コア - バグ #3438

hrpsys-base の rtm.py::reeadDataPort() を呼ぶと disconnect() で稀に落ちる

2016/01/29 04:53 - n-ando

ステータス:	新規	**開始日:**	2016/01/29
優先度:	通常	**期日:**	
担当者:		**進捗率:**	90%
カテゴリ:		**予定工数:**	0.00時間
対象バージョン:			

説明

https://github.com/fkanehiro/hrpsys-base/issues/905

関係しているリビジョン

リビジョン 2719 - 2016/05/13 10:14 - n-ando

Now data value is stored to PortProfile.properties. refs #3438

リビジョン 2719 - 2016/05/13 10:14 - n-ando

Now data value is stored to PortProfile.properties. refs #3438

履歴

#1 - 2016/01/29 04:54 - n-ando

1回は OutPortCorbaCdrProvider.cpp のデストラクタで落ちていることを確認。
logger の呼び出しが最後

#2 - 2016/01/29 09:40 - n-ando

- 進捗率 を 0 から 10 に変更

おそらくここ。OutPortCorbaCdrProvider のデストラクタ内で死んでいる。

```
(gdb) where
#0  do_lookup_x (new_hash=new_hash@entry=1732989254, old_hash=old_hash@entry=0x7ffff26af0c0,
    result=result@entry=0x7ffff26af0d0, scope=<optimized out>, i=i@entry=0, flags=flags@entry=5,
    skip=skip@entry=0x0, undef_map=undef_map@entry=0x7ffff7ff7a08) at dl-lookup.c:228
#1  0x00007ffff7de4991 in _dl_lookup_symbol_x (
    undef_name=0x7ffff6fd3470 "_ZN7omniORB6loggerC1EPKc", undef_map=0x7ffff7ff7a08,
    ref=ref@entry=0x7ffff26af188, symbol_scope=0x7ffff7ff7d60, version=0x0,
    type_class=type_class@entry=1, flags=5, skip_map=skip_map@entry=0x0) at dl-lookup.c:737
#2  0x00007ffff7de9557 in _dl_fixup (l=<optimized out>, reloc_arg=<optimized out>)
    at ../elf/dl-runtime.c:111
#3  0x00007ffff7df0515 in _dl_runtime_resolve () at ../sysdeps/x86_64/dl-trampoline.S:45
#4  0x00007ffff7087f3a in omniServant::~omniServant() () from /usr/lib/libomniORB4.so.1
#5  0x00007ffff7a6e5bd in RTC::OutPortCorbaCdrProvider::~OutPortCorbaCdrProvider (
    this=0x7fffe0006050, __in_chrg=<optimized out>, __vtt_parm=<optimized out>)
    at OutPortCorbaCdrProvider.cpp:68
#6  0x00007ffff7a6eab8 in RTC::OutPortCorbaCdrProvider::~OutPortCorbaCdrProvider (
    this=0x7fffe0006050, __in_chrg=<optimized out>, __vtt_parm=<optimized out>)
    at OutPortCorbaCdrProvider.cpp:100
#7  0x00007ffff7a6f952 in coil::Destructor<RTC::OutPortProvider, RTC::OutPortCorbaCdrProvider> (
    obj=@0x7fffe000bd18: 0x7fffe0006050) at ../../../src/lib/coil/include/coil/Factory.h:80
#8  0x00007ffff7a2b801 in coil::Factory<RTC::OutPortProvider, std::string, std::less<std::string>, RTC::OutPortProvider* (*)(), void (*)(RTC::Out
tProvider*&)>::deleteObject (this=0x69d100,
    obj=@0x7fffe000bd18: 0x7fffe0006050) at ../../../src/lib/coil/include/coil/Factory.h:343
#9  0x00007ffff7a2b319 in RTC::OutPortPullConnector::disconnect (this=0x7fffe000b8d0)
    at OutPortPullConnector.cpp:99
#10 0x00007ffff7a2b18f in RTC::OutPortPullConnector::~OutPortPullConnector (this=0x7fffe000b8d0,
    __in_chrg=<optimized out>) at OutPortPullConnector.cpp:68
#11 0x00007ffff7a2b1ec in RTC::OutPortPullConnector::~OutPortPullConnector (this=0x7fffe000b8d0,
    __in_chrg=<optimized out>) at OutPortPullConnector.cpp:69
#12 0x00007ffff7a4feec in RTC::OutPortBase::unsubscribeInterfaces (this=0x6a7b20,
    connector_profile=...) at OutPortBase.cpp:691
#13 0x00007ffff7a5ad12 in RTC::PortBase::notify_disconnect (this=0x6a7b20,
    connector_id=0x7fffd4000b10 "4446e7af-045d-43d6-a451-8a9178b987c3") at PortBase.cpp:422
#14 0x00007ffff7a8e737 in _0RL_lcfn_bf82f9885dac07a6_b4000000(omniCallDescriptor*, omniServant*) ()
```

```
    from /home/n-ando/tmp/OpenRTM-aist-1.1.1/src/lib/rtm/.libs/libRTC-1.1.1.so
#15 0x00007ffff7092ae3 in omni::omniOrbPOA::dispatch(omniCallDescriptor&, omniLocalIdentity*) ()
    from /usr/lib/libomniORB4.so.1
#16 0x00007ffff7077de6 in omniLocalIdentity::dispatch(omniCallDescriptor&) ()
    from /usr/lib/libomniORB4.so.1
#17 0x00007ffff70868d5 in omniObjRef::_invoke(omniCallDescriptor&, bool) ()
    from /usr/lib/libomniORB4.so.1
#18 0x00007ffff7a9de79 in RTC::_objref_PortService::notify_disconnect(char const*) ()
    from /home/n-ando/tmp/OpenRTM-aist-1.1.1/src/lib/rtm/.libs/libRTC-1.1.1.so
#19 0x00007ffff7a5a6ae in RTC::PortBase::disconnect (this=0x6a7b20,
    connector_id=0x7fffd4000b10 "4446e7af-045d-43d6-a451-8a9178b987c3") at PortBase.cpp:368
#20 0x00007ffff7a8e6a7 in _0RL_lcfn_bf82f9885dac07a6_84000000(omniCallDescriptor*, omniServant*) ()
    from /home/n-ando/tmp/OpenRTM-aist-1.1.1/src/lib/rtm/.libs/libRTC-1.1.1.so
#21 0x00007ffff70a1361 in omniCallHandle::upcall(omniServant*, omniCallDescriptor&) ()
    from /usr/lib/libomniORB4.so.1
```

