OpenRTM-aist (Python) - 機能 #3400

①　コンポーネント操作関数セットの実装

2015/12/22 09:43 - n-ando

ステータス:	終了	**開始日:**	2015/12/22
優先度:	通常	**期日:**	2016/03/25
担当者:	miyamoto	**進捗率:**	100%
カテゴリ:		**予定工数:**	30.00時間
対象バージョン:	RELEASE_1_2_0		

説明

rtshellライク[1]に、コンポーネントの各種操作を簡単に行うことができる関数セットを実装すること。
[1] rtshell: http://openrtm.org/openrtm/ja/node/869

関係しているリビジョン

リビジョン 632 - 2016/02/01 12:58 - miyamoto

[incompat,new_func,new_file,->RELENG_1_2] add CORBA_RTCUtil.py. refs #3400

リビジョン 644 - 2016/02/01 19:22 - miyamoto

[incompat,new_func,new_file,->RELENG_1_2] add CORBA_RTCUtil.py. refs #3400

リビジョン 655 - 2016/02/25 04:31 - miyamoto

[incompat,new_func,->RELENG_1_2] add disconnect_by_portref_connector_name() and disconnect_by_portref_connector_id(), disconnect_all_by_ref(). refs #3400

リビジョン 661 - 2016/02/26 00:25 - miyamoto

[incompat,new_func,->RELENG_1_2] add disconnect_all_by_name() and disconnect_by_portname_connector_name(), disconnect_by_portname_connector_id(). refs #3400

リビジョン 668 - 2016/02/27 10:34 - miyamoto

[incompat,bugfix,func,->RELENG_1_2] bug fix. refs #3400

リビジョン 672 - 2016/02/27 11:04 - miyamoto

[compat,bugfix,->RELENG_1_2] bug fix. refs #3400

リビジョン 675 - 2016/02/27 22:54 - miyamoto

[incompat,new_func,->RELENG_1_2] add get_connector_ids_by_portref() and get_connector_names_by_portref(). refs #3400

リビジョン 678 - 2016/02/28 17:04 - miyamoto

[incompat,bugfix,func,->RELENG_1_2] The bug of get_state() has been fixed. refs #3400.

履歴

#1 - 2016/01/14 16:14 - miyamoto

- 期日 を 2016/03/25 にセット

- 担当者 を miyamoto にセット

- 対象バージョン を RELEASE_1_2_0 にセット

- 進捗率 を 0 から 50 に変更

- 予定工数 を 30.00時間 にセット

#2 - 2016/01/14 18:34 - n-ando

作業内容についても簡単に記載してください。よろしくお願いいたします。

#3 - 2016/01/14 21:26 - miyamoto

- ファイル test_CORBA_RTCUtil.py を追加

すみません。使い方がよく分かっていませんでした。
以下で作業内容を説明します。

実装

CORBA_RTCUtil.pyを作成し、以下の関数を実装した。

RTC基本操作関数

- get_component_profile:RTC のプロパティを取得する
- is_existing:RTCが終了しているかを判定する
- get_actual_ec:対象のRTCから指定したIDの実行コンテキストを取得する

実行コンテキスト操作関数

- get_ec_id:対象のRTCから指定した実行コンテキストのIDを取得する
- activate:対象のRTCを指定IDの実行コンテキストでアクティブ化する
- deactivate:対象のRTCを指定IDの実行コンテキストで非アクティブ化する
- reset:対象のRTCを指定IDの実行コンテキストでリセットする
- get_state:対象のRTCを指定IDの実行コンテキストでの状態を取得する
- is_in_inactive:対象のRTCを指定IDの実行コンテキストでの状態がINACTIVE_STATEかを判定する
- is_in_active:対象のRTCを指定IDの実行コンテキストでの状態がACTIVE_STATEかを判定する
- is_in_error:対象のRTCを指定IDの実行コンテキストでの状態がERROR_STATEかを判定する
- get_default_rate:実行コンテキストのデフォルトの周期をプロファイルから取得する
- get_current_rate:実行コンテキストの周期を取得する
- set_current_rate:実行コンテキストの周期を設定する
- get_participants_rtc:対象の実行コンテキストに参加しているRTCのリストを返す

