WSDL 2.0 extensibility in Woden
- WSDL4J to Woden Comparison
- WSDL 1.1 to 2.0 Converter
- Woden Roadmap

ApacheCon
Europe 06

What is WSDL 2.0?

- Web Services Description Language 2.0
- Emerging W3C standard to supersede WSDL 1.1
- Specification describes:
  - WSDL Component model with XML mappings
  - Syntactic and semantic validation rules
  - Extensibility

ApacheCon
Europe 06

What is Apache Woden?

- An Apache incubator project
  - under the Apache Web Services project
  - http://incubator.apache.org/woden/
  - http://wiki.apache.org/ws/FrontPage/Woden
- A WSDL 2.0 processor
  - Implementation of the W3C WSDL 2.0 spec
  - Originated from WSDL4J, the WSDL 1.1 processor
- Project Goals
  1. Help validate the WSDL 2.0 spec
     - in response to W3C Call for Implementations
  2. Read/create/modify WSDL 2.0 documents
  3. Convert WSDL 1.1 to WSDL 2.0

ApacheCon
Europe 06

WSDL 2.0 history (& future)

- W3C Web Services Description Working Group formed - January, 2002

- First Last Call - August, 2003

- Second Last Call - August, 2005

- Candidate Recommendation - March, 2006
  - WSDL 2.0 Interoperability Event July 2006

- Proposed Recommendation - November, 2006 (planned)

- W3C Recommendation - January, 2007 (planned)

ApacheCon Europe 06

Apache Woden history

- April 2005 Woden began incubation
  - initial code contribution was WSDL4J
- June 2006 Milestone 5 release
- July 2006 W3C WSDL 2.0 interop event
  - Validate the spec and implementations
- 1.0 release possibly 3Q06
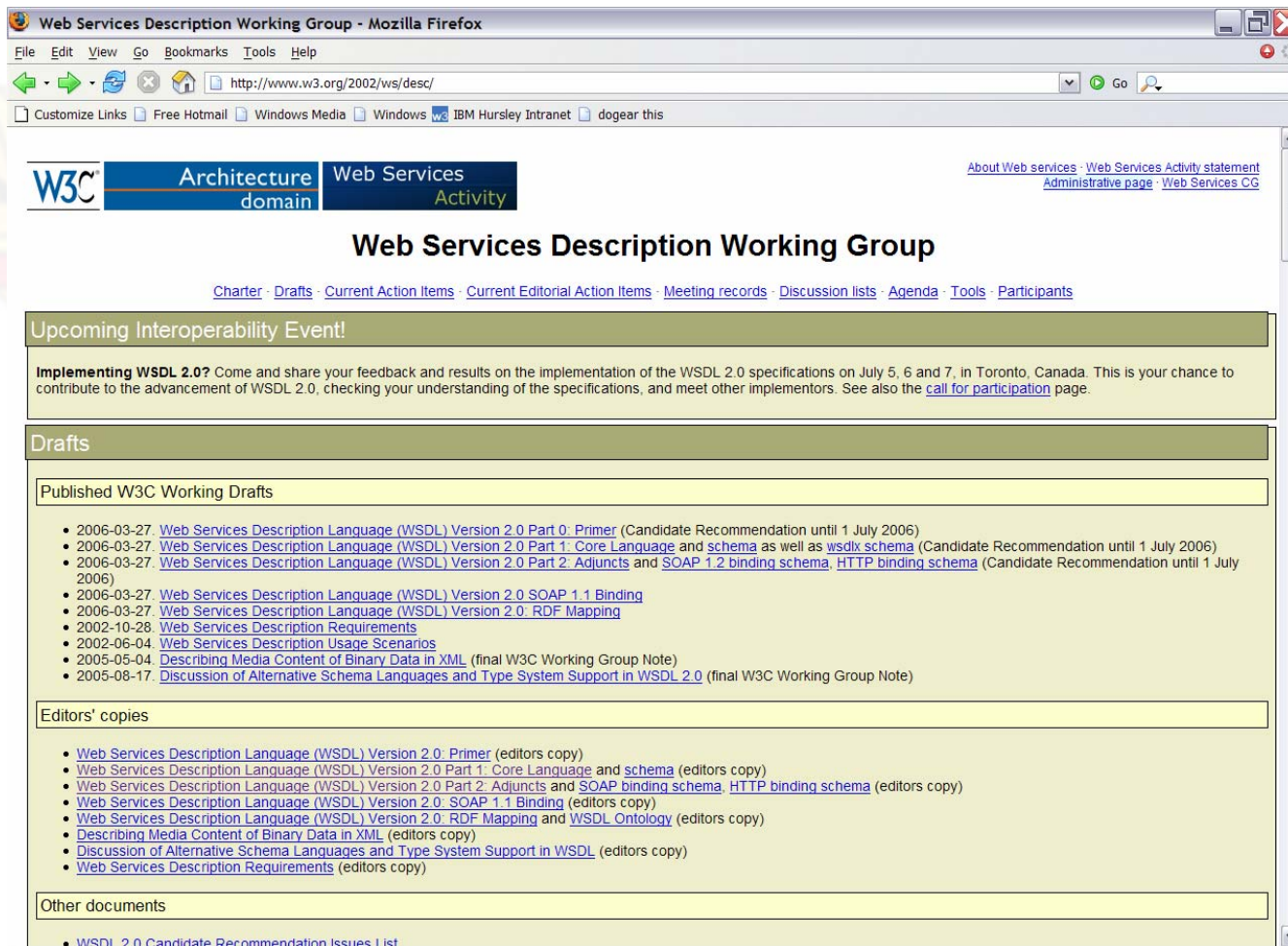
ApacheCon
Europe 06

Agenda

- Background to WSDL 2.0 and Apache Woden
- **WSDL 2.0 Overview**
- WSDL 1.1 to WSDL 2.0 Comparison
- Woden API Overview
- WSDL 2.0 validation in Woden
- WSDL 2.0 extensibility in Woden
- WSDL4J to Woden Comparison
- WSDL 1.1 to 2.0 Converter
- Woden Roadmap

ApacheCon
Europe 06

WSDL 2.0 Specification

- Consists of several documents:
  - Primer (tutorial, non-normative)
  - Part 1: Core Language (Component model)
  - Part 2: Adjuncts (extensions)
  - Others (SOAP 1.1 Binding, RDF mapping, ...)
- http://www.w3.org/2002/ws/desc/

ApacheCon Europe 06

Editor's copies are most up-to-date

ApacheCon
Europe 06

Part 1: Core Language

- The WSDL 2.0 "Component Model"
- An abstract view of the WSDL:
  - Focus on the web service information
    - not the XML infoset details
  - Describes WSDL Components and their properties
    - Not WSDL elements and attributes
    - But it does define the XML representation
  - A 'flattened' view of the WSDL
    - Importing or including of documents is not exposed
    - Imported or included content is aggregated at 'top level'

ApacheCon Europe 06

Component Model – Top Level Components
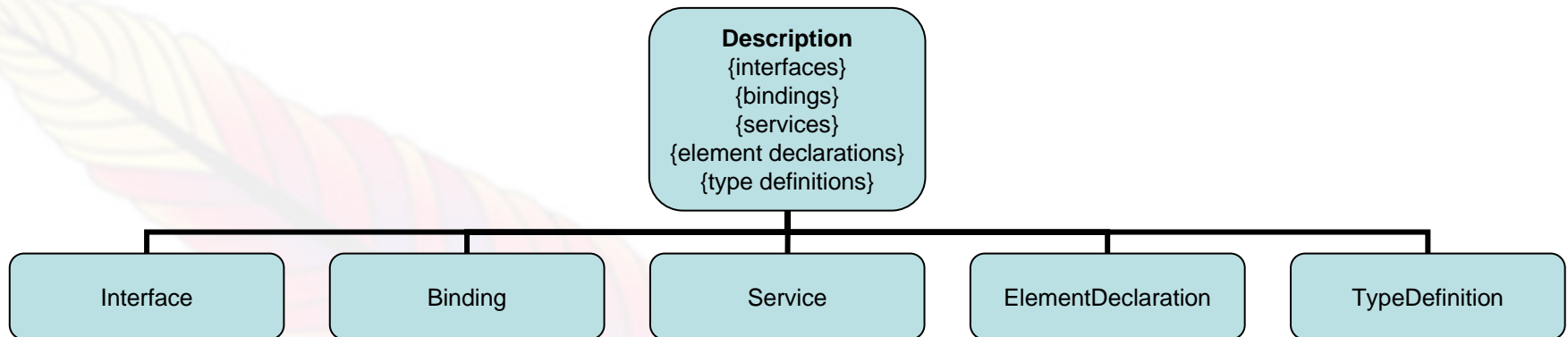
- •Description is a container for 2 types of top-level components

    - ▪WSDL components (Interface, Binding, Service)

    - ▪Type system components (ElementDeclaration, TypeDefinition)

- •Top-level components:

