Robotics Service Bus - Bug #2193
Crash in CPP introspection when destructing a participant

03/09/2015 01:37 PM - V. Richter

Status:	Resolved	**Start date:**	03/09/2015
Priority:	Normal	**Due date:**	
Assignee:	J. Wienke	**% Done:**	100%
Category:	Introspection	**Estimated time:**	0.00 hour
Target version:	rsb-0.11		

Description

0x00007fffea74c861 in (anonymous namespace)::handleParticipantDestroyed (participant=0xaa0520) at /media/local_data/jenkins/jobs/rsb-cpp-0.11-toolkit-lsp-csra-nightly/workspace/src/rsb/introspection/Plugin.cpp:122
122 /media/local_data/jenkins/jobs/rsb-cpp-0.11-toolkit-lsp-csra-nightly/workspace/src/rsb/introspection/Plugin.cpp: No such file or directory.
(gdb) bt
#0 0x00007fffea74c861 in (anonymous namespace)::handleParticipantDestroyed (participant=0xaa0520) at /media/local_data/jenkins/jobs/rsb-cpp-0.11-toolkit-lsp-csra-nightly/workspace/src/rsb/introspection/Plugin.cpp:122
#1 0x00007ffff768d321 in operator() (a0=0xaa0520, this=<optimized out>) at /usr/include/boost/function/function_template.hpp:767
#2 operator()<boost::signals::detail::connection_slot_pair> (slot=..., this=<optimized out>) at /usr/include/boost/signals/signal_template.hpp:119
#3 dereference (this=<optimized out>) at /usr/include/boost/signals/detail/slot_call_iterator.hpp:61
#4 dereference<boost::signals::detail::slot_call_iterator<boost::signals::detail::call_bound1<void>::caller<rsb::Participant*, boost::function1<void, rsb::Participant*> >, boost::signals::detail::named_slot_map_iterator> > (f=...)
at /usr/include/boost/iterator/iterator_facade.hpp:514
#5 operator* (this=<optimized out>) at /usr/include/boost/iterator/iterator_facade.hpp:639
#6 postfix_increment_proxy (x=..., this=<synthetic pointer>) at /usr/include/boost/iterator/iterator_facade.hpp:145
#7 operator++<boost::signals::detail::slot_call_iterator<boost::signals::detail::call_bound1<void>::caller<rsb::Participant*, boost::function1<void, rsb::Participant*> >, boost::signals::detail::named_slot_map_iterator>, boost::signals::detail::unusable, boost::single_pass_traversal_tag, const boost::signals::detail::unusable&, long int> (i=...) at /usr/include/boost/iterator/iterator_facade.hpp:728
#8 operator()<boost::signals::detail::slot_call_iterator<boost::signals::detail::call_bound1<void>::caller<rsb::Participant*, boost::function1<void, rsb::Participant*> >, boost::signals::detail::named_slot_map_iterator> > (
this=<optimized out>, last=..., first=...) at /usr/include/boost/last_value.hpp:49
#9 boost::signal1<void, rsb::Participant*, boost::last_value<void>, int, std::less<int>, boost::function1<void, rsb::Participant*> >::operator() (this=<optimized out>, a1=a1@entry=0xaa0520)
at /usr/include/boost/signals/signal_template.hpp:354
#10 0x00007ffff768bc33 in rsb::Participant::~Participant (this=0xaa0520, __vtt_parm=<optimized out>, __in_chrg=<optimized out>) at /media/local_data/jenkins/jobs/rsb-cpp-0.11-toolkit-lsp-csra-nightly/workspace/src/rsb/Participant.cpp:56
#11 0x00007ffff768546d in rsb::Listener::~Listener (this=0xaa0520, __in_chrg=<optimized out>, __vtt_parm=<optimized out>) at /media/local_data/jenkins/jobs/rsb-cpp-0.11-toolkit-lsp-csra-nightly/workspace/src/rsb/Listener.cpp:55
#12 0x00007ffff7685539 in rsb::Listener::~Listener (this=0xaa0520, __in_chrg=<optimized out>, __vtt_parm=<optimized out>) at /media/local_data/jenkins/jobs/rsb-cpp-0.11-toolkit-lsp-csra-nightly/workspace/src/rsb/Listener.cpp:56
#13 0x0000000000447ede in boost::detail::sp_counted_base::release (this=0xa8e1c0) at /usr/include/boost/smart_ptr/detail/sp_counted_base_gcc_x86.hpp:146
#14 0x00007ffff76c3ad5 in ~shared_count (this=<optimized out>, __in_chrg=<optimized out>) at /usr/include/boost/smart_ptr/detail/shared_count.hpp:371
#15 ~shared_ptr (this=<optimized out>, __in_chrg=<optimized out>) at /usr/include/boost/smart_ptr/shared_ptr.hpp:328
#16 reset (this=0xa9ffe0) at /usr/include/boost/smart_ptr/shared_ptr.hpp:619
#17 rsb::patterns::Method::deactivate (this=0xa9fd20) at /media/local_data/jenkins/jobs/rsb-cpp-0.11-toolkit-lsp-csra-nightly/workspace/src/rsb/patterns/Server.cpp:52
#18 0x00007ffff76caa39 in rsb::patterns::LocalServer::~LocalServer (this=0xaa6f00, __in_chrg=<optimized out>,