ポート操作関数

- get_port_names:対象のRTCの保持しているポートの名前のリストを取得する
- get_inport_names:対象のRTCの保持しているインポートの名前のリストを取得する
- get_outport_names:対象のRTCの保持しているアウトポートの名前のリストを取得する
- get_svcport_names:対象のRTCの保持しているサービスポートの名前のリストを取得する
- get_port_by_name:対象のRTCから指定した名称のポートを取得する
- get_connector_names:対象のポートが保持するコネクタの名前のリストを取得する
- get_connector_ids:対象のポートが保持するコネクタのIDのリストを取得する
- create_connector:指定したポートを接続するためのコネクタプロファイルを取得する
- already_connected:対象のポート同士が接続されているかを判定する
- connect:対象のポート同士を接続する
- connect_multi:対象のポートと指定したリスト内の全てのポートとを接続する
- connect_by_name:指定した名前のポート同士を接続する
- disconnect:指定したコネクタを切断する
- disconnect_by_connector_name:対象のポートから指定した名前のコネクタを切断する
- disconnect_by_connector_id:対象のポートから指定したIDのコネクタを切断する
- disconnect_by_port_name:対象のポートが指定した名前のポートと接続されている場合に切断する

コンフィギュレーション操作関数

- get_configuration:対象のRTCのコンフィギュレーションオブジェクトを取得する
- get_parameter_by_key:対象のRTCから指定のコンフィギュレーションセット名，パラメータ名のコンフィギュレーションパラメータを取得する
- get_active_configuration:アクティブなコンフィギュレーションセットを取得する
- set_configuration:対象のRTCのコンフィギュレーションパラメータを設定する

テスト

添付したテスト用コードでテストを行った。
テスト用コードは以下の動作を行う。

setUp関数

マネージャ初期化
self.manager = OpenRTM_aist.Manager.init(sys.argv)
self.manager.setModuleInitProc(MyModuleInit)
self.manager.activateManager()

test_Component

get_component_profile関数で正常にプロパティが取得できているか確認
compProf = OpenRTM_aist.get_component_profile(self.comp1)
self.assertEqual(compProf.getProperty("implementation_id"), "TestComp1")

is_existing関数でRTCが終了しているか確認

```
ret = OpenRTM_aist.is_existing(self.comp1)
self.assertFalse(ret)
```

test_EC関数

get_actual_ec関数で実行コンテキストを取得できているか、get_ec_id関数でIDを取得できているかの確認
```
ec = OpenRTM_aist.get_actual_ec(self.comp1,0)
ec_id = OpenRTM_aist.get_ec_id(self.comp1, ec)
self.assertEqual(ec_id, 0)
```

get_default_rate関数で実行周期が取得できてるかの確認
```
rate = OpenRTM_aist.get_default_rate(ec)
self.assertEqual(int(rate), 1000)
```

set_current_rate関数で実行周期を設定できているか、get_current_rate関数で実行周期を取得できているかの確認
```
rate = OpenRTM_aist.get_current_rate(ec)
self.assertEqual(int(rate), 100)
rate = OpenRTM_aist.get_current_rate(ec)
self.assertEqual(int(rate), 100)
```

activate関数でアクティブ化できているか、is_in_active関数でACTIVE_STATEかどうかを正常に判定できているかの確認
```
ret = OpenRTM_aist.activate(self.comp1,0)
self.assertEqual(ret, RTC.RTC_OK)
state = OpenRTM_aist.is_in_active(self.comp1, 0)
self.assertTrue(state)
```

deactivate関数で非アクティブ化できているか、is_in_inactive関数でINACTIVE_STATEかどうかを正常に判定できているかの確認
```
ret = OpenRTM_aist.deactivate(self.comp1,0)
self.assertEqual(ret, RTC.RTC_OK)
state = OpenRTM_aist.is_in_inactive(self.comp1, 0)
self.assertTrue(state)
```

get_state関数で状態を取得できているかの確認
```
ret = [None]
ans = OpenRTM_aist.get_state(self.comp1, 0, ret)
self.assertEqual(ret[0], RTC.INACTIVE_STATE)
```

test_Port関数

get_port_names関数でポートオブジェクトを取得できているかの確認
```
port_names = OpenRTM_aist.get_port_names(self.comp1)
self.assertTrue("TestComp10.out" in port_names)
```

get_inport_names関数でポートオブジェクトを取得できているかの確認
```
inport_names = OpenRTM_aist.get_inport_names(self.comp2)
self.assertTrue("TestComp20.in" in inport_names)
```

