

# 青梅商工会議所主催 RTM講習会

日時:2011年7月25日(月) 10:30~17:30

場所:産業技術総合研究所 中央第2 本部情報棟1F ネットワーク会議室



10:30- 11:15	<b>第1部:RTミドルウェアの概略紹介</b>
	担当:神徳 徹雄(産業技術総合研究所)
	概要:RTミドルウェア, RTコンポーネントの概要説明
11:15- 12:00	<b>第2部:RTミドルウェアの概略, 導入方法の紹介</b>
	担当:栗原真二(産業技術総合研究所)
	概要:サンプルシステムを用いた概略紹介. RTミドルウェアの導入方法について紹介
13:00- 14:00	<b>第3部:RTミドルウェアを用いたシステム構築方法の紹介</b>
	担当:坂本武志(株式会社グローバルアシスト)
	概要:簡単なサンプルの動作(実習1)基本的な仕組み, 機能の紹介
14:15- 15:15	<b>第4部:RTコンポーネントの作成方法の紹介</b>
	担当:坂本武志(株式会社グローバルアシスト), 栗原真二(産業技術総合研究所)
	概要:サンプルコンポーネントの作成(実習2)RTCの設計方法の紹介
15:30- 16:00	<b>第5部:OpenRTM-aistコマンドラインツール rtshellの利用方法</b>
	担当:Geoffrey Biggs(産業技術総合研究所)
	概要:rtshellの紹介と, システムの実行. ログのとりかた, ログの再生方法の紹介
16:15- 17:30	<b>第6部:RTミドルウェアの便利な機能</b>
	担当:片見 剛人(富士ソフト株式会社)
	概要:ネットワークで接続されたシステムの紹介(実習3)ホームページプロジェクト登録の例. 質疑応答

# 第3部 RTミドルウェアを用いたシステム構築方法 の紹介

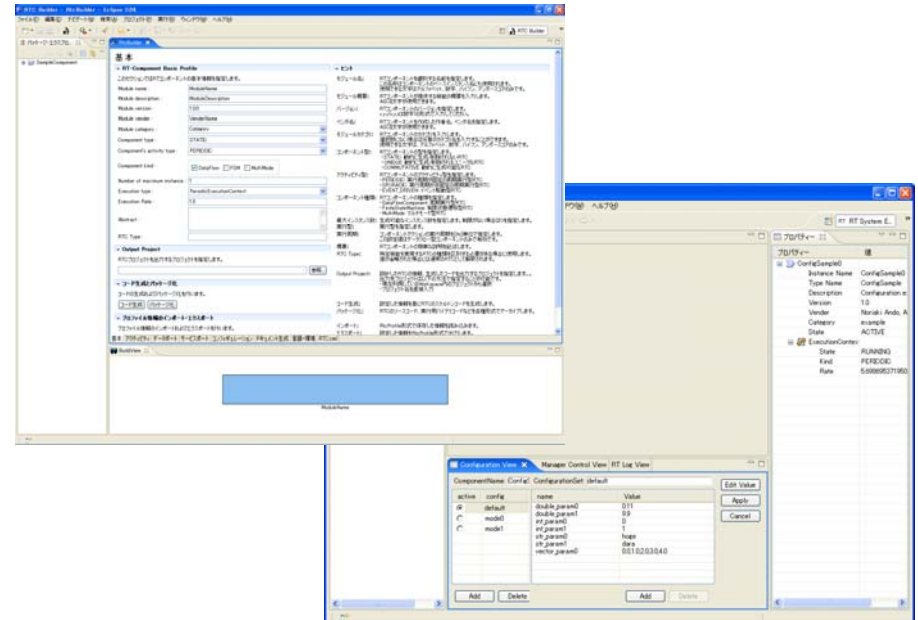
株式会社 グローバルアシスト  
坂本 武志



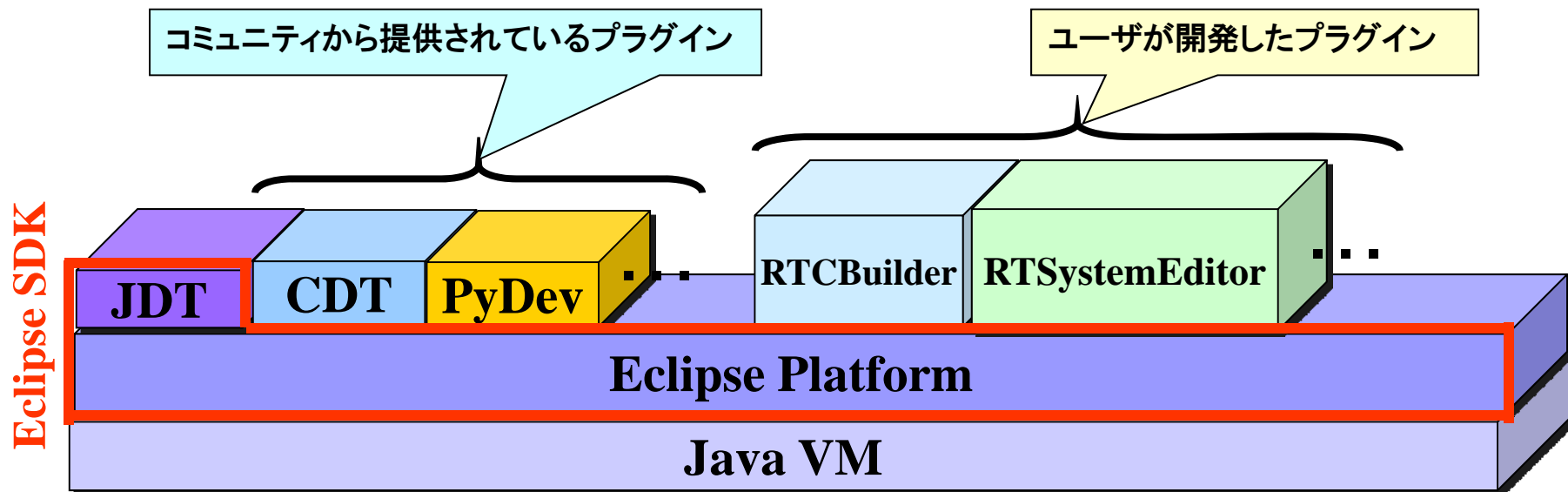
- 次世代ロボット知能ソフトウェアプラットフォーム
  - <http://www.openrtp.jp/wiki/>
  - システム設計, シミュレーション, 動作生成, シナリオ生成などをサポート
- OpenRT Platformツール群
  - コンポーネント開発, システム開発における各開発フェーズの作業支援
  - 開発プラットフォームにEclipseを採用

## ■ 構成

- **RTCビルダ**
  - **RTCデバッガ**
  - **RTシステムエディタ**
  - **ロボット設計支援ツール**
  - **シミュレータ**
  - **動作設計ツール**
  - **シナリオ作成ツール**
- など



- オープンソース・コミュニティで開発されている統合開発環境
  - マルチプラットフォーム対応. WindowsやLinuxなど複数OS上で利用可能
  - 「Plug-in」形式を採用しており, 新たなツールの追加, 機能のカスタマイズが可能
  - RCP(Rich Client Platform)を利用することで, 簡単に単独アプリ化が可能



# RTCBuilder, RTSystemEditorのインストール

## ■ ダウンロードし、解凍するだけ

- Javaの実行環境については、別途インストールが必要

The screenshot shows the OpenRTM-aist website. The main content area is titled "OpenRTM Eclipse tools 1.0-RELEASE". It includes a navigation menu on the left, a main text area with a globe icon, and a table of contents. The table of contents lists various packages, and a section titled "全部入りパッケージ" (All-in-one package) contains a table of download links for different operating systems and languages.

ホーム - ダウンロード - ツール - Eclipse tools 1.0-RELEASE

### OpenRTM Eclipse tools 1.0-RELEASE

これまで、OpenRTM-aistのツールとして開発されてきた RTCBuilder (旧RtcTemplate) および RTSystemEditor (旧 RtcLink) は、OpenHRP3やその他のツールと統合開発環境を構成する OpenRT Platform に組み込まれることになりました。こちらでは、RTSystemEditor 及び RTCBuilder のみを配布していますが、将来的には様々なツールを一括で提供する予定です。

現在の RTSystemEditor 及び RTCBuilder の最新バージョンは 1.0.0 です。

#### Table of contents

- 全部入りパッケージ
- バイナリ
- RTSystemEditor/RTCBuilderデベロップ
- Eclipse/JDK/JRE等
- 過去のバージョン

#### 全部入りパッケージ

Eclipse-3.4.2 [Ganymede SR2]		
Eclipse3.4.2+RTSE+RTCB Windows用全部入り	eclipse342_rtmttools100release_win32_ja.zip MD5:A52450B24F0A1C59402D5340D9FA8D56	2010.06.01
Eclipse3.4.2+RTSE+RTCB (英語版) Windows用全部入り	eclipse342_rtmttools100release_win32_en.zip MD5:2A1895F0E01D874E35CDDC29EFDCE1DE7	2010.06.01
Eclipse3.4.2+RTSE+RTCB Linux用全部入り	eclipse342_rtmttools100release_linux_ja.tar.gz MD5:FD54B638BB72A351D92ACD22CD4099C7	2010.06.01
Eclipse3.4.2+RTSE+RTCB (英語版) Linux用全部入り	eclipse342_rtmttools100release_linux_en.tar.gz MD5:4B1F4ACEE7F8E99B9C36D08068DBE5E1	2010.06.01
Eclipse3.4.2+RTSE+RTCB MacOSX用全部入り	eclipse342_rtmttools100release_macosx_ja.tar.gz MD5:19277C8E1E672688347C6767B57D9D1F	2010.06.10

