Robotics Service Bus - Bug #2181
Calling rpc with double attribute results in an exception.

02/12/2015 09:19 PM - M. Pohling

Status:	Resolved	**Start date:**	02/12/2015
Priority:	Normal	**Due date:**	
Assignee:	J. Wienke	**% Done:**	100%
Category:	Java	**Estimated time:**	0.00 hour
Target version:	rsb-0.11		

Description

The following exception is thrown if a Double type is transmitted via rsb rpc call.

```
remoteServer.callAsync(MethodName, Double);

java.nio.ReadOnlyBufferException
    at java.nio.ByteBuffer.array(ByteBuffer.java:961)
    at rsb.converter.DoubleConverter.deserialize(DoubleConverter.java:82)
    at rsb.converter.DoubleConverter.deserialize(DoubleConverter.java:39)
    at rsb.protocol.ProtocolConversion.fromNotification(ProtocolConversion.java:231)
    at rsb.transport.socket.SocketInPushConnector.handle(SocketInPushConnector.java:142)
    at rsb.transport.socket.BusBase.handleLocally(BusBase.java:349)
    at rsb.transport.socket.BusBase.handleOutgoing(BusBase.java:408)
    at rsb.transport.socket.RefCountingBus.handleOutgoing(RefCountingBus.java:116)
    at rsb.transport.socket.SocketOutConnector.push(SocketOutConnector.java:116)
    at rsb.eventprocessing.DefaultOutRouteConfigurator.publishSync(DefaultOutRouteConfigurator.java:89)
    at rsb.Informer$InformerStateActive.send(Informer.java:157)
    at rsb.Informer.send(Informer.java:295)
    at rsb.patterns.RemoteMethod.call(RemoteMethod.java:172)
    at rsb.patterns.RemoteServer.callAsyncEvent(RemoteServer.java:427)
    at rsb.patterns.RemoteServer.callAsyncData(RemoteServer.java:441)
    at rsb.patterns.RemoteServer.callAsync(RemoteServer.java:202)
    at de.citec.jul.rsb.RSBRemoteService.callMethodAsync(RSBRemoteService.java:188)
    at de.citec.dal.hal.al.AmbientLightRemote.setBrightness(AmbientLightRemote.java:58)
    at de.citec.dal.hal.al.AmbientLightRemoteTest.testSetBrightness(AmbientLightRemoteTest.java:201)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:606)
    at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java:47)
    at org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java:12)
    at org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java:44)
    at org.junit.internal.runners.statements.InvokeMethod.evaluate(InvokeMethod.java:17)
    at org.junit.internal.runners.statements.FailOnTimeout$StatementThread.run(FailOnTimeout.java:74)
```

Associated revisions

Revision f0a0745c - 02/13/2015 06:08 PM - J. Wienke

ConnectorCheck.create{In,Out}Connector accept a converter map

Prepares to define the required converters from the outside.

refs #2181

Revision 9b90c769 - 02/13/2015 06:10 PM - J. Wienke

ConnectorCheck.create{In,Out}Connector accept a converter map

Prepares to define the required converters from the outside.

refs #2181

(cherry picked from commit f0a0745c27309386b9a01d6215588fcddbbeaeb9)

Revision 5d4cba78 - 02/13/2015 06:13 PM - J. Wienke

Handle read-only restrictions of ByteBuffers correctly

It is invalid to call array() on read-only ByteBuffer instances. This
was done in converters like the DoubleConverter to access the wire data
for deserialization. However, in case of the socket transport when
handling send-receive cycles in the same process, the ByteBuffer that
ended up at the receiving side for deserialization was such a read only
instance.

This commit works around this issue by using non-read-only instances all
the time. Ultimately, this is still a hacky solution because ByteBuffers
with their statefull reading interface are actually a bad choice for
Converters, because multiple converters might read them in parallel.

fixes #2181

- ConnectorCheck: new test case for this issue
- SocketInPushConnector: always create writable ByteBuffer instances

Revision c4d816db - 02/13/2015 06:14 PM - J. Wienke

Handle read-only restrictions of ByteBuffers correctly

It is invalid to call array() on read-only ByteBuffer instances. This
was done in converters like the DoubleConverter to access the wire data
for deserialization. However, in case of the socket transport when
handling send-receive cycles in the same process, the ByteBuffer that
ended up at the receiving side for deserialization was such a read only
instance.

This commit works around this issue by using non-read-only instances all
the time. Ultimately, this is still a hacky solution because ByteBuffers
with their statefull reading interface are actually a bad choice for
Converters, because multiple converters might read them in parallel.

fixes #2181

- ConnectorCheck: new test case for this issue
- SocketInPushConnector: always create writable ByteBuffer instances

(cherry picked from commit 364c30111198c8bef0d1e58226b3bb2c1bd7168f)

History

#1 - 02/13/2015 05:04 PM - J. Wienke

- Description updated

#2 - 02/13/2015 06:14 PM - J. Wienke

- Status changed from New to Resolved

- % Done changed from 0 to 100

Applied in changeset commit:rsb-java|5d4cba78edd6eedbb9837d8a1126fde4c8bb8ac2.