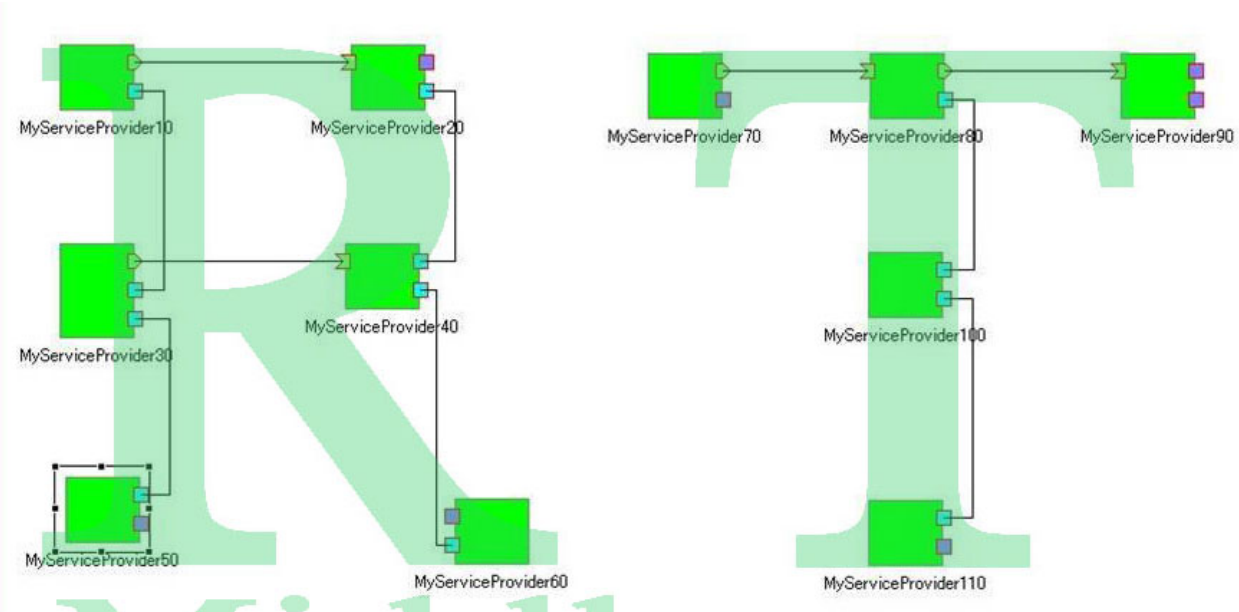
The tutorial of the ‘Connector Module’ for RT-Component

Version 1.0

(There is a Japanese version, too)



Contents

１．　Outline and Benefit of　‘Connector Module’

- This connector module can connect ports and control activating of components instead of RTC-link.

- You can make and write your connector module easily even though you don’t know python and CORBA.

- When you use this module, you can save time connect many components.

２．　Environment for development

This module was made by the following composition on November,2007.

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

３．　How to use ‘Connector Module’

This chapter shows how to use Connector Module.

※　hostname：ufrg01，username：ufrg

１．Get the ‘ConnectorModule.tar.gz’

２．Make an arbitrary directory（<WorkDIR>）

３．Unpack the ‘ConnectorModule.tar.gz’ in <WorkDIR>.

※　Ex.) Unpacking module in ‘test’ directory.

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

４．There are three directories in this ConnectorModule. There are 'common' 'example' and 'tutorial'. The detail of this module is shown in the Reference(page.18).

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

５．Move to an example directory in the Connector Module.

In example directory, there are four directories.