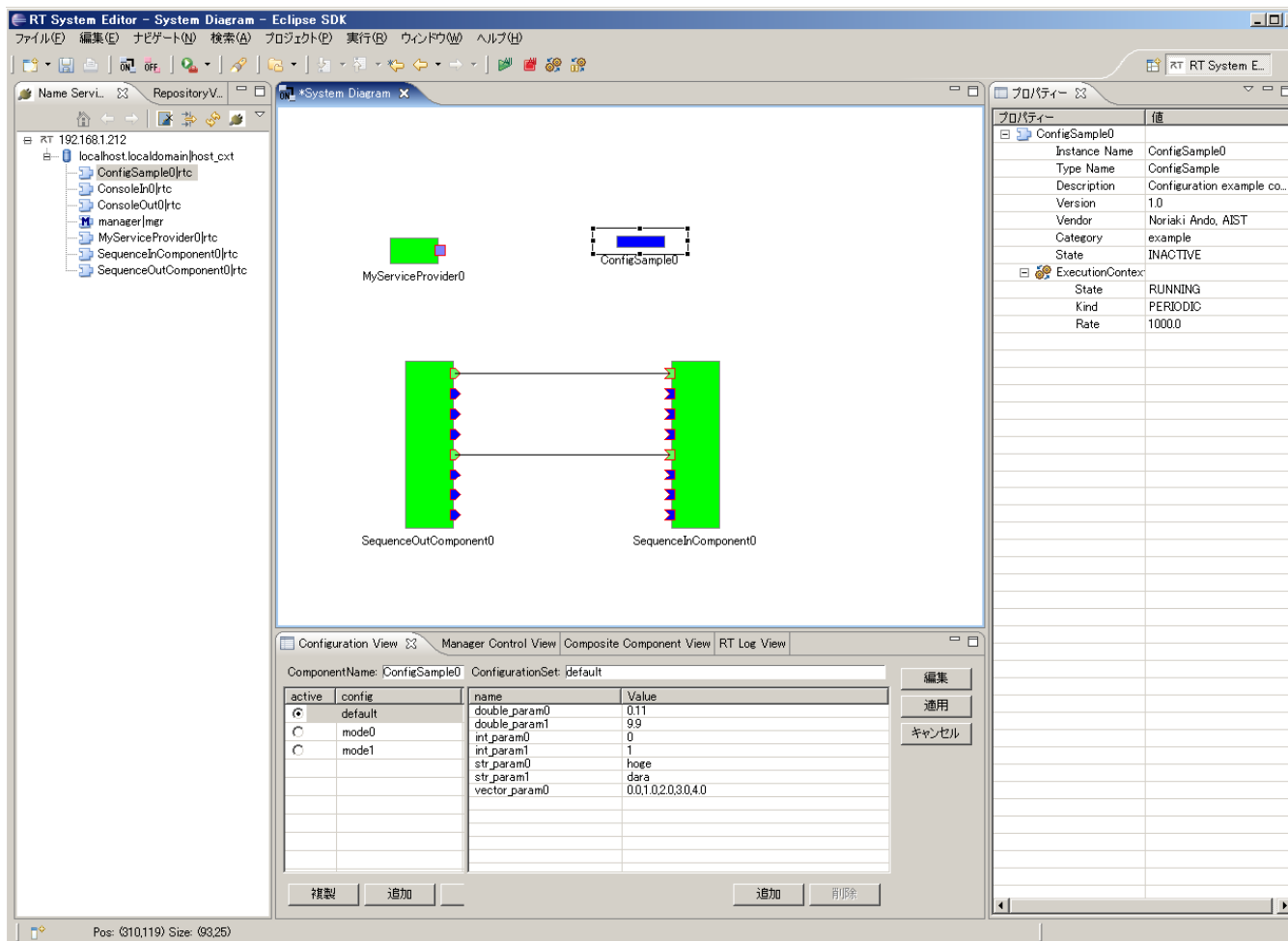
# RTSystemEditorについて

---



## ■ RTSystemEditorとは？

- RTコンポーネントを組み合わせて、RTシステムを構築するためのツール





# 画面構成

The screenshot shows the RT System Editor interface. The main window displays a System Diagram with components like MyServiceProvider0, ConfigSample0, and SequenceOutComponent0. A pink box labeled "システムエディタ" (System Editor) is overlaid on the diagram. To the left, a Name Service View shows a tree structure of components. To the right, a Property View shows configuration details for ConfigSample0.

プロパティ	値
Instance Name	ConfigSample0
Type Name	ConfigSample
Description	Configuration example co...
Version	1.0
Vendor	Moriaki Ando, AIST
Category	example
State	INACTIVE
ExecutionContext	
State	RUNNING
Kind	PERIODIC
Rate	10000

ネームサービスビュー

プロパティビュー

コンフィギュレーションビュー

The screenshot shows the Manager Control View with buttons for "Loadable Modules", "Loaded Modules", "Active Components", "Create", "Fork", and "Shutdown".

マネージャビュー

The screenshot shows the Composite Component View with a table of components and their ports.

component	port
ConsoleIn0	ConsoleIn0.out
ConsoleOut0	ConsoleOut0.in

コンジットコンポーネントビュー

The screenshot shows the Execution Context View with a table of execution contexts.

component	time	level	component	logger	message
owned0	2011-04-28	ERROR	Notify0	RTC	test log!
participate0	2011-04-28	ERROR	Notify1	RTC	test log!
	2011-04-28	ERROR	Notify0	RTC	test log!
	2011-04-28	ERROR	Notify1	RTC	test log!
	2011-04-28	ERROR	Notify0	RTC	test log!
	2011-04-28	ERROR	Notify1	RTC	test log!
	2011-04-28	ERROR	Notify0	RTC	test log!
	2011-04-28	ERROR	Notify1	RTC	test log!
	2011-04-28	ERROR	Notify0	RTC	test log!
	2011-04-28	ERROR	Notify1	RTC	test log!

実行コンテキストビュー

The screenshot shows the Log View with a table of log messages.

component	time	level	component	logger	message
owned0	2011-04-28	ERROR	Notify0	RTC	test log!
participate0	2011-04-28	ERROR	Notify1	RTC	test log!
	2011-04-28	ERROR	Notify0	RTC	test log!
	2011-04-28	ERROR	Notify1	RTC	test log!
	2011-04-28	ERROR	Notify0	RTC	test log!
	2011-04-28	ERROR	Notify1	RTC	test log!
	2011-04-28	ERROR	Notify0	RTC	test log!
	2011-04-28	ERROR	Notify1	RTC	test log!
	2011-04-28	ERROR	Notify0	RTC	test log!
	2011-04-28	ERROR	Notify1	RTC	test log!

ログビュー

## ■ Naming Serviceの起動

### ■ [スタート]メニューから

[プログラム]→[OpenRTM-aist 1.1]→[C++]→[tools]→[Start Naming Service]

## ■ CosoleInCompの起動

### ■ [スタート]メニューから起動

[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]

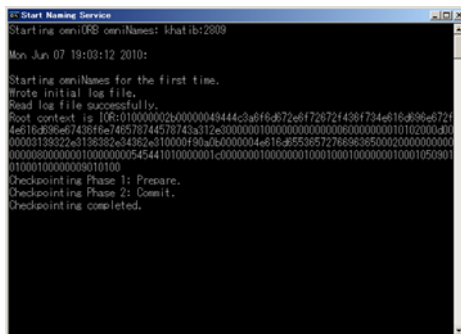
→[examples]→ [ConsoleInComp.exe]

## CosoleOutCompの起動

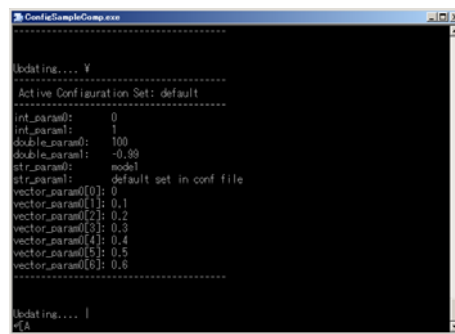
### [スタート]メニューから起動

[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]

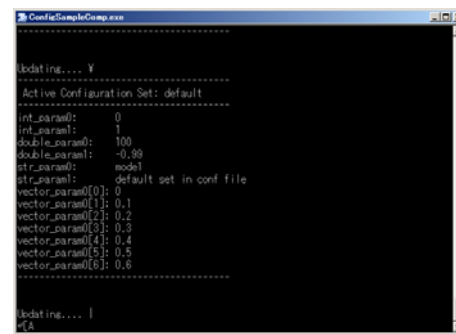
→[examples]→ [ConsoleOutComp.exe]



```
Start Naming Service
Starting omniORB omniNames: khatib:2309
Mon Jun 07 19:03:12 2010:
Starting omniNames for the first time.
Wrote initial ice files.
Read ice file successfully.
Root context is [OR:01000002b00000049444c3a6f6a672e6f1726721436f734e616a696e672f4e616d696e67436f6e746578744578743a312e3000001000000000000000000010102000d00000119322e3138232c34352c310000f90a7b5000004e816a95538572766983850002000000000000000001000000054544101000001c00000010000001000100000001000105090101000100000009010100]
Checkpointing Phase 1: Prepare.
Checkpointing Phase 2: Commit.
Checkpointing completed.
```



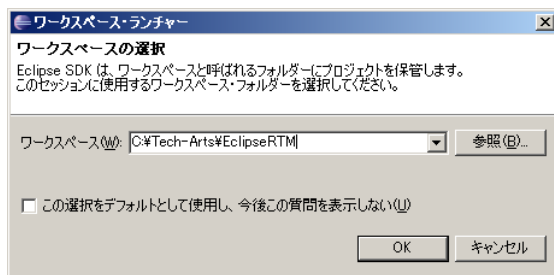
```
ConfigSampleComp.exe
Updating... V
-----
Active Configuration Set: default
-----
int_param0: 0
int_param1: 1
double_param0: 100
double_param1: -0.89
str_param0: model
str_param1: default set in conf file
vector_param0[0]: 0
vector_param0[1]: 0.1
vector_param0[2]: 0.2
vector_param0[3]: 0.3
vector_param0[4]: 0.4
vector_param0[5]: 0.5
vector_param0[6]: 0.6
-----
Updating... |
%A
```



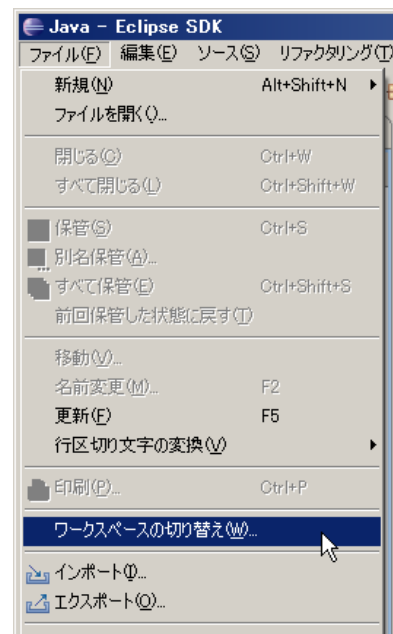
```
ConfigSampleComp.exe
Updating... V
-----
Active Configuration Set: default
-----
int_param0: 0
int_param1: 1
double_param0: 100
double_param1: -0.89
str_param0: model
str_param1: default set in conf file
vector_param0[0]: 0
vector_param0[1]: 0.1
vector_param0[2]: 0.2
vector_param0[3]: 0.3
vector_param0[4]: 0.4
vector_param0[5]: 0.5
vector_param0[6]: 0.6
-----
Updating... |
%A
```

- Windowsの場合
  - Eclipse.exeをダブルクリック
- Unix系の場合
  - ターミナルを利用してコマンドラインから起動
    - Ex) `$ /usr/local/Eclipse/eclipse`

## ■ ワークスペースの選択(初回起動時)



## ■ ワークスペースの切替(通常時)



### ※ワークスペース

Eclipseで開発を行う際の作業領域

Eclipse上でプロジェクトやファイルを作成するとワークスペースとして指定したディレクトリ以下に実際のディレクトリ, ファイルを作成する

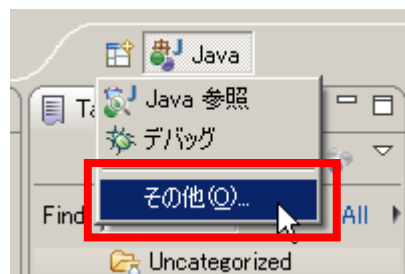
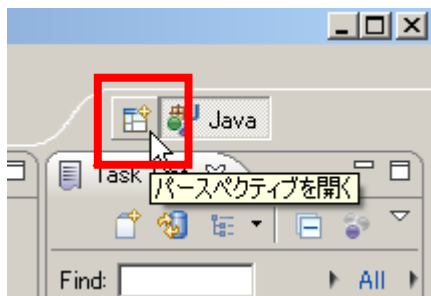
- 初期画面のクローズ
  - 初回起動時のみ



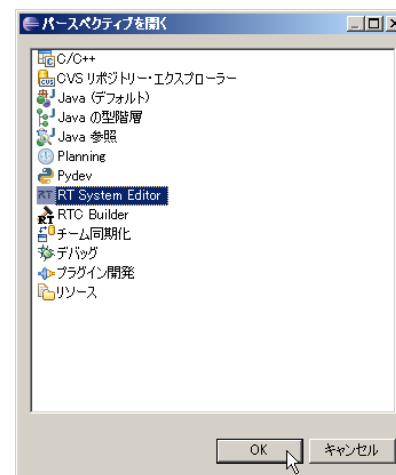
※パースペクティブ  
Eclipse上でツールの構成を管理する単位  
メニュー, ツールバー, エディタ, ビューなど  
使用目的に応じて組み合わせる  
独自の構成を登録することも可能

