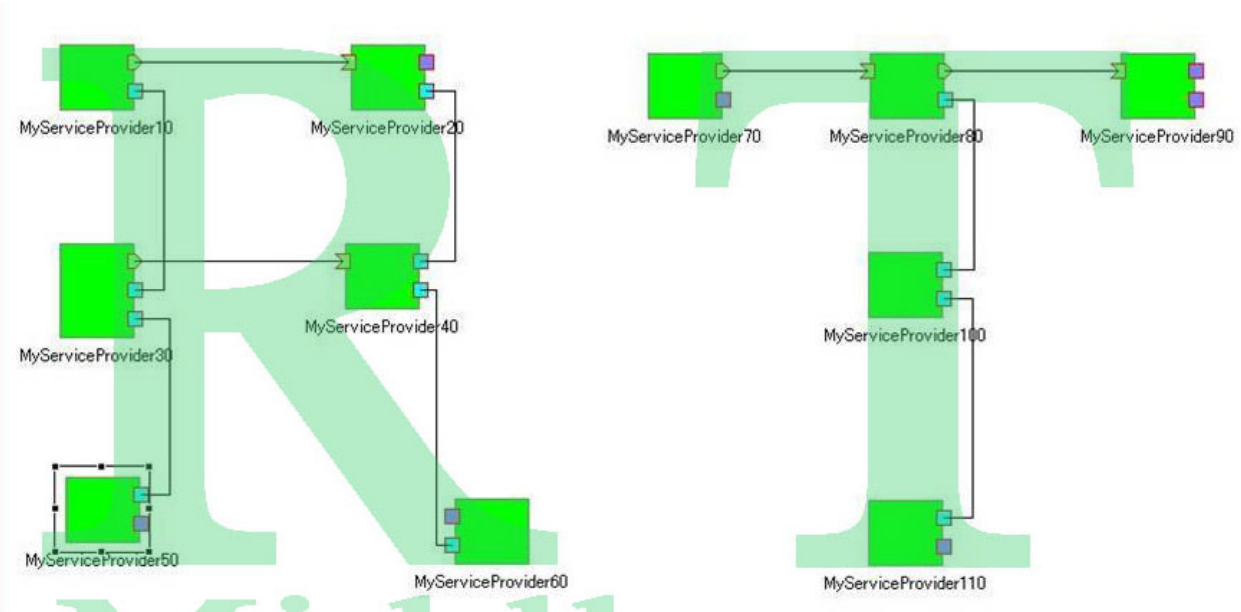
# RT コンポーネント用

# コネクタモジュール取扱説明書

Version 1.0

(There is an English version, too)



# 目次

## １．　特長と利点

・ このコネクタモジュールは，通常 rtc-link 上から操作していた”ポート間の接続”と ”アクティビティの変更”をコンソール上から行うことが可能です.

・ python や CORBA を意識せずに，任意のコネクタモジュールを作成することができます．また，基本的に１ファイルでコネクタモジュールを新規作成できます.

・ このコネクタモジュールを使用することにより，デバッグの度にコンポーネントを接続する煩わしさや，多数のコンポーネントを立ち上げる手間を大幅に削減することができます.

## ２．　開発環境

2007 年 11 月現在，このモジュールは以下のシステム構成になっています.

- OS：Vine3.2 (kernel 2.4.31-0v11.8smp)
- CPU：Intel Pentium D (2.8GHz)
- Memory：2.0GB
- RT-Middleware のバージョン：OpenRTM-aist.0.4.1-RELEASE
- Python のバージョン：python02.3.4-0v18.1
- omniORBpy のバージョン：omniORBpy-2.7-1

# ３．　使用方法

この章では，ConnectorModule を入手してから使用するまでの流れを説明します．
※　ホスト名：ufrg01，ユーザ名：ufrg で行った場合の例を示しています．

１．ConnectorModule.tar.gz を入手する．

２．任意ディレクトリを作成する　（以降<WorkDIR>とする）

３．<WorkDIR>に，tar.gz ファイルを展開する．
※　例では test ディレクトリに展開している．

```
[ufrg@ufrg01 test]$ ls
ConnectorModule.tar.gz
[ufrg@ufrg01 test]$ tar xzvf ConnectorModule.tar.gz
```

４．展開してできた ConnectorModule ディレクトリに 3 つのディレクトリができていることを確認する．各ディレクトリの構成は 18 ページの補足資料に示すツリーのとおりです．

```
[ufrg@ufrg01 test]$ ls
ConnectorModule/  ConnectorModule.tar.gz
[ufrg@ufrg01 test]$ cd ConnectorModule
[ufrg@ufrg01 ConnectorModule]$ ls
common/  example/  tutorial/
[ufrg@ufrg01 ConnectorModule]$
```

５．コネクタモジュール例のディレクトリへ移動する．
データポート用コネクタモジュール，サービス用コネクタモジュールなど
４つのディレクトリが確認できる．

```
[ufrg@ufrg01 test]$ cd ConnectorModule
[ufrg@ufrg01 ConnectorModule]$ ls
common/  example/  tutorial/
[ufrg@ufrg01 ConnectorModule]$ cd example/
[ufrg@ufrg01 example]$ ls
DS/  DataPort/  NamingServices/  ServicePort/
[ufrg@ufrg01 example]$
```

６．別ウインドウにて，OpenRTM-aist-0.4.1 のサンプルコンポーネントを起動する．
（例：SimpleIO のコンポーネント ConsoleInComp，ConsoleOutComp など）