(Dataport, ServicePort, DS, NamingServices)

```
[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. Open another console. And, activate the sample components in OpenRTM-aist-0.4.1.

Ex.) : SimpleIO = ConsoleInComp, ConsoleOutComp.

7. Execute the module's shell after activating the component.

Ex.) : In case of SimpleIO

Execute the DataPort shell in DataPort directory.

４．　‘Connector Module’ Example

(A)　How to connect DataPort　　Ex.) SimpleIO

１．Activate your Naming Service.

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

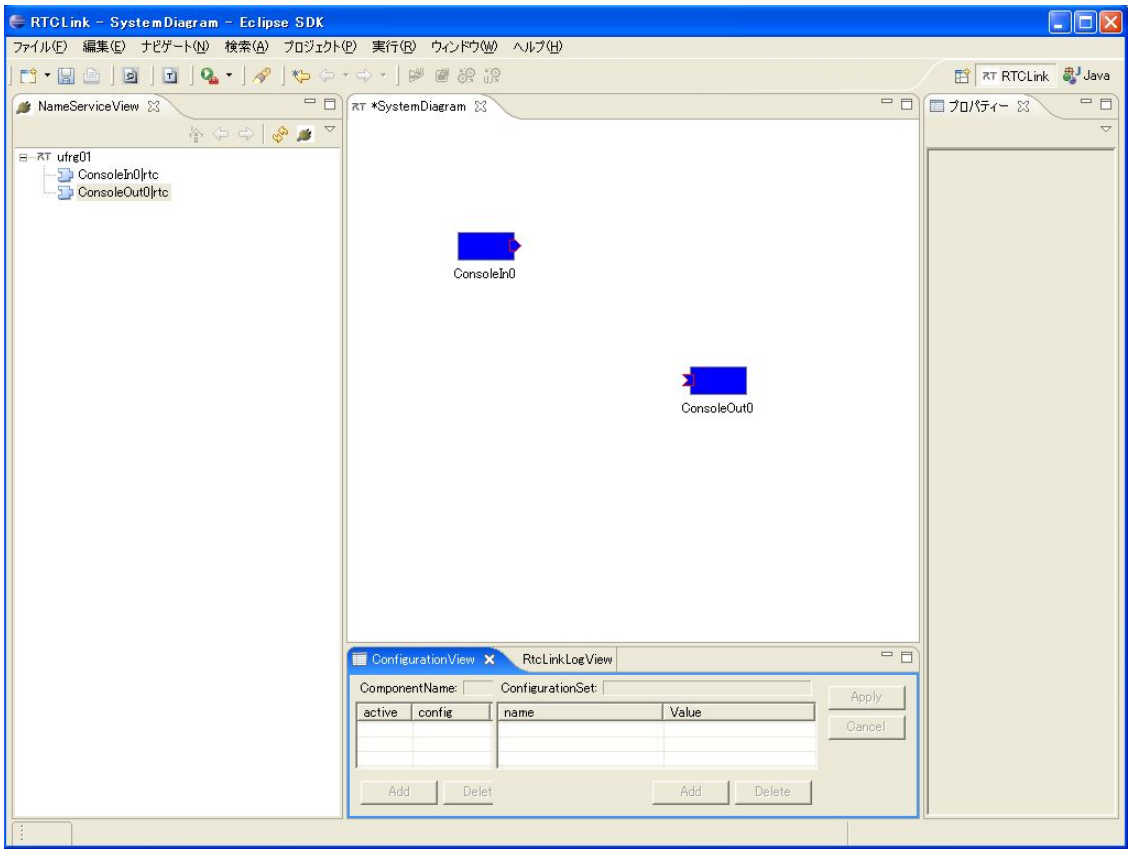
２．Activate the ConsoleIn component.

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

３．Activate the ConsoleOut component

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

