Functional Specification for Choice Mappings

Project ID: 7875/Choice Mappings for OXM

Version: November 1st, 2007

Status: Review Draft

Author: Matt MacIvor

Project Manager: david.twelves

Change Log		
Version	**Reviewers**	**Changes**
11/1/07		Initial draft for review

Table of Contents

1. Project Overview

The goal of this feature is to provide EclipseLink OXM support for mapping to elements within a choice or to elements that are part of a substitution group.

Typically in EclipseLink OXM, an attribute is mapped to a single xpath and has a specific type. With a substitution group, or a choice, the name of the element determines the type, so a single java attribute would be mapped to multiple xpaths and each xpath would have an associated type. The support for choice is required for JAXB 2.0 and substitution groups are required for SDO.

2. Requirements

2.1. Functionality

This feature must support the following use cases:

Use Case 1 – inheritance

```
<element name=”addr”  type=”address”/>
<element name=”usAddr” type=“usAddress”  substitutionGroup=”addr”/>
<element name=”cdnAddr” type=”cdnAddress”  substitutionGroup=”addr”/>

<complexType name=”employee”>
      <element ref=”addr”/>
</complexType>

Valid XML Docs
<employee>
   <addr>
      <street>theStreet</street>
   </addr>
</employee>

<employee>
   <cdnAddr>
      <street>theStreet</street>
      <postalCode>123</postalCode>
   </cdnAddr>
</employee>

Java:
Employee object has attribute named address set to an instance of
CanadianAddress

<employee>
   <addr xsi:type=”cdnAddress”>
      <street>theStreet</street>
      <postalCode>123</postalCode>
   </addr>
</employee>

Java:
Employee object has attribute named address set to an instance of
CanadianAddress
```

Use Case 2 - inheritance + local vs. global element same name

```
<element name=”addr”  type=”address”/>
<element name=”usAddr” type=“usAddress”  substitutionGroup=”addr”/>
<element name=”cdnAddr” type=”cdnAddress”  substitutionGroup=”addr”/>

<complexType name=”employee”>
      <element ref=”addr”/>
</complexType>

<complexType name=”company”>
      <element name=”addr” type=”address”/>
</complexType>

the “addr” under company can NOT be substituted

XML doc example we should read/write
<root>
   <employee>
      <cdnAddr>
         <street>theStreet</street>
         <postalCode>123</postalCode>
      </cdnAddr>
   </employee>
   <company>
      <addr xsi:type=”cdnAddress”>
         <street>theStreet</street>
         <postalCode>123</postalCode>
      </addr>
   </company>
</root>
```

Java
Employee Object has attribute named “addr” set to an instance of CanadianAddress
Company Object has property named “addr” set to an instance of CanadianAddress

Use Case 3 – no inheritance simple

```
<element name=”theName”  type=”xsd:string”/>
<element name=”theTest” type=“xsd:string” substitutionGroup=”theName”/>

<complexType name=”employee”>
      <element ref=” theName”/>
</complexType>

<employee>
      <theName>test</theName>
</employee>
```

Java
Employee Object has attribute named “theName” set to value “test”

```
<employee>
      <theTest>test2</theTest>
</employee>
```

Java:
Employee Object has attribute named “theName” set to value of “test2”

Use Case 4 – no inheritance simple collection

```
<element name=”theName”  type=”xsd:string”/>
<element name=”theTest” type=“xsd:string” substitutionGroup=”theName”/>

<complexType name=”employee”>
<element ref=” theName” maxOccurs=”unbounded”/>
</complexType>

<employee>
      <theName>test1</theName>
      <theTest>test2</theTest >
      <theName>test3</theName>
</employee>
```

Java
Employee Object has a collection attribute named “theName” set to value of List size 3(test1, test2, test3)

Use Case 5– no inheritance complex object

```
<element name=”addr”  type=”xsd:address”/>
<element name=”cAddr” type=“xsd:address” substitutionGroup=”addr”/>

<complexType name=”employee”>
<element ref=”addr”/>
</complexType>

<employee>
      <addr>
            <street>theStreet</street>
</addr>
</employee>
```

SDO
Employee DataObject has property named “addr” set to value of the address

```
<employee>
      <cAddr>
            <street>theStreet</street>
      </cAddr>
</employee>
```

Java
Employee Object has property named “addr” set to value of the address

Use Case 6– no inheritance compositeCollectionMapping

```
<element name="addr"  type="xsd:address"/>
<element name="cAddr" type="xsd:address" substitutionGroup="addr"/>

<complexType name="employee">
      <element ref="addr" maxOccurs="unbounded"/>
</complexType>

<employee>
      <addr>
            <street>theStreet</street>
      </addr>
      <cAddr>
            <street>theStreet</street>
      </cAddr>
</employee>
```

Java
Employee Object has collection attribute named "addr" set to value of List size 2 (addr, cAddr)

Use Case 7– Simple Choice

```
<complexType name="employee">
      <choice>
            <element name="stringTest" type="xsd:string"/>
            <element name="intTest" type="xsd:int"/>
      </choice>
</complexType>

XMLDoc 1
<employee>
      <stringTest>theString</stringTest>
</employee>

XMLDoc 2
<employee>
      <intTest>10</intTest>
</employee>
```

Use Case 8– Complex Choice

```
<complexType name="employee">
      <choice>
            <element name="phone" type="phoneType"/>
            <element name="address" type="addressType"/>
      </choice>
</complexType>

XMLDoc 1
<employee>
      <phone>
            <number>(613)123-4567</number>
      </phone>
</employee>

XMLDoc 2
<employee>
      <address>
            <street>theStreet</street>
      </address>
```

```
</employee>
```