７．コンポーネントを起動後，実行したコンポーネントに適したモジュールの
シェルを実行する．
例：SimpleIO の場合，
以下のように DataPort ディレクトリのモジュールを実行する．

## ４．　使用例

（ア）Data Port 間の接続（例：SimpleIO）

１．任意のネーミングサービスを起動する．

```
[ufrg@ufrg01 ufrg]$ rtm-naming
```

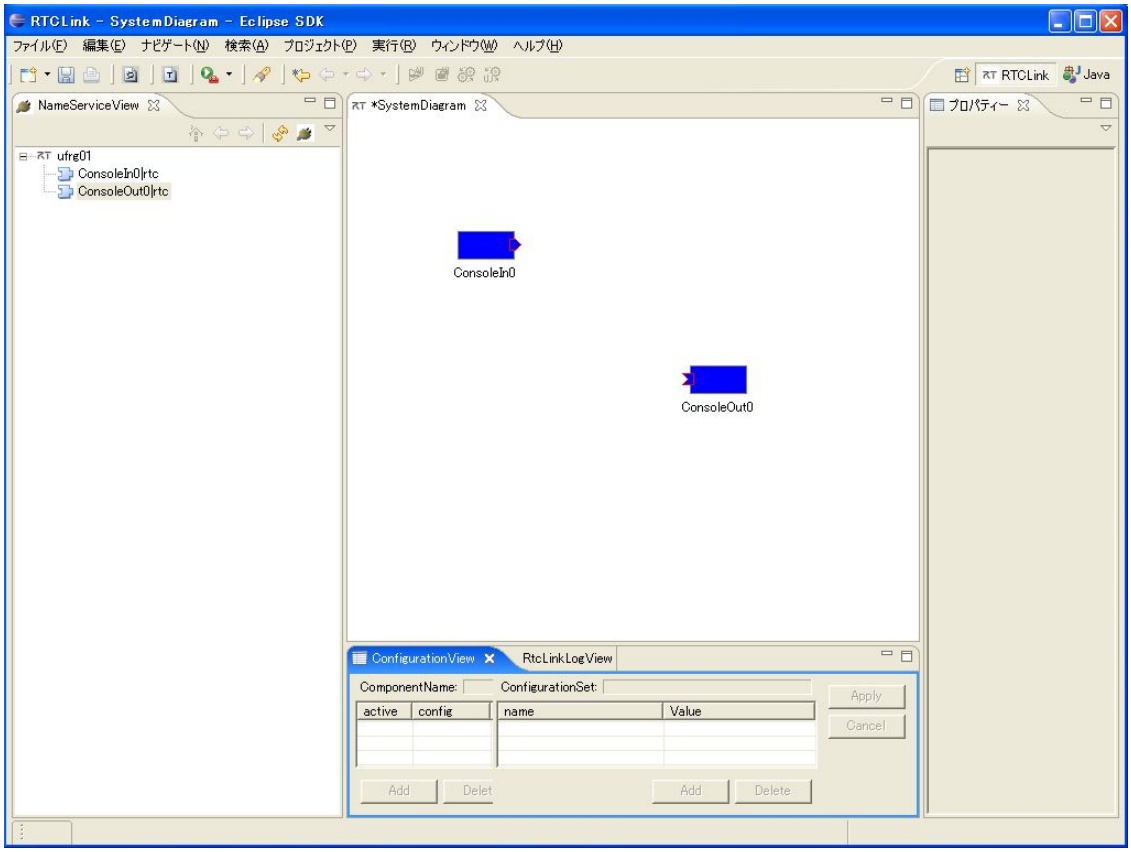
２．ConsoleIn コンポーネントを起動する．

```
[ufrg@ufrg01 SimpleIO]$ ./ConsoleInComp -f rtc.conf
```

３．ConsoleOut コンポーネントを起動する．

```
[ufrg@ufrg01 SimpleIO]$ ./ConsoleOutComp -f rtc.conf
```