４．Your can check the components on the rtc-link.



５．Move to the directory for DataPort directory.

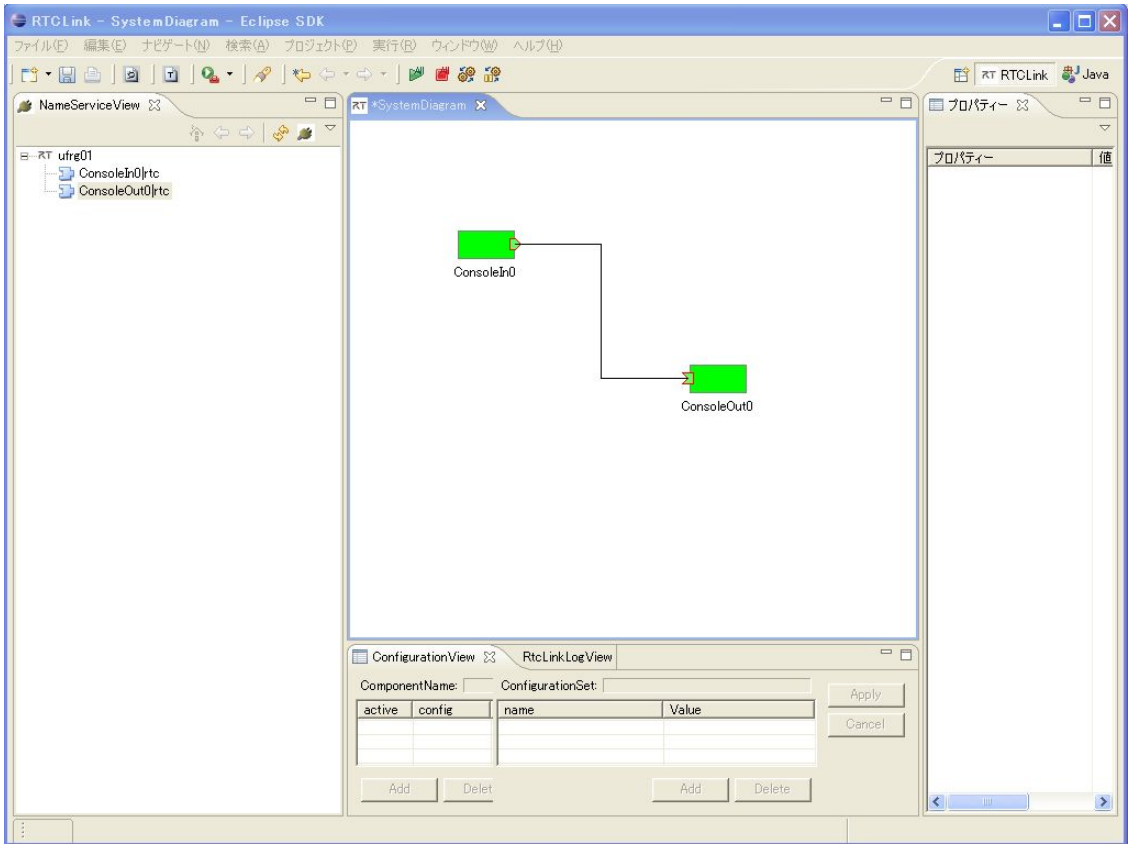
```
[ufrg@ufrg01 test]$ cd ConnectorModule/example/
[ufrg@ufrg01 example]$ ls
DS/  DataPort/  NamingServices/  ServicePort/
[ufrg@ufrg01 example]$ cd DataPort/
```

６．Execute the python file (run.sh file).

Then, the connector module connects ports and activates the components automatically.

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

７．You can check the connect and the activate on the rtc-link.



(B)　How to connect ServicePort　Ex.) SimpleService

1. Activate your Naming Service.

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

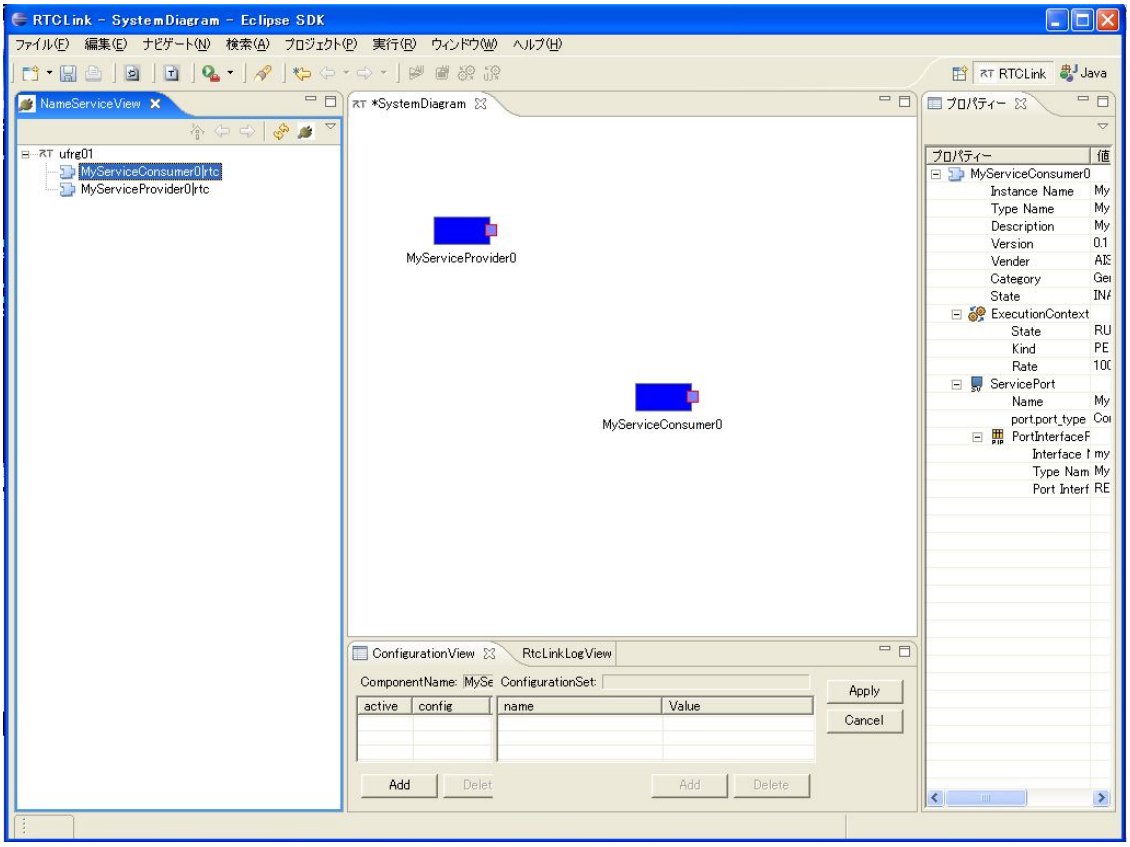
2. Activate the ‘SimpleServiceProvider’.