## ■ パースペクティブの切り替え

①画面右上の「パースペクティブを開く」を選択し、一覧から「その他」を選択

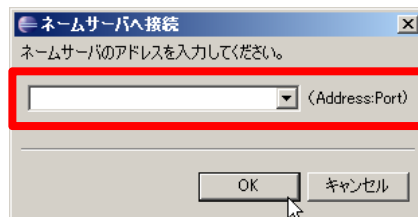
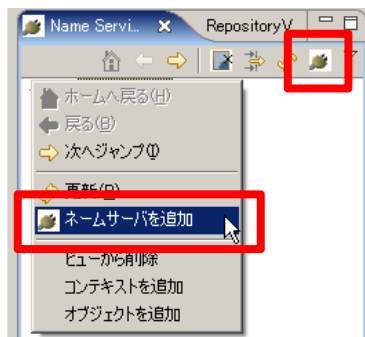


②一覧画面から対象ツールを選択

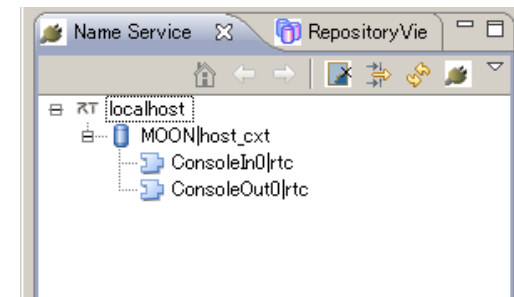


# RTシステム構築の基本操作

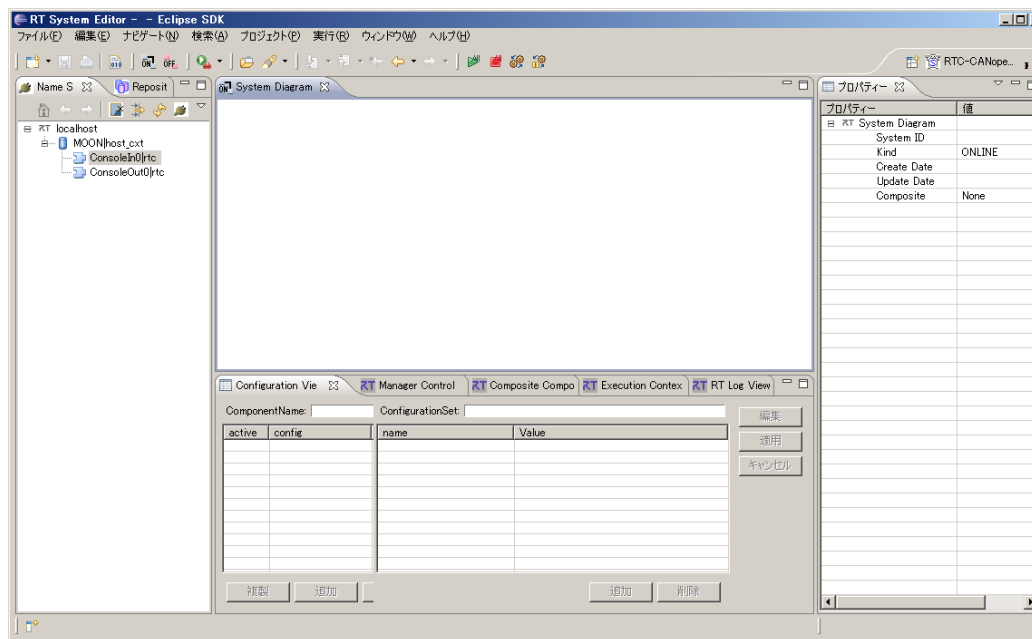
## ■ ネームサービスへ接続



※対象ネームサーバのアドレス、ポートを指定  
→ポート省略時のポート番号は  
設定画面にて設定可能

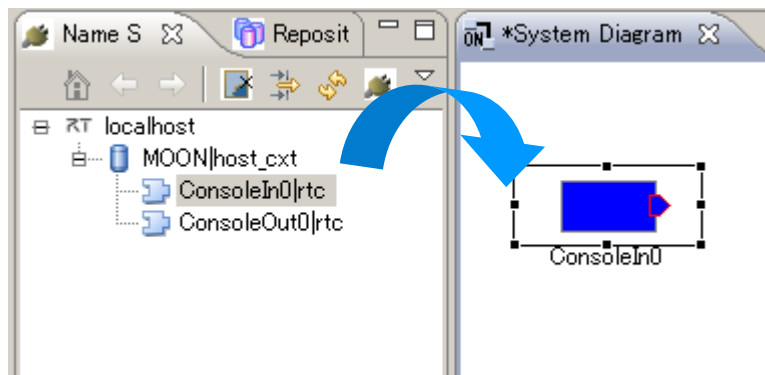


## ■ システムエディタの起動

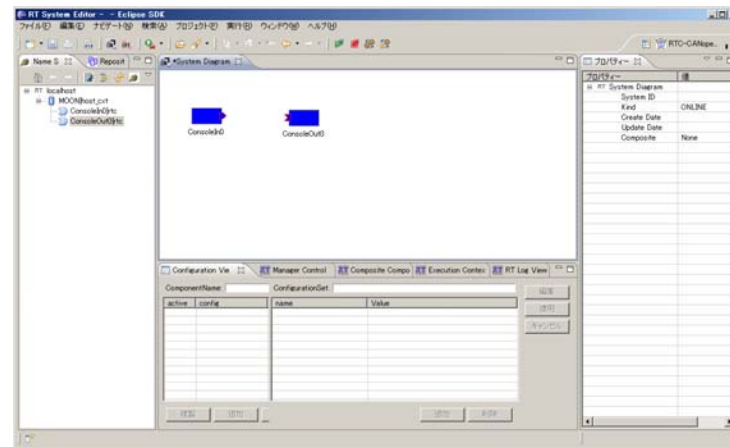


# RTシステム構築の基本操作

## ■ RTコンポーネントの配置

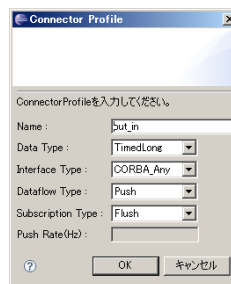
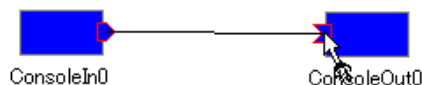


※ネームサービスビューから対象コンポーネントをドラッグアンドドロップ

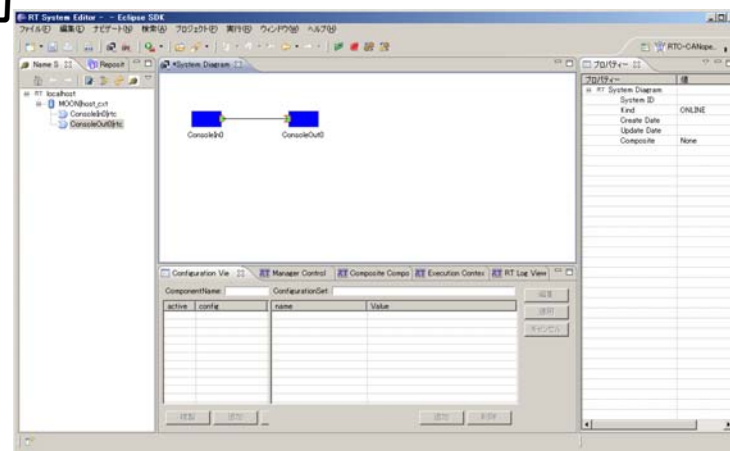


## ■ ポートの接続

①接続元のポートから接続先の②接続プロフィールを入力  
ポートまでドラッグ



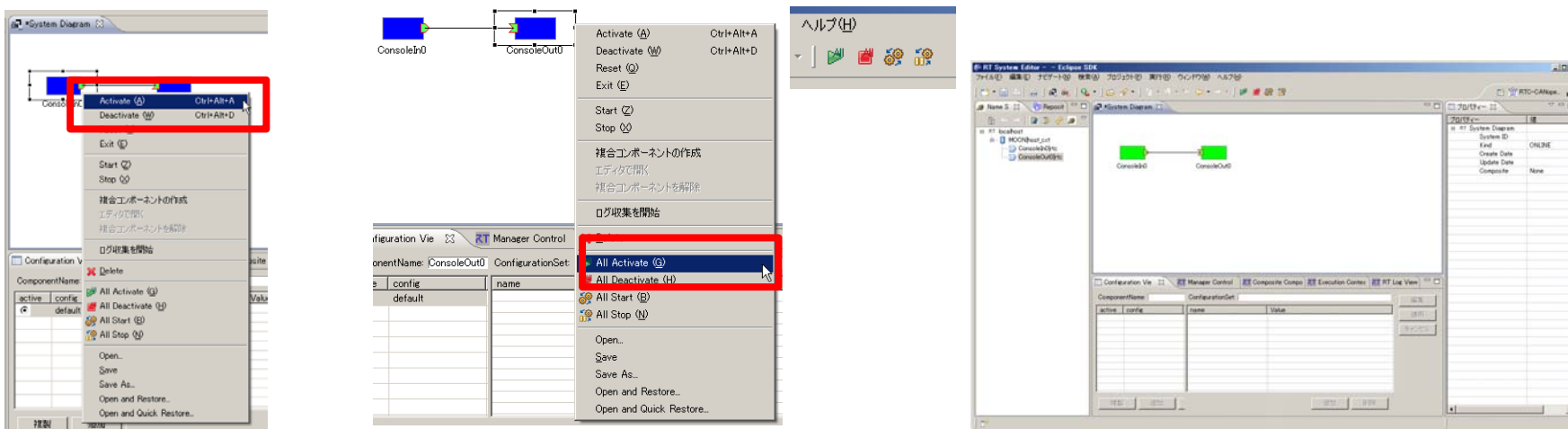
※ポートのプロパティが異なる場合など、  
接続不可能なポートの場合にはアイコンが変化



# RTシステム構築の基本操作

## ■ コンポーネントの起動

※各RTC単位で起動する場合 ※全てのRTCを一括で起動する場合



## ■ サンプルの実行

① ConsoleIn側で数字を入力

② ConsoleOut側が表示

```
ConsoleInComp.exe
dataport.interface_type: corba_cdr
-----
Connector Listener: ON_CONNECT
Profile:name: ConsoleIn0_out_ConsoleOut0.in
Profile:id: 2e25e2b7-cb49-43e1-8b0a-634f0a8dfc75
Profile:properties:
- data_type: IDL:RTC/TimeLong:1.0
- interface_type: corba_cdr
- dataflow_type: push
- subscription_type: flush
- publisher
- push_policy: all
- serializer
- cdr
- endian: little.big
- corba_cdr
- inport_ior: IOR:010000001a0000004944c3a4f70856e52544d2f496e506f72744364723a
812e30000000100000000000000000000001010200040000003139322e3136382e31312e390000
e090e00000f0e3eb194e000152e00000000400002000000000000000000000000000000000054
544101000001c00000010000000100010001000000010001050901010001000000009010100
-----
Please input number: 128
```