４．rtc-link 上で，コンポーネントが起動されていることを確認する．



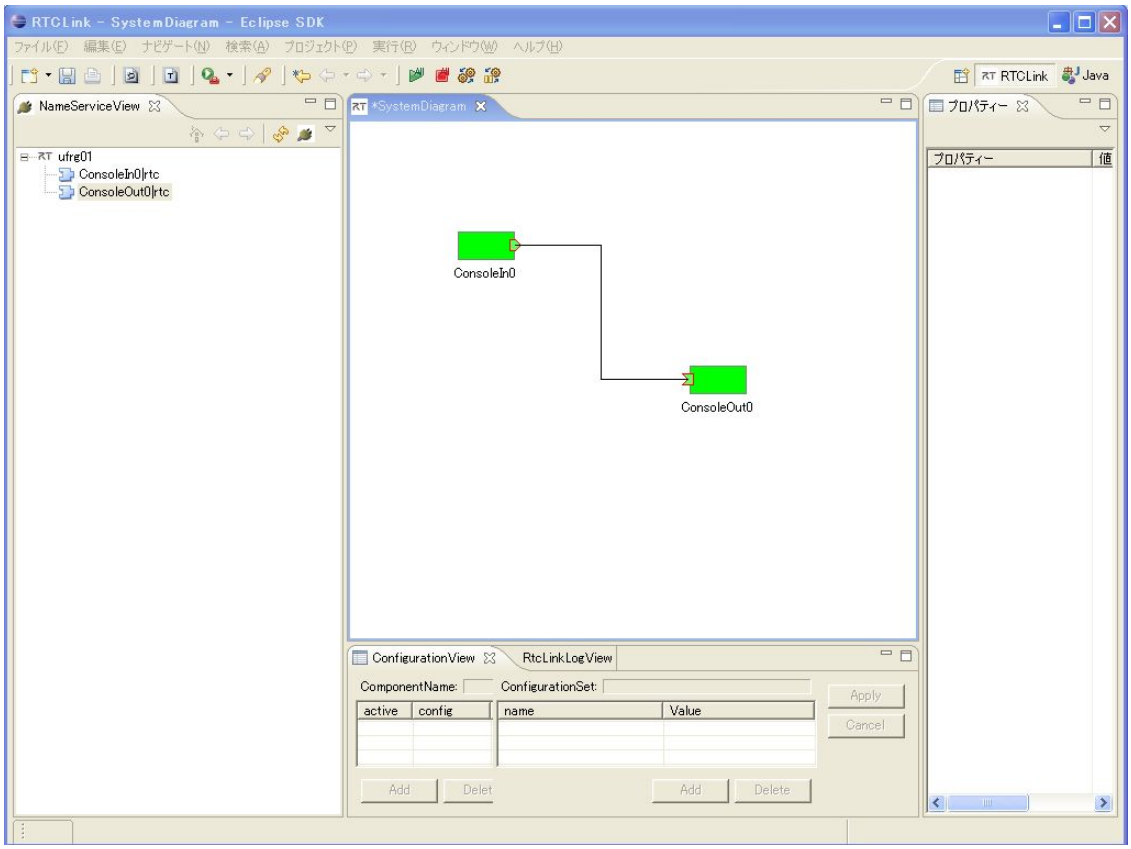
5．データポート用コネクタモジュールのディレクトリへ移動する．

```
[ufrg@ufrg01 test]$ cd ConnectorModule/example/
[ufrg@ufrg01 example]$ ls
DS/  DataPort/  NamingServices/  ServicePort/
[ufrg@ufrg01 example]$ cd DataPort/
```

6．python ファイルを実行する．

```
[ufrg@ufrg01 DataPort]$ ls
DataConnector.py  m.sh*  run.sh*
[ufrg@ufrg01 DataPort]$ ./run.sh
```

7．rtc-link 上でコンポーネントが接続されたことを確認できる．



（イ）ServicePort 間の接続（例：SimpleService）

1．任意のネーミングサービスを起動する．

```
[ufrg@ufrg01 ufrg]$ rtm-naming
```

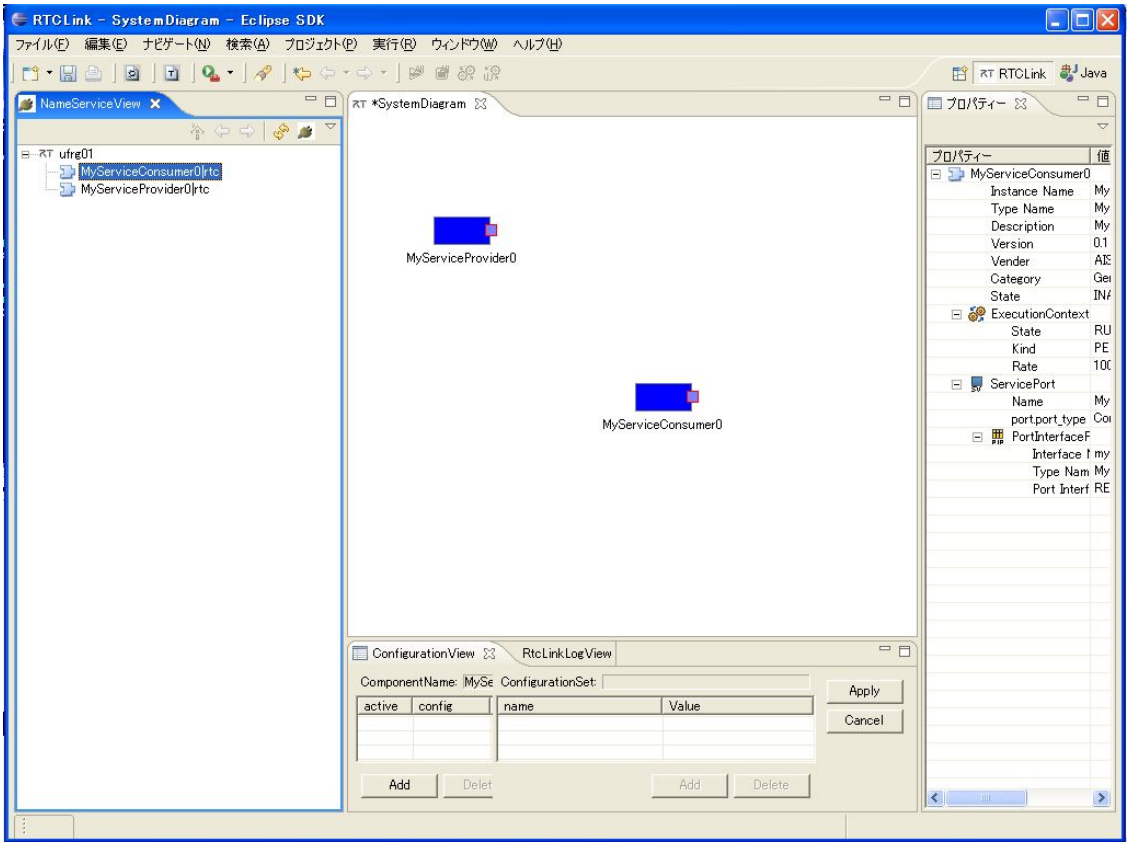
2．SimpleServiceProvider を起動する．

```
[ufrg@ufrg01 SimpleService]$ ./MyServiceProviderComp -f rtc.conf
```