    - •Do not have a parent

    - •May contain "nested" components

ApacheCon
Europe 06

XML representation – Description
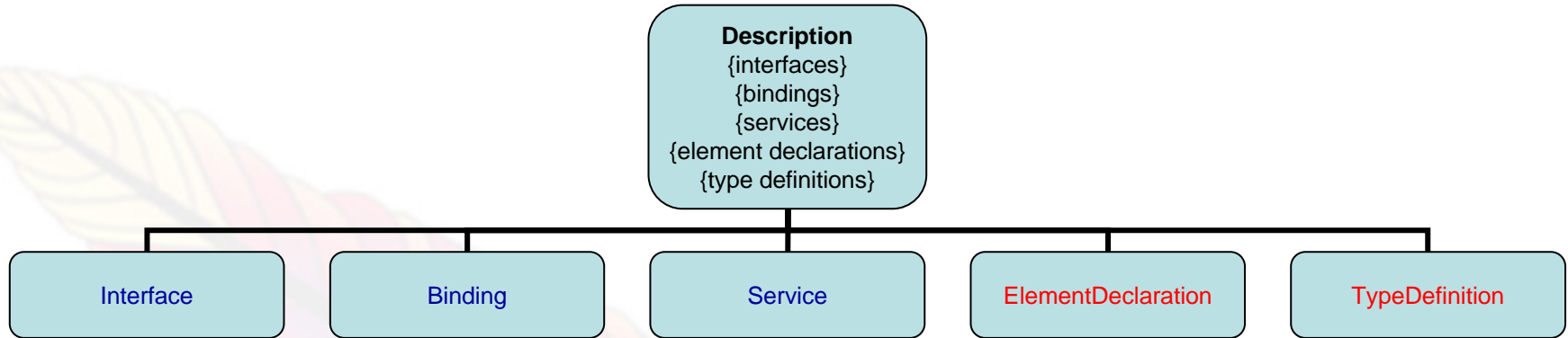
```
            ┌─────────────────────────┐
            │      Description          │
            │      {interfaces}         │
            │      {bindings}           │
            │      {services}           │
            │  {element declarations}   │
            │   {type definitions}      │
            └─────────────────────────┘
```

Interface	Binding	Service	ElementDeclaration	TypeDefinition

```
<?xml version="1.0" encoding="utf-8" ?>
<description
  targetNamespace=…
  xmlns=http://www.w3.org/2006/01/wsdl … >

 <types>
  <xs:schema …
    <xs:element name=…
    <xs:complexType name=…
  </xs:schema>
 </types>

 <interface> … </interface>

 <binding> … </binding>

 <service> … </service>

</description>
```

12

ApacheCon
Europe 06

Component Model – Type system components

```
                    ┌─────────────────────┐
                    │    Description      │
                    └─────────────────────┘
                              │
                ┌─────────────┴─────────────┐
    ┌───────────────────┐        ┌───────────────────┐
    │ ElementDeclaration │        │  TypeDefinition    │
    │      {name}        │        │      {name}        │
    │     {system}       │        │     {system}       │
    └───────────────────┘        └───────────────────┘
```
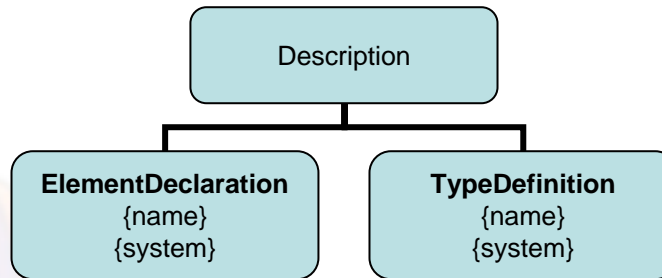
- Abstract wrapper for some underlying type system
  - Typically W3C XML Schema
  - Could be some other XML based type system, like RelaxNG
  - Could be a non-XML type system, like DTD
- ElementDeclaration – a global schema element declaration
- TypeDefinition – a global schema type definition

13

ApacheCon
Europe 06

XML representation – Type system components

```xml
<?xml version="1.0" encoding="utf-8" ?>
<description
   xmlns="http://www.w3.org/2006/01/wsdl"
   targetNamespace= . . . >

 <types>
  <xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://greath.example.com/2004/schemas/resSvc"
   xmlns="http://greath.example.com/2004/schemas/resSvc">

    <xs:element name="checkAvailability" type="tCheckAvailability"/>
    <xs:complexType name="tCheckAvailability">
     <xs:sequence>
       <xs:element  name="checkInDate" type="xs:date"/>
       <xs:element  name="checkOutDate" type="xs:date"/>
       <xs:element  name="roomType" type="xs:string"/>
     </xs:sequence>
    </xs:complexType>
    <xs:element name="checkAvailabilityResponse" type="xs:double"/>
    <xs:element name="invalidDataError" type="xs:string"/>

  </xs:schema>
 </types>
 . . .
</description>
```

```
                    ┌──────────────────┐
                    │    Description    │
                    └──────────────────┘
             ┌─────────────┴─────────────┐
┌─────────────────────┐      ┌─────────────────────┐
│  ElementDeclaration  │      │   TypeDefinition    │
│       {name}         │      │       {name}        │
│      {system}        │      │      {system}       │
└─────────────────────┘      └─────────────────────┘
```

{system} = http://www.w3.org/2001/XMLSchema
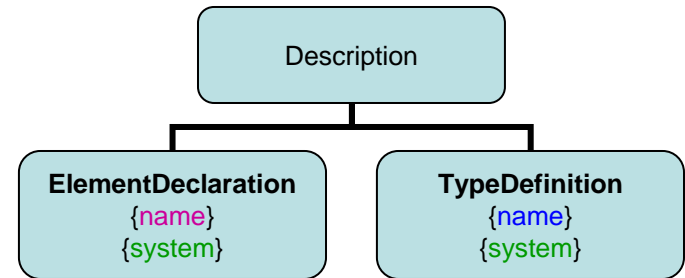
ElementDeclaration {name} =
{http://greath.example.com/2004/schemas/resSvc}checkAvailability

TypeDefinition {name} =
{http://greath.example.com/2004/schemas/resSvc} tCheckAvailability

ApacheCon
Europe 06

Component Model – nested Components
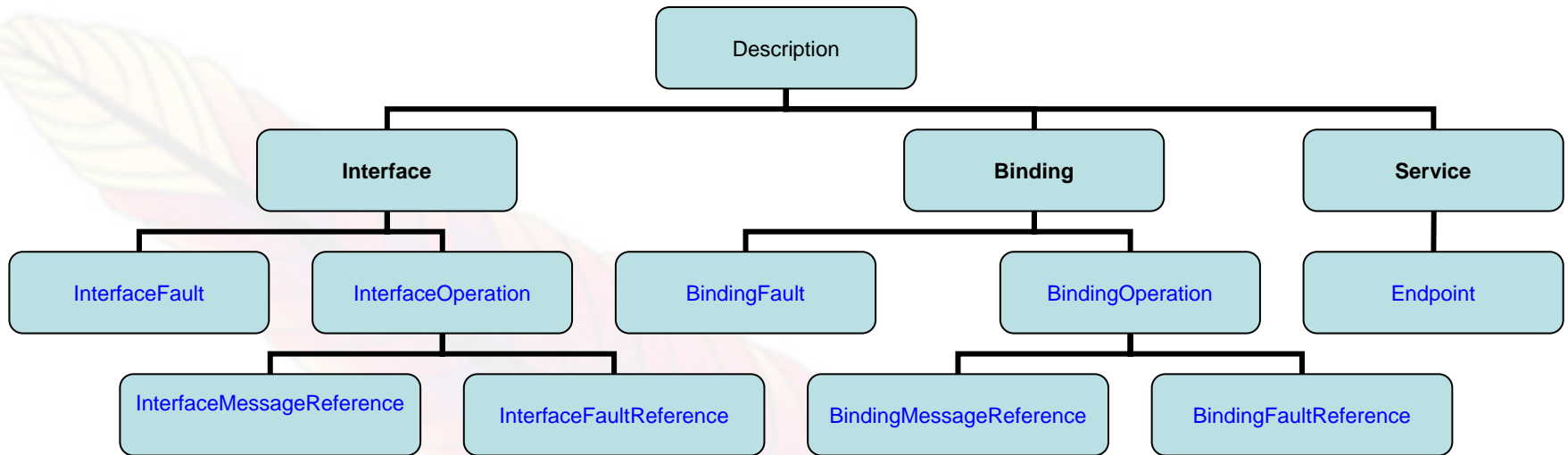
- **Nested components**:
  - Below top-level components (Interface, Binding, Service)
  - Have a parent
  - May contain other nested components
- Note symmetry between Interface and Binding nested components

ApacheCon
Europe 06

Interface components – WHAT the service does

```xml
<?xml version="1.0" encoding="utf-8" ?>
<description targetNamespace=…
  xmlns="http://www.w3.org/2006/01/wsdl"  . . . >

<types>
 <xs:schema …>
   <xs:element name="invalidDataError" …/>
   <xs:element name="checkAvailability" …/>
   …
   </xs:schema>
</types>

<interface  name = "reservationInterface"
    extends=…  styleDefault=…>

 <fault name = "invalidDataFault"
       element = "ghns:invalidDataError"/>

 <operation name="opCheckAvailability"
      pattern="http://www.w3.org/2006/01/wsdl/in-out"
      style="http://www.w3.org/2006/01/wsdl/style/iri">
   <input messageLabel="In"
       element="ghns:checkAvailability" />
   <output messageLabel="Out"
       element="ghns:checkAvailabilityResponse" />
   <infault ref="tns:invalidDataFault" messageLabel="In"/>
   <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
 </operation>

</interface>
. . .
</description>
```


Description

Interface
{name}
{extended interfaces}
{interface faults}
{interface operations}

InterfaceFault
{name}
{element declaration}
{parent}

InterfaceOperation
{name}
{message exchange pattern}
{interface message references}
{interface fault references}
{style}
{parent}