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

3. Activate the ‘SimpleServiceConsumer’.

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

4. You can check your components on the rtc-link.



５． Move to the directory for ServicePort directory.

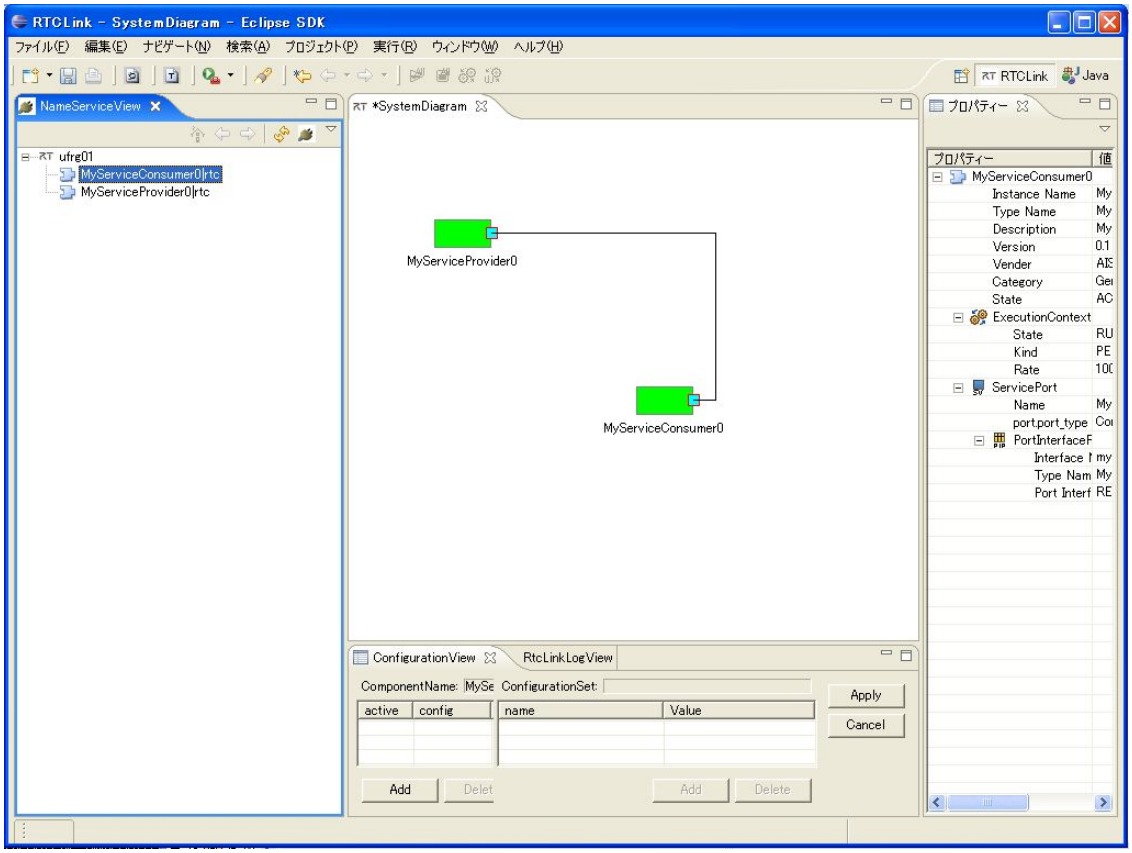
```
[ufrg@ufrg01 test]$ cd ConnectorModule/example/
[ufrg@ufrg01 example]$ ls
DS/  DataPort/  NamingServices/  ServicePort/
[ufrg@ufrg01 example]$ cd ServicePort/
```

６． Execute the python file (run.sh file).

Then, the connector module connects ports and activates the components automatically.

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

７． You can check the connect and the activate on the rtc-link.



(C) How to connect DataPort / ServicePort