3．SimpleServiceConsumer を起動する．

```
[ufrg@ufrg01 SimpleService]$ ./MyServiceConsumerComp -f rtc.conf
```

4．rtc-link 上で，コンポーネントが起動されていることを確認する．



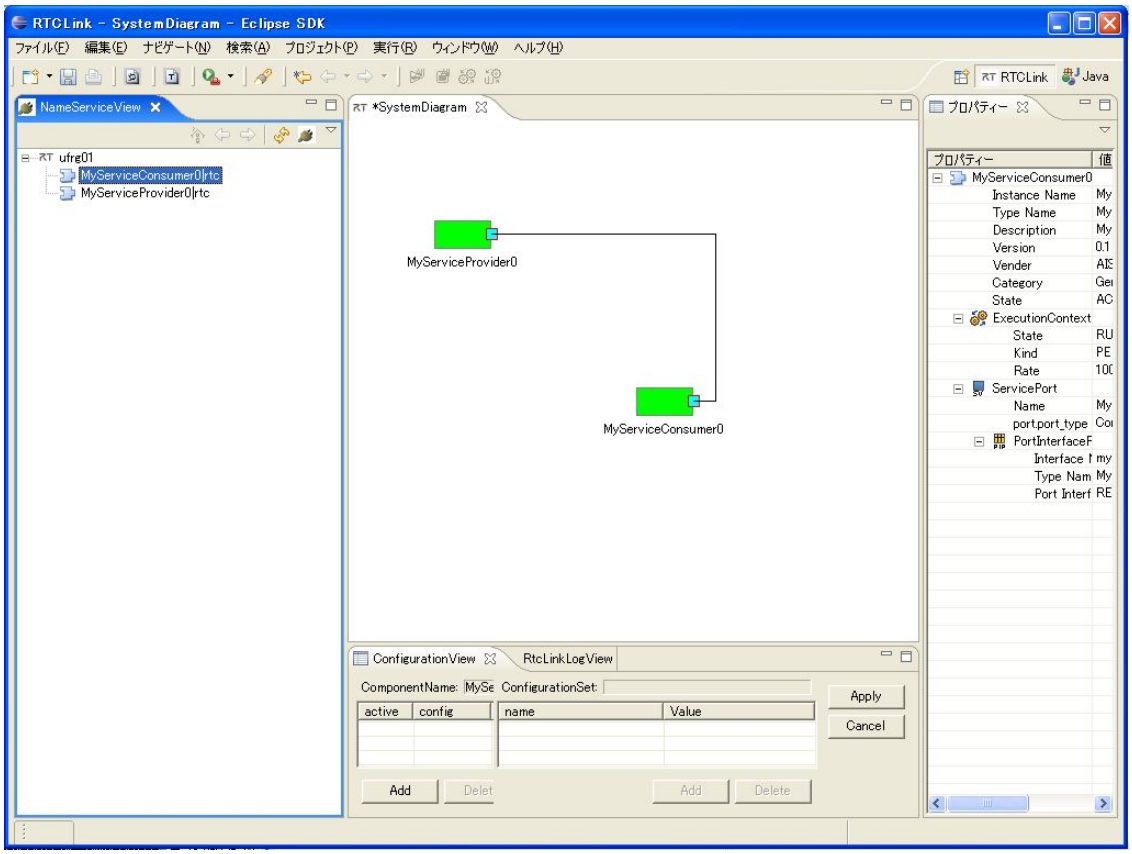
5．サービスポート用モジュールのディレクトリへ移動する．

```
[ufrg@ufrg01 test]$ cd ConnectorModule/example/
[ufrg@ufrg01 example]$ ls
DS/  DataPort/  NamingServices/  ServicePort/
[ufrg@ufrg01 example]$ cd ServicePort/
```

6．python ファイルを実行する．

```
[ufrg@ufrg01 ServicePort]$ ls
ServiceConnector.py  m.sh*  run.sh*
[ufrg@ufrg01 ServicePort]$ ./run.sh
```

7．rtc-link 上でコンポーネントが接続され，Activate されたことを確認する．



(ウ)　DataPort / ServicePort を持つコンポーネントの接続

※　事前に DataPort×1 と ServicePort×1 を持つコンポーネントを 1 組用意する．
例では，
OutPort×1 と ServiceProvider×1 を持つ MyServiceProvider と
InPort×1 と ServiceConsumer×1 を持つ MyServiceConsumer を用意している．