この時のスレッドの状態。

```
(gdb) info thread
  Id   Target Id         Frame
* 16    Thread 0x7ffff26b0700 (LWP 17597) "lt-SeqOutComp" do_lookup_x (
    new_hash=new_hash@entry=1732989254, old_hash=old_hash@entry=0x7ffff26af0c0,
    result=result@entry=0x7ffff26af0d0, scope=<optimized out>, i=i@entry=0, flags=flags@entry=5,
    skip=skip@entry=0x0, undef_map=undef_map@entry=0x7ffff7ff7a08) at dl-lookup.c:228
  7     Thread 0x7ffff2eb1700 (LWP 24415) "lt-SeqOutComp" 0x00007ffff7093e6e in omni::omniOrbPOA::lastInvocationHasCompleted(o
alIdentity*) () from /usr/lib/libomniORB4.so.1
  6     Thread 0x7ffff36b2700 (LWP 24395) "lt-SeqOutComp" 0x00007ffff73637eb in __libc_recv (fd=12,
    buf=0x7fffdc000a98, n=8192, flags=-1) at ../sysdeps/unix/sysv/linux/x86_64/recv.c:33
  5     Thread 0x7ffff3eb3700 (LWP 24391) "lt-SeqOutComp" 0x00007ffff6092f3d in nanosleep ()
    at ../sysdeps/unix/syscall-template.S:81
  4     Thread 0x7ffff48c0700 (LWP 24385) "lt-SeqOutComp" pthread_cond_timedwait@@GLIBC_2.3.2 ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/pthread_cond_timedwait.S:238
  3     Thread 0x7ffff50c1700 (LWP 24384) "lt-SeqOutComp" 0x00007ffff60bf12d in poll ()
    at ../sysdeps/unix/syscall-template.S:81
  2     Thread 0x7ffff58c2700 (LWP 24383) "lt-SeqOutComp" 0x00007ffff60c3da3 in select ()
    at ../sysdeps/unix/syscall-template.S:81
  1     Thread 0x7ffff7fc97c0 (LWP 24379) "lt-SeqOutComp" pthread_cond_wait@@GLIBC_2.3.2 ()
    at ../nptl/sysdeps/unix/sysv/linux/x86_64/pthread_cond_wait.S:185
```