※ You have to make components which have DataPort × 1 and ServicePort × 1 beforehand.

This example readied

MyServiceProvider which have OutPort × 1 and ServiceProvider × 1

MyServiceConsumer which have InPort × 1 and ServiceConsumer × 1

1. Activaete your Naming Service.

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

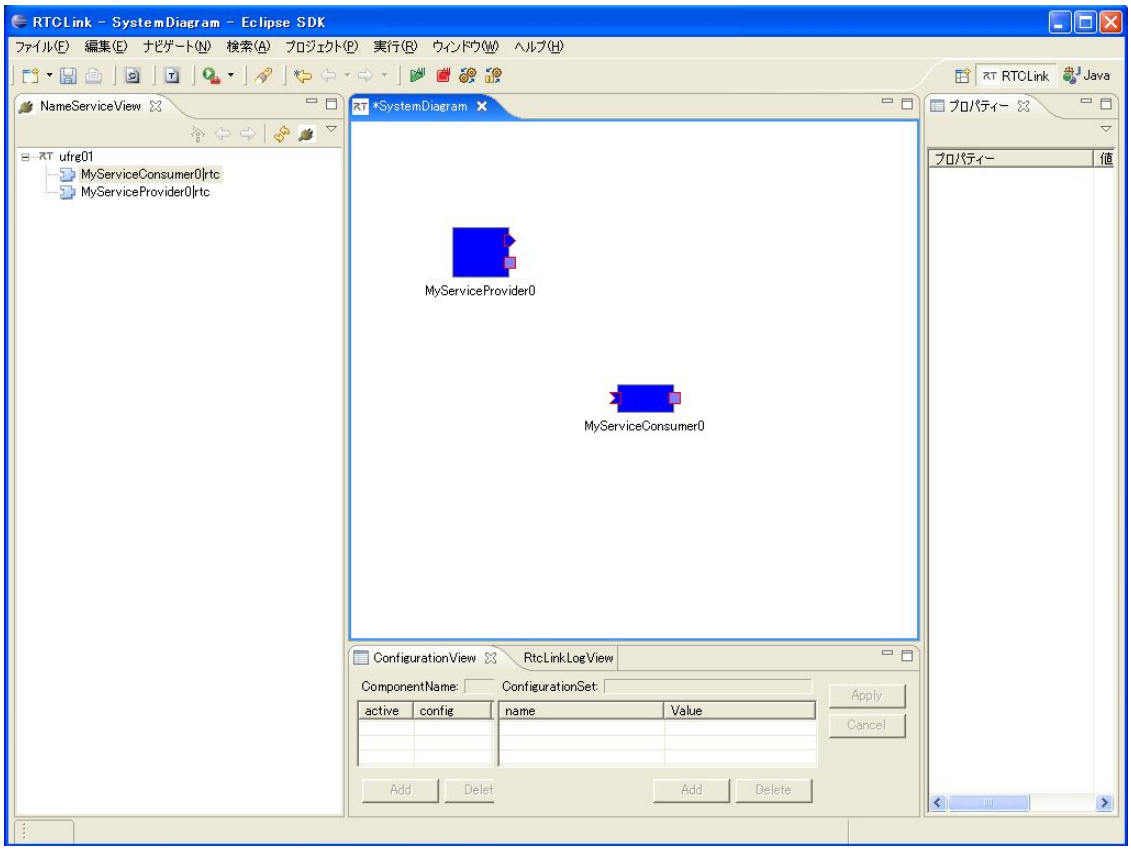
2. Activate the ‘SimpleServiceProvider’.

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

3. Activate the ‘SimpleServiceConsumer’.

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

4. You can check your components on the rtc-link.



５．Move the directory for DS directory.

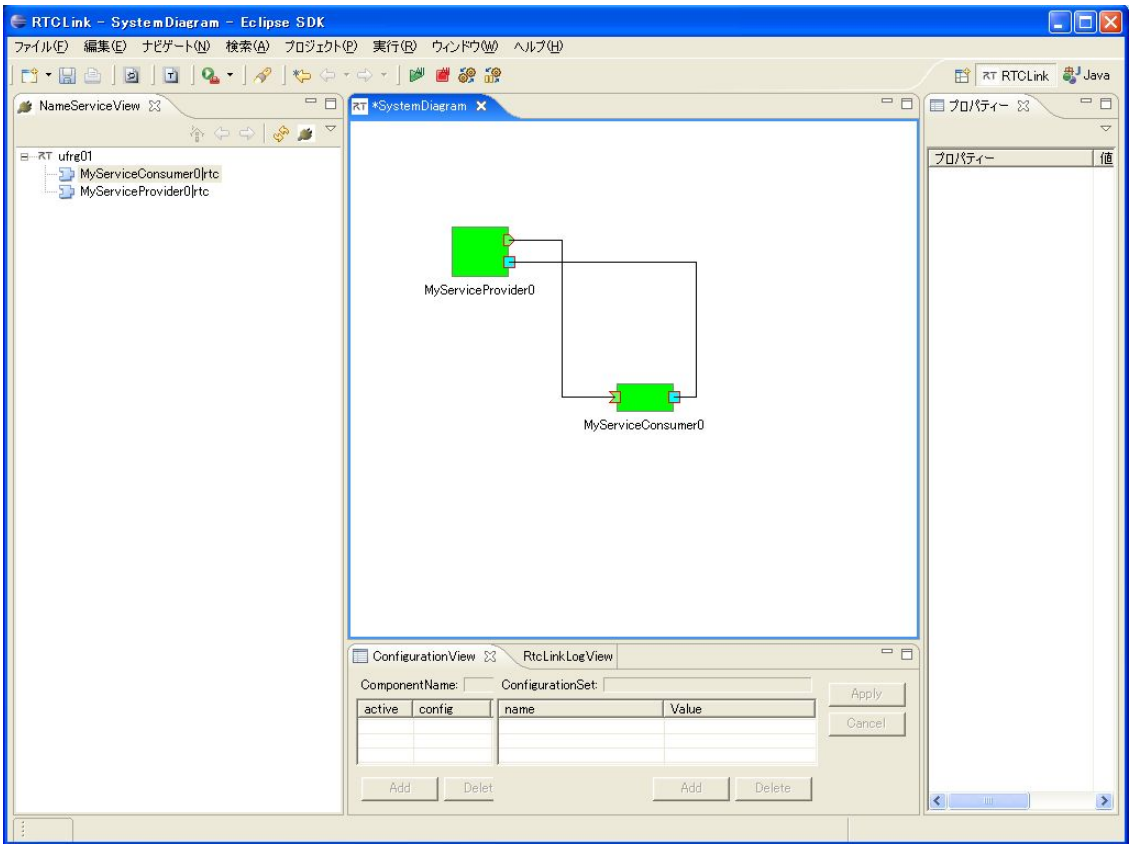
```
[ufrg@ufrg01 test]$ cd ConnectorModule/example/
[ufrg@ufrg01 example]$ ls
DS/  DataPort/  NamingServices/  ServicePort/