１．任意のネーミングサービスを起動する．

```
[ufrg@ufrg01 ufrg]$ rtm-naming
```

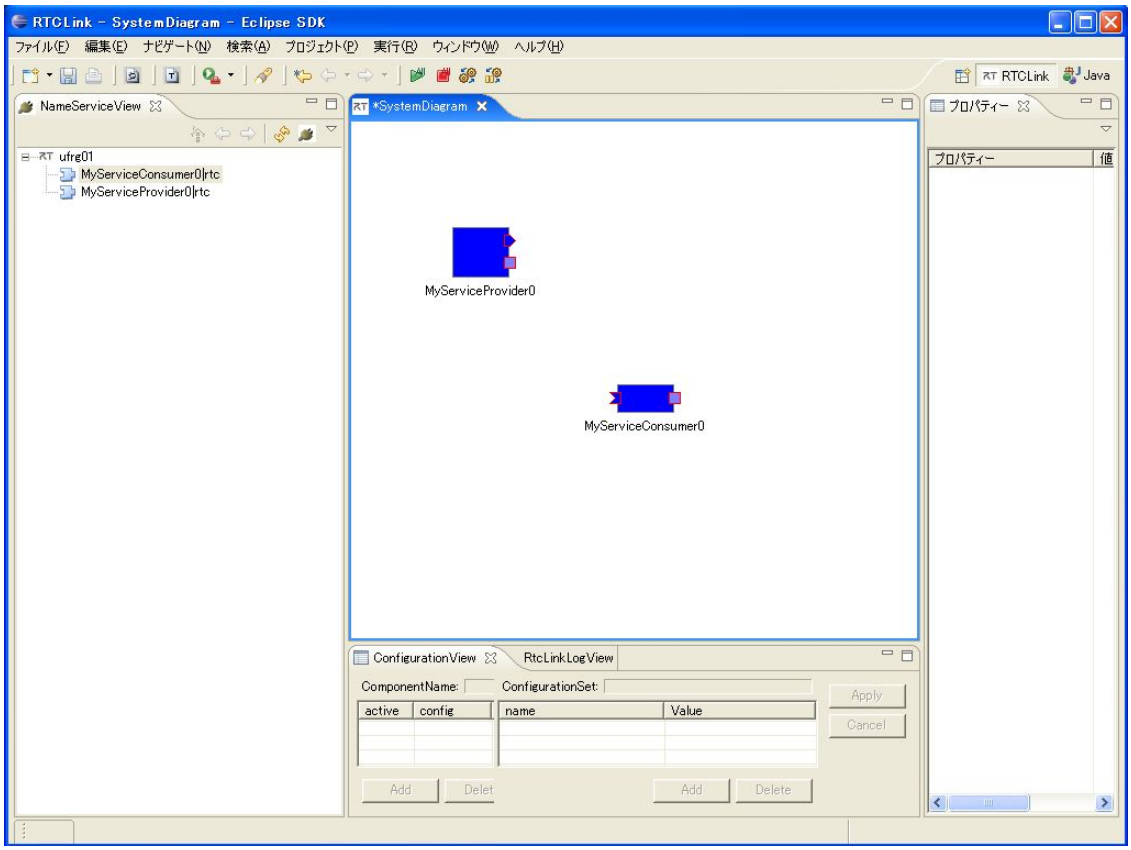
２．SimpleServiceProvider を起動する．

```
[ufrg@ufrg01 Data_Service]$ ./MyServiceProviderComp -f rtc.conf
```

３．SimpleServiceConsumer を起動する．

```
[ufrg@ufrg01 Data_Service]$ ./MyServiceConsumerComp -f rtc.conf
```

４．rtc-link 上で，コンポーネントが起動されていることを確認する．



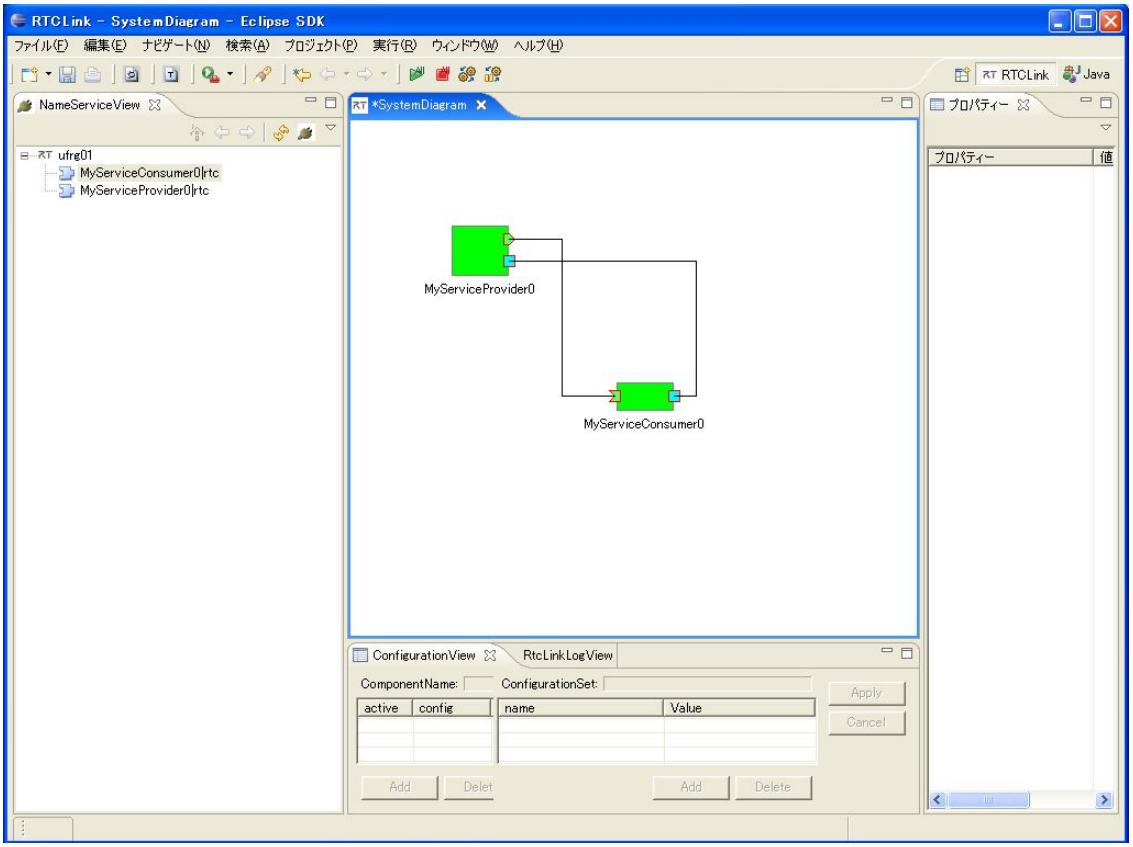
５．データ/サービス用モジュールのディレクトリへ移動する．

```
[ufrg@ufrg01 test]$ cd ConnectorModule/example/
[ufrg@ufrg01 example]$ ls
DS/  DataPort/  NamingServices/  ServicePort/
[ufrg@ufrg01 example]$ cd DS/
```

