JMX Management

Configuring JMX Integration

To enable JMX integration, register an InstrumentationManager extension with the CXF bus. Using Spring XML on Tomcat, the following minimal XML snippet will enable JMX integration.

```
<import resource="classpath:META-INF/cxf/cxf.xml"/>
...

<bean id="org.apache.cxf.management.InstrumentationManager"
  class="org.apache.cxf.management.jmx.InstrumentationManagerImpl">
  <property name="enabled" value="true" />
  <property name="bus" ref="cxf" />
  <property name="usePlatformMBeanServer" value="true" />
</bean>
```

The default InstrumentationManager accepts the following configuration options:

Name	Value	Default
enabled	If the JMX integration should be enabled or not	false
bus	The CXF bus instance to register the JMX extension with	None
server	An optional reference to an MBeanServer instance to register MBeans with. If not supplied, an MBeanServer is resolved using the "usePlatformMBeanServer" and/or "serverName" optoins.	None
usePlatformMBeanServer	If true and no reference to an MBeanServer is supplied, the JMX extension registers MBeans with the platform MBean server.	false
serverName	If supplied, usePlatformMBeanServer is false, and no reference to an MBeanServer is supplied, the JMX extension registers MBeans with the MBean server carrying this name.	None
createMBServerConnectorFactory	If true, a connector is created on the MBeanServer.	true
threaded	Determines if the creation of the MBean connector is performed in this thread or in a separate thread. Only relevant if createMBServerConnectorFactory is true.	false
daemon	Determines if the MBean connector creation thread is marked as a daemon thread or not. Only relevant if createMBServerConnectorFactory is true.	false
JMXServiceURL	The URL of the connector to create on the MBeanServer. Only relevant if createMBServerConnectorFactory is true.	service:jmx:rmi:///jndi/rmi://localhost:9913/jmxrmi

The MBean instrumentation provided by the above configuration will provide generic information about the WSDL supported by the web service as well as web service administration commands. To see performance metrics of the SOAP call processing, further configuration is required – these are disabled by default to avoid unnecessary runtime overhead.

If you're using Maven, make sure you have the following dependency added to the pom.xml for the web service provider:

```
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-management</artifactId>
    <version>${cxf.version}</version>
</dependency>
```

Example Configuration

Enable JMX integration by adding the following XML to your CXF Spring context.

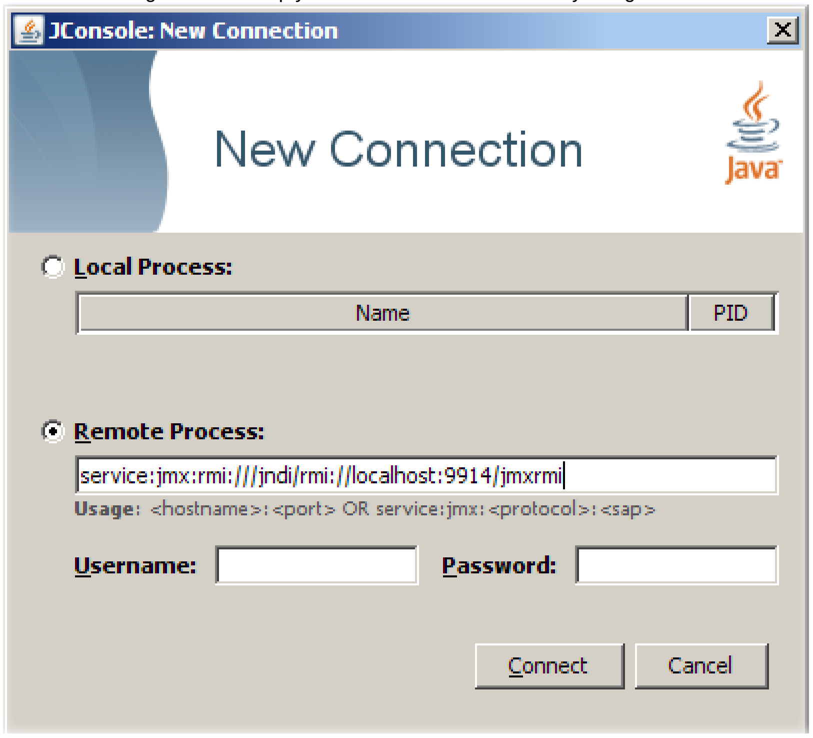
```
<bean id="org.apache.cxf.management.InstrumentationManager"
  class="org.apache.cxf.management.jmx.InstrumentationManagerImpl">
  <property name="bus" ref="cxf" />
  <property name="enabled" value="true" />
  <property name="JMXServiceURL " value="service:jmx:rmi:///jndi/rmi://localhost:9914/jmxrmi" />
</bean>
```

Starting from 2.5.2, an equivalent configuration of the above instrumentation manager can be directly made within the bus configuration using the corresponding property names having the "bus.jmx" prefix, as in

```
<cxf:bus bus="cxf">
  <cxf:properties>
    <entry key="bus.jmx.enabled" value="true"/>
    <entry key="bus.jmx.JMXServiceURL" value="service:jmx:rmi:///jndi/rmi://localhost:9914/jmxrmi" />
  </cxf:properties>
</cxf:bus>
```