#3 - 2016/01/29 10:08 - n-ando

- 進捗率 を 10 から 20 に変更

omniORBのトレースメッセージを取ってみた。デストラクタ付近の関数呼び出しと並べてみると。

```
~OutPortCorbaCdrProvider(void)
PortableServer::ObjectId_var oid;
oid = _default_POA()->servant_to_id(this);
_default_POA()->deactivate_object(oid);
>> omniORB: State root<2133> (active) -> deactivating
>> omniORB: Object is still busy -- etherealise later. <- deactivateに失敗して後回しにされている。
＃デストラクタ抜ける
>> omniORB: ObjRef(IDL:OpenRTM/OutPortCdr:1.0) -- deleted.
>> omniORB: ERROR -- A servant has been deleted that is still activated.
>>            id: root<2133> (deactivating)
>> omniORB: sendChunk: to giop:tcp:[::ffff:10.211.55.4]:39503 28 bytes
>> omniORB:
>> 4749 4f50 0102 0101 1000 0000 b661 0000 GIOP.........a..
>> 0000 0000 0000 0000 0000 0000                    ............
>> omniORB: POA(RootPOA) etherealising object root<2133> (deactivating).
>>   id:
>> omniORB: Assertion failed.  This indicates a bug in the application
>> using omniORB, or maybe in omniORB itself.
>>   file: ../../../../../src/lib/omniORB/orbcore/omniServant.cc
>>   line: 223
>>   info: activation_found
>> omniORB: WARNING -- method 'get' raised an unexpected
>> exception (not a CORBA exception).
```

以下、その付近の全メッセージ。

```
omniORB: giopWorker task execute.
omniORB: inputMessage: from giop:tcp:[::ffff:10.211.55.4]:39503 105 bytes
omniORB:
4749 4f50 0102 0100 5d00 0000 b661 0000 GIOP....]....a..
```

```
0300 0000 0000 0000 0e00 0000 fedf b7aa ................
5600 003e 7000 0000 0002 d900 0b00 0000 V..>p...........
6469 7363 6f6e 6e65 6374 0000 0000 0000 disconnect.......
2500 0000 6264 6461 3336 3566 2d34 3939 %...bdda365f-499
622d 3465 3336 2d62 3437 342d 3139 3330 b-4e36-b474-1930
3239 3563 6131 3431 00                  295ca141.
1
2
3
4
omniORB: State root<2133> (active) -> deactivating
omniORB: Object is still busy -- etherealise later.
5
omniORB: ObjRef(IDL:OpenRTM/OutPortCdr:1.0) -- deleted.
omniORB: ERROR -- A servant has been deleted that is still activated.
          id: root<2133> (deactivating)
omniORB: sendChunk: to giop:tcp:[::ffff:10.211.55.4]:39503 28 bytes
omniORB:
4749 4f50 0102 0101 1000 0000 b661 0000 GIOP.........a..
0000 0000 0000 0000 0000 0000           ............
omniORB: POA(RootPOA) etherealising object root<2133> (deactivating).
  id:
omniORB: Assertion failed. This indicates a bug in the application
using omniORB, or maybe in omniORB itself.
  file: ../../../../../src/lib/omniORB/orbcore/omniServant.cc
  line: 223
  info: activation_found
omniORB: WARNING -- method 'get' raised an unexpected
  exception (not a CORBA exception).
omniORB: sendChunk: to giop:tcp:[::ffff:10.211.55.4]:39503 68 bytes
omniORB:
4749 4f50 0102 0101 3800 0000 b461 0000 GIOP....8....a..
0200 0000 0000 0000 1e00 0000 4944 4c3a ............IDL:
6f6d 672e 6f72 672f 434f 5242 412f 554e omg.org/CORBA/UN
4b4e 4f57 4e3a 312e 3000 3939 0100 4d4f KNOWN:1.0.99..MO
0100 0000                               ....
omniORB: inputMessage: from giop:tcp:[::ffff:10.211.55.4]:39503 72 bytes
omniORB:
4749 4f50 0102 0100 3c00 0000 b861 0000 GIOP....<....a..
0300 0000 0000 0000 0e00 0000 fedf b7aa ................
5600 003e 7000 0000 0003 d900 1100 0000 V..>p...........
6765 745f 706f 7274 5f70 726f 6669 6c65 get_port_profile
0000 0000 0000 0000                     ........
omniORB: SocketCollection idle. Sleeping.
omniORB: inputMessage: from giop:tcp:[::ffff:10.211.55.3]:31380 110 bytes
omniORB:
4749 4f50 0102 0000 0000 0062 0000 0311 GIOP.......b....
0300 0000 0000 0000 0000 000e fedf b7aa ................
5600 003e 7000 0000 0000 0000 0000 000e V..>p...........
5f6e 6f6e 5f65 7869 7374 656e 7400 0000 _non_existent...
0000 0003 0000 0001 0000 000c 0000 0000 ................
0001 0001 0001 0109 0000 0011 0000 0002 ................
0002 0000 4e45 4f00 0000 0002 0014      ....NEO.........
```