６．python ファイルを実行する．

```
[ufrg@ufrg01 DS]$ ls
DS_Connector.py  m.sh*  run.sh*
[ufrg@ufrg01 DS]$ ./run.sh
```

７．rtc-link 上でコンポーネントが接続され，Activate されたことを確認する．



（エ）複数の Naming Service を使用する場合

| |
|---|
| Naming Service 1 ：ホスト名：ufrg01，ユーザ名：ufrg<br>Naming Service 2 ：ホスト名：ufmrp02-arm，ユーザ名：ufrg |

1．Naming Service 1 を起動する．
（例では，ホスト ufrg01 ユーザ ufrg が Naming Service を起動）

```
[ufrg@ufrg01 ufrg]$ rtm-naming
```

2．SimpleServiceProvider を起動する．

```
[ufrg@ufrg01 SimpleService]$ ./MyServiceProviderComp -f rtc.conf
```

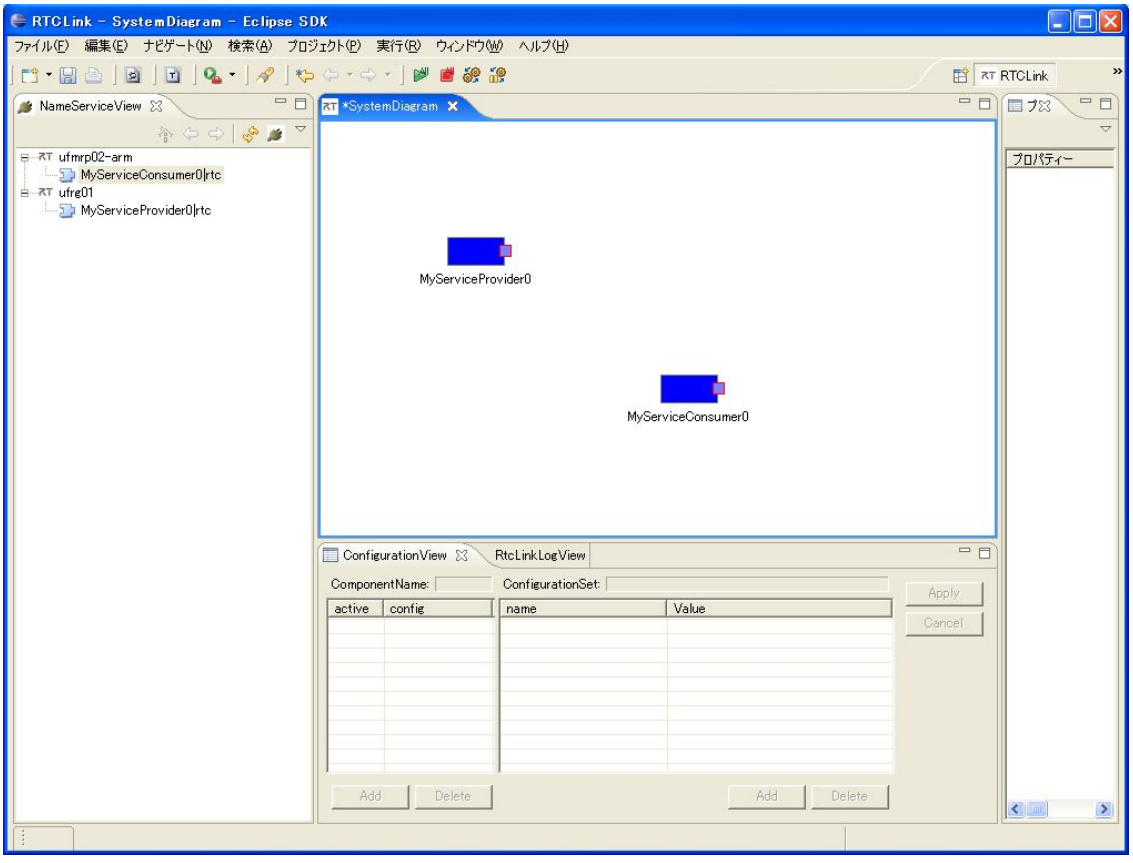
3．NamingService 2 を起動する．
（例では，ホスト ufmrp02-arm ユーザ ufrg が Naming Service を起動）

```
[ufrg@ufmrp02-arm ufrg]$ rtm-naming
```