InterfaceMessageReference
{message label}
{direction}
{message content model}
{element declaration}
{parent}

InterfaceFaultReference
{interface fault}
{message label}
{direction}
{parent}

ApacheCon
Europe 06

Binding components – HOW to connect

```xml
<?xml version="1.0" encoding="utf-8" ?>
<description targetNamespace= …
    xmlns="http://www.w3.org/2006/01/wsdl" …>

<types> … </types>
<interface  name = "reservationInterface" > … </interface>

<binding name="reservationSOAPBinding"
        interface="tns:reservationInterface"
        type="http://www.w3.org/2006/01/wsdl/soap"
        …">

    <fault ref="tns:invalidDataFault"
            wsoap:code="soap:Sender"/>


    <operation ref="tns:opCheckAvailability"  …>
      <input>…</input>
      <output>….</output>
      <infault> …. </infault>
      <outfault> …. </outfault>
    </operation>

</binding>
. . .
</description>
```

Description

Binding
{name}
{interface}
{type}
{binding faults}
{binding operations}

BindingFault
{interface fault}
{parent}

BindingOperation
{interface operation}
{binding message references}
{binding fault references}
{parent}

BindingMessageReference
{interface message reference}
{parent}

BindingFaultReference
{interface fault reference}
{parent}

17

ApacheCon
Europe 06

Service components – WHERE to connect

```xml
<?xml version="1.0" encoding="utf-8" ?>
<description targetNamespace=…
  xmlns="http://www.w3.org/2006/01/wsdl" …>

<types> … </types>

<interface  name="reservationInterface" >
 …
</interface>

<binding name="reservationSOAPBinding"
      interface="tns:reservationInterface"
      … >
 …
</binding>

<service name="reservationService"
      interface="tns:reservationInterface">
  <endpoint name="reservationEndpoint"
        binding="tns:reservationSOAPBinding"
        address ="http://greath.example.com/2004/reservation"/>
</service>
</description>
```

Description

Service
{name}
{interface}
{endpoints}

Endpoint
{name}
{binding}
{address}
{parent}

ApacheCon
Europe 06
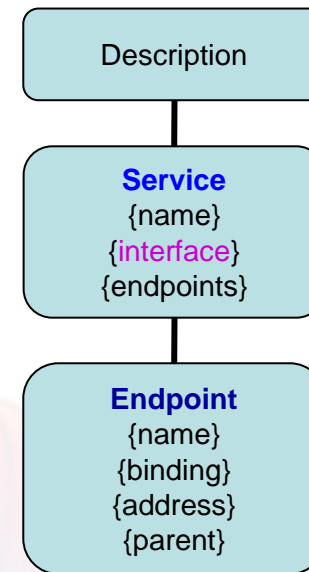
Feature & Property - configuration

```
                            ┌─────────────────────┐
                            │    Description      │
                            └─────────────────────┘
          ┌─────────────────────────┼──────────────────────────┐
┌──────────────────┐    ┌──────────────────┐    ┌──────────────────┐
│    Interface     │    │    Binding       │    │    Service       │
│   {features}     │    │   {features}     │    │   {features}     │
│  {properties}    │    │  {properties}    │    │  {properties}    │
└──────────────────┘    └──────────────────┘    └──────────────────┘
```

| Interface {features} {properties} | Binding {features} {properties} | Service {features} {properties} |

| InterfaceFault {features} {properties} | InterfaceOperation {features} {properties} | BindingFault {features} {properties} | BindingOperation {features} {properties} | Endpoint {features} {properties} |

| InterfaceMessageReference {features} {properties} | InterfaceFaultReference {features} {properties} | BindingMessageReference {features} {properties} | BindingFaultReference {features} {properties} |

| Feature {ref} {required} {parent} | Property {ref} {value constraint} {value} {parent} |

- Every component except Description may have Feature and Property components
- Provide additional processing information for a component
  - Eg: reliability, correlation, routing, retries
- Different to Woden features and properties
  - Those are scoped to WSDLReader or WSDLWriter, not to WSDL components

ApacheCon
Europe 06

Feature & Property XML Representation

```
<interface name="reservationInterface">
  <operation name="makeReservation">
    ….
    <feature uri="http://example.com/securityFeature" required="true"/>
  </operation>
</interface>


<interface name="reservationInterface">
  <operation name="makeReservation">
    <property uri="http://example.com/securityFeature/securityLevel">
      <value>5</value>
    </property>
    . . .
  </operation>
</interface>




<types>
  <schema>
    <simpleType name="securityLevelConstraint">
      …
    </simpleType>
  </schema>
</types>
. . .
<binding>
    ….
    <property uri="http://example.com/securityFeature/securityLevel">
      <constraint type="tns:securityLevelConstraint"/>
    </property>
</binding>
```
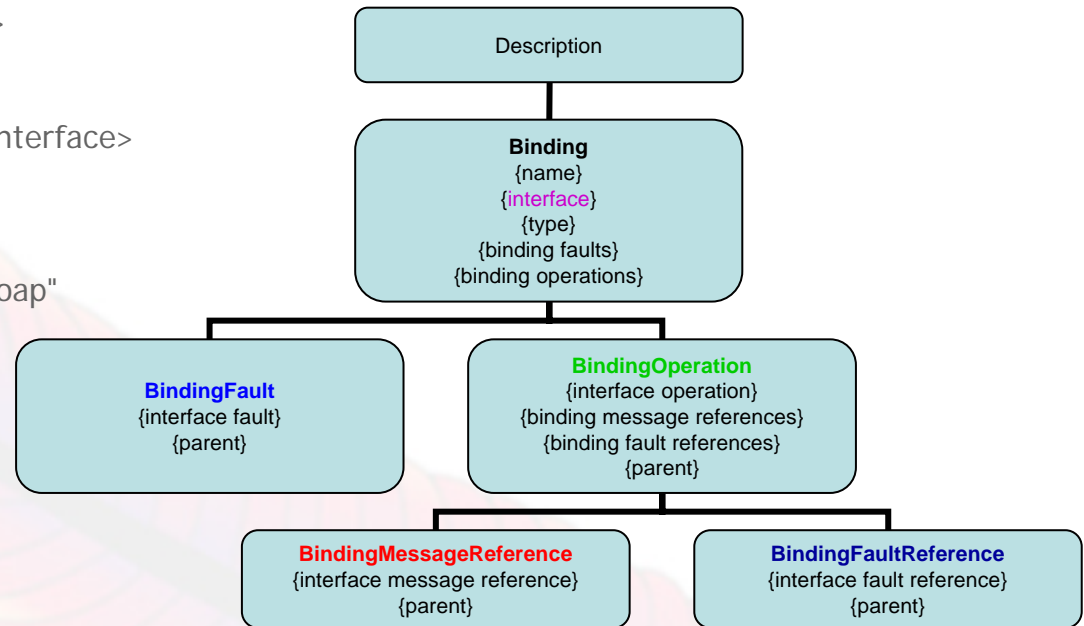
WSDL component

Feature
{ref}
{required}
{parent}

Property
{ref}
{value constraint}
{value}
{parent}

ApacheCon
Europe 06

Message Exchange Patterns

- MEP defines a contract between service and client
- MEP defined as an exchange of messages indicating:
  - sequence and cardinality
  - optionality
  - sender and receiver
  - fault propagation ruleset
    - Fault Replaces Message
    - Message Triggers Fault
    - No Faults
- WSDL 2.0 defines 8 MEPs
  - In-Only, Robust In-Only, In-Out, In-Optional-Out, Out-Only, Robust Out-Only, Out-In, Out-Optional-In

ApacheCon
Europe 06

Example MEP: In-Out

In-Out

This pattern consists of exactly two messages, in order, as follows:

1.   A message:

indicated by a <u>Interface Message Reference</u> component whose {<u>message label</u>} is "In" and {<u>direction</u>} is "in" received from some node N

2.   A message:

indicated by a <u>Interface Message Reference</u> component whose {<u>message label</u>} is "Out" and {<u>direction</u>} is "out" sent to node N

This pattern uses the rule <u>2.2.1 Fault Replaces Message</u>.