#4 - 2016/01/29 10:11 - n-ando

- 進捗率 を 20 から 30 に変更

omniORB: Object is still busy -- etherealise later.でググってみると、以下の記事がヒット。
但しちょっと古い。

deactivate_object() に加えて _remove_ref() も呼ばないとだめか？

"Object is still busy -- etherealise later."
The POA is still using one of your servants...

"ERROR -- A servant has been deleted that is still activated."
But you seem to have deleted it!!

This sounds like your servant lifecycle design is incorrect. The POA
must be finished with an active servant before it can be deleted. It is
very hard to know when the POA is finished with a servant - you can call
`deactivate_object()' - but the POA can still keep hold of the servant
for an arbitary time (until ongoing calls are complete). Henning &
Vinoski section 11.9 (pp499-502) is the best reference I know on this.

Solutions:

1. RefCountServantBase

The approach I always use is to use the mix-in class `RefCountServantBase'. This implements a simple reference counting mechanism. RefCountServantBase::_remove_ref() deletes the object when the ref count falls to zero. The POA is refcount-aware, so it calls _add_ref() & _remove_ref() as required.

When you are finished with a servant you need to do two things (in either order):

* call deactivate_object() in the object's POA. See this file for an example of code to call deactivate_object() for you:

http://cvs.sf.net/viewcvs.py/omniifr/omniifr/repository/IRObject.cc?rev=1.1&view=auto

* call _remove_ref() to release your own reference to the servant. (The newly contructed object always has a ref count of '1'.) You can do this as soon as the obect is activated, if you don't want to refer to the servant except though the POA.

2. POA deactivate

Your problem seems to occur when you are deactivating a POA, or perhaps shutting down the ORB.

If you make sure that the POA is completely deactivated BEFORE you delete any servant objects, then you should be safe. Call POA:: (or ORB::) deactivate() with the `wait_for_completion' parameter set to TRUE. Only then should you delete your servant objects.

3. Servant Activator

You can install a servant activator in your POA. The POA will then call incarnate() when it wants a new servant, and etherealize() when the servant can be deleted.

See OmniEvents::ProxyPullSupplierManager for an example of this approach.

Let me know how you get on.

