



FUSE™ Mediation Router

Migrating to 1.6

Version 1.6
April 2009

Migrating to 1.6

Version 1.6

Publication date 17 Jul 2009

Copyright © 2001-2009 Progress Software Corporation and/or its subsidiaries or affiliates.

Legal Notices

Progress Software Corporation 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 Progress Software Corporation, 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.

Progress, IONA, IONA Technologies, the IONA logo, Orbix, High Performance Integration, Artix, FUSE, and Making Software Work Together are trademarks or registered trademarks of Progress Software Corporation and/or its subsidiaries in the US and other countries.

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 US 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 Progress Software Corporation 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. Progress Software Corporation 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.

All products or services mentioned in this manual are covered by the trademarks, service marks, or product names as designated by the companies who market those products.

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 Progress Software Corporation. No third-party intellectual property right liability is assumed with respect to the use of the information contained herein. Progress Software Corporation 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

- 1. Component Changes 7
 - Apache CXF Component 8
 - Other Components 10
- 2. Data Converters 11

List of Examples

1.1. Specifying the serviceClass Property	8
1.2. Turning Off the Policy Engine	8
2.1. Using Unmashed Message Data	11

Chapter 1. Component Changes

There are two major changes to the FUSE Mediation Router components.

Apache CXF Component	8
Other Components	10

Apache CXF Component

systemClass property

When using the Apache CXF component as a producer, or `to` endpoint, the value of the `serviceClass` property must be a Java interface representing the service endpoint interface (SEI). You can no longer use a Java class as the SEI.

[Example 1.1 on page 8](#) shows an example specifying the `systemClass` property for a Apache CXF producer implementing the `HelloWorld` interface.

Example 1.1. Specifying the serviceClass Property

```
<beans ...>

  <cxf:cxfEndpoint id="routerEndpoint" ... />

  <cxf:cxfEndpoint id="serviceEndpoint"
    serviceClass="org.apache.hello_world_soap_http.Hello
World"
    ... />

  <camelContext id="camel" xmlns="http://activemq.apache.org/camel/schema/spring">
    <route>
      <from uri="cxf:bean:routerEndpoint" />
      <to uri="cxf:bean:serviceEndpoint" />
    </route>
  </camelContext>

</beans>
```

Policy engine

The Apache CXF policy engine is turned on by default. This means that WS-Policy assertions contained in WSDL documents are evaluated. If the runtime supports a given policy assertion, it will act accordingly.

To disable the policy engine you can add the configuration snippet in [Example 1.2 on page 8](#) to your application's configuration.

Example 1.2. Turning Off the Policy Engine

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:cxf="http://cxf.apache.org/core"
  xmlns:p="http://cxf.apache.org/policy"
```



```

        xsi:schemaLocation="
http://cxf.apache.org/core http://cxf.apache.org/schemas/core.xsd
http://cxf.apache.org/policy http://cxf.apache.org/schemas/policy.xsd
http://www.springframework.org/schema/beans http://www.springframe
work.org/schema/beans/spring-beans-2.0.xsd">
    ...
    <cxf:bus>
        <cxf:features>
            <p:policies enabled="false" />
        </cxf:features>
    </cxf:bus>
    ...
</beans>

```

JMS

The Apache CXF JMS transport defaults to using the JMS 1.1 API. If you are using a JMS 1.0 provider you can change this behavior by setting the `useJms11` configuration attribute to `false`.

Other Components

JBI component

Stream caching is turned on by default in 1.6. It is no longer necessary to include the `streamCaching()` processor to routes that use JBI endpoints if the message body is read multiple times.

Jetty component

The `camel-jetty` producer, or `to` endpoint, is no longer provided. It should be replaced by a `camel-http` producer.



Tip

This API change also means that the dependency on the `jetty-client` module is removed.

Chapter 2. Data Converters

The JAXB data marshaller behavior has changed and does not always return the expected result for unmarshalled `JAXBElement` objects.

Overview

When using the JAXB data format in conjunction with XJC to create Java classes from XML Schema, the run time will default to using the instance value provided in the XML Schema for `JAXBElements`. The actual value of the associated element in the unmarshalled message is ignored.

Work around

If you want to get the value of the element mapped to a `JAXBElement` object from the unmarshalled message body, you need to set the `JaxbDataFormat` object's `ignoreJAXBElement` property to `false`.

[Example 2.1 on page 11](#) shows a route that uses the unmarshalled message values.

Example 2.1. Using Unmarshaled Message Data

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="direct:start"/>
    <marshal>
      <jaxb ignoreJAXBElement="false" />
    </marshal>
    <to uri="mock:result"/>
  </route>
</camelContext>
```