An operation using this MEP has a {<u>message exchange pattern</u> } property with the value "**http://www.w3.org/2006/01/wsdl/in-out**".

```
<operation name="bookTickets"
           pattern="http://www.w3.org/2006/01/wsdl/in-out">
    <input messageLabel="In" element="rsv:booking"/>
    <output messageLabel="Out" element="rsv:confirmation"/>
    <outfault ref="tns:invalidData" messageLabel="Out"/>

</operation>
```

ApacheCon
Europe 06

Agenda

- Background to WSDL 2.0 and Apache Woden
- WSDL 2.0 Overview
- **WSDL 1.1 to WSDL 2.0 Comparison**
- Woden API Overview
- WSDL 2.0 validation in Woden
- WSDL 2.0 extensibility in Woden
- WSDL4J to Woden Comparison
- WSDL 1.1 to 2.0 Converter
- Woden Roadmap

ApacheCon
Europe 06

WSDL 1.1 / 2.0 differences

WSDL 1.1	WSDL 2.0
<definitions>	<description>
<portType>	<interface>
<binding>	<binding>
<types>	<types>
<service>	<service>
<port>	<endpoint>
<message>	Defined within <operation>

ApacheCon
Europe 06

GrainOfSandCount diffs

WSDL 1.1

```
<wsdl:definitions targetNamespace="http://grainsofsand.beach"
            xmlns="http://schemas.xmlsoap.org/wsdl/" ...
    <wsdl:types>
...
    </wsdl:types>

    <wsdl:message name="getGrainCountRequest">
        <wsdl:part element="tns:getGrainCount" name="parameters" />
    </wsdl:message>

    <wsdl:message name="getGrainCountResponse">
        <wsdl:part element="tns:getGrainCountResponse" name="parameters" />
    </wsdl:message>

    <wsdl:portType name="GrainCount">
        <wsdl:operation name="getGrainCount">
            <wsdl:input message="tns:getGrainCountRequest"
                name="getGrainCountRequest" />
            <wsdl:output message="tns:getGrainCountResponse"
                name="getGrainCountResponse" />
        </wsdl:operation>
    </wsdl:portType>
```

WSDL 2.0

```
<description xmlns="http://www.w3.org/2006/01/wsdl"
                targetNamespace="http://grainsofsand.beach" ...
    <types>
...
    </types>
    <interface name="GrainCount">
        <operation name="getGrainCount"
                    pattern="http://www.w3.org/2006/01/wsdl/in-out">
            <input messageLabel="In" element="tns:getGrainCount"/>
            <output messageLabel="Out" element="tns:getGrainCountResponse"/>
        </operation>
    </interface>
```

ApacheCon
Europe 06

`<binding>` diffs

WSDL 1.1

```
<wsdl:binding name="GrainCountSoapBinding" type="tns:GrainCount">
  <wsdlsoap:binding style="document"
                    transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="getGrainCount">
    <wsdlsoap:operation soapAction="" />
    <wsdl:input name="getGrainCountRequest">
      <wsdlsoap:body use="literal" />
    </wsdl:input>
    <wsdl:output name="getGrainCountResponse">
      <wsdlsoap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

WSDL 2.0

```
<binding name="GrainCountSoapBinding"
         interface="tns:GrainCount"
         type="http://www.w3.org/2006/01/wsdl/soap"
         wsoap:version="1.1"
         wsoap:protocol="http://www.w3.org/2006/01/soap11/bindings/HTTP">
  <operation ref="tns:getGrainCount" wsoap:action="">
  </operation>
</binding>
```

ApacheCon
Europe 06

<service> diffs

WSDL 1.1

```
<wsdl:service name="GrainCountService">
  <wsdl:port name="GrainCount"
    binding="tns:GrainCountSoapBinding">
    <wsdlsoap:address
      location="http://localhost:9080/GrainCount" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

WSDL 2.0

```
<service name="GrainCountService"
         interface="tns:GrainCount">
  <endpoint name="GrainCount"
            binding="tns:GrainCountSoapBinding"
            address="http://localhost:9080/GrainCount">
    </endpoint>
  </service>
</description>
```

ApacheCon
Europe 06

WSDL vs WSDL

- Operation overloading

  `WSDL 1.1` used names of `<operation>`, `<input>` and `<output>` elements

  `WSDL 2.0` must be uniquely named within an `<interface>`

- targetNamespace

  `WSDL 1.1` optional. If none no top level elements can be referred to by other WSDLs.

  `WSDL 2.0` mandatory

ApacheCon
Europe 06

WSDL vs WSDL

- Definition of <service>
  - `WSDL 1.1` `<service>`s can have multiple `<port>`s. Each `<port>` can be bound to a different `<portType>`
  - `WSDL 2.0` `<service>`s bound to single `<interface>`

- Ordering of top level elements
  - `WSDL 1.1` children of `<definitions>` may appear in any order
  - `WSDL 2.0` order is strictly defined

ApacheCon
Europe 06

WSDL vs WSDL

- Definition of faults

  `WSDL 1.1` scoped at the `<operation>`

  `WSDL 2.0` scoped at the `<interface>` – enables reuse across operations

- Imports

  `WSDL 1.1` WSDL `<import>` same or different `targetNamespace`

  `WSDL 2.0` Use `<include>` for the same tns

  Use `<import>` for different tns

  Consistent with XML Schema

ApacheCon
Europe 06

Agenda

- Background to WSDL 2.0 and Apache Woden
- WSDL 2.0 Overview
- WSDL 1.1 to WSDL 2.0 Comparison
- **Woden API Overview**
- WSDL 2.0 validation in Woden
- WSDL 2.0 extensibility in Woden
- WSDL4J to Woden Comparison
- WSDL 1.1 to 2.0 Converter
- Woden Roadmap

ApacheCon
Europe 06

Apache Woden packages

Copyright © IBM Corp 2006

ApacheCon
Europe 06
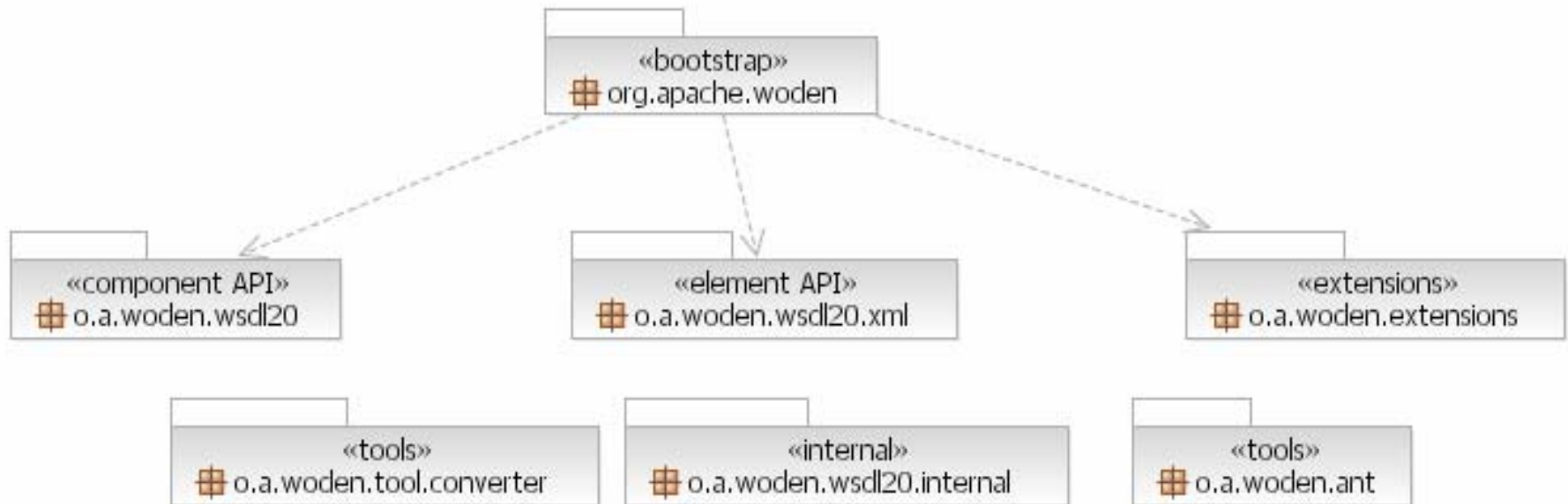
Apache Woden WSDL 2.0 processor APIs

- Component model API
  - Mirror the component model of WSDL 2.0
    - Large service descriptions are spread across multiple .wsdl files
    - One root Description object representing a merge of all .wsdl files imported and included
    - Good for applications working at this abstract level e.g wsdl2java
  - Read only view (today)
- Element model API
  - Mirror the XML representation of WSDL 2.0
    - And hence the physical (file) representation
  - Read / write

ApacheCon
Europe 06

Represent a WSDL 2.0 document(s) in java object form

«Java Class» **WSDLFactory**
● newInstance ( ) : WSDLFactory
● newInstance (factoryImplName : String) : WSDLFactory
● newWSDLReader ( ) : WSDLReader
● newDescription ( ) : DescriptionElement
● newPopulatedExtensionRegistry ( ) : ExtensionRegistry

«use»

«Java Interface» **WSDLReader**
● readWSDL (wsdlURI : String) : DescriptionElement
● readWSDL (wsdlURI : String, errorHandler : ErrorHandler) : DescriptionElement
● getErrorReporter ( ) : ErrorReporter
● setFactoryImplName (factoryImplName : String) : void
● getFactoryImplName ( ) : String
● setExtensionRegistry (extReg : ExtensionRegistry) : void
● getExtensionRegistry ( ) : ExtensionRegistry
● setFeature (name : String, value : boolean) : void
● getFeature (name : String) : boolean
● setProperty (name : String, value : Object) : void
● getProperty (feature : String) : Object

```
WSDLFactory factory = WSDLFactory.newInstance();
WSDLReader reader   = factory.newWSDLReader();

reader.setFeature(WSDLReader.FEATURE_VALIDATION, true);

// Element model
DescriptionElement descElement =
    reader.readWSDL("file:///GrainOfSandCounter-2.wsdl");

// Component model
Description desc = descElement.toComponent();
```

ApacheCon
Europe 06

Model Hopping

- `<component model class>.toElement()`
  - Yields the associated Element-model object
- `DescriptionElement.toComponent()`
  - DescriptionElement only
  - Yields everything in scope of the <description>
    - Top level and imported/included descriptions
  - Could yield a Description that doesn't validate
    - toComponent() on a partially complete description