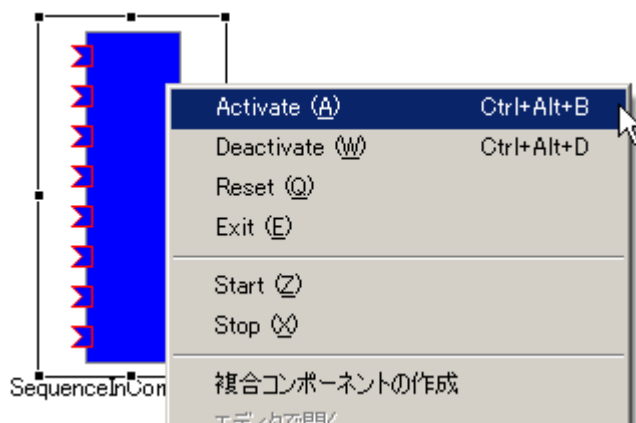
```
ConsoleOutComp.exe
- dataflow_type: push
- subscription_type: flush
- publisher
- push_policy: all
- serializer
- cdr
- endian: little.big
- provider:
-----
Data Listener: ON_RECEIVED
Profile:name: ConsoleIn0_out_ConsoleOut0.in
Profile:id: 2e25e2b7-cb49-43e1-8b0a-634f0a8dfc75
Data: 128
-----
Data Listener: ON_BUFFER_WRITE
Profile:name: ConsoleIn0_out_ConsoleOut0.in
Profile:id: 2e25e2b7-cb49-43e1-8b0a-634f0a8dfc75
Data: 128
-----
Received: 128
TimeStamp: 0[s] 0[ns]
```

※停止はDeactivateを実行

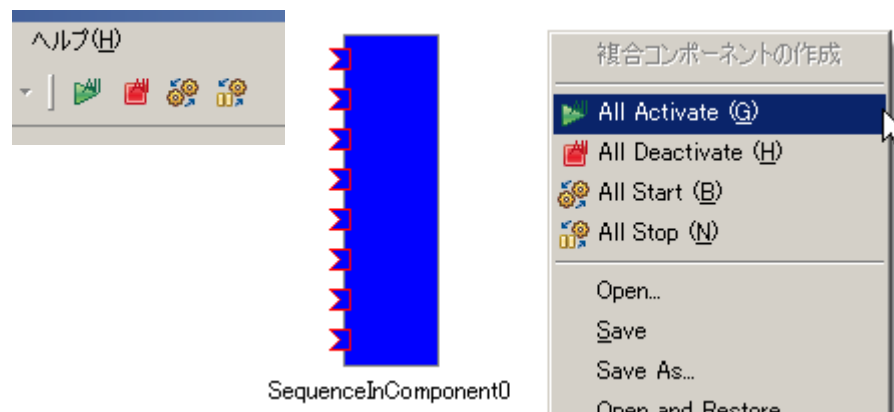
※RTC間の接続を切る場合には  
接続線をDelete  
もしくは、右クリックメニューから「Delete」を選択

アクション名	説明
Activate	対象RTCを活性化する
Deactivate	対象RTCを非活性化する
Reset	対象RTCをエラー状態からリセットする
Exit	対象RTCの実行主体(ExecutionContext)を停止し, 終了する
Start	実行主体(ExecutionContext)の動作を開始する
Stop	実行主体(ExecutionContext)の動作を停止する

## ■各コンポーネント単位での動作変更



## ■全コンポーネントの動作を一括変更



※ポップアップメニュー中でのキーバインドを追加

※単独RTCのActivate/Deactivateについては, グローバルはショートカットキ一定義を追加



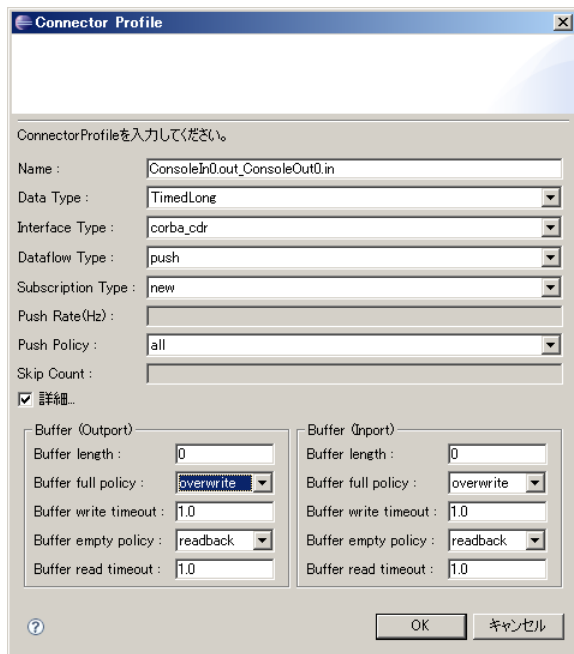
# 接続プロファイル(DataPort)について

項目	設定内容
Name	接続の名称
DataType	ポート間で送受信するデータの型. ex)TimedOctet, TimedShortなど
InterfaceType	データを送受信するポートの型. ex)corba_cdrなど
DataFlowType	データの送信方法. ex)push, pullなど
SubscriptionType	データ送信タイミング. <b>送信方法がPush</b> の場合有効. New, Periodic, Flushから選択
Push Rate	データ送信周期(単位:Hz). <b>SubscriptionTypeがPeriodic</b> の場合のみ有効
Push Policy	データ送信ポリシー. <b>SubscriptionTypeがNew, Periodic</b> の場合のみ有効. all, fifo, skip, newから選択
Skip Count	送信データスキップ数. <b>Push PolicyがSkip</b> の場合のみ有効

- SubscriptionType
  - New : バッファ内に新規データが格納されたタイミングで送信
  - Periodic : 一定周期で定期的にデータを送信
  - Flush : バッファを介さず即座に同期的に送信
- Push Policy
  - all : バッファ内のデータを一括送信
  - fifo : バッファ内のデータをFIFOで1個ずつ送信
  - skip : バッファ内のデータを間引いて送信
  - new : バッファ内のデータの最新値を送信(古い値は捨てられる)

# 接続プロファイル(DataPort)について

項目	設定内容
Buffer length	バッファの大きさ
Buffer full policy	データ書き込み時に、バッファフルだった場合の処理。 overwrite, do_nothing, blockから選択
Buffer write timeout	データ書き込み時に、タイムアウトイベントを発生させるまでの時間(単位:秒)
Buffer empty policy	データ読み出し時に、バッファが空だった場合の処理。 readback, do_nothing, blockから選択
Buffer read timeout	データ読み出し時に、タイムアウトイベントを発生させるまでの時間(単位:秒)



※OutPort側のバッファ, InPort側のバッファそれぞれに設定可能  
※timeoutとして「0.0」を設定した場合は、タイムアウトしない

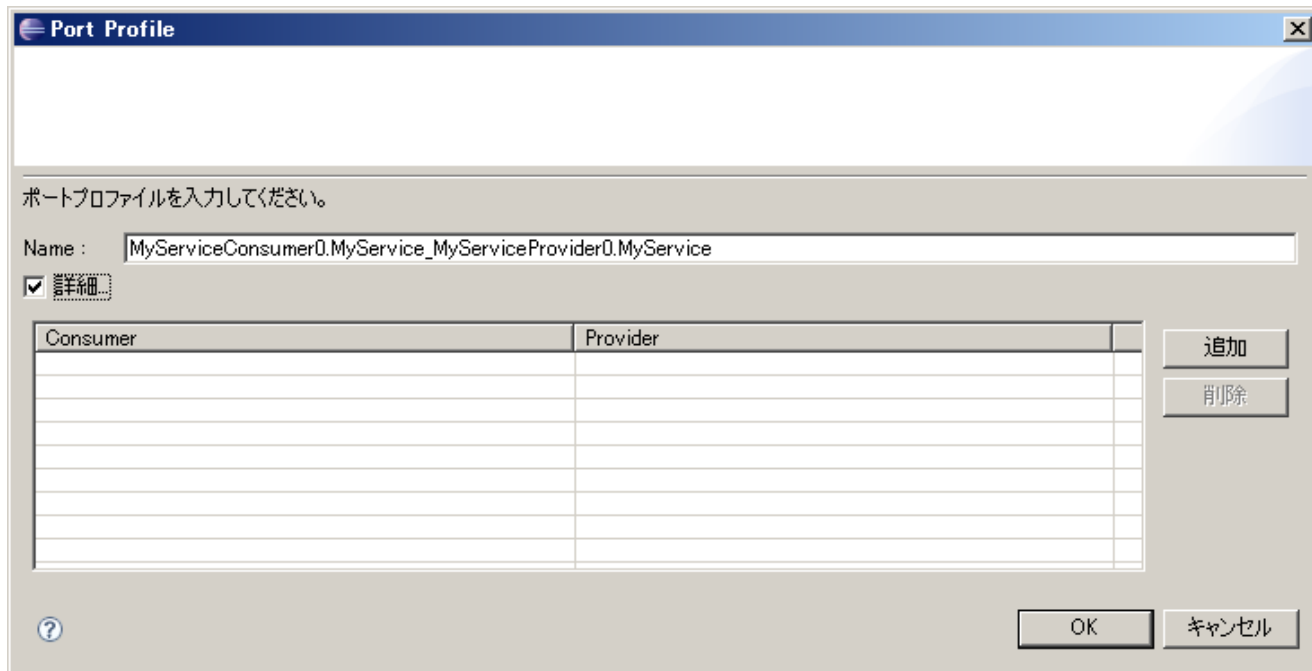
## ■ Buffer Policy

- overwrite : 上書き
- readback : 最後の要素を再読み出し
- block : ブロック
- do\_nothing : なにもしない

※Buffer Policy = Block+timeout時間の指定で、一定時間後読み出し/書き込み不可能な場合にタイムアウトを発生させる処理となる

# 接続プロファイル(ServicePort)について

項目	設定内容
Name	接続の名称
インターフェース情報	接続するインターフェースを設定。 接続対象のServicePortに複数のServiceInterfaceが定義されていた場合、どのインターフェースを実際に接続するかを指定