4．SimpleServiceConsumer を起動する．

```
[ufrg@ufmrp02-arm SimpleService]$ ./MyServiceConsumerComp -f rtc.conf
```

5．rtc-link 上で，各 Naming Service 上においてコンポーネントが起動されていることを確認する．



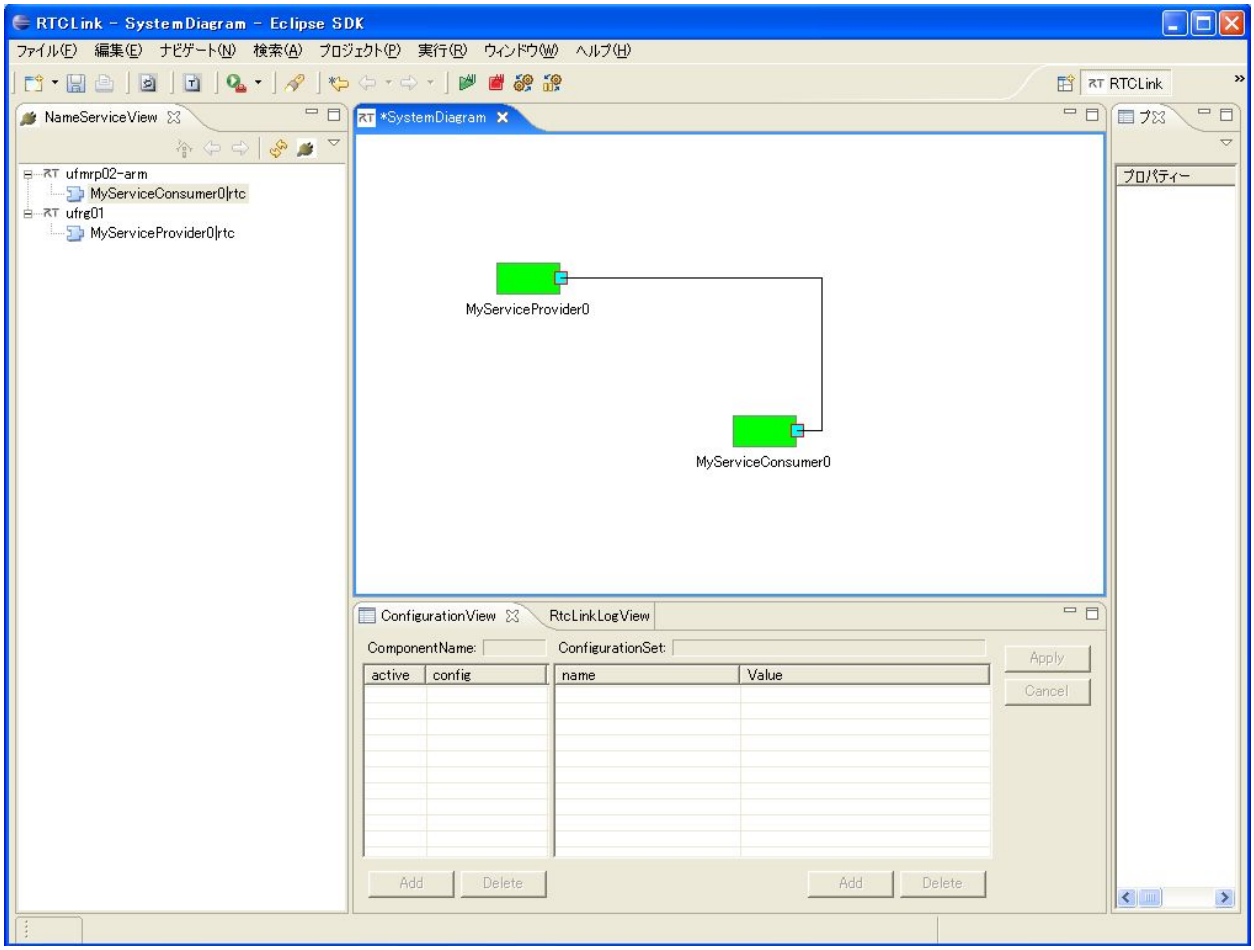
６．複数の NamingService 用モジュールのディレクトリへ移動する．

```
[ufrg@ufrg01 test]$ cd ConnectorModule/example/
[ufrg@ufrg01 example]$ ls
DS/  DataPort/  NamingServices/  ServicePort/
[ufrg@ufrg01 example]$ cd NamingServices/
```

７．python ファイルを実行する．

```
[ufrg@ufrg01 NamingServices]$ ls
NS_Connector.py  m.sh*  run.sh*
[ufrg@ufrg01 NamingServices]$ ./run.sh
```