[ufrg@ufrg01 example]$ cd DS/
```

６．Execute python file (run.sh file).

Then, the connector module connects ports and activates the components automatically.

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

７．You can check the connecting and the activate on the rtc-link.



(D) Several Naming Services

Naming Service 1 : hostname : ufrg01, username : ufrg Naming Service 2 : hostname : ufmrp02-arm, username : ufrg

1. Activate the Naming Service 1.

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

2. Activate the 'SimpleServiceProvider' in ufrg01.

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

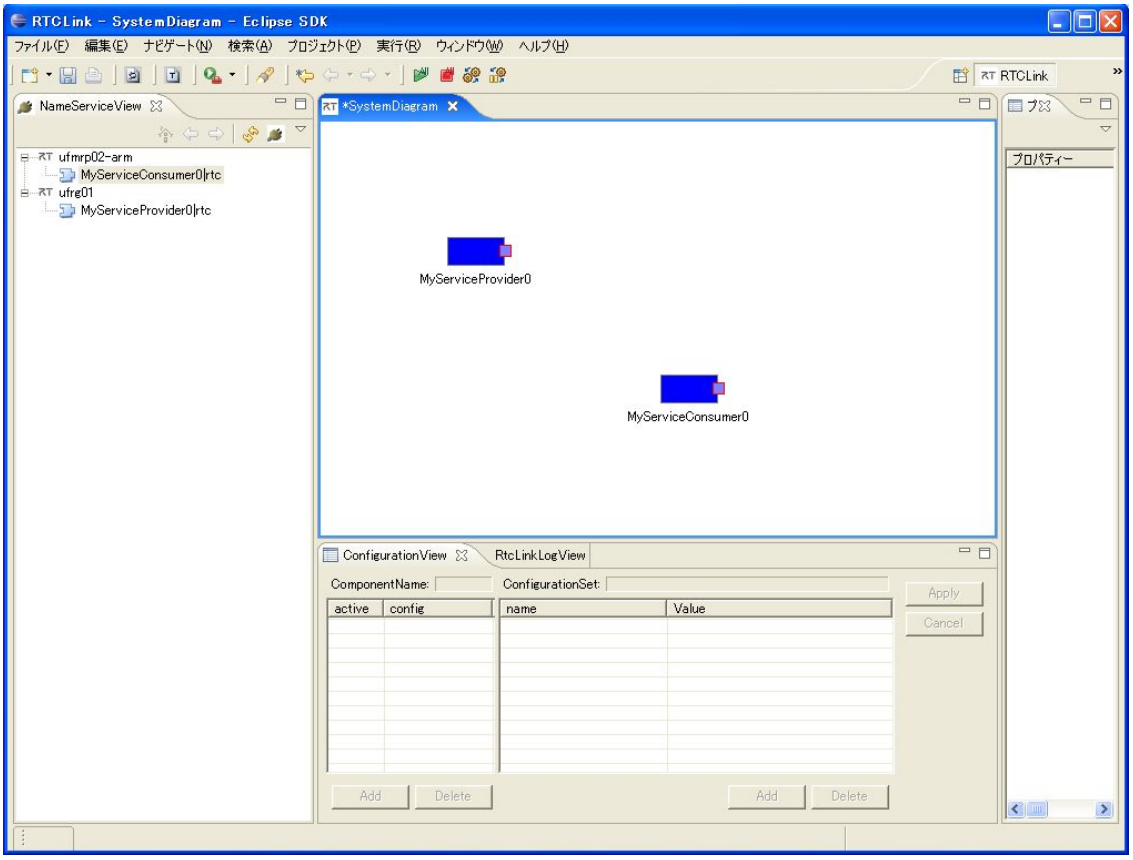
3. Activate the NamingService 2.

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

4. Activate the 'SimpleServiceConsumer' in ufmrp02-arm.