ApacheCon
EuRope 06

Example code

```
// Retrieve a particular service component from the description
// component's {services} property. Use the service component's
// {name} property as the key
QName name = new QName("http://grainsofsand.beach",
                       "GrainCountService");
Service svc = desc.getService(name);

// service component's {interface} property:
Interface iface = svc.getInterface();

// description component's {interfaces} property:
Interface[] ifaces = desc.getInterfaces();

// Change {name} property of an {endpoint} property of the
// service component.
NCName epName = new NCName("GrainCount");
Endpoint epnt = svc.getEndpoint(epName);
NCName newEpName = new NCName("GrainWarehouse");
epnt.toElement().setName(newEpName);
```
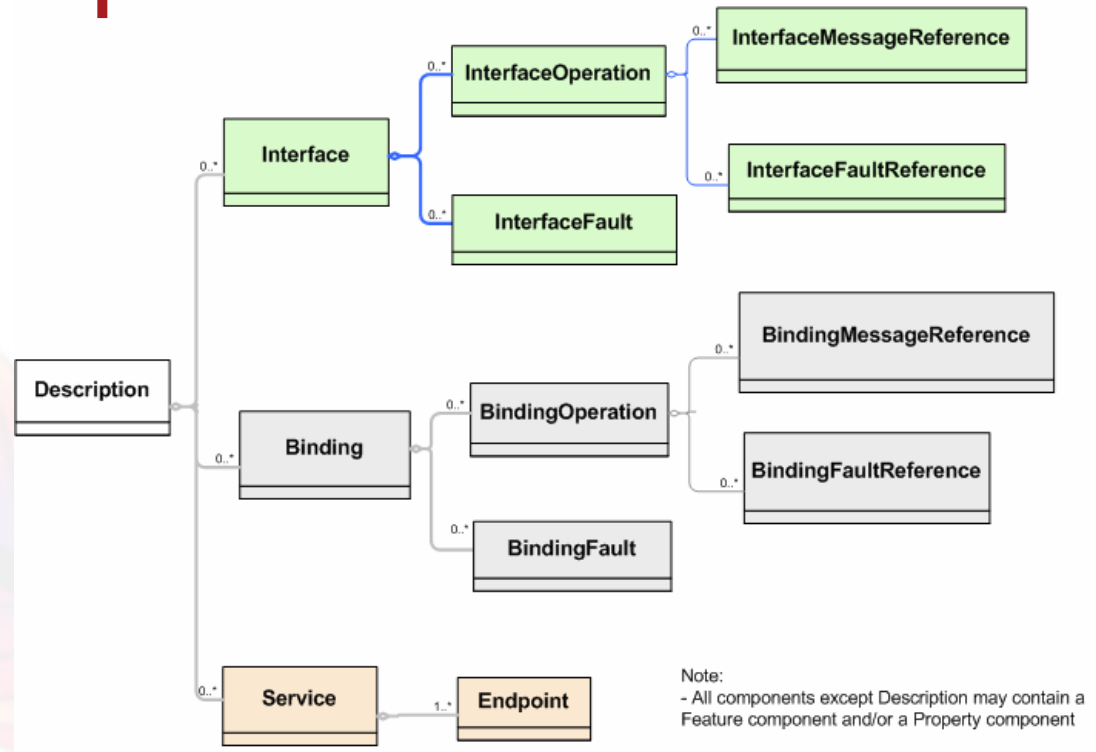
36

ApacheCon
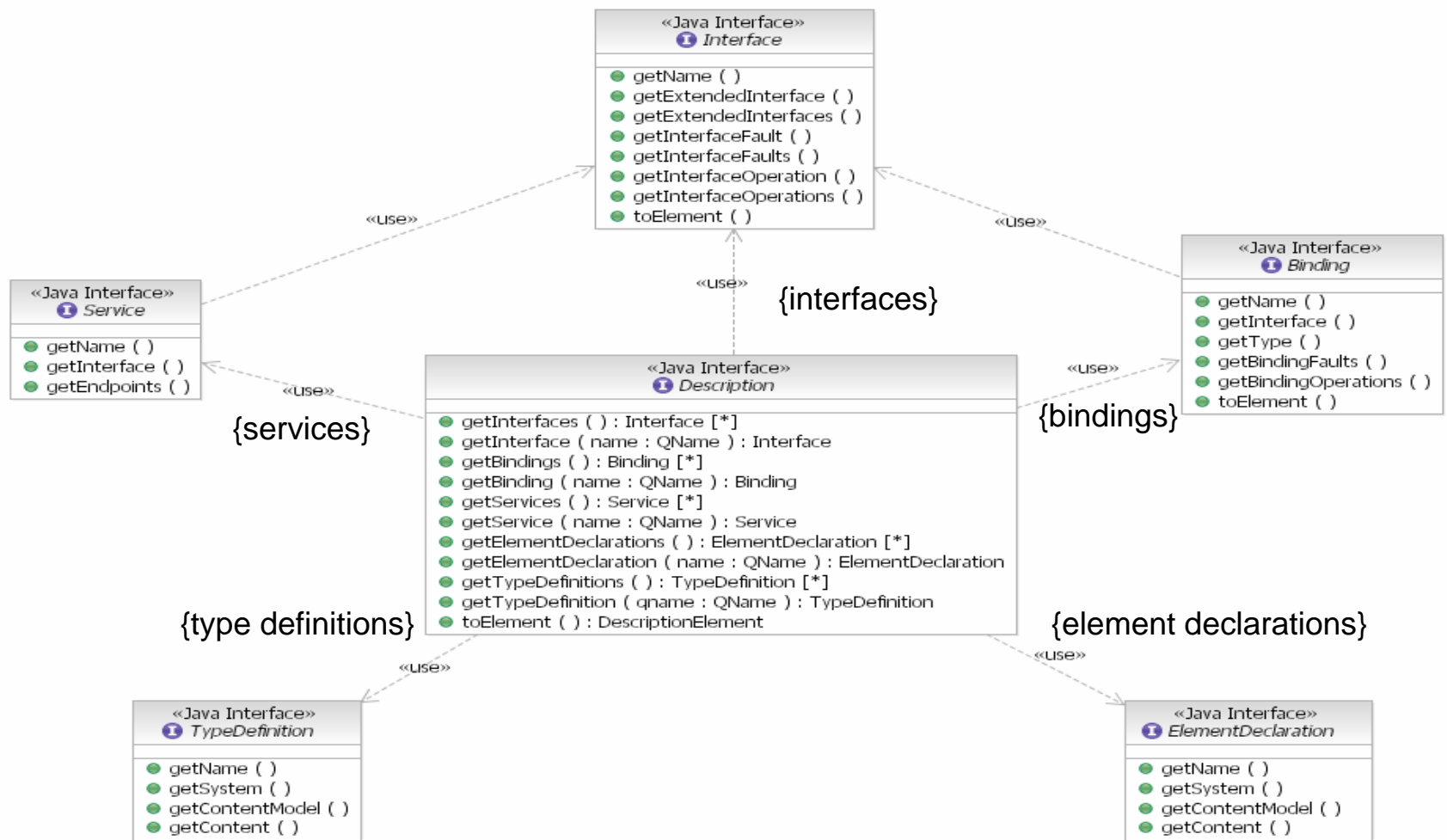Europe 06

WSDL 2.0 recap

- WSDL 2.0 defines properties of the Description component
  - WSDL 2.0
    - Interface
    - Binding
    - Service
  - Types
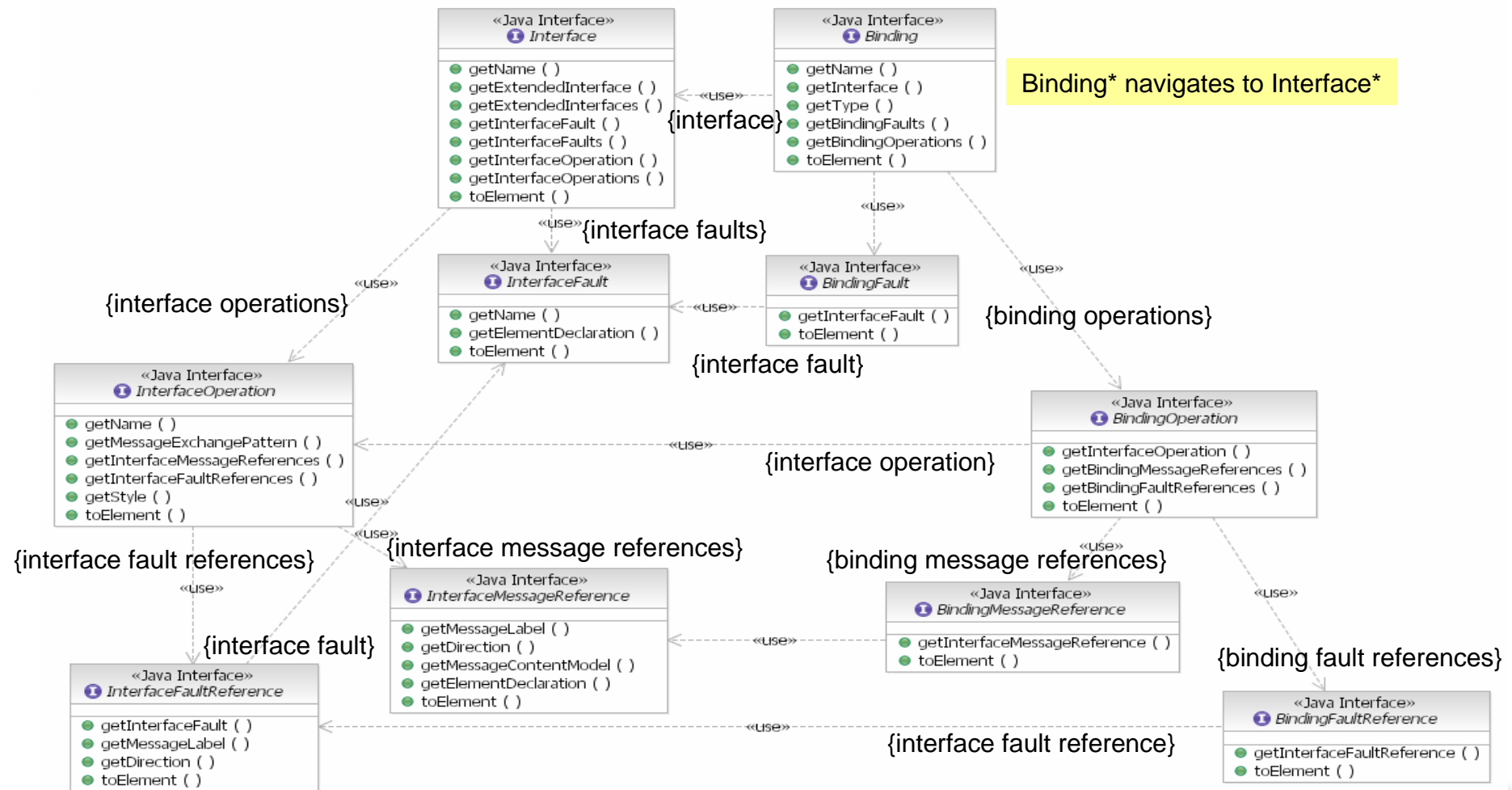   - Element Declaration
    - Type Definition