８．rtc-link 上でコンポーネントが接続され，Activate されたことを確認する．



## ５．　実装方法

この章では，任意のコンポーネント用モジュールの実装方法を紹介します.
例として，DataPort モジュールのソースファイル DataConnector.py を示します.

```
from Connector_Header import *

#-------------------------------------------------------------
def get_comp_info():
    for j in range(len(comp_lists)):
        Components[j] = a.GetObjectReference(comp_lists[j+1])
        Lists[j]  = a.GetPortsObject(Components[j])
        Activity[j]  = a.GetExecutionContextServices(Components[j])

        print '\n---------------------------------------'
        print 'Component[',j,'] = [',a.GetComponentProfile(Components[j]).type_name,']'
        print '---------------------------------------'

        i = 0
        print 'PortLists of the [',comp_lists[j+1],'] are as follows\n'
        for p in Lists[i]:
            print 'Lists[',i,'] =',p.get_port_profile().name
            i = i+1
        print '\n'
#-------------------------------------------------------------

a = RTCConnector()

# Please write your naming service                                  A
#================================================
a.Connect2Nameservice("ufrg01.a02.aist.go.jp:2809")
#================================================
a.GetComponentLists()

# Please write your component lists in "comp_lists"                 B
#================================================
comp_lists = { 1:'ConsoleIn0', 2:'ConsoleOut0' }
#================================================

get_comp_info()

# Please ready the connect                                          C
#================================================
con_in  = Lists[0] # con_in = Outport of 'ConsoleIn'
con_out = Lists[1] # con_out= Inport  of 'ConsoleOut'

# Connect                                                           D
#================================================
a.PortConnect(con_in[0],con_out[0])

# Please ready the activate                                         E
#================================================
act_in  = Activity[0] # act_in  = ConsoleIn
act_out = Activity[1] # act_out = ConsoleOut

# Activate                                                          F
#================================================
act_in[0].activate_component(Components[0])
act_out[0].activate_component(Components[1])
```

事前準備：python ファイルを開く.

ここでは，DataPort のモジュールファイルを test ディレクトリにコピーして編集します.

```
[ufrg@ufrg01 example]$ ls
DS/  DataPort/  NamingServices/  ServicePort/
[ufrg@ufrg01 example]$ mkdir test
[ufrg@ufrg01 example]$ cp DataPort/* ./test/
[ufrg@ufrg01 example]$ cd test/
[ufrg@ufrg01 test]$ ls
DataConnector.py  m.sh*  run.sh*
[ufrg@ufrg01 test]$ jed DataConnector.py
```

(A)　　Naming Service の入力

使用するネーミングサービスを入力します.
下の例では”ufrg01.a02.aist.go.jp:2809”を入力しています.

```
# Please write your naming service
#================================================
a.Connect2Nameservice("ufrg01.a02.aist.go.jp:2809")
#================================================
```

(B)　　起動したコンポーネントリストの入力

起動した全コンポーネントを入力します．下の例では，Open-RTM-aist のサンプル SimpleIO なので，”ConsoleIn0” “ConsoleOut0”となっています.

```
# Please write your component lists in "comp_lists"
#================================================
comp_lists = { 1:'ConsoleIn0', 2:'ConsoleOut0' }
#================================================
```

(C)　　各ポートへの変数の割り当て

各ポートに任意の変数を割り当てます．下の例では，ConsoleIn コンポーネントの Inport を con_in，ConsoleOut コンポーネントの Outport を con_out としています．

```
# Please ready the connect
#================================================
con_in  = Lists[0] # con_in = Outport of 'ConsoleIn'
con_out = Lists[1] # con_out= Inport  of 'ConsoleOut'
```

(D)　　ポートの接続

(エ)で割り当てた各ポートの変数間を接続します．下の例では，ConsoleIn の con_in と CosoleOut の con_out を接続しています．

```
# Connnect
#================================================
a.PortConnect(con_in[0],con_out[0]);
```

(E)　　各コンポーネントへの変数の割り当て

各コンポーネントに任意の変数の割り当てを行います．下の例では，ConsoleIn コンポーネントを act_in，ConsoleOut コンポーネントを act_out としています．

```
# Please ready the activate
#================================================
act_in  = Activity[0] # act_in  = ConsoleIn
act_out = Activity[1] # act_out = ConsoleOut
```

(F)　　コンポーネントの起動方法

コンポーネントを Activate します．下の例では，ConsoleIn である act_in と ConsoleOut であるを act_out を Activate しています．

```
# Activate
#================================================
act_in[0].activate_component(Components[0])
act_out[0].activate_component(Components[1])
```

# ６．　Q&A

Q.　Windows 上でも使用可能ですか？
A.　可能です．WindowsXP 上にて動作確認致しました．

Q.　各ポートの接続を解除することはできますか？
A.　可能です．例えば DataConnector.py の場合，“a.Disconnect(con_in[0])”とすることで，ポート con_in[0]への接続を解除できます．

Q.　RT コンポーネントの作成方法を教えてください．
A.　RT ミドルウェア公式ページをご覧ください．
参考 URL:
http://www.is.aist.go.jp/rt/OpenRTM-aist/html/E3839EE3838BE383A5E382A2E383AB/RTE382B3E383B3E3839DE383BCE3838DE383B3E38388E4BD9CE68890.html

Q.　複数の Naming Service を使用する場合にはどうすればいいのでしょうか？
A.　使用例『（エ）複数の Naming Service を使用する場合』をご覧ください．
3 つ以上の Naming Service を使用する場合は，
<WorkDIR>/test/ConnectorModule/common/Connector_Header.py
にある配列を増やせば可能です．

Q.　多数(100 個以上)のコンポーネントを動かす場合にも対応可能なのでしょうか？
A.　デフォルトの設定では 20 個まで対応可能です．それ以上の場合は，
<WorkDIR>/test/ConnectorModule/common/Connector_Header.py
にある配列を増やせば可能です．

Q.　rtc-link を起動しないと，コネクタモジュールは使用できないのですか？
A.　いいえ，使用可能です．3 章 使用方法などでは，Connect や Activate の様子を明確に示すために rtc-link を起動していますが，rtc-link を起動しなくても使用可能です．

## ７．　補足（モジュールツリー）