__vtt_parm=<optimized out>)
at /media/local_data/jenkins/jobs/rsb-cpp-0.11-toolkit-lsp-csra-nightly/workspace/src/rsb/patterns/LocalServer.cpp:140
#19 0x00007ffff76cab49 in rsb::patterns::LocalServer::~LocalServer (this=0xaa6f00, __in_chrg=<optimized out>,
__vtt_parm=<optimized out>)
at /media/local_data/jenkins/jobs/rsb-cpp-0.11-toolkit-lsp-csra-nightly/workspace/src/rsb/patterns/LocalServer.cpp:142
#20 0x0000000000447ede in boost::detail::sp_counted_base::release (this=0xaa4070) at
/usr/include/boost/smart_ptr/detail/sp_counted_base_gcc_x86.hpp:146
#21 0x00000000004489c8 in ~shared_count (this=0xaa5dd8, __in_chrg=<optimized out>) at
/usr/include/boost/smart_ptr/detail/shared_count.hpp:371
#22 ~shared_ptr (this=0xaa5dd0, __in_chrg=<optimized out>) at /usr/include/boost/smart_ptr/shared_ptr.hpp:328
#23 ~RsbSink (this=0xaa5d90, __in_chrg=<optimized out>) at
/media/local_data/jenkins/jobs/sitinf-b0.4-toolkit-lsp-csra-nightly/workspace/src/io/RsbSink.h:110
#24 checked_delete<sitinf::io::RsbSink<rst::classification::Situation> > (x=0xaa5d90) at
/usr/include/boost/checked_delete.hpp:34
#25 boost::detail::sp_counted_impl_p<sitinf::io::RsbSink<rst::classification::Situation> >::dispose (this=<optimized out>) at
/usr/include/boost/smart_ptr/detail/sp_counted_impl.hpp:78
#26 0x0000000000447ede in boost::detail::sp_counted_base::release (this=0xaa15b0) at
/usr/include/boost/smart_ptr/detail/sp_counted_base_gcc_x86.hpp:146
#27 0x00007ffff5823259 in ?? () from /lib/x86_64-linux-gnu/libc.so.6
#28 0x00007ffff58232a5 in exit () from /lib/x86_64-linux-gnu/libc.so.6
#29 0x00007ffff5808ecc in __libc_start_main () from /lib/x86_64-linux-gnu/libc.so.6
#30 0x000000000043f227 in _start ()

Associated revisions

Revision 51f8506d - 03/09/2015 02:00 PM - J. Wienke

Prevent printing uninitialized memory in introspection

Prevent printing out memory which might not be initialized yet or might
have been destructed already. Since concrete subclasses of Participant
might not have been initialized yet or are already destructed when the
introspection is notified about them, we must avoid using the print
methods of the concrete subclasses, which would access invalid memory.

refs #2193

(cherry picked from commit 6b0f39164f7c3897759bbc8a2b261d99921102a0)

Revision 773bbc17 - 03/09/2015 02:00 PM - J. Wienke

Prevent printing uninitialized memory in introspection

Prevent printing out memory which might not be initialized yet or might
have been destructed already. Since concrete subclasses of Participant
might not have been initialized yet or are already destructed when the
introspection is notified about them, we must avoid using the print
methods of the concrete subclasses, which would access invalid memory.

refs #2193

Revision 493f80cc - 03/27/2015 06:24 PM - J. Moringen

fixes #2193: Improved introspection destruction

History

#1 - 03/09/2015 01:45 PM - J. Wienke

This line points to the logger debug statement. I suspect this is caused by printing out the whole participant in this logging statement. Since concrete participant subclasses are already destructed at the point in time where the participant is being printed, their print methods will access invalid memory.