Port Profile

ポートプロファイルを入力してください。

Name : MyServiceConsumer0.MyService\_MyServiceProvider0.MyService

詳細

Consumer	Provider

追加  
削除

? OK キャンセル

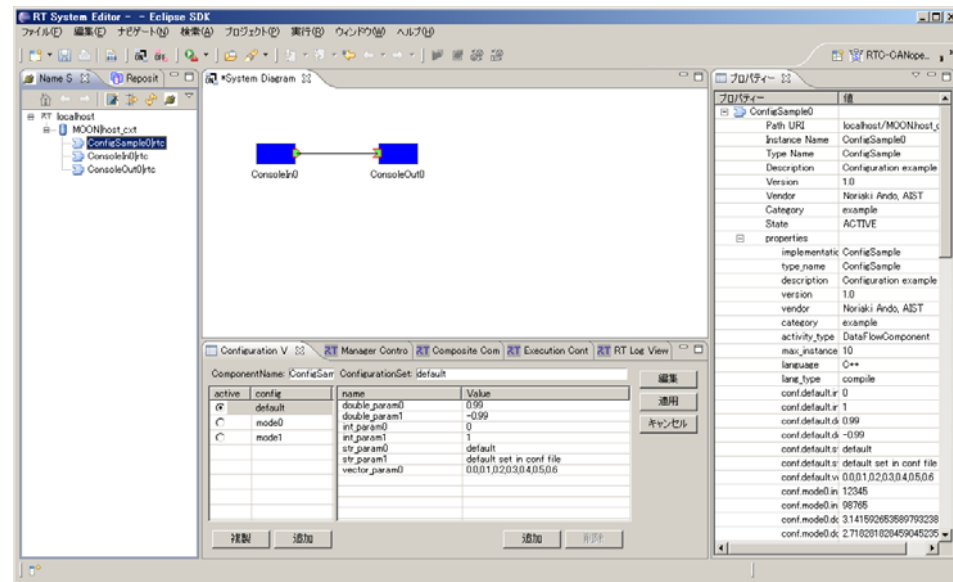
## ■ ConfigSampleCompの起動

### ■ [スタート]メニューから起動

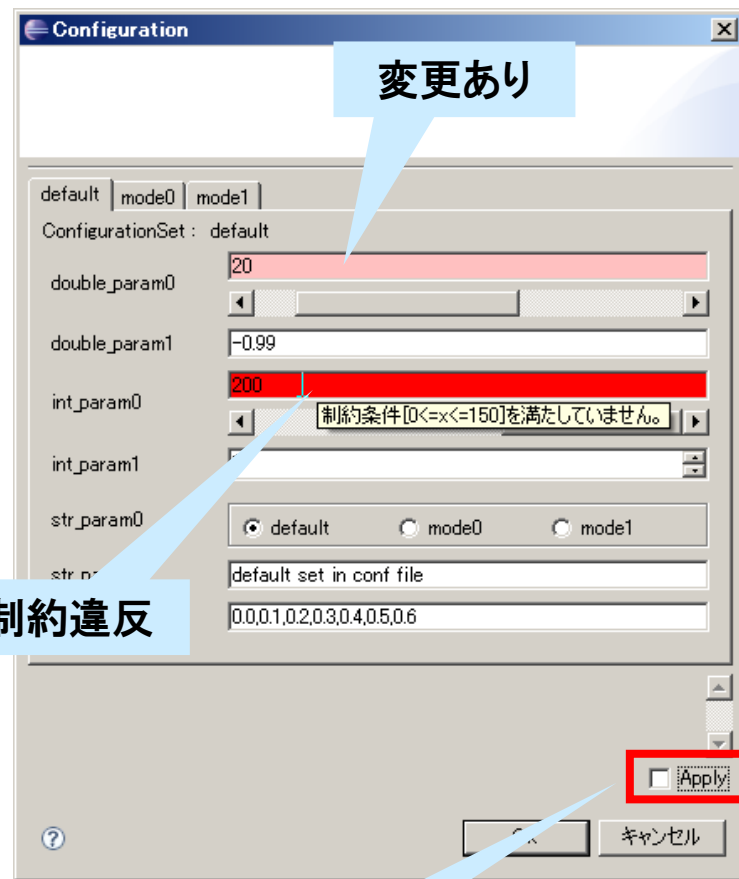
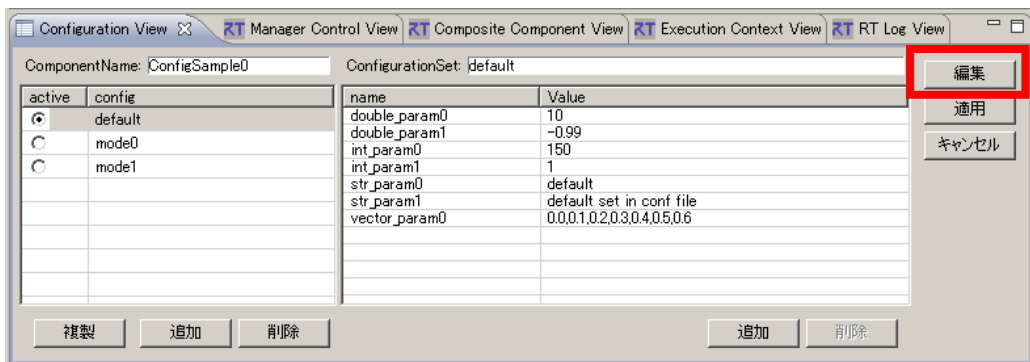
[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]

→[examples]→ [ConfigSampleComp.exe]

```
ConfigSampleComp.exe
-----
Updating... ¥
-----
Active Configuration Set: default
-----
int_param0:      0
int_param1:      1
double_param0:   100
double_param1:   -0.99
str_param0:      model
str_param1:      default set in conf file
vector_param0[0]: 0
vector_param0[1]: 0.1
vector_param0[2]: 0.2
vector_param0[3]: 0.3
vector_param0[4]: 0.4
vector_param0[5]: 0.5
vector_param0[6]: 0.6
-----
Updating... |
^C
```



## ■ RTコンポーネントのコンフィギュレーション情報の確認/編集



※「編集」ボタンにより、各種コントロールを用いた一括編集が可能

※「Apply」チェックボックスがONの場合、設定値を変更すると即座にコンポーネントに反映  
→テキストボックスからフォーカス外れる、ラジオボタンを選択する、スライダーを操作する、スピナを変更する、などのタイミング

※コンフィギュレーション情報を複数保持している場合、上部のタブで編集対象を切り替え

# コンフィギュレーション情報の設定方法

- rtc.conf内

[カテゴリ名]. [コンポーネント名]. config\_file: [コンフィギュレーションファイル名]

※例) example.ConfigSample.config\_file: configsample.conf

- コンフィギュレーションファイル内

- コンフィギュレーション情報

conf. [コンフィグセット名]. [コンフィグパラメータ名] : [デフォルト値]

※例) conf.mode0.int\_param0: 123

- Widget情報

conf. \_\_widget\_\_. [コンフィグパラメータ名] : [Widget名]

※例) conf.\_\_widget\_\_.str\_param0: radio

- 制約情報

conf. \_\_constraints\_\_. [コンフィグパラメータ名] : [制約情報]

※例) conf.\_\_constraints\_\_.str\_param0: (bar,foo,foo,dara)

conf. \_\_[コンフィグセット名]. [コンフィグパラメータ名] : [制約情報]

※例) conf.\_\_mode1.str\_param0: (bar2,foo2,dara2)

RTCの利用者が設定するのではなく、RTC開発者、RTC管理者が設定することを想定。

RTCBuilderを使用することで設定可能

## ■ CameraViewerCompの起動

### ■ [スタート]メニューから起動

[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]  
→[opencv-rtcs]→ [CameraViewerComp.exe]

## ■ OpenCVCameraCompの起動

### ■ [スタート]メニューから起動

[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]  
→[opencv-rtcs]→ [DirectShowCamComp.exe]

## ■ 画像処理用コンポーネントの起動

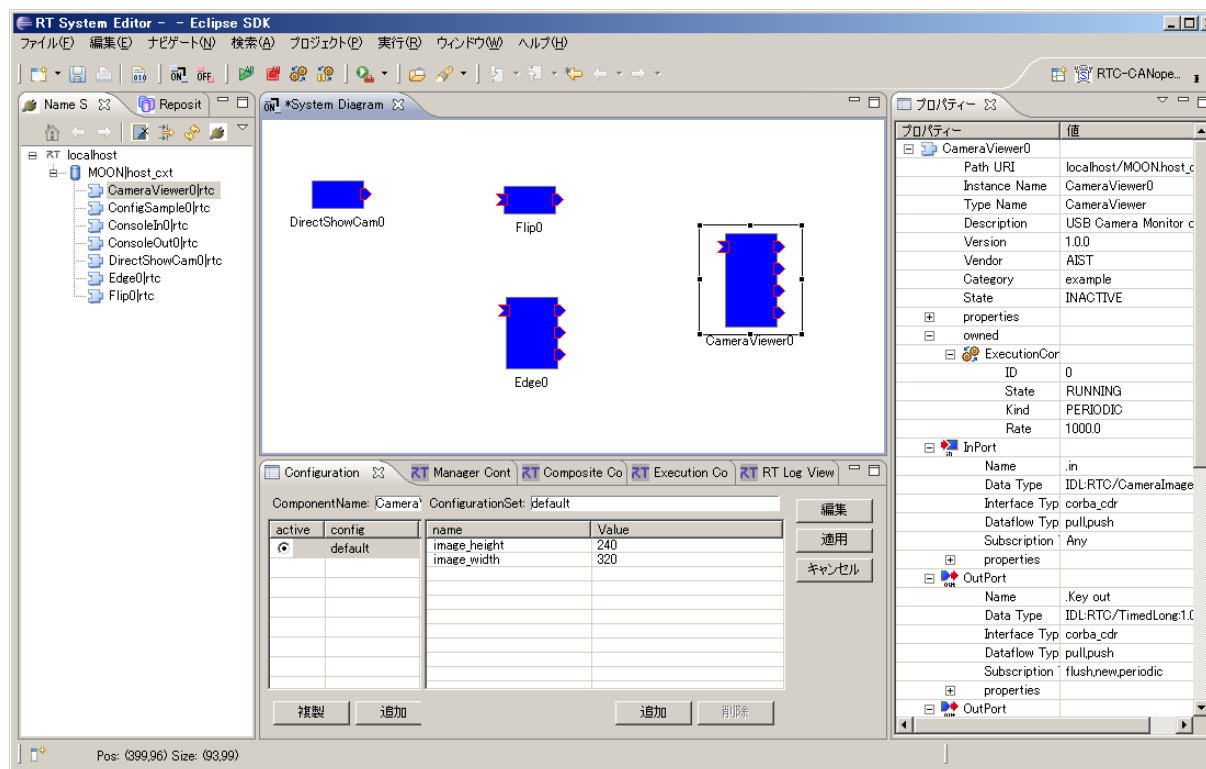
### ■ [スタート]メニューから起動

[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]  
→[opencv-rtcs]→ [FlipComp.exe]

[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]  
→[opencv-rtcs]→ [EdgeComp.exe]

