



FUSE Services Framework

Using the XML Binding

Version 2.0
December 2007

Making Software Work Together™

Using the XML Binding

IONA Technologies

Version 2.0

Published 19 Mar 2008

Copyright © 2001-2008 IONA Technologies PLC

Trademark and Disclaimer Notice

IONA Technologies PLC and/or its subsidiaries may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this publication. Except as expressly provided in any written license agreement from IONA Technologies PLC, the furnishing of this publication does not give you any license to these patents, trademarks, copyrights, or other intellectual property. Any rights not expressly granted herein are reserved.

IONA, IONA Technologies, the IONA logo, Orbix, High Performance Integration, Artix, FUSE, and Making Software Work Together are trademarks or registered trademarks of IONA Technologies PLC and/or its subsidiaries.

Java and J2EE are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. CORBA is a trademark or registered trademark of the Object Management Group, Inc. in the United States and other countries. All other trademarks that appear herein are the property of their respective owners.

While the information in this publication is believed to be accurate, IONA Technologies PLC makes no warranty of any kind to this material including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. IONA shall not be liable for errors contained herein, or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Copyright Notice

No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, photocopying, recording or otherwise, without prior written consent of IONA Technologies PLC. No third-party intellectual property right liability is assumed with respect to the use of the information contained herein. IONA Technologies PLC assumes no responsibility for errors or omissions contained in this publication. This publication and features described herein are subject to change without notice. Portions of this document may include Apache Foundation documentation, all rights reserved.

Table of Contents

Using XML Documents	7
Index	13

List of Examples

1. Valid XML Binding Message	8
2. Invalid XML Binding Message	9
3. Invalid XML Document	9
4. XML Binding with rootNode set	10
5. XML Document generated using the rootNode attribute	10
6. Using <code>xformat:body</code>	10

Using XML Documents

Summary

The pure XML payload format provides an alternative to the SOAP binding by allowing services to exchange data using straight XML documents without the overhead of a SOAP envelope.

XML binding namespace

The extensions used to describe XML format bindings are defined in the namespace `http://cxf.apache.org/bindings/xformat`. FUSE Services Framework tools use the prefix `xformat` to represent the XML binding extensions. Add the following line to your contracts:

```
xmlns:xformat="http://cxf.apache.org/bindings/xformat"
```

Hand editing

To map an interface to a pure XML payload format do the following:

1. Add the namespace declaration to include the extensions defining the XML binding. See XML binding namespace.
2. Add a standard WSDL `binding` element to your contract to hold the XML binding, give the binding a unique `name`, and specify the name of the WSDL `portType` element that represents the interface being bound.
3. Add an `xformat:binding` child element to the `binding` element to identify that the messages are being handled as pure XML documents without SOAP envelopes.
4. Optionally, set the `xformat:binding` element's `rootNode` attribute to a valid QName. For more information on the effect of the `rootNode` attribute see XML messages on the wire.
5. For each operation defined in the bound interface, add a standard WSDL `operation` element to hold the binding information for the operation's messages.
6. For each operation added to the binding, add the `input`, `output`, and `fault` children elements to represent the messages used by the operation.

These elements correspond to the messages defined in the interface definition of the logical operation.

7. Optionally add an `xformat:body` element with a valid `rootNode` attribute to the added `input`, `output`, and `fault` elements to override the value of `rootNode` set at the binding level.



Note

If any of your messages have no parts, for example the output message for an operation that returns void, you must set the `rootNode` attribute for the message to ensure that the message written on the wire is a valid, but empty, XML document.

XML messages on the wire

When you specify that an interface's messages are to be passed as XML documents, without a SOAP envelope, you must take care to ensure that your messages form valid XML documents when they are written on the wire. You also need to ensure that non-FUSE Services Framework participants that receive the XML documents understand the messages generated by FUSE Services Framework.

A simple way to solve both problems is to use the optional `rootNode` attribute on either the global `xformat:binding` element or on the individual message's `xformat:body` elements. The `rootNode` attribute specifies the QName for the element that serves as the root node for the XML document generated by FUSE Services Framework. When the `rootNode` attribute is not set, FUSE Services Framework uses the root element of the message part as the root element when using doc style messages, or an element using the message part name as the root element when using rpc style messages.

For example, if the `rootNode` attribute is not set the message defined in Example 1, "Valid XML Binding Message" would generate an XML document with the root element `lineNumber`.

Example 1. Valid XML Binding Message

```
<type ...>
...
<element name="operatorID" type="xsd:int"/>
...
```

```
</types><message name="operator"><part name="lineNumber" element="ns1:operatorID"/>
</message>
```

For messages with one part, FUSE Services Framework will always generate a valid XML document even if the `rootNode` attribute is not set. However, the message in Example 2, “Invalid XML Binding Message” would generate an invalid XML document.

Example 2. Invalid XML Binding Message

```
<types>
  ...
  <element name="pairName" type="xsd:string"/>
  <element name="entryNum" type="xsd:int"/>
  ...
</types>

<message name="matildas">
  <part name="dancing" element="ns1:pairName"/>
  <part name="number" element="ns1:entryNum"/>
</message>
```

Without the `rootNode` attribute specified in the XML binding, FUSE Services Framework will generate an XML document similar to Example 3, “Invalid XML Document” for the message defined in Example 2, “Invalid XML Binding Message”. The generated XML document is invalid because it has two root elements: `pairName` and `entryNum`.

Example 3. Invalid XML Document

```
<pairName>
  Fred&Linda
</pairName>
<entryNum>
  123
</entryNum>
```

If you set the `rootNode` attribute, as shown in Example 4, “XML Binding with `rootNode` set” FUSE Services Framework will wrap the elements in the specified root element. In this example, the `rootNode` attribute is defined for the entire binding and specifies that the root element will be named `entrants`.

Example 4. XML Binding with rootNode set

```
<portType name="danceParty">
  <operation name="register">
    <input message="tns:matildas" name="contestant"/>
  </operation>
</portType>

<binding name="matildaXMLBinding" type="tns:dancingMatildas">
  <xmlformat:binding rootNode="entrants"/>
  <operation name="register">
    <input name="contestant"/>
    <output name="entered"/>
  </operation>
</binding>
```

An XML document generated from the input message would be similar to Example 5, “XML Document generated using the rootNode attribute”. Notice that the XML document now only has one root element.

Example 5. XML Document generated using the rootNode attribute

```
<entrants>
  <pairName>
    Fred&Linda
  </pairName>
  <entryNum>
    123
  </entryNum>
</entrants>
```

Overriding the binding's rootNode attribute setting

You can also set the `rootNode` attribute for each individual message, or override the global setting for a particular message, by using the `xformat:body` element inside of the message binding. For example, if you wanted the output message defined in Example 4, “XML Binding with rootNode set” to have a different root element from the input message, you could override the binding's root element as shown in Example 6, “Using `xformat:body`”.

Example 6. Using `xformat:body`

```
<binding name="matildaXMLBinding" type="tns:dancingMatildas">
  <xmlformat:binding rootNode="entrants"/>
  <operation name="register">
    <input name="contestant"/>
    <xformat:body/>
  </operation>
</binding>
```

```
<output name="entered">
  <xformat:body rootNode="entryStatus"/>
</output>
</operation>
</binding>
```

Index

B

bindings
XML, 7

X

xformat:binding, 7
rootNode, 7
xformat:body, 8
rootNode, 8