#2 - 03/09/2015 01:45 PM - J. Wienke

- Status changed from New to In Progress

- Assignee changed from J. Moringen to J. Wienke

#3 - 03/09/2015 01:57 PM - J. Wienke

Maybe this is not the explanation. Listener does not implement printContents. Therefore only the print method of Participant itself exists and this one won't access memory of the Listener instance. Still, we should ensure in the plugin that this can never happen.

#4 - 03/09/2015 02:00 PM - J. Wienke

- % Done changed from 0 to 10

Viktor, can you try again after these fixes whether the problem persists? I suspect this isn't fixed, but I am also not sure about the origin now.

#5 - 03/10/2015 12:42 PM - V. Richter

Tested. No it does not seem to solve it (assuming I rebuilt the right components).

It seems to come from deleting Participants after the corresponing Factory is out of scope. The following code reproduces the error:

```
#include <rsb/Factory.h>

rsb::patterns::RemoteServerPtr server;

int main(int n, char**ppc){
  server = rsb::getFactory().createRemoteServer("/");
  return 0;
}
```

Is it intended that the Factory **must** be held all the time?

#6 - 03/10/2015 01:05 PM - V. Richter

V. Richter wrote:

> *Tested. No it does not seem to solve it (assuming I rebuilt the right components).*

> *It seems to come from deleting Participants after the corresponing Factory is out of scope. The following code reproduces the error:*
>
> *[...]*
>
> *Is it intended that the Factory **must** be held all the time?*

Looking into my environment configuration I found that the following is needed in the environment:

```
export RSB_PLUGINS_CPP_LOAD=rsbintrospection
```

#7 - 03/19/2015 11:11 PM - J. Wienke

V. Richter wrote:

> *Tested. No it does not seem to solve it (assuming I rebuilt the right components).*
>
> *It seems to come from deleting Participants after the corresponing Factory is out of scope. The following code reproduces the error:*
>
> *[...]*
>
> *Is it intended that the Factory **must** be held all the time?*

This shouldn't be the case. The factory is statically maintained inside the RSB code and one and only one instance should exist all the time. You only receive a reference to that instance in the getFactory call.

#8 - 03/19/2015 11:13 PM - J. Wienke

V. Richter wrote:

> *Looking into my environment configuration I found that the following is needed in the environment:*

This is always needed for using the introspection. Without the plugin, introspection simply isn't available. So I assume it only crashes in case the introspection module is loaded?

#9 - 03/20/2015 09:34 AM - V. Richter

J. Wienke wrote:

> *V. Richter wrote:*
>
> > *Looking into my environment configuration I found that the following is needed in the environment:*
>
> *This is always needed for using the introspection. Without the plugin, introspection simply isn't available. So I assume it only crashes in case the introspection module is loaded?*

Yes. The code snippet crashes with the environment variable set and does not without setting the variable.

#10 - 03/20/2015 11:45 AM - J. Wienke

Ok, I can reproduce the issue.

#11 - 03/20/2015 11:58 AM - J. Wienke

Ouch, this is a problem of static deconstruction order. The logging factory is already destroyed at the time the participant is going to be deconstructed. Therefore, logging does not work anymore.

#12 - 03/20/2015 12:14 PM - J. Wienke

I have no idea how to solve this. We would have to eliminate all references to static variables inside the introspection plugin to be sure that such a thing doesn't happen. But that is basically impossible. Even the introspection sender instance of the plugin could already be destroyed at the time the participant is going to be destructed.

Any ideas Jan?

#13 - 03/27/2015 06:30 PM - J. Moringen

- Status changed from In Progress to Resolved
- % Done changed from 10 to 100

Applied in changeset commit:rsb-cpp|493f80cc77517c7b7966eac37e454a7a26df3398.