# システムの構築

- 以下のコンポーネントをエディタ上に配置
  - DirectShowCam
  - Flip
  - Edge
  - CameraViewer

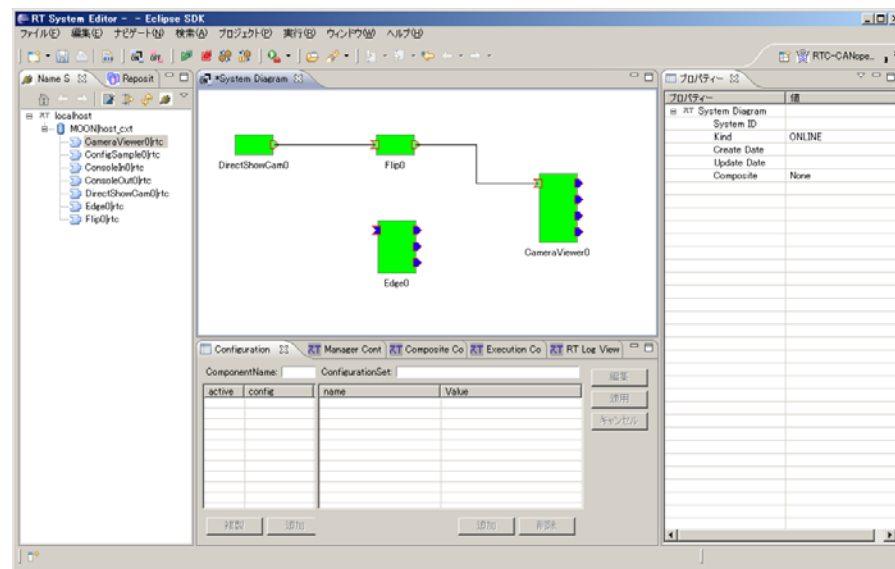




# システム構成の変更

## Flip側との接続

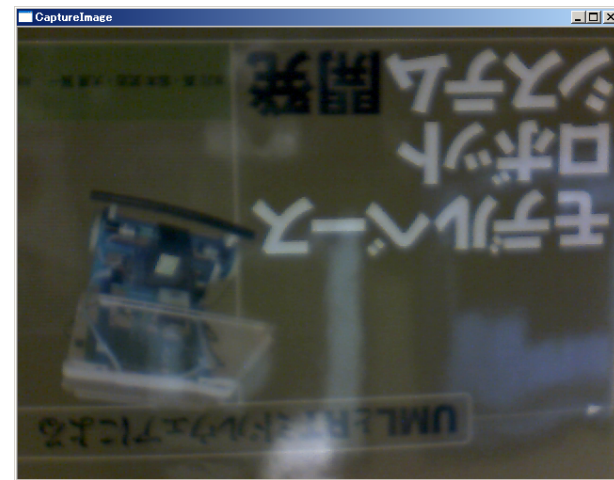
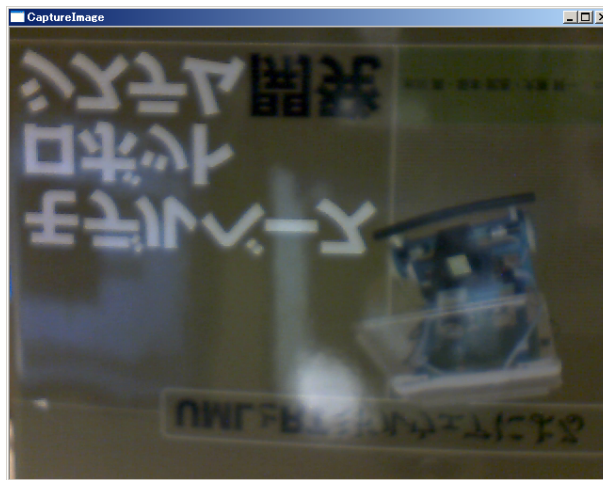
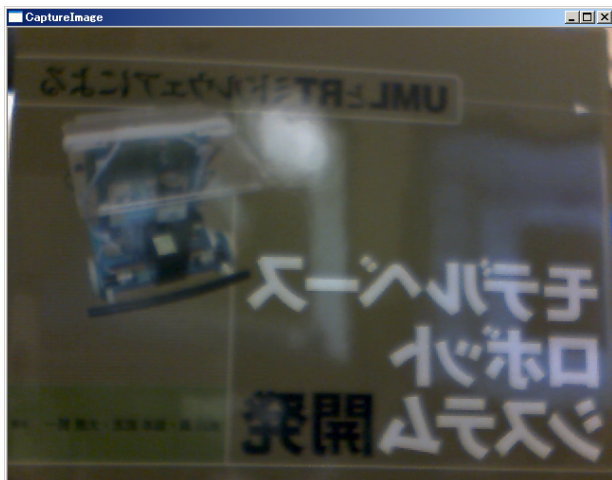
- DirectShowCam → Flip  
→ CameraViewerと接続  
(接続プロファイルはデフォルト設定)
- AllActivateを実行



flip\_mode=1

flip\_mode=0

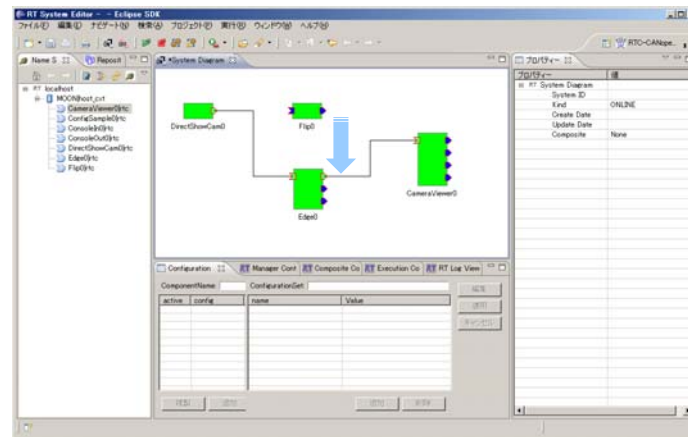
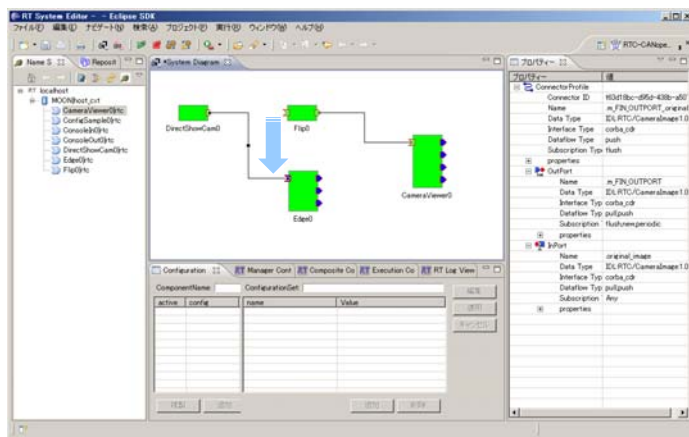
flip\_mode=-1



# システム構成の変更

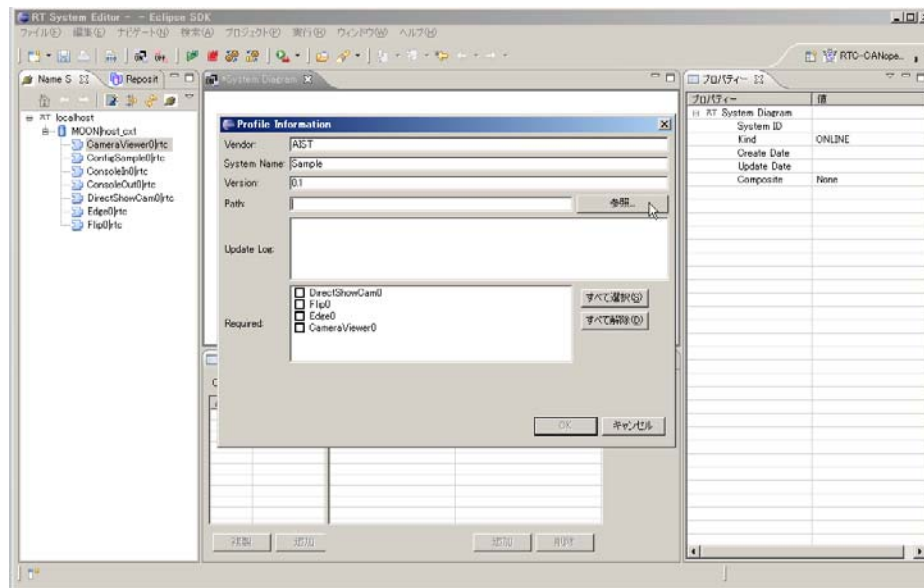
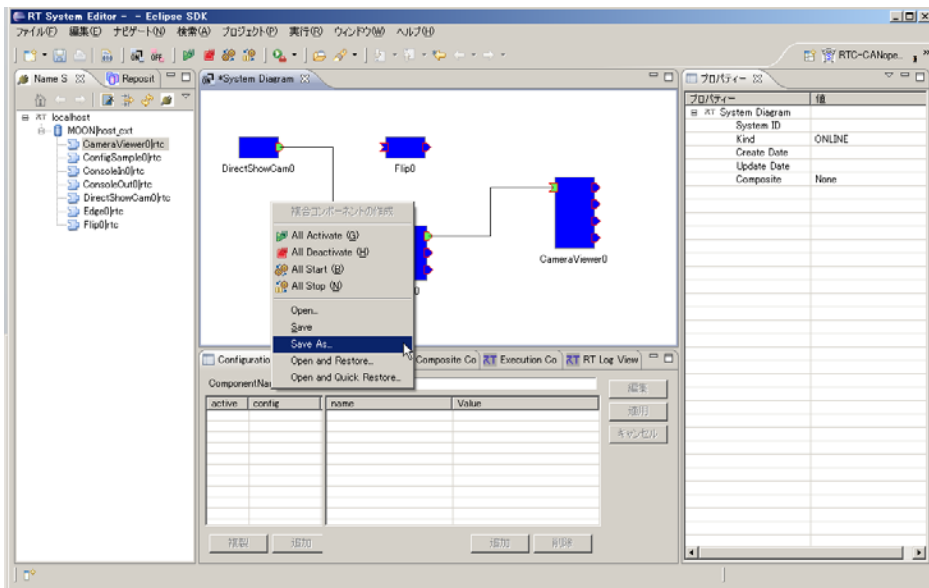
## Edge側への差し替え

- Flipに繋がっている接続線を選択
- Flip側のPort部分に表示されているハンドルをEdge側のPortに繋ぎ替え
- 接続プロファイルはデフォルト設定のまま



# システム構成の保存・復元

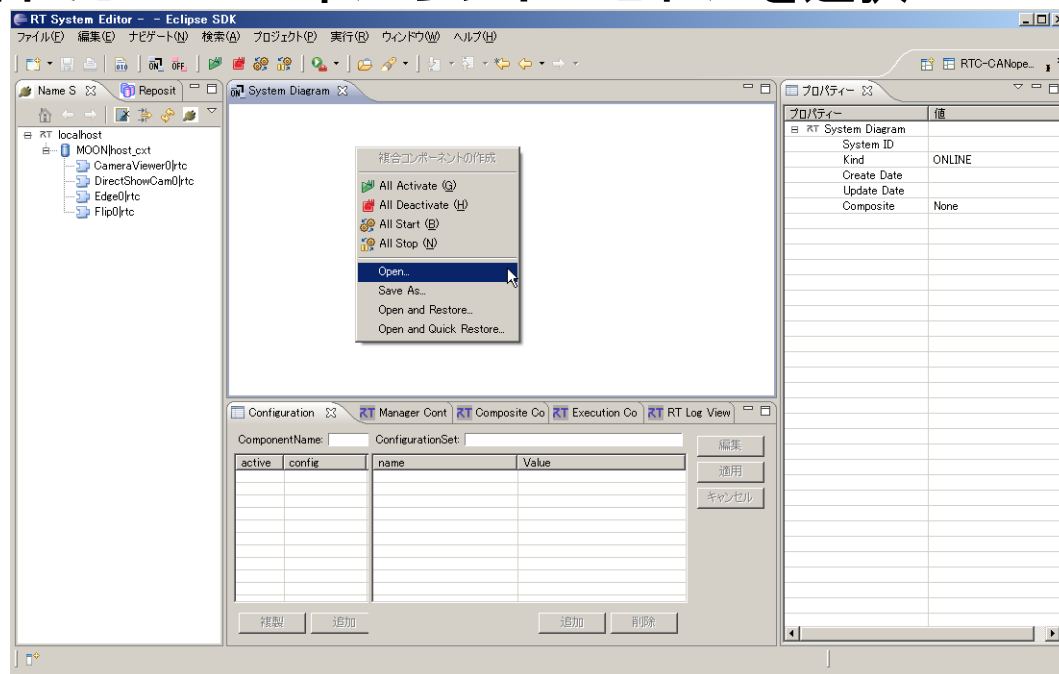
- エディタ上で右クリック
- 表示されたメニュー内から「Save As…」を選択
- 【Profile Information】画面にて、ベンダ名，システム名，バージョン番号，保存先ファイル名を指定



※指定したファイルにXML形式(RtsProfile)で保存

# システム構成の保存・復元

- システムエディタを閉じる
- 各コンポーネントを一度終了し、再起動
- エディタ上で右クリックし、表示されたメニュー中から以下のどれかを選択
  - 「Open」
  - 「Open and Restore」
  - 「Open and Quick Restore」
- 【ファイル選択画面】にて、先ほど保存したファイルを指定