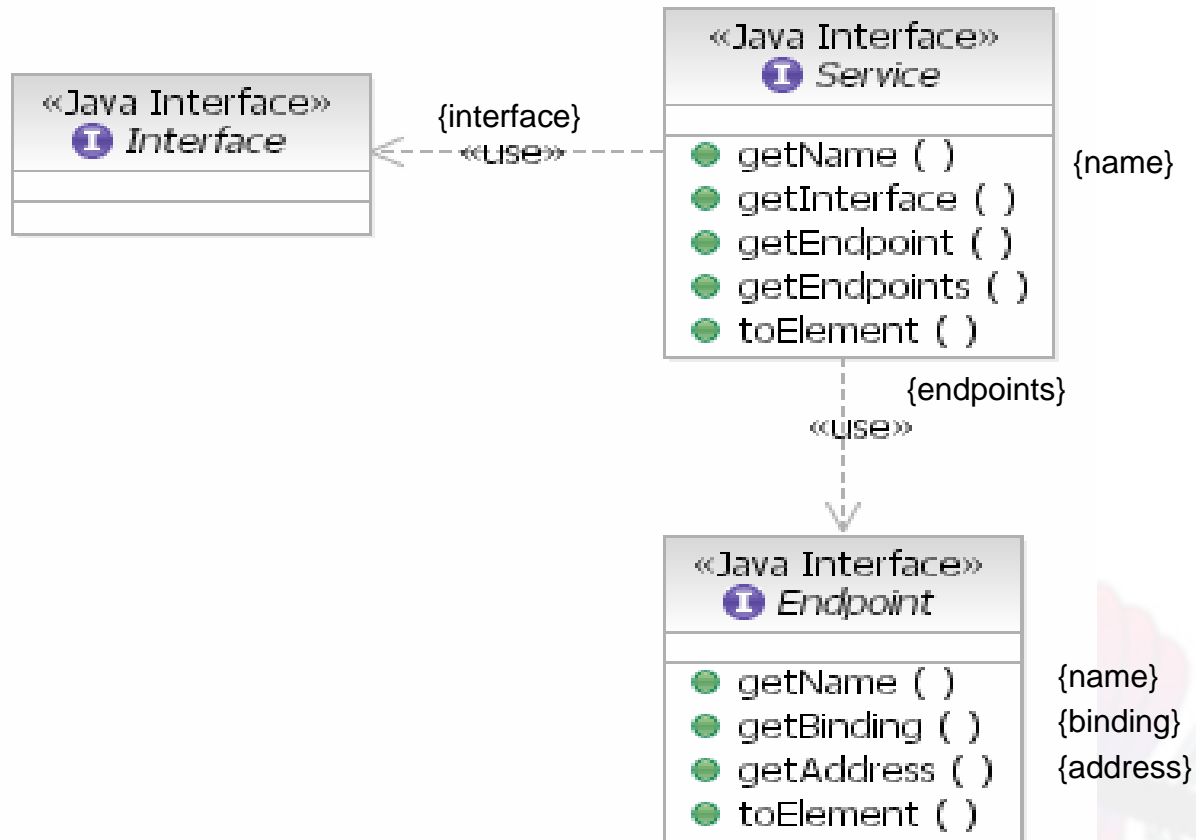

Note:
- All components except Description may contain a Feature component and/or a Property component

[2] W3C Copyright Notice

ApacheCon
Europe 06

Component model: containment hierarchy

«Java Interface»
ⓘ Interface

- getName ( )
- getExtendedInterface ( )
- getExtendedInterfaces ( )
- getInterfaceFault ( )
- getInterfaceFaults ( )
- getInterfaceOperation ( )
- getInterfaceOperations ( )
- toElement ( )

«use»

«use»

«use»

{interfaces}

«Java Interface»
ⓘ Binding

- getName ( )
- getInterface ( )
- getType ( )
- getBindingFaults ( )
- getBindingOperations ( )
- toElement ( )

«Java Interface»
ⓘ Service

- getName ( )
- getInterface ( )
- getEndpoints ( )

«use»

{services}

«Java Interface»
ⓘ Description

- getInterfaces ( ) : Interface [*]
- getInterface (name : QName) : Interface
- getBindings ( ) : Binding [*]
- getBinding (name : QName) : Binding
- getServices ( ) : Service [*]
- getService (name : QName) : Service
- getElementDeclarations ( ) : ElementDeclaration [*]
- getElementDeclaration (name : QName) : ElementDeclaration
- getTypeDefinitions ( ) : TypeDefinition [*]
- getTypeDefinition (qname : QName) : TypeDefinition
- toElement ( ) : DescriptionElement

«use»

{bindings}

{type definitions}

«use»

{element declarations}

«use»

«Java Interface»
ⓘ TypeDefinition

- getName ( )
- getSystem ( )
- getContentModel ( )
- getContent ( )

«Java Interface»
ⓘ ElementDeclaration

- getName ( )
- getSystem ( )
- getContentModel ( )
- getContent ( )

38

ApacheCon Europe 06

Interface and Binding components

Binding* navigates to Interface*

Copyright © IBM Corp 2006

Service

DescriptionElement

«Java Interface»
DescriptionElement

- setDocumentBaseURI ( )
- getDocumentBaseURI ( )
- setTargetNamespace ( )
- getTargetNamespace ( )
- addNamespace ( )
- removeNamespace ( )
- getNamespace ( )
- getNamespaces ( )
- createDocumentationElement ( )
- createImportElement ( )
- createIncludeElement ( )
- createTypesElement ( )
- createInterfaceElement ( )
- createInterfaceFaultElement ( )
- createInterfaceOperationElement ( )
- createInterfaceFaultReferenceElement ( )
- createInterfaceMessageReferenceElement ( )
- createBindingElement ( )
- createBindingFaultElement ( )
- createBindingOperationElement ( )
- createBindingFaultReferenceElement ( )
- createBindingMessageReferenceElement ( )
- createFeatureElement ( )
- createServiceElement ( )
- createEndpointElement ( )
- createPropertyElement ( )
- addImportElement ( )
- getImportElements ( )
- addIncludeElement ( )
- getIncludeElements ( )
- setTypesElement ( )
- getTypesElement ( )
- addInterfaceElement ( )
- getInterfaceElements ( )
- addBindingElement ( )
- getBindingElements ( )
- addServiceElement ( )
- getServiceElements ( )
- setExtensionRegistry ( )
- getExtensionRegistry ( )
- toComponent ( )

- Container and factory for child elements

```
<description
      targetNamespace="xs:anyURI" >
  <documentation />*
  [ <import /> | <include /> ]*
  <types />?
  [ <interface /> | <binding /> | <service /> ]*
</description>
```

[3] W3C Copyright Notice

ApacheCon
Europe 06

41

Element model: containment hierarchy


```
<description
    targetNamespace="xs:anyURI" >
    <documentation />*
    [ <import /> | <include /> ]*
    <types />?
    [ <interface /> | <binding />
        | <service /> ]*
</description>
```

ApacheCon
Europe 06

Agenda

- Background to WSDL 2.0 and Apache Woden
- WSDL 2.0 Overview
- WSDL 1.1 to WSDL 2.0 Comparison
- Woden API Overview
- **WSDL 2.0 validation in Woden**
- WSDL 2.0 extensibility in Woden
- WSDL4J to Woden Comparison
- WSDL 1.1 to 2.0 Converter
- Woden Roadmap

ApacheCon
Europe 06

WSDL 2.0 validation rules

- Infoset syntax
  - defined by WSDL 2.0 XML schema
  - http://www.w3.org/2006/01/wsdl/wsdl20.xsd
- Sematic rules
  - Defined in the WSDL 2.0 spec as **assertions**
  - Some apply to a WSDL Document (Infoset)
    - QName refs, name scope, import/include visibility, etc
  - Some apply to Component model
    - Model structure, data relationships, scope, etc

ApacheCon
Europe 06

WSDL 2.0 assertions

WSDL Document examples:

InterfaceMessageReference-1205002 The type of the element *attribute information item* is a union of *xs:QName* and *xs:token* where the allowed token values are *#any*, *#none*, or *#other*.