-Alex

#5 - 2016/01/29 10:37 - n-ando

_remove_ref() を追加すると、常時以下のようなメッセージが表示されるようになった。

```
oid = _default_POA()->servant_to_id(this);
>> omniORB: Adding root<14974> (activating) to object table.
>> omniORB: State root<14974> (activating) -> active
>> omniORB: Application check failed. This indicates a bug in the application
>>    using omniORB.    See the comment in the source code for more info.
>>    file: ../../../../../src/lib/omniORB/orbcore/portableserver.cc
>>    line: 273
>>    info: _pd_refCount > 0
_default_POA()->deactivate_object(oid);
this->_remove_ref();
>> omniORB: State root<14974> (active) -> deactivating
>> omniORB: POA(RootPOA) etherealising object root<14974> (deactivating).
>> id: IDL:OpenRTM/OutPortCdr:1.0
>> omniORB: State root<14974> (deactivating) -> etherealising
>> omniORB: Removing root<14974> (etherealising) from object table
>> omniORB: Object table entry root<14974> (dead) deleted.
>> omniORB: ServantBase has zero ref count -- deleted.
```

portableserver.cc の273行目を見てみる。

http://sourceforge.net/p/omniorb/svn/HEAD/tree/tags/4_1_6/omniORB/src/lib/omniORB/orbcore/portableserver.cc#l273

```
void
PortableServer::ServantBase::_add_ref()
```

```
{
    omni_tracedmutex_lock l(ref_count_lock);
    // If the reference count is 0, then the object is either in the
    // process of being deleted by _remove_ref, or has already been
    // deleted. It is too late to be trying to _add_ref now. If the
    // reference count is less than zero, then _remove_ref has been
    // called too many times.
    OMNIORB_USER_CHECK(_pd_refCount > 0);

    _pd_refCount++;
}
void
PortableServer::ServantBase::_remove_ref()
{
    ref_count_lock.lock();
    int done = --_pd_refCount > 0;
    ref_count_lock.unlock();
    if( done ) return;

    if( _pd_refCount < 0 ) {
        omniORB::logs(1, "ServantBase has negative ref count!");
        return;
    }

    omniORB::logs(15, "ServantBase has zero ref count -- deleted.");

    try {
        delete this;
    }
    catch (...) {
        omniORB::logs(1, "ERROR: Servant destructor threw an exception.");
    }
}
```

しかし、4.2.0 以降では、だいぶ変わっている模様。

```
void
PortableServer::ServantBase::_add_ref()
{
    _pd_refCount.inc();
}
void
PortableServer::ServantBase::_remove_ref()
{
    int val = _pd_refCount.dec();

    if (val > 0)
        return;

    if (val < 0) {
        omniORB::logs(1, "ServantBase has negative ref count!");
        return;
    }

    omniORB::logs(15, "ServantBase has zero ref count -- deleted.");

    try {
        delete this;
    }
    catch (...) {
        omniORB::logs(1, "Error: Servant destructor threw an exception.");
    }
}
```

#6 - 2016/01/29 10:43 - n-ando

4.1.6 の _add_ref() 内の _pd_refCount にはロックかかってないのでは？
4.2.0 の方では _pd_refCount.inc(), _pd_refCount.dec() に置き換わっている。このオブジェクトは omnithread/atomic.h で定義されていて、Linuxでは gcc の __sync_add_and_fetch などを使って以下のように実装されている。

```
class _OMNITHREAD_NTDLL_ omni_refcount {
public:
    inline omni_refcount(int start) : count(start) {}
    inline ~omni_refcount() {}
```

```
    // Atomically increment reference count and return new value
    inline int inc() {
        return __sync_add_and_fetch(&count, 1);
    }

    // Atomically decrement reference count and return new value
    inline int dec() {
        return __sync_sub_and_fetch(&count, 1);
    }

    // Return snapshot of current value. Real count may have changed by
    // the time the value is looked at!
    inline int value() {
        return count;
    }

private:
    volatile int count;
};
```

ちなみに、Windowsでは、

```
class _OMNITHREAD_NTDLL_ omni_refcount {
public:
    inline omni_refcount(int start) : count(start) {}
    inline ~omni_refcount() {}

    // Atomically increment reference count and return new value
    inline int inc() {
        return InterlockedIncrement(&count);
    }

    // Atomically decrement reference count and return new value
    inline int dec() {
        return InterlockedDecrement(&count);
    }

    // Return snapshot of current value. Real count may have changed by
    // the time the value is looked at!
    inline int value() {
        return count;
    }

private:
    volatile LONG count;
};
```