Use Case 9– Simple and Complex Choice

```
<complexType name=”employee”>
      <choice>
            <element name=”street” type=”xsd:string”/>
            <element name=”address” type=”addressType”/>
      </choice>
</complexType>

XMLDoc 1
<employee>
      <street>theStreet</street>
</employee>

XMLDoc 2
<employee>
      <address>
            <street>theStreet</street>
      </address>
</employee>
```

Use Case 10– Choice unbounded

```
<complexType name=”employee”>
      <choice maxOccurs=”unbounded”>
            <element name=”street” type=”xsd:string”/>
            <element name=”address” type=”addressType”/>
      </choice>
</complexType>

XMLDoc 1
<employee>
      <street>theStreet</street>
      <address>
            <street>theStreet</street>
      </address>
</employee>
```

Use Case 11– Choice unbounded element

```
<complexType name=”employee”>
   <choice>
      <element name=”street” type=”xsd:string” maxOccurs=”unbounded”/>
      <element name=”address” type=”addressType”/>
   </choice>
</complexType>

XMLDoc 1
<employee>
      <street>theStreet</street>
      <street>anotherStreet</street>
<employee>

XMLDoc 2
<employee>
      <address>
            <street>theStreet</street>
      </address>
<employee>
```

XMLRoot:
The choice mappings will also support the use of XMLRoots to preserve the element name.

2.2. Performance

- Overall performance of EclipseLink OXM must not be negatively impacted by this feature
- Performance of new mappings must be inline with current standards of EclipseLink OXM Performance.

2.3. Manageability

Does this project make the product easier or more difficult for the developer/administrator to manage in the following aspects:

Note: If manageability is not an issue for this project you must make a statement to that effect.

2.3.1. Session/Project Configuration

- This will require changes to deployment xml to support the new mappings. A user will now be able to add new choice based mappings to their projects

2.3.2. Performance Tuning

- No Performance Tuning issues

2.4. Reliability

2.5. Security

No security requirements.

2.6. Compatibility

2.6.1. Internationalization

Any new exception messages will need to be internationalized.

3. Client Interfaces

This section describes the project, as seen by the client (either a paying customer, another Oracle component, or another company's component). There should be sufficient detail here for technical writers to document the project in product manuals. Include real-world examples of the syntax and of the interface use. Discuss the expected typical usage and values.

Note: If there are no changes to any user interfaces you must make a statement to that effect.

3.1. Command-line Tools

No effect on command-line tools

3.2. Public APIs/Classes

This feature will add two new public mapping classes.

XmlChoiceMapping – This mapping should be used for any Choice or Substitution group with only a single occurance:

- public setAttributeName(String) – Sets the name of the attribute for this mapping
- public setGetMethodName(String) – sets the name of the getMethod to be used when accessing this attribute
- public setSetMethodName(String) – sets the name of the setMethod to be used when accessing this attribute
- public addChoiceElement(String xpath, Class referenceClass) – Configures the type of value to be read in if the specified choice element (xpath) is found in the document. This can be either a simple (String, Integer) value or a complex one.
- public addComplexElement(String xpath, String referenceClassName) – Configures the reference class to be used when building a complex value from the specified choice element. Takes the class name as a string instead of the class. For MW.

XmlChoiceCollectionMapping – This mapping should be used for a choice where any element in the choice could be a collection, or a choice with a maxOccurs > 1 or a substitution group reference with more than a single occurance:

- public setAttributeName(String) – Sets the name of the attribute for this mapping
- public setGetMethodName(String) – sets the name of the getMethod to be used when accessing this attribute
- public setSetMethodName(String) – sets the name of the setMethod to be used when accessing this attribute
- public addChoiceElement(String xpath, Class referenceType) – Configures the type of a simple value to be read in if the specified choice element (xpath) is found in the document. This can be a simple or a complex type.
- public addChoiceElement(String xpath, String referenceClassName) – Configures the reference class to be used when building a complex value from the specified choice element. Takes the class name instead of the class itself. For MW.
- public setContainerClass(Class containerClass) – Sets the container type to use for this collection mapping
- public setContainerClassName(String className) – As above but takes the class name instead of the class.
- public setContainerPolicy(ContainerPolicy policy) – Sets the container policy to be used by this mapping to be the provided policy.

3.3. Configuration Files

Project Deployment XML will be modified to include new mappings.

4. Dependencies and Effects

4.1. Software

No new software dependencies will be added..

4.2. User Documentation

A section should be added to the user docs to describe this mapping and it’s usage.

5. Compatibility & Migration

A user that was making use of a work around to handle the choice or substitution group use cases in a previous release will be able to migrate to use the newly provided mappings.

A. References

Indicate all references for the project, giving them a unique code you can refer to in the text of the specification, for example, "see [JAXB] ." The following references are examples; replace them with your own references.

Table 6–1 References

[JAXB]	https://jaxb.dev.java.net/

C. Open Issues

Issue Log			
Issue #	**Name**	**Description**	**Status**
1			OPEN/CLOSED
2			
3			

D. Decisions

Decision Log		
Decision #	**Name**	**Description**
1	Policy vs Mapping	Decided to implement this as it's own mapping rather than as a policy on a separate mapping as a means to support both complex and simple values in the same choice.
2		
3		