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

5. You can check component in each Naming Service on the rtc-link.



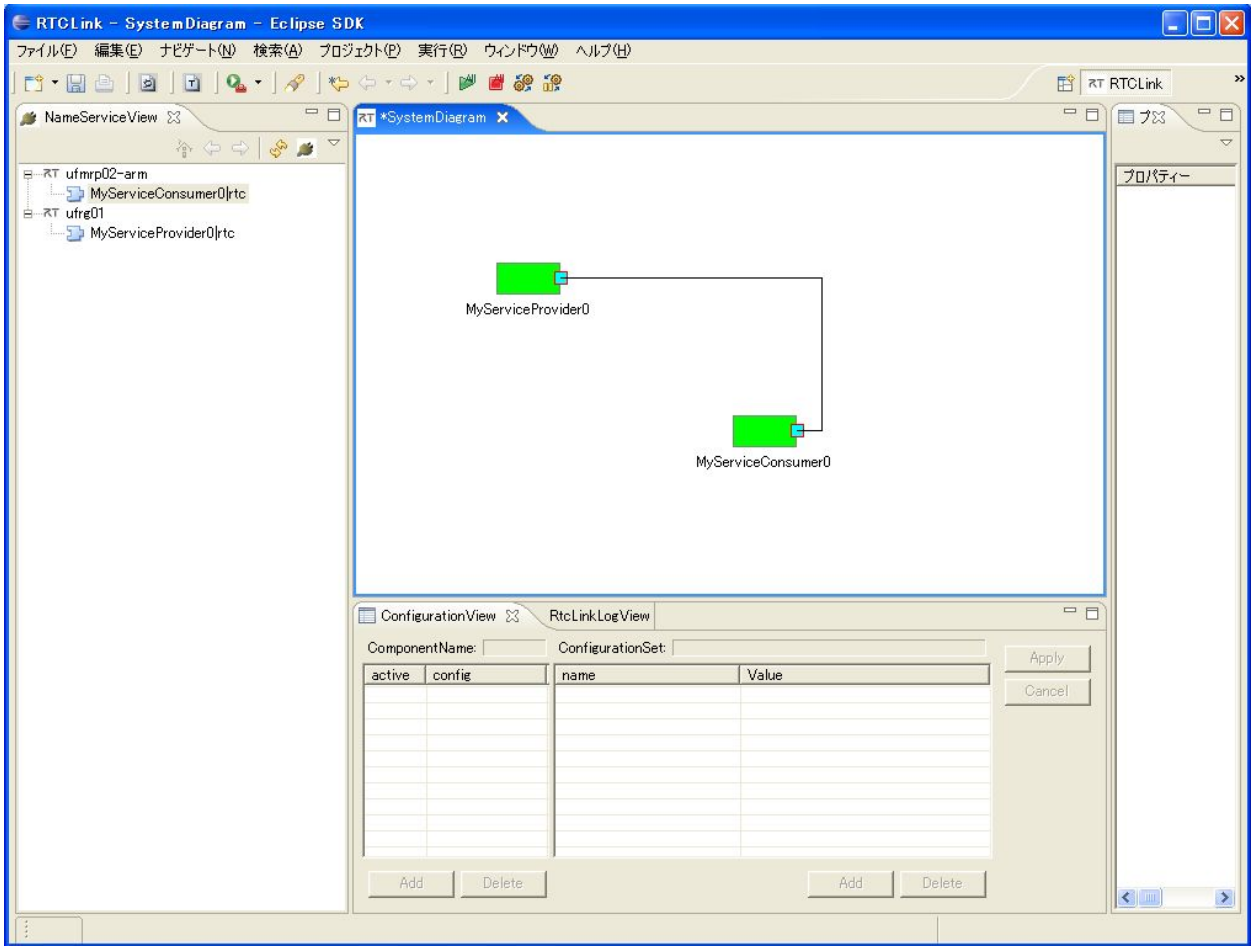
6. Move the directory for Naming Services directory.

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

7. Execute the python file (run.sh file).
   Then, the connector module connects ports and activates the components automatically.

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

8. You can check the connect and the activate on the rtc-link.



５．　How to make ‘Connector Module’.

This chapter shows how to make your connector module.

DataConnector.py for DataPort module is displayed as following.

Ready：Open the python file.

This example uses the DataPort module’s files.

