# OpenRTM-aist (C++) - バグ #2176

## RTCがActive状態で終了(exit)する際のon_deactivate,on_finalizeの呼び出し順序の不整合

2011/06/29 21:57 - kurihara

| | | | |
|---|---|---|---|
| ステータス: | 終了 | 開始日: | 2011/06/29 |
| 優先度: | 通常 | 期日: | |
| 担当者: | | 進捗率: | 100% |
| カテゴリ: | | 予定工数: | 0.00時間 |
| 対象バージョン: | | | |

**説明**

株式会社セック　小田桐様からの報告

RTCがActive状態で終了(exit)する際、タイミングやRTCの周期の値に
よっては、on_deactivateが呼ばれない、もしくは、
on_finalizeが呼ばれた後にon_deactivateが呼ばれる場合があります。
原因は、RTObject_impl::exit内で、deactivate_componentを
呼び出した後、実際にRTCがInactive状態に遷移するのを
待っていないためです。

**履歴**

#1 - 2011/06/29 22:13 - kurihara

- プロジェクト を OpenRTM-aist から OpenRTM-aist (C++) に変更

#2 - 2011/07/01 00:34 - n-ando

- ステータス を 新規 から 終了 に変更
- 進捗率 を 0 から 100 に変更

PeriodicExecutionContext::deactivate_component(LightweightRTObject_ptr comp) 内で、

it->_sm.m_sm.goTo(INACTIVE_STATE);

のようにINACTIVE_STATEへの遷移を実行しているが、実行コンテキストは別スレッドで実行されているため実際にINACTIVEなっているかどうかを調べる必要がある。
以下のコードを追加

```
---  PeriodicExecutionContext.cpp            (revision  2191)
+++  PeriodicExecutionContext.cpp            (working  copy)
@@  -374,14  +374,35  @@
            CompItr  it;
            it  =  std::find_if(m_comps.begin(),  m_comps.end(),
                                find_comp(comp));
-          if  (it  ==  m_comps.end())
-              return  RTC::BAD_PARAMETER;
-
+          if  (it  ==  m_comps.end())  {  return  RTC::BAD_PARAMETER;  }
            if  (!(it->_sm.m_sm.isIn(ACTIVE_STATE)))
-              return  RTC::PRECONDITION_NOT_MET;
+          {
+              return  RTC::PRECONDITION_NOT_MET;
+          }

            it->_sm.m_sm.goTo(INACTIVE_STATE);
-          return  RTC::RTC_OK;
+          int  count(0);
+          const  double  usec_per_sec(1.0e6);
+          double  sleeptime(10.0  *  usec_per_sec  /  get_rate());
+          RTC_PARANOID(("Sleep  time  is  %f  [us]",  sleeptime));
+          while  (it->_sm.m_sm.isIn(ACTIVE_STATE))
+          {
```

```
+                     RTC_TRACE(("Waiting to be the INACTIVE state"));
+                     coil::usleep(sleeptime);
+                     if (count > 1000)
+                       {
+                         RTC_ERROR(("The component is not responding."));
+                         break;
+                       }
+                     ++count;
+                   }
+                 if (it->_sm.m_sm.isIn(INACTIVE_STATE))
+                   {
+                     RTC_TRACE(("The component has been properly deactivated."));
+                     RTC::RTC_OK;
+                   }
+                 RTC_ERROR(("The component could not be deactivated."));
+                 return RTC::RTC_ERROR;
   #else // ORB_IS_RTORB
                for (int i(0); i < (int)m_comps.size(); ++i)
                  {
@@ -392,7 +413,29 @@
                                        return RTC::PRECONDITION_NOT_MET;
                                      }
                                    m_comps.at(i)._sm.m_sm.goTo(INACTIVE_STATE);
-                                   return RTC::RTC_OK;
+                                   int count(0);
+                                   const double usec_per_sec(1.0e6);
+                                   double sleeptime(10.0 * usec_per_sec / get_rate());
+                                   RTC_PARANOID(("Sleep time is %f [us]", sleeptime));
+                                   while (m_comps.at(i)._sm.m_sm.isIn(ACTIVE_STATE))
+                                     {
+                                       RTC_TRACE(("Waiting to be the INACTIVE state"));
+                                       coil::usleep(sleeptime);
+
+                                       if (count > 1000)
+                                         {
+                                           RTC_ERROR(("The component is not responding."));
+                                           break;
+                                         }
+                                       ++count;
+                                     }
+                                   if (m_comps.at(i)._sm.m_sm.isIn(INACTIVE_STATE))
+                                     {
+                                       RTC_TRACE(("The component has been properly deactivated."));
+                                       RTC::RTC_OK;
+                                     }
+                                   RTC_ERROR(("The component could not be deactivated."));
+                                   return RTC::RTC_ERROR;
                                  }
                              }
                            return RTC::BAD_PARAMETER;
```