Schema-0017 The referenced schema MUST contain a targetNamespace *attribute information item* on its xs:schema *element information item*.

Component model examples:

Interface-0030 For each Interface component in the {interfaces} property of a Description component, the {name} property MUST be unique.

InterfaceFaultReference-0043 The value of this property MUST match the name of a placeholder message defined by the message exchange pattern.

ApacheCon
Europe 06

Woden WSDL Validation

- Validation is 'off' by default.

- Enable validation before reading the WSDL:
  ```
  reader.setFeature(WSDLReader.FEATURE_VALIDATION, true);
  DescriptionElement desc = reader.readWSDL("http://...");
  ```

- Schema validation is done by the XML parser
  - by Xerces in the Woden DOM implementation

- Semantic validation is done by Woden
  - on the WSDL object model
  - assertions are checked against the model

- All errors are reported in one pass
  - Woden does not stop-on-first-error like WSDL4J

- The WSDL object model is always returned
  - even if it has errors.

46

ApacheCon
Europe 06

Reporting WSDL Errors

- `ErrorReporter` reports 3 levels of severity
  - **Warning**, **Error**, **Fatal Error**
  - Processing continues after Warning or Error
  - Processing terminates after Fatal Error

- Default `ErrorHandler` prints messages to System.out
- Can customize error handling
  - Re-implement `ErrorHandler` with required behaviour
  - Set via `ErrorReporter.setErrorHandler`

- Error messages retrieved from ResourceBundle
  - Parameterized for insertion of error details
  - NLS support via 'locale'

ApacheCon
Europe O6

WSDL Error vs WSDLException

- WSDL 'warning' or 'error' do not terminate processing
  - Reported as per ErrorHandler implementation
  - Processing continues

- `WSDLException` terminates processing
  - WSDL 'fatal error'
    - WSDL document not found
    - root element is not <description>
  - Woden config error
    - invalid WSDLReader Feature or Property
    - Extension registration error
  - Wrapper for checked exceptions
    - MalformedURLException, SAXException, etc
    - NOTE: runtime exceptions are NOT wrapped

ApacheCon Europe 06

Agenda

- Background to WSDL 2.0 and Apache Woden
- WSDL 2.0 Overview
- WSDL 1.1 to WSDL 2.0 Comparison
- Woden API Overview
- WSDL 2.0 validation in Woden
- **WSDL 2.0 extensibility in Woden**
- WSDL4J to Woden Comparison
- WSDL 1.1 to 2.0 Converter
- Woden Roadmap

ApacheCon
Europe 06

WSDL 2.0 Extensibility

- Every WSDL 2.0 element is extensible
  - Via extension elements and extension attributes
- Every WSDL 2.0 component is extensible
  - Via extension properties

- Extension properties are derived from extension elements and attributes
  - But no differentiation in the Component model

- WSDL 2.0 defines extensions for:
  - SOAP bindings
  - HTTP bindings
  - Interface operation 'safety'
  - 'rpc' operation style

ApacheCon
Europe 06

Extension Property vs Element and Attribute

Component extension properties do not differentiate extension elements and attributes.

```
<binding name="reservationBinding"
    interface="tns:reservation"
    type="http://www.w3.org/2006/01/wsdl/soap"
    wsoap:version="1.2"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP"
    wsoap:mepDefault="http://www.w3.org/2003/05/soap/mep/request-response">

    <wsoap:module ref="http://example.org" required="true" />
    </wsoap:module>
    ...
</binding>
```

Binding

{type} is SOAP

{soap version}

{soap underlying protocol}

{soap mep default}

{soap modules}

ApacheCon Europe 06
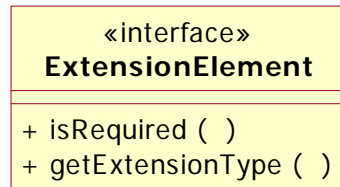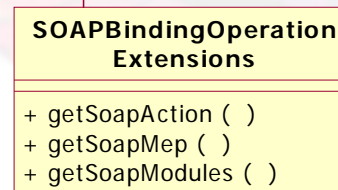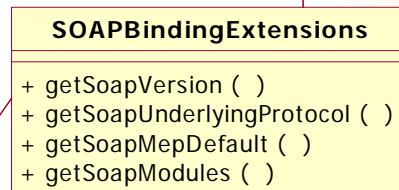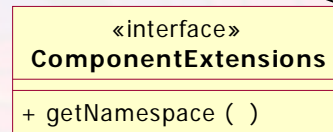
WSDL extensions in Woden

1. Application developer implements some Woden interfaces
   - to create read/write logic for extension elements and attributes
   - to represent the extension properties of a Component for a particular namespace

2. Application registers these implementations in extension registry

3. Woden will query extension registry and invoke extension-specific logic when extensions are encountered

ApacheCon
EuROPE O6

Step 1: Implement extensions

Example:
<wsoap:module>
extension element

«interface»
ExtensionElement

+ isRequired ( )
+ getExtensionType ( )

«interface»
ExtensionDeserializer

+ unmarshal ( )

«interface»
ExtensionSerializer

+ marshall ( )

SOAPModuleElement

SOAPModuleDeserializer

SOAPModuleSerializer

These implementations support
the <wsoap:module> element
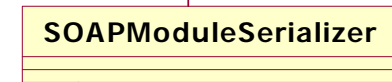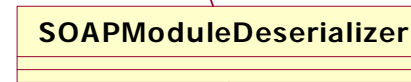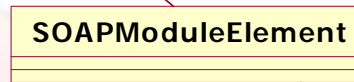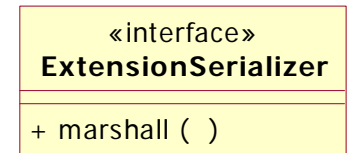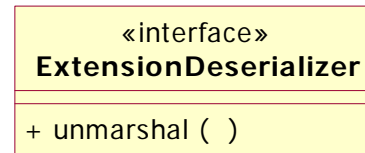
Example:

Binding component
extensions

BindingOperation
component extensions

«interface»
ComponentExtensions

+ getNamespace ( )

SOAPBindingExtensions

+ getSoapVersion ( )
+ getSoapUnderlyingProtocol ( )
+ getSoapMepDefault ( )
+ getSoapModules ( )

**SOAPBindingOperation
Extensions**

+ getSoapAction ( )
+ getSoapMep ( )
+ getSoapModules ( )

Binding extension properties
{soap version}, {soap underlying
protocol}, {soap mep default} a...

BindingOperation extension
properties {soap action}, {soap
mep} and {soap modules}

ApacheCon
Europe 06

53

Step 2: Register extensions

- Woden registers extensions for the 4 extension types defined by WSDL 2.0
  - SOAP, HTTP, operation safety, rpc style
- Application registers any application-specific extensions

- Register classes that parse extension elements
  - `registerExtElement`
  - `registerDeserializer`
  - `registerSerializer`
- Register classes parse extension attributes
  - `registerExtAttributeType`
- Register Component extension wrapper classes
  - `registerComponentExtension`

«interface» **ExtensionRegistry**
+ registerExtElement ( )   + registerDeserializer ( )   + registerSerializer ( )   + registerExtAttributeType ( )   + registerComponentExtension ( )   + createExtElement ( )   + queryDeserializer ( )   + querySerializer ( )   + createExtAttribute ( )   + createComponentExtension ( )

ApacheCon
Europe 06

Step 3: Invoke extensions

- At read-time or write-time, when Woden encounters an extension (something outside the WSDL namespace) it invokes the registered extension logic to handle it.

- Get objects to parse an extension element
  - `createExtElement`
  - `queryDeserializer`
  - `querySerializer`
- Get objects to parse an extension attribute
  - `createExtAttribute`
- Get a Component extension wrapper object
  - `createComponentExtension`

«interface» **ExtensionRegistry**
+ registerExtElement ( ) + registerDeserializer ( ) + registerSerializer ( ) + registerExtAttributeType ( ) + registerComponentExtension ( ) + createExtElement ( ) + queryDeserializer ( ) + querySerializer ( ) + createExtAttribute ( ) + createComponentExtension ( )

ApacheCon Europe 06

Agenda

- Background to WSDL 2.0 and Apache Woden
- WSDL 2.0 Overview
- WSDL 1.1 to WSDL 2.0 Comparison
- Woden API Overview
- WSDL 2.0 validation in Woden
- WSDL 2.0 extensibility in Woden
- **WSDL4J to Woden Comparison**
- WSDL 1.1 to 2.0 Converter
- Woden Roadmap

ApacheCon Europe 06

WSDL4J-Woden comparison (1)

- Similarities:
  - Factory, Reader, Features/Properties, Extension Registry
  - Ideas from WSDL4J have been reused
    - but little code, mainly implementation utility classes
- WSDL object models
  - JWSDL API in WSDL4J (javax.wsdl)
  - Component and Element APIs in Woden
- XML parser strategy
  - JAXP / DOM in WSDL4J
  - Pluggable XML parser in Woden (goal!)
    - DOMWSDLReader based on Xerces/DOM
    - StAXWSDLReader development is underway

ApacheCon
Europe 06

WSDL4J-Woden comparison (2)

