Progress FUSE

FUSE™ Mediation Router

Migrating to 1.6

Version 1.6
April 2009

PROGRESS SOFTWARE

Migrating to 1.6

Version 1.6

Publication date 17 Jul 2009

Legal Notices

Table of Contents

List of Examples

Chapter 1. Component Changes

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

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://act
ivemq.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 snipit 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 classed 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 unmarshaled message is ignored.

Work around

If you want to get the value of the element mapped to a `JAXBElement` object form the unmarshaled 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 unmarshaled message values.

Example 2.1. Using Unmashaled Message Data

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