```
[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) Definition of the Naming Service

Input your Naming Service.

This example inputs ”ufrg01.a02.aist.go.jp:2809”.

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

(B) Input of the component lists

Input your components.

This example inputs ‘ConsoleIn0’, ‘ConsoleOut0’ from SimpleIO.

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

(C) Input the variable of each port

Input the variable of each port.

This example defines, input by ‘con_in = outport of ConsoleIn’,outport by ‘con_out=inport of ConsoleOut’ respeaking.

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

(D) Connecting the port

Connect each port decided (C).

This example connects ‘con_in’ and ‘con_out’.

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

(E) Input of the variable for each port

Input the variable of each component.

This example inputs ‘act_in = ConsoleIn’, ’act_out = ConsoleOut’

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

(F) Activating the component

Activate the components.

This example activates ‘act_in’ and ‘act_out’.

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

6. Question & Answer

- Can I use this module at the Windows system?
  - Yes, you can. I checked at the WindowsXP.

- Can this module disconnect each port?
  - Yes, it can.
  - Ex.) In case of DataConnector.py, if you write “a.Disconnect(con_in[0])”, you can disconnect the port ‘con_in[0]’.

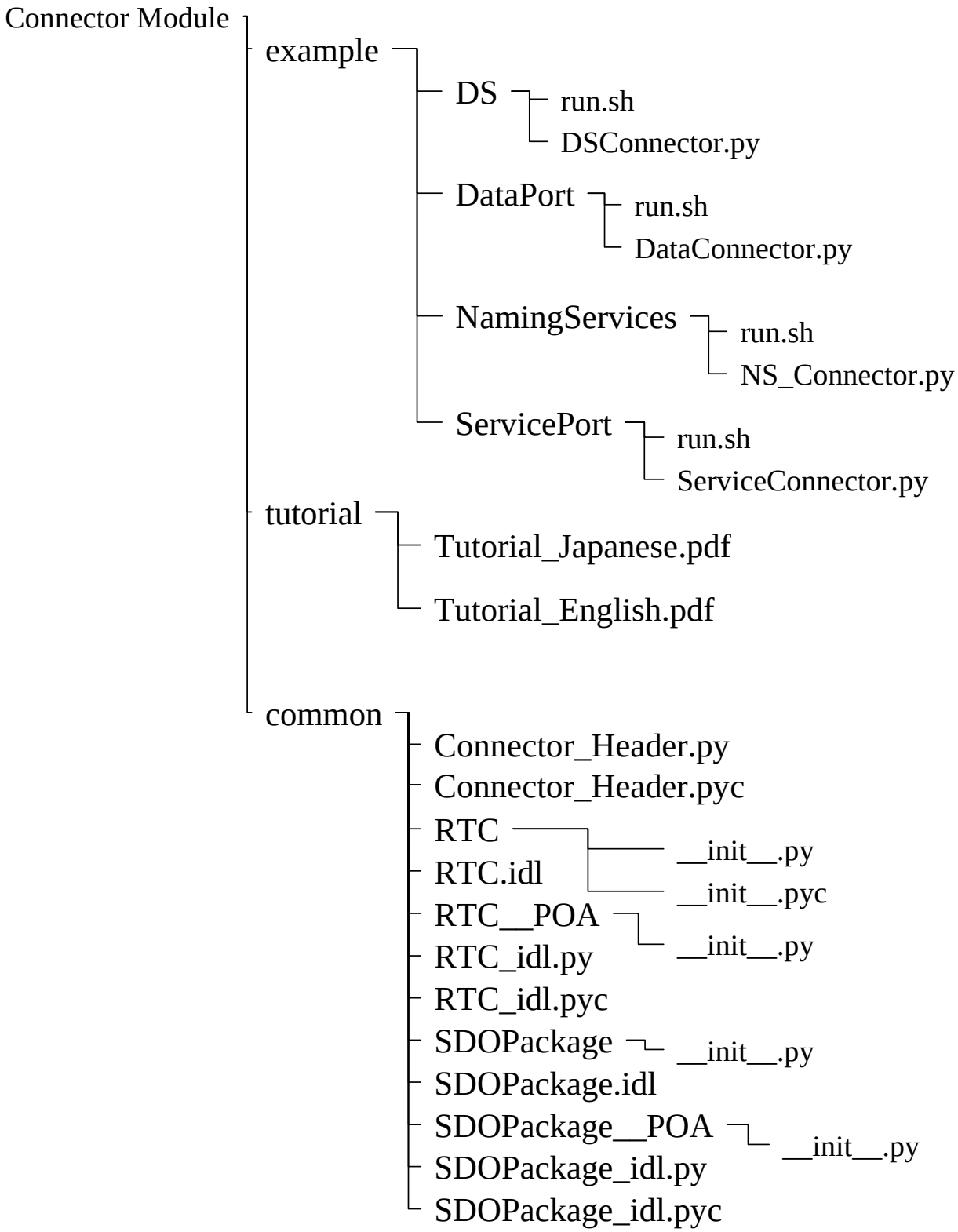
- Would you tell me how to make the RT-Component?
  - Please check the Webpage of RT-Middleware
  - URL:
    http://www.is.aist.go.jp/rt/OpenRTM-aist/html/E3839EE3838BE383A5E382A2E383AB/RTE382B3E383B3E3839DE383BCE3838DE383B3E38388E4BD9CE68890.html

- Can this module use several Naming Service?
  - Please show the chapter 5 ’ (D) Several Naming Service’.
    If you want to use more than three Naming Services, you have to modify the array in the <WorkDIR>/test/ConnectorModule/common/Connector_Header.py

- Can this module use more than 100 components?
  - Yes, you can. But, default is until 20 components. If you run more than 20 components, you have to modify the array of the
  - <WorkDIR>/test/ConnectorModule/common/Connector_Header.py

-

7. Reference


Connector Module
example
DS
run.sh
DSConnector.py
DataPort
run.sh
DataConnector.py
NamingServices
run.sh
NS_Connector.py
ServicePort
run.sh
ServiceConnector.py
tutorial
Tutorial_Japanese.pdf
Tutorial_English.pdf
common
Connector_Header.py
Connector_Header.pyc
RTC
RTC.idl
RTC__POA
RTC_idl.py
RTC_idl.pyc
SDOPackage
SDOPackage.idl
SDOPackage__POA
SDOPackage_idl.py
SDOPackage_idl.pyc
__init__.py
__init__.pyc
__init__.py
__init__.py
__init__.py