- Validation
  - WSDL4J:
    - Partial validation at parse-time in the Reader
    - Terminate on error, so report only first error found
  - Woden:
    - Validate entire WSDL object model and report all errors
    - Reader returns the WSDL object model, valid or invalid
- Error handling
  - WSDL4J
    - System.out.println
    - English language message strings embedded in source code
  - Woden
    - Customizable `ErrorHandler` (default behaviour is System.out.println)
    - Messages in ResourceBundle for Internationalization and parameterization
    - 3 levels of error; warning, error, fatal error

ApacheCon
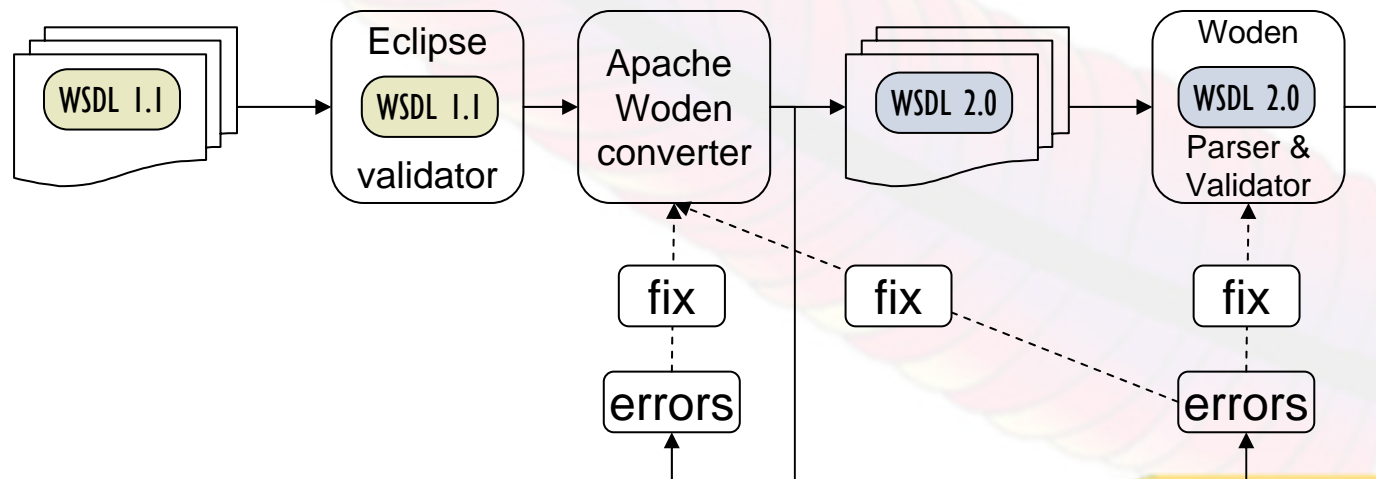Europe 06

WSDL4J-Woden comparison (3)

- Extension architecture
  - WSDL4J
    - Supports only extension elements and attributes (WSDL infoset)
    - Register Deser/Serializer/Type for extension elements
    - Reader/Writer handle extension attributes
  - Woden
    - Supports extension elements and attributes in Element API
    - These map to extension properties in Component API
    - Register 3 things!
      - Deser/Serialzer/Type for extension elements
      - XMLAttr subtype for extension attributes
      - ComponentExtensions subtype for extension properties (by Namespace)
- Supported WSDL versions
  - WSDL 1.1 in WSDL4J
  - WSDL 2.0 and (soon) WSDL 1.1 conversion in Woden
    - Read WSDL 1.1 doc, return WSDL 2.0 object model
    - Migration path for existing WSDL4J / WSDL 1.1 applications
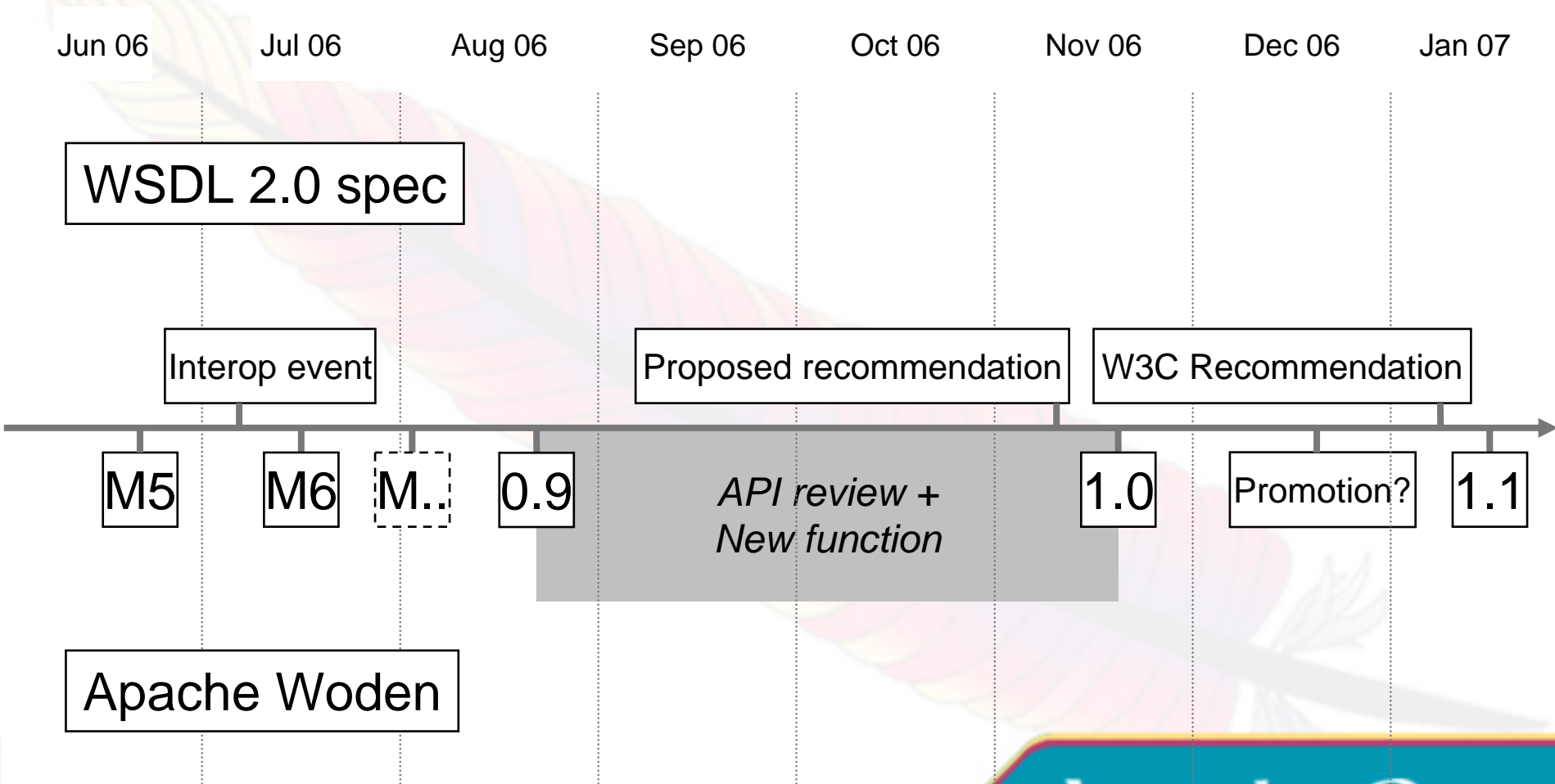
ApacheCon
Europe 06

Agenda

- Background to WSDL 2.0 and Apache Woden
- WSDL 2.0 Overview
- WSDL 1.1 to WSDL 2.0 Comparison
- Woden API Overview
- WSDL 2.0 validation in Woden
- WSDL 2.0 extensibility in Woden
- WSDL4J to Woden Comparison
- **WSDL 1.1 to 2.0 Converter**
- Woden Roadmap

ApacheCon
Europe 06

Apache Woden - Converter

- Converts WSDL 1.1 to WSDL 2.0
  - aid to WSDL 2.0 adoption
- Developed independently
- Will evolve to use Woden parser given WSDL 1.1 support in Woden
  - WSDL 2.0 documents ->in memory Woden objects -> serialize to WSDL 2.0 doc
- Basis of (future) test harness

ApacheCon Europe 06

Apache Woden roadmap

| Jun 06 | Jul 06 | Aug 06 | Sep 06 | Oct 06 | Nov 06 | Dec 06 | Jan 07 |

WSDL 2.0 spec

Interop event

Proposed recommendation

W3C Recommendation

M5   M6   M..   0.9

*API review +
New function*

1.0   Promotion?   1.1

Apache Woden

ApacheCon
Europe 06

Completing the function

- WSDLWriter
  - Serialize WSDL 2.0 object models to file
- WSDL 1.1 to 2.0 conversion
- Updatable component model
  - Add modifier methods
- Support for other parsers – StAX / SAX

- We need you – users and contributors
  - woden-dev-subscribe@ws.apache.org

ApacheCon
Europe 06

W3C Copyright Notice

The content in this document referred to below is Copyright W3C:

[1] MEP In-Out template
(WSDL 2.0 Part 2 Adjuncts, http://www.w3.org/TR/2006/CR-wsdl20-adjuncts-20060327/)

[2] WSDL 2.0 Component model diagram
(WSDL 2.0 Primer, http://www.w3.org/TR/2006/CR-wsdl20-primer-20060327/)

[3] Description Component XML representation
(WSDL 2.0 Part1 Core Language, http://www.w3.org/TR/2006/CR-wsdl20-20060327 /)

ApacheCon
Europe 06