※ Open: 使用していたRTCのみを読み込み

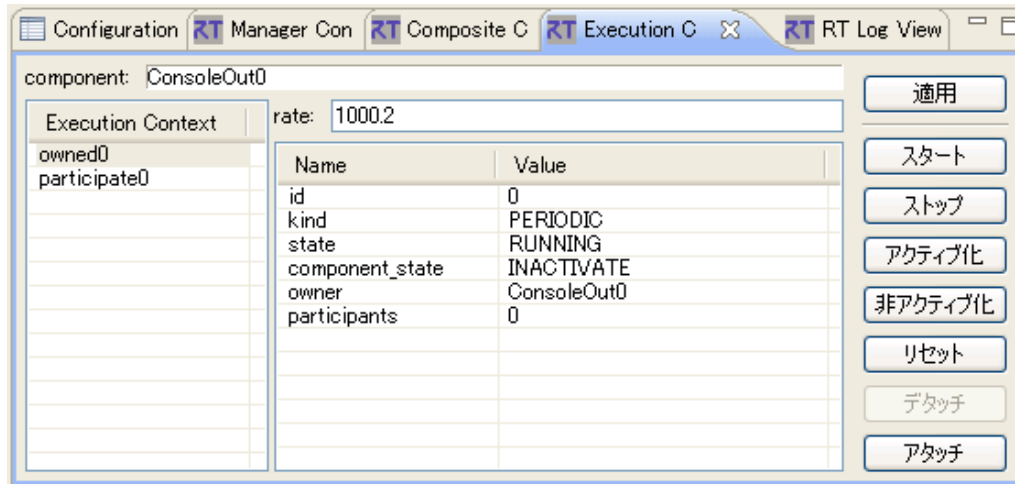
Open and Restore: 使用していたRTCを読み込むと同時に、接続、コンフィギュレーションセットの内容も復帰

Open and Quick Restore: 読み込み内容はOpen and Restoreと同様。該当RTCを検索する際にIORのみを使用

# 補足説明

---

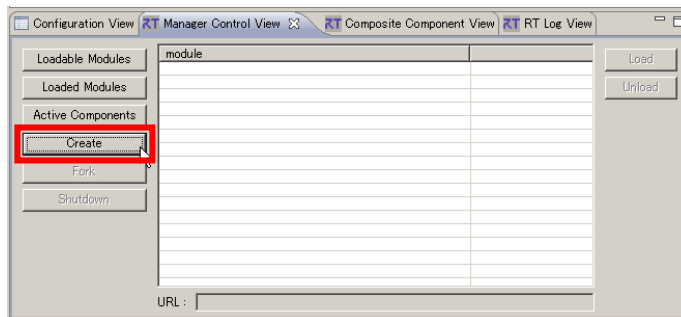
## ■ RTコンポーネントが属する実行コンテキスト(EC)を一覧表示



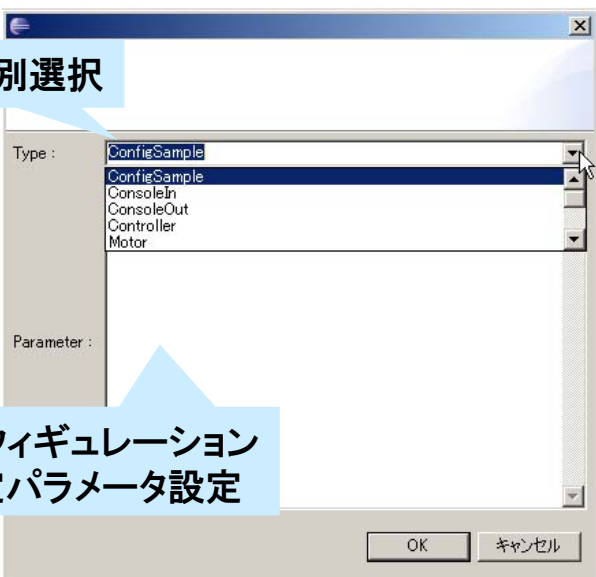
属性名	説明
id	ECのID. オンラインの場合には, context_handleを表示
kind	ECの種別(PERIODIC/EVENT_DRIVEN/OTHER)
state	ECの状態(RUNNING/STOPPING)
component state	対象RTCの状態(ACTIVE/INACTIVE/ERROR)
owner	対象ECを所有しているオーナーRTCのインスタンス名
participants	対象ECに参加中のRTCの数

※対象ECの実行周期の変更, EC自身の動作開始/終了, 新規RTCへのアタッチ, アタッチ済みRTCのデタッチも可能

## ■ RTコンポーネントの新規インスタンスの生成

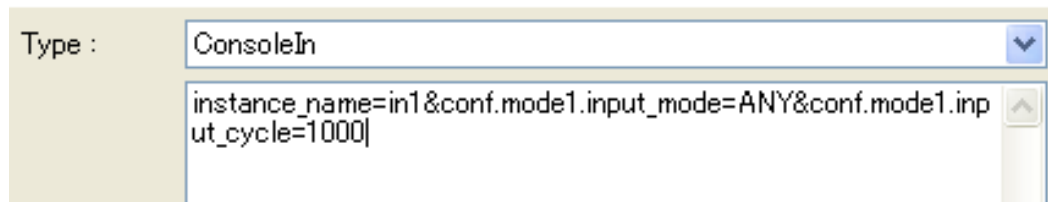


RTC種別選択



コンフィギュレーション  
指定パラメータ設定

- コンフィギュレーション指定パラメータ
  - `conf. [ConfigSet名]. [Configパラメータ名]=[設定値]`の形式にてConfigurationSetの値も設定可能





## ■ 選択したRTCから収集したログ情報を一覧表示

component	time	level	component	logger	message
<input checked="" type="checkbox"/> Notify0	2011-04-28 ...	ERROR	Notify0	RTC	test log!
<input checked="" type="checkbox"/> Notify1	2011-04-28 ...	ERROR	Notify1	RTC	test log!
	2011-04-28 ...	ERROR	Notify0	RTC	test log!
	2011-04-28 ...	ERROR	Notify1	RTC	test log!
	2011-04-28 ...	ERROR	Notify0	RTC	test log!
	2011-04-28 ...	ERROR	Notify1	RTC	test log!
	2011-04-28 ...	ERROR	Notify0	RTC	test log!
	2011-04-28 ...	ERROR	Notify1	RTC	test log!
	2011-04-28 ...	ERROR	Notify0	RTC	test log!
	2011-04-28 ...	ERROR	Notify1	RTC	test log!
	2011-04-28 ...	ERROR	Notify0	RTC	test log!
	2011-04-28 ...	ERROR	Notify1	RTC	test log!
	2011-04-28 ...	ERROR	Notify0	RTC	test log!
	2011-04-28 ...	ERROR	Notify1	RTC	test log!
	2011-04-28 ...	ERROR	Notify0	RTC	test log!
	2011-04-28 ...	ERROR	Notify1	RTC	test log!

※近日機能追加予定

## ● ログ収集の開始/停止

Left screenshot: Context menu with 'ログ収集を開始' (Start log collection) highlighted.

Right screenshot: Context menu with 'ログ収集を停止' (Stop log collection) highlighted.

## ● ログ情報のフィルタリング

The screenshot shows the log view with a filter applied to the 'component' column. The filter is set to 'Notify1'. The table below shows the filtered results.

time	level	component	logger	message
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!
2011-04-28 ...	ERROR	Notify1	RTC	test log!

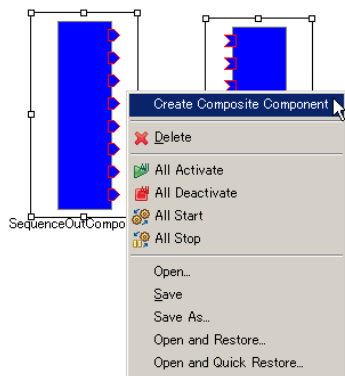


# 複合コンポーネント

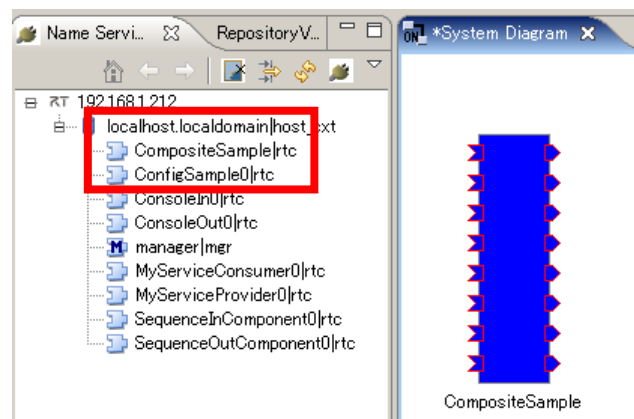
- 複数のRTCをまとめて、1つのRTCとして扱うための仕組み

- 複合コンポーネントの作成方法

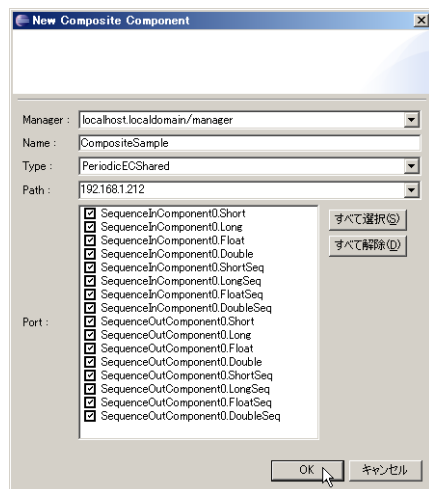
① 複数RTCを選択している状態で右クリック



③ 複合コンポーネントを生成



② 複合コンポーネントのプロパティを設定



項目	設定内容
Manager	複合コンポーネントを制御するマネージャを選択
Name	複合コンポーネントのインスタンス名を入力
Type	複合コンポーネントの型を選択
Path	複合コンポーネントのパスを入力
Port	外部に公開するポートを選択

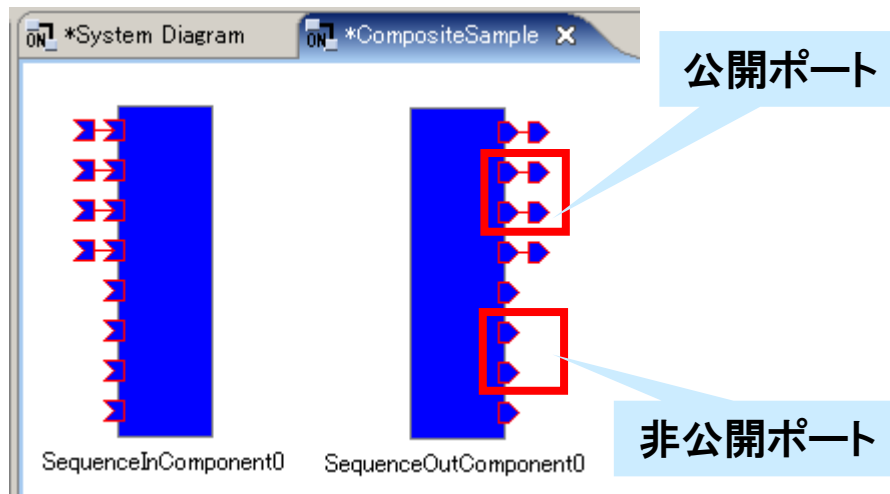
※生成対象複合コンポーネント外部と接続されているPortは強制的に公開されます

## ■ 複合コンポーネントのタイプについて

タイプ名	説明
PeriodicECShared	実行主体であるExecutionContextのみを共有. 各子コンポーネントはそれぞれの状態を持つ
PeriodicStateShared	実行主体であるExecutionContextと状態を共有
Grouping	便宜的にツール上のみでグループ化