でも、gcc と VC 以外は？

#7 - 2016/01/29 12:19 - n-ando

試しにomniORB-4.2.0 で同じことをやってみる。

```
n-ando@ubuntu1404:~/tmp/test42/RELENG_1_1/OpenRTM-aist/examples/SeqIO$ export LD_LIBRARY_PATH=/usr/local/lib
n-ando@ubuntu1404:~/tmp/test42/RELENG_1_1/OpenRTM-aist/examples/SeqIO$ ldd .libs/lt-SeqOutComp
        linux-vdso.so.1 =>  (0x00007fff07ff8000)
        libRTC-1.1.2.so => /home/n-ando/tmp/test42/RELENG_1_1/OpenRTM-aist/src/lib/rtm/.libs/libRTC-1.1.2.so (0x00007f32cbfee000)
        libcoil-1.1.2.so => /home/n-ando/tmp/test42/RELENG_1_1/OpenRTM-aist/src/lib/coil/lib/.libs/libcoil-1.1.2.so (0x00007f32cbdd6000
        libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f32cbb91000)
        libomniORB4.so.2 => /usr/local/lib/libomniORB4.so.2 (0x00007f32cb7c8000)
        libomnithread.so.4 => /usr/local/lib/libomnithread.so.4 (0x00007f32cb5c1000)
        libomniDynamic4.so.2 => /usr/local/lib/libomniDynamic4.so.2 (0x00007f32cb0a4000)
        libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007f32cada0000)
        libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007f32cab89000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f32ca7c4000)
        libuuid.so.1 => /lib/x86_64-linux-gnu/libuuid.so.1 (0x00007f32ca5bf000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f32ca3ba000)
        libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f32ca0b4000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f32cc518000)
```

4.2.0 は /usr/local 以下にインストール。ソースはRELENG_1_1の最新（4.2対応がされている）

が、やっぱり omniORB: Object is still busy -- etherealise later. がでる。

1

```
2
3
4
omniORB: State root<6458> (active) -> deactivating
omniORB: Object is still busy -- etherealise later.
5
omniORB: ObjRef(IDL:OpenRTM/OutPortCdr:1.0) -- deleted.
omniORB: Error: a servant has been deleted that is still activated.
         id: root<6458> (deactivating)
omniORB: ObjRef(IDL:omg.org/RTC/PortService:1.0) -- deleted.
omniORB: sendChunk: to giop:tcp:[::ffff:10.211.55.4]:47281 28 bytes
omniORB:
4749 4f50 0102 0101 1000 0000 7828 0100 GIOP.....x(..
0000 0000 0000 0000 0000 0000                    ............
omniORB: POA(RootPOA) etherealising object root<6458> (deactivating).
  id:
omniORB: Assertion failed.  This indicates a bug in the application
using omniORB, or maybe in omniORB itself.
  file: omniServant.cc
  line: 168
  info: activation_found
omniORB: Warning: method 'get' raised an unexpected exception (not a CORBA exception).
omniORB: sendChunk: to giop:tcp:[::ffff:10.211.55.4]:47281 68 bytes
omniORB:
```

ちなみに ~OutPortCorbaCdrProvider() に _remove_ref() を入れると、そもそもreadDataPort から値が返ってこない。

```
omniORB: State root<14975> (active) -> deactivating
omniORB: POA(RootPOA) etherealising object root<14975> (deactivating).
  id: IDL:OpenRTM/OutPortCdr:1.0
omniORB: State root<14975> (deactivating) -> etherealising
omniORB: Removing root<14975> (etherealising) from object table
omniORB: Object table entry root<14975> (dead) deleted.
omniORB: ServantBase has zero ref count -- deleted.
1
2
3

Program received signal SIGSEGV, Segmentation fault.
[Switching to Thread 0x7fffee7fc700 (LWP 4124)]
0x00007ffff612d65a in __GI__IO_default_xsputn (f=0x7fffedffd600, data=0x7ffff7221ef7, n=0)
        at genops.c:447
447         genops.c: そのようなファイルやディレクトリはありません.
```