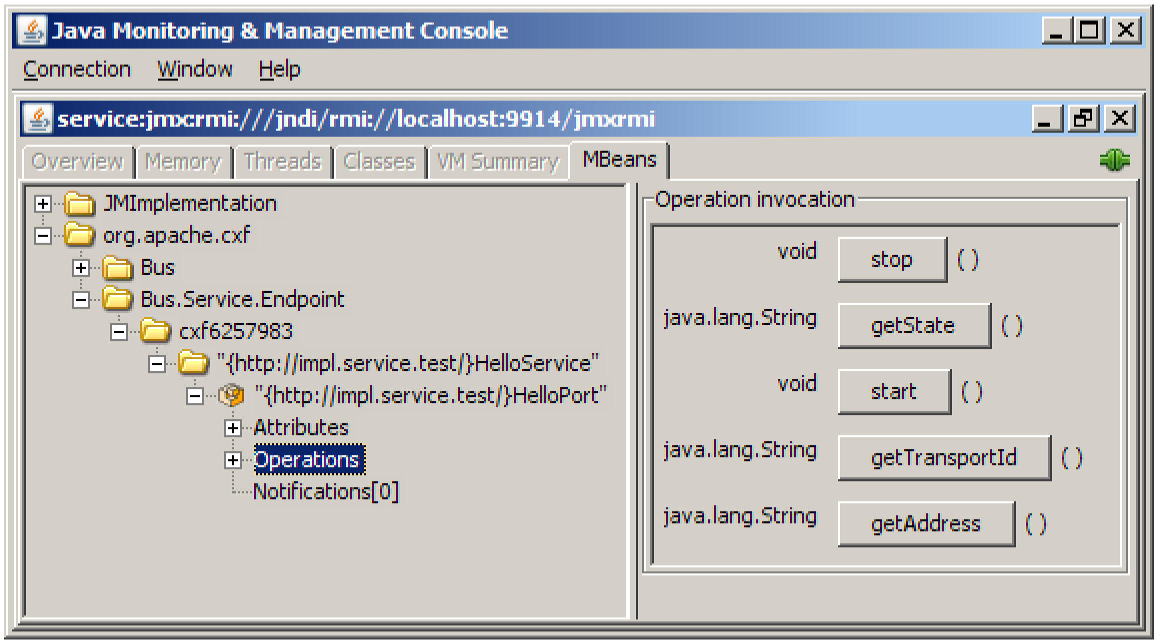
Changes in CXF 2.5.x
Starting from 2.5.0, if a MBeanServer is available in the Spring context or as an OSGi server (when running in OSGi), the InstrumentationManger will be automatically enabled and will use that MBeanServer and the CXF MBeans will be registered. Therefore, the instrumentation manager configuration shown above is not needed in such cases.

To test the configuration start up your service and connect to it by using JConsole from the JDK.



```
service:jmx:rmi:///jndi/rmi://localhost:9914/jmxrmi
```

Then you can browse to your endpoint:


Java Monitoring & Management Console
Connection
Window
Help
service:jmxrmi:///jndi/rmi://localhost:9914/jmxrmi
Overview
Memory
Threads
Classes
VM Summary
MBeans
Operation invocation

```
JMImplementation
org.apache.cxf
  Bus
  Bus.Service.Endpoint
    cxf6257983
      "{http://impl.service.test/}HelloService"
        "{http://impl.service.test/}HelloPort"
          Attributes
          Operations
          Notifications[0]
```

```
void              stop ( )
java.lang.String  getState ( )
void              start ( )
java.lang.String  getTransportId ( )
java.lang.String  getAddress ( )
```

Configuring CXF to Use the ServiceMix 4 MBeanServer

If you are embedding a CXF service in a ServiceMix 4 container, the configuration is slightly different from above. You don't want to start a new MBeanServer and you probably don't want to create additional connectors as the container manages both of these for you. You can get a reference to the container's MBeanServer through the OSGi framework and inject this reference into the JMX integration extension. Don't forget to add the Spring OSGI namespace and schemaLocation to your CXF configuration file if they are not already present.

```
<!-- OSGi namespace and schemaLocation required -->
<beans ...
       xmlns:osgi="http://www.springframework.org/schema/osgi"
       ...
       xsi:schemaLocation="...
       http://www.springframework.org/schema/osgi  http://www.springframework.org/schema/osgi/spring-osgi.xsd">

...

<!-- Grab a reference to the current MBeanServer -->
<osgi:reference id="mbeanServer" interface="javax.management.MBeanServer" cardinality="0..1"/>

<bean id="org.apache.cxf.management.InstrumentationManager"
  class="org.apache.cxf.management.jmx.InstrumentationManagerImpl">
  <property name="bus" ref="cxf" />
  <property name="enabled" value="true" />
  <!-- Unless you really want to open an additional connector, set this to false -->
  <property name="createMBServerConnectorFactory " value="false" />

  <!-- Inject the reference to the MBeanServer -->
  <property name="server" ref="mbeanServer" />
</bean>
```

How to get web service performance metrics (Request/Response time, number of calls, etc.)?

The CXF management module also provides a feature (the Performance.Counter.Server MBean) which provides aggregate statistics for services running in the CXF Bus. It is not enabled by default to avoid unnecessary runtime overhead during web service call processing.

Here is the configuration snippet that you should add to your Spring context file to be able to view this information:

```
    <!-- Wiring the counter repository -->
    <bean id="CounterRepository" class="org.apache.cxf.management.counters.CounterRepository">
        <property name="bus" ref="cxf" />
    </bean>
```

The CounterRepository collects the following metrics: invocations, checked application faults, unchecked application faults, runtime faults, logical runtime faults, total handling time, max handling time, and min handling time. Note a SOAP call will need to occur against the web service before you will see the MBean within your JMX monitoring software.