Robotics Service Bus - Bug #1153
unknown data types cause crash when received

08/29/2012 02:15 AM - R. Haschke

Status:	Rejected	**Start date:**	08/29/2012
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	C++	**Estimated time:**	0.00 hour
Target version:	rsb-0.11		

Description

An unknown data type send over a channel will cause an existing program to crash, if it did not register an appropriate handler. Often in development, a data type is added later on in the evolution of a system. It would be nice, if a monitoring application (e.g. logger), would not crash if not yet modified and recompiled appropriately. Rather, a graceful error message would be nice.

Related issues:

Related to Robotics Service Bus - Tasks # 1035: Implement error handling subs...	**New**	**06/25/2012**
Related to Robotics Service Bus - Bug # 344: C++ Spread: Transport-level erro...	**New**	**06/05/2011**
Related to Robotics Service Bus - Bug # 515: Data handlers cannot deal with u...	**New**	**08/19/2011**

History

#1 - 08/29/2012 03:23 AM - J. Moringen

- Status changed from New to Feedback

- Target version set to rsb-0.9

Both loggers should not crash, no matter which payloads they receive. Does one of the loggers crash in this situation? If so, please submit a separate issue describing the circumstances.

Otherwise, I consider this a duplicate of #515. Is this correct?

#2 - 08/29/2012 09:54 AM - R. Haschke

I did not tested with loggers. Because the loggers probably do not unpack the data by default, it's not an issue there.
However, having any application listening on a specific channel and later on adding a new datatype which is send over this channel, will cause the terminate.

This is because the rsb machinery attempts to extract all payloads before dispatching to any handlers. If at this point no suitable converter is available, the code throws an exception, which is not handled somewhere.

I have observed this issue, when sending XOPData, while receiving with rsb_listener from rsb/examples/listener.

This is similar but different to issue #515 because there, the dispatching to a typed handler was the issue, causing a segfault.

#3 - 08/29/2012 01:10 PM - J. Moringen

I see. I think the current behavior of the C++ implementation is printing an exception and aborting processing of the particular event that caused the error. Did you see different behavior?

As a result, I think this is more related to #344 and #1035.

Can you supply the program output/error message and backtrace (if available).

Can you also state the desired behavior?

#4 - 08/30/2012 01:07 PM - R. Haschke

Example output: Exception is thrown but never catched

Listener setup finished. Waiting for messages on scope Scope[/]
terminate called after throwing an instance of 'rsc::runtime::NoSuchObject'
  what(): No converter for wire-schema or data-type `.rst.xml.XOPData'.
Available converters: {.*: **rsb::converter::ByteArrayConverter[wireType = std::string, wireSchema = .**, dataType = bytearray] at 0x9be8810
.rsb.protocol.EventId: *EventIdConverter[wireType = std::string, wireSchema = dummy, dataType = rsb::EventId] at 0x9bea290
bool: *rsb::converter::BoolConverter[wireType = std::string, wireSchema = bool, dataType = bool] at 0x9be93c0
uint64: *rsb::converter::Uint64Converter[wireType = std::string, wireSchema = uint64, dataType = unsigned long long] at 0x9be97d8
utf-8-string: *rsb::converter::StringConverter[wireType = std::string, wireSchema = utf-8-string, dataType = std::string] at 0x9be8d40
void: *rsb::converter::VoidConverter[wireType = std::string, wireSchema = void, dataType = void] at 0x9be8858}

Desired behaviour would be to throw an error message and continue operation.

#5 - 08/30/2012 01:14 PM - J. Wienke

Which transport are you using?

#6 - 04/17/2013 02:54 PM - J. Moringen

- Target version changed from rsb-0.9 to rsb-0.10

#7 - 12/10/2013 11:47 PM - J. Moringen

- Target version changed from rsb-0.10 to rsb-0.11

#8 - 01/20/2014 05:51 PM - J. Moringen

- Category changed from Specification to C++
- Status changed from Feedback to Rejected

The underlying issue seems to be a missing converter. This causes the C++ implementation to abort. For the current architecture and (C++) implementation, this is expected behavior.
For future improvements, see

- Being able to continue despite the error will be enabled by #1035.
- The general topic of dealing with unexpected data types is discussed in #515.