get_outport_names関数でポートオブジェクトを取得できているかの確認
```
outport_names = OpenRTM_aist.get_outport_names(self.comp1)
self.assertTrue("TestComp10.out" in outport_names)
```

get_svcport_names関数でポートオブジェクトを取得できているかの確認
```
svcport_names = OpenRTM_aist.get_svcport_names(self.comp1)
self.assertTrue("TestComp10.service" in svcport_names)
```

get_port_by_name関数でポートオブジェクトを取得できているかの確認
```
rtc1_port_out = OpenRTM_aist.get_port_by_name(self.comp1,"TestComp10.out")
pp = rtc1_port_out.get_port_profile()
self.assertEqual(pp.name, "TestComp10.out")
```

connect関数でポートを接続できているか、already_connectedで接続されているかの判定を正常に行えているかを確認
```
ret = OpenRTM_aist.connect("con1",prop,rtc1_port_out,rtc2_port_in)
self.assertEqual(ret, RTC.RTC_OK)
ret = OpenRTM_aist.already_connected(rtc1_port_out,rtc2_port_in)
self.assertTrue(ret)
```

get_connector_names関数でコネクタの名前のリストを取得できているかの確認
```
con_names = OpenRTM_aist.get_connector_names(rtc1_port_out)
self.assertTrue("con1" in con_names)
```

disconnect関数でコネクタを切断できているかの確認
```
ret = OpenRTM_aist.disconnect(rtc1_port_out.get_connector_profiles()[0])
self.assertEqual(ret, RTC.RTC_OK)
ret = OpenRTM_aist.already_connected(rtc1_port_out,rtc2_port_in)
self.assertFalse(ret)
```

connect_multi関数でポートを接続できているかの確認
ret = OpenRTM_aist.connect_multi("con2",prop,rtc1_port_out,[rtc1_port_in,rtc2_port_in])
self.assertEqual(ret, RTC.RTC_OK)
ret = OpenRTM_aist.already_connected(rtc1_port_out,rtc1_port_in)
self.assertTrue(ret)

connect_by_name関数でポートを接続できているか確認

ret = OpenRTM_aist.connect_by_name("con3",prop,self.comp1,"TestComp10.out",self.comp2,"TestComp20.in")
self.assertEqual(ret, RTC.RTC_OK)
ret = OpenRTM_aist.already_connected(rtc1_port_out,rtc2_port_in)
self.assertTrue(ret)

disconnect_by_connector_name関数でコネクタを切断できているか確認
ret = OpenRTM_aist.disconnect_by_connector_name(rtc1_port_out, "con3")
self.assertEqual(ret, RTC.RTC_OK)
ret = OpenRTM_aist.already_connected(rtc1_port_out,rtc2_port_in)
self.assertFalse(ret)

disconnect_by_connector_id関数でコネクタを切断できているか確認
ret = OpenRTM_aist.connect("con1",prop,rtc1_port_out,rtc2_port_in)
ret = OpenRTM_aist.disconnect_by_connector_id(rtc1_port_out,rtc1_port_out.get_connector_profiles()[0].connector_id)
self.assertEqual(ret, RTC.RTC_OK)
ret = OpenRTM_aist.already_connected(rtc1_port_out,rtc2_port_in)
self.assertFalse(ret)

disconnect_by_port_name関数でコネクタを切断できているか確認
ret = OpenRTM_aist.connect("con1",prop,rtc1_port_out,rtc2_port_in)
ret = OpenRTM_aist.disconnect_by_port_name(rtc1_port_out,"TestComp20.in")
self.assertEqual(ret, RTC.RTC_OK)
ret = OpenRTM_aist.already_connected(rtc1_port_out,rtc2_port_in)
self.assertFalse(ret)

test_configuration関数

get_parameter_by_key関数でコンフィギュレーションパラメータを取得できているか確認
ret = OpenRTM_aist.get_parameter_by_key(self.comp1,"default","test1")
self.assertEqual(ret,"0")

set_configuration関数でコンフィギュレーションパラメータ設定できているかを確認
ret = OpenRTM_aist.set_configuration(self.comp1,"default","test1","1")
self.assertTrue(ret)
ret = OpenRTM_aist.get_parameter_by_key(self.comp1,"default","test1")
self.assertEqual(ret,"1")