## ■ 複合コンポーネントエディタ

- 複合コンポーネントをダブルクリックすることで表示



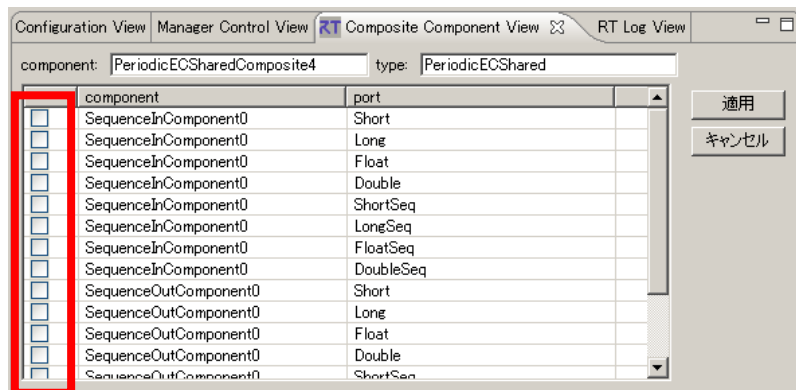
- ※エディタ内に別RTCをDnDすることで、子コンポーネントの追加が可能  
→追加したRTCのポートは全て非公開に設定
- ※エディタ内のRTCを削除することで、子コンポーネントの削除が可能  
→削除されたRTCは、親エディタに表示

# 複合コンポーネント

## ■ 公開ポートの設定

### ● 複合コンポーネントビュー

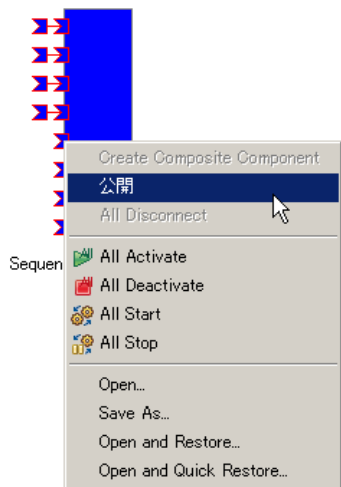
ポート公開情報



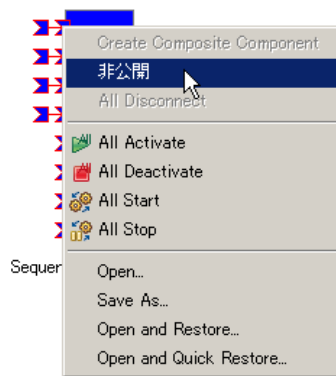
※ポート公開情報を変更し、「適用」をクリック

### ● 複合コンポーネントエディタ

※非公開ポートを「公開」



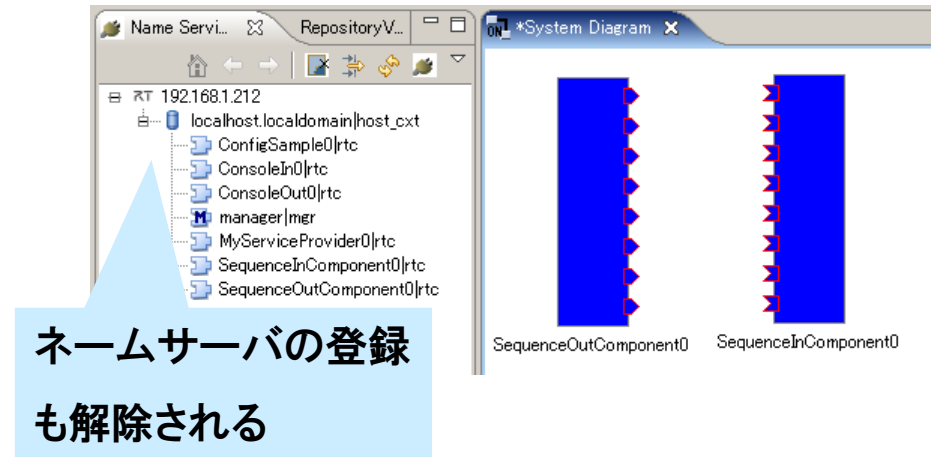
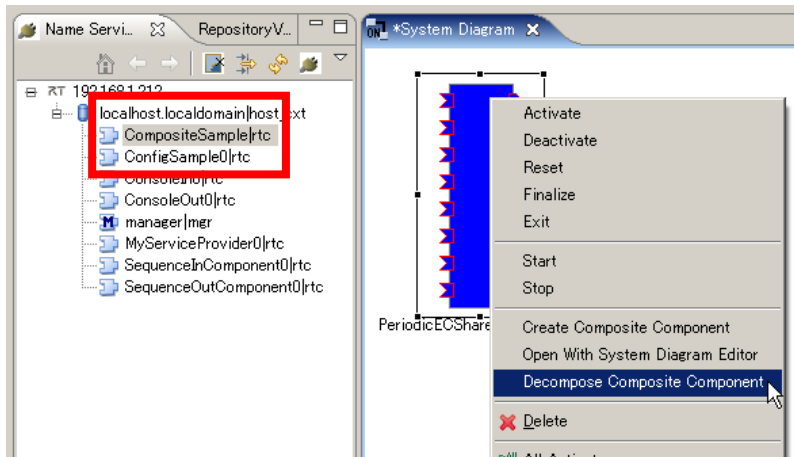
※公開ポートを「非公開」



外部コンポーネントと接続されているポートを「非公開」に設定することはできません

## ■ 複合コンポーネントの解除

- ① 複合RTCを右クリックし、複合コンポーネントの解除を選択
- ② 複合コンポーネントが分解され、内部のRTCが表示



※ エディタ上で、(Deleteキーなどで)単純に削除した場合は、エディタから表示が消えるのみ複合コンポーネントは解除されない

# オフラインエディタ

- RTコンポーネントの仕様を用いてRTシステムを構築
  - 実際のRTコンポーネントが動作している必要はない

The screenshot displays the RT System Editor interface with several key components highlighted by dashed lines and callouts:

- リポジトリレビュー (Repository Review):** A pink callout box pointing to the left-hand repository browser showing a project tree with components like CameraComponent, ImageProcess, and ImageViewer.
- オフライン・システムエディタ (Offline System Editor):** A blue callout box pointing to the central workspace containing an offline system diagram with components CameraComponent\_1, ImageProcess\_1, and ImageViewer\_1.
- プロパティビュー (Properties View):** An orange callout box pointing to the right-hand panel showing the configuration details for ImageProcess\_1.
- コンフィギュレーションビュー (Configuration View):** A pink callout box pointing to the bottom panel showing a table for configuring the selected component.

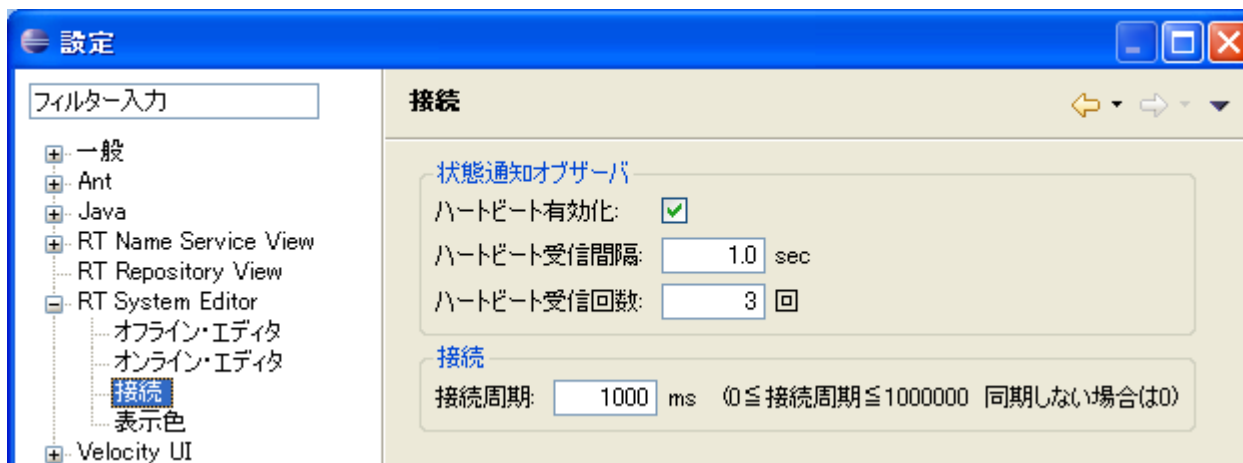
active	config	name	Value

プロパティ	値
Instance Name	ImageProcess_1
Type Name	ImageProcess
Description	Image Processing RTC
Version	1.0.0
Vendor	AIST
Category	Sample
InPort	
Name	Din
Data Type	RTC:TimedOctetSeq
OutPort	
Name	Dout
Data Type	RTC:TimedOctetSeq
ServicePort	
Name	CapPort
PortInterface	
Interface	CameraInfo
Type Name	CameraInfo
Port Inter	REQUIRED

## ■ 接続 - 状態通知オブザーバ

### ● RTCの生存確認用オブザーバに関する設定

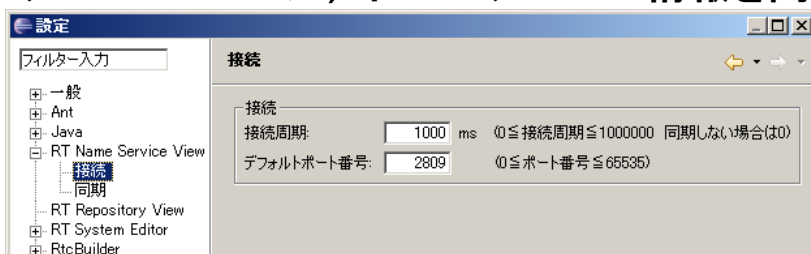
- RTSE側から生存確認を行うのではなく, RTC側から通知(ハートビート)を行う形
- OpenRTM-aist-1.1以降で対応



- ハートビート有効化: ハートビートによる生存確認機能の有効化
- ハートビート受信間隔: ハートビートの受信間隔. この間隔以内にRTC側からハートビートが送られてこない場合生存確認失敗と判断
- ハートビート受信回数: この回数を超えて生存確認に失敗した場合, 対象RTCに異常が発生したと判断

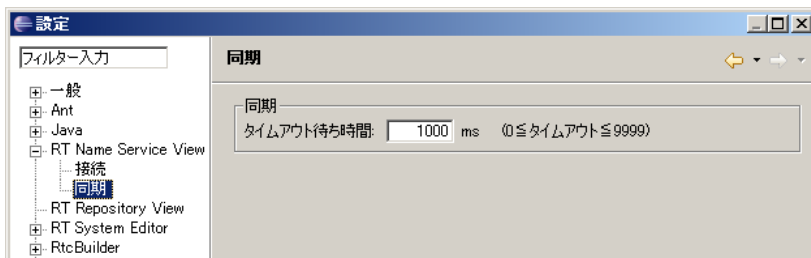
## ■ 「RT Name Service View」－「接続」【接続周期】

- ネームサービスビューが、ネームサーバに情報を問い合わせる周期



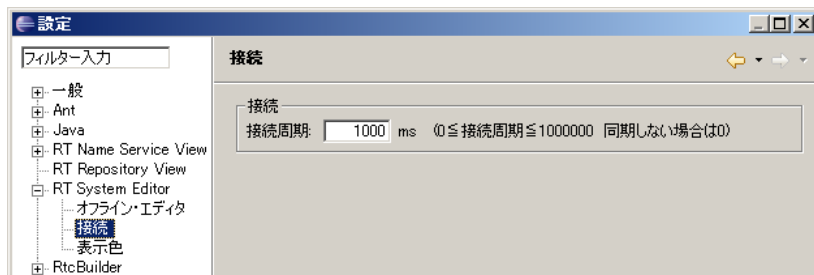
## ■ 「RT Name Service View」－「同期」【タイムアウト待ち時間】

- ネームサービスビューが、リモートオブジェクトのレスポンスを待つ時間



## ■ 「RT System Editor」－「接続」【接続周期】

- システムエディタが、ネームサーバに情報を問い合わせる周期



**【接続周期】をゼロに設定すると  
ネームサーバとの同期を行わない**

# 青梅商工会議所主催 RTM講習会

