Robotics Service Bus - Enhancement #2050
allow shallow data copy for converter arguments?

10/09/2014 10:57 AM - R. Haschke

Status:	New	**Start date:**	10/09/2014
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	C++	**Estimated time:**	0.00 hour
Target version:	rsb-1.0		

Description

Hi Sebastian and all,

Why did you decided to pass data to converters by ref instead of shared_ptr?
This requires to copy data over and over. Instead one could easily also pass the shared_ptr,
e.g. in source:rsb-cpp|src/rsb/transport/socket/InPullConnector.cpp#L90:

```
        boost::shared_ptr<string> wireData = static_pointer_cast<string>(event->getData());
    string wireSchema = event->getMetaData().getUserInfo("rsb.wire-schema");
    AnnotatedData d = getConverter(wireSchema)->deserialize(wireSchema, *wireData);
```

and then reuse the pointer instead of reallocating, e.g. in source:rsb-cpp/src/rsb/converters/ByteArrayConverter.cpp#L49:

```
      AnnotatedData ByteArrayConverter::deserialize(const std::string& /*wireSchema*/,
                                          const string& wire) {
        return make_pair(getDataType(), boost::shared_ptr<string>(new string(wire)));
    }
```

Associated revisions

Revision 0ecb696b - 10/09/2014 04:17 PM - R. Haschke

Improved ByteArrayConverter string copying

Use the direct copy constructor instead of a workaround through c
strings.

This also adds unit tests for the converter.

refs #2050

(cherry picked from commit 5dab393ffb2ab114d320422c80567283039aebfa)

History

#1 - 10/09/2014 12:49 PM - J. Moringen

- Description updated

- Target version set to rsb-0.12

#2 - 10/09/2014 04:17 PM - J. Wienke

Since most converters actually deserialize data, this is not a problem for them. The copy operation for the ByteArrayConverter is actually intended to decouple different handlers so that one handler mangling the data cannot influence another handler attached to the same listener.

Apart from that, I have cherry-picked your commit improving the converter to the master branch since this is a simple improvement anyway.

#3 - 10/11/2014 12:02 AM - R. Haschke

I see your argument decoupling handlers. But: first, you could also enforce that by providing const pointers. And second, the argument doesn't hold: the very same (copied) string is handed over to all handlers - no decoupling.

Yes, most converters actually convert and throw away the wire string. However, not all do that. And one could easily think of converters that happily index to the original wire string, e.g. using http://www.boost.org/doc/libs/1_37_0/libs/range/doc/utility_class.html#iter_range. For example, the Micro Protocol Buffers library (https://github.com/haberman/upb) directly provides access to the protocol buffer wire in an online/dynamic fashion. As most actual converters, use the ProtocolBufferConverter only as an intermediate step to convert to the final domain type, it would be actually beneficial to for all those converters!

Finally, one could postpone the conversion to later stages. Currently all incoming messages are eagerly converted - even if never used by any handler. That could currently easily avoided by triggering the conversion only if handlers are present.
An application that would actually benefit from lazy conversion is my HSM state machine that needs to access the event in two different languages (HSM receives events at C++ level and dispatches them to an embedded python interpreter). Hence both languages need to access the original wire string and do the deserialization themselves. Currently to achieve this goal, the message would be copied 4 times (with a language switch after data string):

original receive buffer -> wire string -> data string -> protocol buffer -> domain type

#4 - 10/13/2014 09:03 PM - R. Haschke

One might even think about not even copying the notification's data content:

source:rsb-cpp|src/rsb/transport/socket/BusConnection.cpp#L252
source:rsb-cpp|src/rsb/transport/socket/Serialization.cpp#L72
By the way: Isn't this code to transform event <-> notification transport agnostic?
There is similar code in source:rsb-cpp/src/rsb/protocol/Notification.cpp#L91, which probably should be reused here.

However, in socket-transport this seems not to be possible as there is only a single messageReceiveBuffer, which is copied to a single Notification and subsequently copied to the wire string. Using several NotificationPtrs instead, we could get rid of the latter copy.

In spread, the Notification is composed from several fragments anyway and a unique NotificationPtr exists for every event:
source:rsb-spread-cpp|src/rsb/transport/spread/Assembly.cpp#L63

Key to less data copying will be a protocol buffer parser that doesn't copy but reference to the wire.

The proposed Converter API will allow to chain several converters in a modular fashion - each operating on the result of the previous one - with arbitrary branching in the conversion tree ;-)

In order to allow type safety for converters acting on the "wire", the converters could require a boost::shared<const WireType> interface, while others use VoidPtr. Constness also ensures, that converters do not modify the original message.

#5 - 04/24/2015 05:38 PM - J. Moringen

- Target version changed from rsb-0.12 to rsb-1.0