#4 - 2016/01/16 00:12 - miyamoto

- ファイル test_CORBA_RTCUtil.py を追加

- 進捗率 を 50 から 60 に変更

以下の関数が仕様と違っていたので修正した。

is_existing:RTCのオブジェクトリファレンスが存在しているかを判定する
get_default_rate:RTCのデフォルトECの実行周期を取得する
get_current_rate:RTCの指定IDのECの実行周期を取得する
set_current_rate:RTCの指定IDのECの実行周期を設定する
get_participants_rtc:RTCのデフォルトECに参加しているRTCのリストを取得する
get_active_configuration:アクティブなコンフィギュレーションセットをkey-valueで取得する

以下の関数を実装した。

is_alive_in_default_ec:RTCのデフォルトECでalive状態かを取得する
set_default_rate:RTCのデフォルトECの実行周期を設定する
add_rtc_to_default_ec:RTCのデフォルトECに指定したRTCを関連付ける
remove_rtc_to_default_ec:RTCのデフォルトECに指定したRTCの関連付けを解除する
get_current_configuration_name:アクティブなコンフィギュレーションセット名を取得する

これに伴いテスト用コードも修正、変更した。

add_rtc_to_default_ec関数でECとRTCを関連付けられるかの確認
ret = OpenRTM_aist.add_rtc_to_default_ec(self.comp1, self.comp2)
self.assertEqual(ret, RTC.RTC_OK)

is_alive_in_default_ec関数でalive状態かを判定できるかの確認
ret = OpenRTM_aist.is_alive_in_default_ec(self.comp1)
self.assertEqual(ret, True)

set_default_rate関数で実行周期を設定できるかの確認
ret = OpenRTM_aist.set_default_rate(self.comp1, 500)
self.assertEqual(ret, RTC.RTC_OK)

get_default_rate関数で実行周期を取得できるかの確認
rate = OpenRTM_aist.get_default_rate(self.comp1)
self.assertEqual(int(rate), 500)

set_default_rate関数で実行周期を設定できるかの確認
ret = OpenRTM_aist.set_current_rate(self.comp2,1000, 100)
self.assertEqual(ret, RTC.RTC_OK)

get_default_rate関数で実行周期を取得できるかの確認
rate = OpenRTM_aist.get_current_rate(self.comp2, 1000)
self.assertEqual(int(rate), 100)

add_rtc_to_default_ec関数でECとRTCの関連付けを解除できるかの確認
ret = OpenRTM_aist.remove_rtc_to_default_ec(self.comp1, self.comp2)
self.assertEqual(ret, RTC.RTC_OK)

get_current_configuration_name関数でコンフィギュレーションセット名を取得できるかの確認
confsetname = OpenRTM_aist.get_current_configuration_name(self.comp1)
self.assertEqual(confsetname,"default")