bt結果。そもそも、_remove_ref() はデストラクタで呼ぶべきではない？

```
#0  0x00007ffff612d65a in __GI__IO_default_xsputn (f=0x7fffedffd600, data=0x7ffff7221ef7, n=0)
      at genops.c:447
#1  0x00007ffff60fbdc0 in _IO_vfprintf_internal (s=s@entry=0x7fffedffd600,
      format=format@entry=0x7ffff7221ef7 "%lu", ap=ap@entry=0x7fffedffd738) at vfprintf.c:1340
#2  0x00007ffff61bb0f4 in ___vsprintf_chk (s=0x7fffdc010fe5 "", flags=1, slen=18446744073709551615,
      format=0x7ffff7221ef7 "%lu", args=args@entry=0x7fffedffd738) at vsprintf_chk.c:84
#3  0x00007ffff61bb04d in ___sprintf_chk (s=<optimized out>, flags=<optimized out>,
      slen=<optimized out>, format=<optimized out>) at sprintf_chk.c:31
#4  0x00007ffff718c69b in pp_key(omniORB::logger&, unsigned char const*, int) ()
    from /usr/local/lib/libomniORB4.so.2
#5  0x00007ffff718c754 in omniORB::logger::operator<<(omniLocalIdentity const*) ()
    from /usr/local/lib/libomniORB4.so.2
#6  0x00007ffff719237b in omniObjTable::newEntry(omniObjKey&, unsigned int) ()
    from /usr/local/lib/libomniORB4.so.2
#7  0x00007ffff71a8d09 in omni::omniOrbPOA::servant_to_id(PortableServer::ServantBase*) ()
    from /usr/local/lib/libomniORB4.so.2
#8  0x00007ffff7aace9a in RTC::OutPortCorbaCdrProvider::~OutPortCorbaCdrProvider() ()
```

#8 - 2016/01/29 13:37 - n-ando

- 進捗率 を 30 から 40 に変更

試しに、deactivate_object() をデストラクタの前に持ってきてみる。
ProviderのFactoryのcoil::Destructorを以下のようにMyDestuctorに置き換え。

```
extern "C"
{
    void MyDestructor(::RTC::OutPortProvider*& obj)
```

```
  {
      if (obj == 0) { return; }
      ::RTC::OutPortCorbaCdrProvider* tmp = dynamic_cast< ::RTC::OutPortCorbaCdrProvider* >(obj);
      if (tmp == 0) { return; }
      try
          {
              PortableServer::ObjectId_var oid;
              std::cout << "1" << std::endl;
              oid = tmp->_default_POA()->servant_to_id(tmp);
              std::cout << "2" << std::endl;
              tmp->_default_POA()->deactivate_object(oid);
          }
      catch (...)
          {
              std::cout << "Exception in deactivate_object()" << std::endl;
          }
      std::cout << "3" << std::endl;
      delete obj;
      std::cout << "4" << std::endl;
      obj = 0;
      std::cout << "5" << std::endl;
  }
  /*!
   * @if jp
   * @brief モジュール初期化関数
   * @else
   * @brief Module initialization
   * @endif
   */
  void OutPortCorbaCdrProviderInit(void)
  {
      RTC::OutPortProviderFactory&
          factory(RTC::OutPortProviderFactory::instance());
      factory.addFactory("corba_cdr",
                                              ::coil::Creator< ::RTC::OutPortProvider,
                                                                              ::RTC::OutPortCorbaCdrProvider>,
                                              MyDestructor);
  }
```

結果は変わらず。
ServantActivatorを導入しなければいけないのか。。。。

#9 - 2016/02/03 22:35 - n-ando

CORBA Objectがアクティブなのに、deactivateしservantをdelete仕様としている
→Servant使用中はdeleteできないようにロックをかけるべき？
→しかし、readDataPortの呼び出しシーケンス的には