#5 - 2016/02/18 12:03 - n-ando

以下の仕様でお願いします。

```
    *  @brief  RTC操作ユーティリティー関数群
    *
    *  RTC操作ユーティリティー関数群
    *  ============================
    *
    *  CORBA_RTCUtilではRTC操作のための種々のインターフェースを提供する
    *
    *  RTC基本操作系
    *  -------------
    *  -  get_component_profile(rtc):
    *           RTCのProfileを取得
    *  -  is_existing(rtc):
    *           RTCが存在しているかを確認
    *  -  is_alive_in_default_ec(rtc):
    *           RTCがデフォルトの実行コンテキストでalive状態かを判定する
    *
    *  ECおよび状態操作系
    *  ------------------
    *  -  get_actual_ec(rtc,  ec_id  =  0):
    *           RTコンポーネントに関連付けした実行コンテキストから指定したIDの
    *           実行コンテキストを取得
    *  -  get_ec_id(rtc,  ec):
    *           対象のRTコンポーネントから指定した実行コンテキストのIDを取得す
    *           る
    *  -  activate(rtc,  ec_id  =  0):
    *           RTCを指定した実行コンテキストでアクティブ化する
    *  -  deactivate(rtc,  ec_id  =  0):
    *           RTCを指定した実行コンテキストで非アクティブ化する
    *  -  reset(rtc,  ec_id  =  0):
    *           RTCを指定した実行コンテキストでリセットする
    *  -  get_state(rtc,  ec_id=0,  rtc  =  []):
    *           対象のRTコンポーネントの指定した実行コンテキストでの状態を取得
    *  -  is_in_inactive(rtc,  ec_id  =  0)
    *           対象のRTコンポーネントの指定した実行コンテキストでINACTIVE状態
    *           かどうか判定
    *  -  is_in_active(rtc,  ec_id  =  0):
    *           対象のRTコンポーネントの指定した実行コンテキストでACTIVE状態か
    *           どうか判定
    *  -  is_in_error(rtc,  ec_id  =  0)
    *           対象のRTコンポーネントの指定した実行コンテキストでERROR状態か
    *           どうか判定
    *  -  get_default_rate(rtc):
```

```
 *           RTCのデフォルトの実行コンテキストの実行周期を取得する
 *   - set_default_rate(rtm,  rate):
 *           RTCのデフォルトの実行コンテキストの実行周期を設定する
 *   - get_current_rate(rtc,  ec_id):
 *           RTCの指定IDの実行コンテキストの周期を取得
 *   - set_current_rate(rtc,  ec_id,  rate):
 *           RTCの指定IDの実行コンテキストの周期を設定
 *   - add_rtc_to_default_ec(localcomp,  othercomp):
 *         対象のRTCのデフォルトの実行コンテキストに指定のRTCを関連付ける
 *   - remove_rtc_to_default_ec(localcomp,  othercomp):
 *         対象のRTCのデフォルトの実行コンテキストの指定のRTCへの関連付け
 *         を解除する
 *   - get_participants_rtc(rtc):
 *           RTCのデフォルトの実行コンテキストに参加しているRTCのリストを取
 *           得する
 *
 *  ポート操作系
 *  ------------
 *   - get_port_names(rtc):
 *           指定したRTCの保持するポートの名前を取得
 *   - get_inport_names(rtc):
 *         指定したRTCの保持するインポートの名前を取得
 *   - get_outport_names(rtc):
 *           指定したRTCの保持するアウトポートの名前を取得
 *   - get_svcport_names(rtc):
 *           指定したRTCの保持するサービスポートの名前を取得
 *   - get_port_by_name(rtc):
 *           対象のRTCから指定した名前のポートを取得
 *   - get_connector_names(rtc,  port_name):
 *           指定したポートの保持しているコネクタの名前のリストを取得
 *   - get_connector_ids(rtc,  name):
 *   - get_connector_ids(port):
 *           指定したポートの保持しているコネクタのIDのリストを取得
 *   - create_connector(name,  prop_arg,  port0,  port1):
 *           指定したポートを接続するためのコネクタプロファイルを取得
 *   - already_connected(port0,  port1):
 *           指定したポート同士が接続されているかを判定
 *   - connect(name,  prop,  port0,  port1):
 *           指定したポートを接続する
 *   - connect_multi(name,  prop,  port,  target_ports):
 *           指定したポートと指定したリスト内のポート全てと接続する
 *   - connect_by_name(name,  prop,  rtc0,  portName0,  rtc1,  portName1):
 *           対象のRTCの指定した名前のポートを接続する
 *   - disconnect(connector_prof):
 *
 *       disconnect_by_portref_connector_id(port_ref,  conn_id)
 *       disconnect_by_portref_connector_name(port_ref,  conn_name)
 *       disconnect_by_portname_connector_id(port_ref,  conn_id)
 *       disconnect_by_portname_connector_name(port_ref,  conn_name)
 *       disconnect_by_portrefs(port0_ref,  port1_ref)
 *       disconnect_all_by_ref(port_ref)
 *       disconnect_all_by_name("rtcloc://hostname/ConsoleIn.out")
 *
 *           指定のコネクタを切断する
 *   - disconnect_by_connector_name(port,  name):
 *           対象のポートで指定した名前のコネクタを切断
 *   - disconnect_by_connector_id(port,  id):
 *           対象のポートで指定したIDのコネクタを切断
 *   - disconnect_by_port_name(port0,  port1):
 *           対象ポートと接続しているポートで指定したポート名と一致した場合
 *           に切断
 *
 *  コンフィギュレーション系
 *  ------------------------
 *   - get_configuration(rtc,  conf_name):
 *           指定したRTコンポーネントのコンフィギュレーション取得
 *   - set_configuration(rtc,  confset_name,  key,  value):
 *           コンフィギュレーションパラメータを設定
 *   - get_parameter_by_key(rtc,  confset_name,  value_name):
 *           指定したコンフィギュレーションセット名、パラメータ名のコンフィ
 *           ギュレーションパラメータを取得
 *   - get_active_configuration_name(rtc):
 *           対象のRTCのアクティブなコンフィギュレーションセット名を取得する
 *   - get_active_configuration(rtc):
 *           アクティブなコンフィギュレーションセットを取得
```

```
    *  -  set_active_configuration(rtc,  value_name,  value):
    *              コンフィギュレーションパラメータを設定
```