- connect
- notify_connect
- get
- disconnect
- deactivate_servant
- delete

の順番のはず。
でも、いちおうgetとdtorが同時に呼ばれる可能性はあるので、ロックをかけてみる。

```
  OutPortCorbaCdrProvider::OutPortCorbaCdrProvider(void)
    : m_buffer(0)
  {
      Guard guard(m_mutex);

:
  OutPortCorbaCdrProvider::~OutPortCorbaCdrProvider(void)
  {
      Guard guard(m_mutex);
              std::cout << "1" << std::endl;
      try
          {
:

  ::OpenRTM::PortStatus
  OutPortCorbaCdrProvider::get(::OpenRTM::CdrData_out data)
      throw (CORBA::SystemException)
```

```
    {
        Guard guard(m_mutex);
        RTC_PARANOID(("OutPortCorbaCdrProvider::get()"));
```

のようにロックをかけてテスト→同様にactivation_foundのアサーションがでる。
ただ、deactivate_object の操作の後に、omniORB: WARNING -- method 'get' raised an unexpected が出ている？？？

```
1
2
3
4
omniORB: State root<223> (active) -> deactivating
omniORB: Object is still busy -- etherealise later.
5
omniORB: ObjRef(IDL:OpenRTM/OutPortCdr:1.0) -- deleted.
omniORB: ERROR -- A servant has been deleted that is still activated.
          id: root<223> (deactivating)
omniORB: POA(RootPOA) etherealising object root<223> (deactivating).
  id:
omniORB: Assertion failed.  This indicates a bug in the application
using omniORB, or maybe in omniORB itself.
  file: ../../../../../src/lib/omniORB/orbcore/omniServant.cc
  line: 223
  info: activation_found
omniORB: WARNING -- method 'get' raised an unexpected
  exception (not a CORBA exception).
omniORB: sendChunk: to giop:tcp:[::ffff:10.211.55.4]:44703 68 bytes
omniORB:
```

#10 - 2016/05/13 10:14 - n-ando

- 進捗率 を 40 から 90 に変更

CORBA内部のエラーのようなので、とりあえず調査はここで終了。
また、rtm.pyのワンショットデータ読み込みは効率が悪いので、PortProfile.properties
のデータを格納しておき、これを読みだす方法に変更することにする。

OutPort.hの変更

```
===================================================================
---  OutPort.h      (リビジョン 2718)
+++  OutPort.h      (作業コピー)
@@ -138,6 +138,8 @@
  #endif
                    m_value(value), m_onWrite(0), m_onWriteConvert(0)
       {
+          addProperty("dataport.data_value", m_value);
+          m_propValueIndex = NVUtil::find_index(m_profile.properties, "dataport.data_value");
       }

       /*!
@@ -209,6 +211,7 @@
                        (*m_onWrite)(value);
                        RTC_TRACE(("OnWrite called"));
                    }
+          m_profile.properties[m_index].value <<= value;

                bool result(true);
                std::vector<const char *> disconnect_ids;
@@ -490,6 +493,8 @@
           coil::TimeMeasure m_cdrtime;

           DataPortStatusList m_status;
+
+          CORBA::Long m_propValueIndex;
       };
  }; // namespace RTC

---  rtm.py.org    2016-05-13 10:12:34.892000000 +0900
+++  rtm.py        2016-05-13 10:12:57.004000000 +0900
@@ -635,11 +635,19 @@
  #  \return data
  #
  def readDataPort(port, timeout=1.0):
-         pprof = port.get_port_profile()
```

```
+           try:
+                   pprof   =   port.get_port_profile()
+           except:
+                   return   None
+           classname   =   None
+           datavalue   =   None
            for   prop   in   pprof.properties:
                    if   prop.name   ==   "dataport.data_type":
                            classname   =   any.from_any(prop.value)
-                           break
+                   if   prop.name   ==   "dataport.data_value":
+                           datavalue   =   any._to_tc_value(prop.value)[1]
+                           return   datavalue
+
            nv1   =   SDOPackage.NameValue("dataport.interface_type",   any.to_any("corba_cdr"))
            nv2   =   SDOPackage.NameValue("dataport.dataflow_type",   any.to_any("Pull"))
            nv3   =   SDOPackage.NameValue("dataport.subscription_type",   any.to_any("flush"))
```