#6 - 2016/02/25 05:19 - miyamoto

- ファイル test_CORBA_RTCUtil.py を追加

- 進捗率 を 60 から 80 に変更

is_existing関数を修正した。
それに伴いテスト用コードも以下のように修正した。

ret = OpenRTM_aist.is_existing(self.comp1)
self.assertFalse(ret)
→
self.assertTrue(ret)

disconnect_by_connector_name関数、disconnect_by_connector_id関数をdisconnect_by_portref_connector_name関数、disconnect_by_portname_connector_id関数に変更した。

disconnect_all_by_ref関数を追加した。
以下のコードでテストを行った。

ret = OpenRTM_aist.disconnect_all_by_ref(rtc1_port_out)
self.assertTrue(ret)
ret = OpenRTM_aist.already_connected(rtc1_port_out,rtc2_port_in)
self.assertFalse(ret)

get_configuration関数を指定した名前のコンフィギュレーションをkey-valueで取得するように変更した。
以下のコードでテストを行った。

conf = OpenRTM_aist.get_configuration(self.comp1, "default")
self.assertEqual(conf.getProperty("test1"),"0")

get_current_configuration_name関数をget_active_configuration_name関数に変更した。

set_active_configuration関数を追加した。
以下のコードでテストを行った。

ret = OpenRTM_aist.set_active_configuration(self.comp1,"test1","2")
self.assertTrue(ret)
ret = OpenRTM_aist.get_parameter_by_key(self.comp1,"default","test1")
self.assertEqual(ret,"2")

#7 - 2016/02/26 00:23 - miyamoto

- ファイル test_CORBA_RTCUtil.py を追加

以下の関数を追加した。

disconnect_all_by_name

指定した名前のポートのコネクタを全て切断する。

以下のコードでテストを行った。

ret = OpenRTM_aist.connect("con1",prop,rtc1_port_out,rtc2_port_in)
ret = OpenRTM_aist.disconnect_all_by_name("rtcloc://localhost:2810/example/TestComp20.in")
self.assertEqual(ret, RTC.RTC_OK)
ret = OpenRTM_aist.already_connected(rtc1_port_out,rtc2_port_in)
self.assertFalse(ret)

disconnect_by_portname_connector_name

指定した名前のポートから名前が一致するコネクタを切断する。

以下のコードでテストを行った。

ret = OpenRTM_aist.connect("con1",prop,rtc1_port_out,rtc2_port_in)
ret = OpenRTM_aist.disconnect_by_portname_connector_name("rtcloc://localhost:2810/example/TestComp20.in","con1")
self.assertEqual(ret, RTC.RTC_OK)
ret = OpenRTM_aist.already_connected(rtc1_port_out,rtc2_port_in)
self.assertFalse(ret)

disconnect_by_portname_connector_id

指定した名前のポートの指定IDのコネクタを切断する。
以下のコードでテストを行った。

```
ret = OpenRTM_aist.connect("con1",prop,rtc1_port_out,rtc2_port_in)
ret =
OpenRTM_aist.disconnect_by_portname_connector_id("rtcloc://localhost:2810/example/TestComp10.out",rtc1_port_out.get_connector_profiles()[0
].connector_id)
self.assertEqual(ret, RTC.RTC_OK)
ret = OpenRTM_aist.already_connected(rtc1_port_out,rtc2_port_in)
self.assertFalse(ret)
```

#8 - 2016/02/28 17:08 - miyamoto

get_state関数の引数の順番を変更した。

```
get_state(rtc, ec_id, state)
→
get_state(state, rtc, ec_id)
```

#9 - 2016/03/17 10:58 - miyamoto

- 進捗率 を 80 から 100 に変更

#10 - 2017/08/30 14:20 - n-ando

- ステータス を 新規 から 終了 に変更

ファイル

test_CORBA_RTCUtil.py	9.28 KB	2016/01/14	miyamoto
test_CORBA_RTCUtil.py	9.84 KB	2016/01/16	miyamoto
test_CORBA_RTCUtil.py	10.5 KB	2016/02/25	miyamoto
test_CORBA_RTCUtil.py	11.5 KB	2016/02/26	miyamoto