# 熊本県産業技術センター
# 第7回技術普及講習会(組み込み技術)
# 「RTミドルウェアの概要と実習」

日時：2011年11月25日(金)　11:00～17:00
場所：熊本県産業技術センター 大会議室

## RTミドルウェア講習会

| | |
|---|---|
| 11:00-<br>11:30 | **第1部：RTミドルウェアの概略紹介** |
| | 担当：安藤 慶昭（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を説明します．また，Web上で自分の作品を公開できる仕組みについて紹介します． |
| 11:30-<br>12:00 | **第2部：RTミドルウェアの概略，導入方法の紹介** |
| | 担当：栗原眞二（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します． |

| | |
|---|---|
| 13:00-<br>13:45 | **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** |
| | 担当：坂本武志（株式会社グローバルアシスト） |
| | 概要:RTコンポーネントを組み合わせてシステムを構築する方法について説明します． |
| 14:00-<br>14:45 | **第4部：RTコンポーネントの作り方** |
| | 担当：坂本武志（株式会社グローバルアシスト） |
| | 概要:RTコンポーネントのテンプレート作成ツールRTCBuilderを用いたコンポーネントの設計と実装について説明します。 |
| 15:00-<br>17:00 | **第5部：RTコンポーネント作成実習** |
| | 担当：栗原眞二（ 産業技術総合研究所） |
| | 概要:参加者全員で実際にコンポーネントを作成してシステムを構築してみます． |

# 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

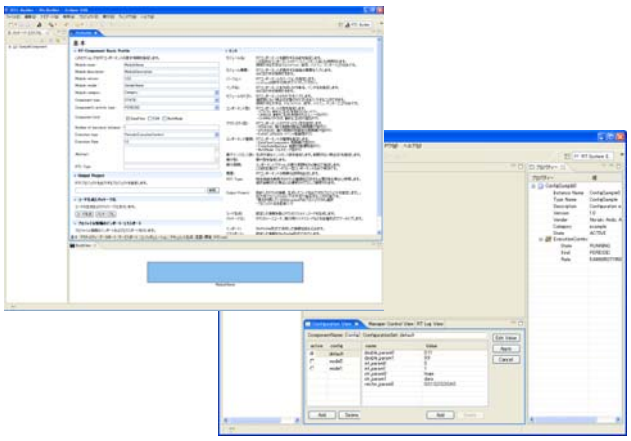
株式会社　グローバルアシスト
坂本 武志



## OpenRT Platform

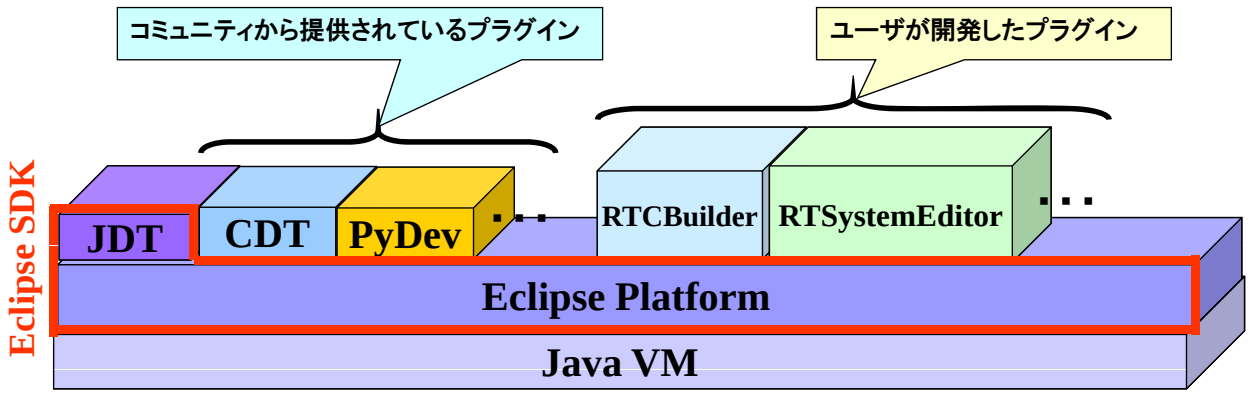
- 次世代ロボット知能ソフトウェアプラットフォーム
  - http://www.openrtp.jp/wiki/
  - システム設計，シミュレーション，動作生成，シナリオ生成などをサポート
- OpenRT Platformツール群
  - コンポーネント開発，システム開発における各開発フェーズの作業支援
  - 開発プラットフォームにEclipseを採用
- 構成
  - RTCビルダ
  - RTCデバッガ
  - RTシステムエディタ
  - ロボット設計支援ツール
  - シミュレータ
  - 動作設計ツール
  - シナリオ作成ツール

など



# 統合開発環境Eclipse

- 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





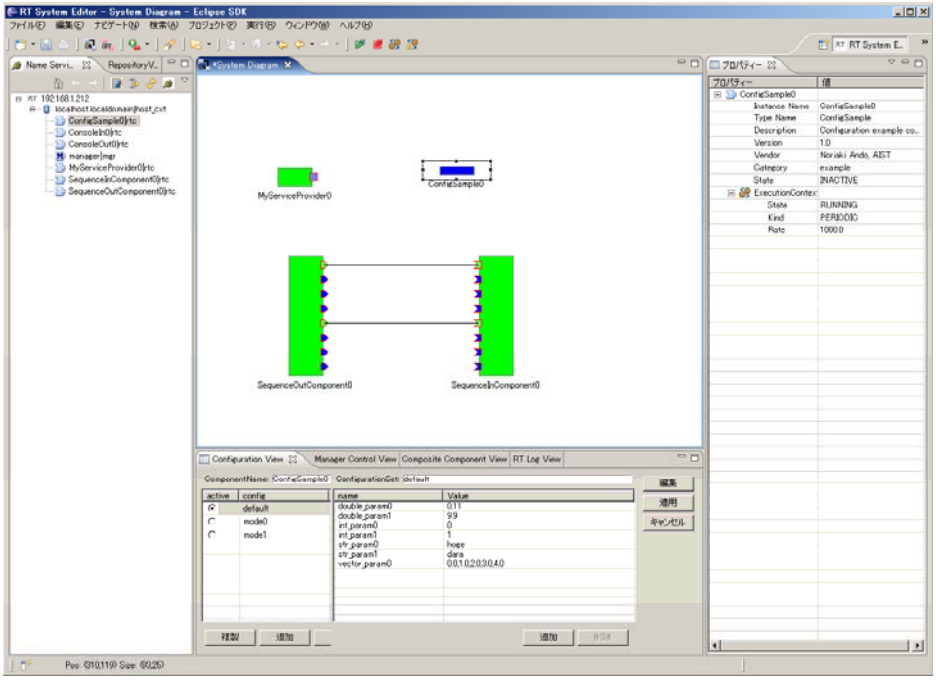

# RTCBuilder,RTSystemEditorのインストール

- 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

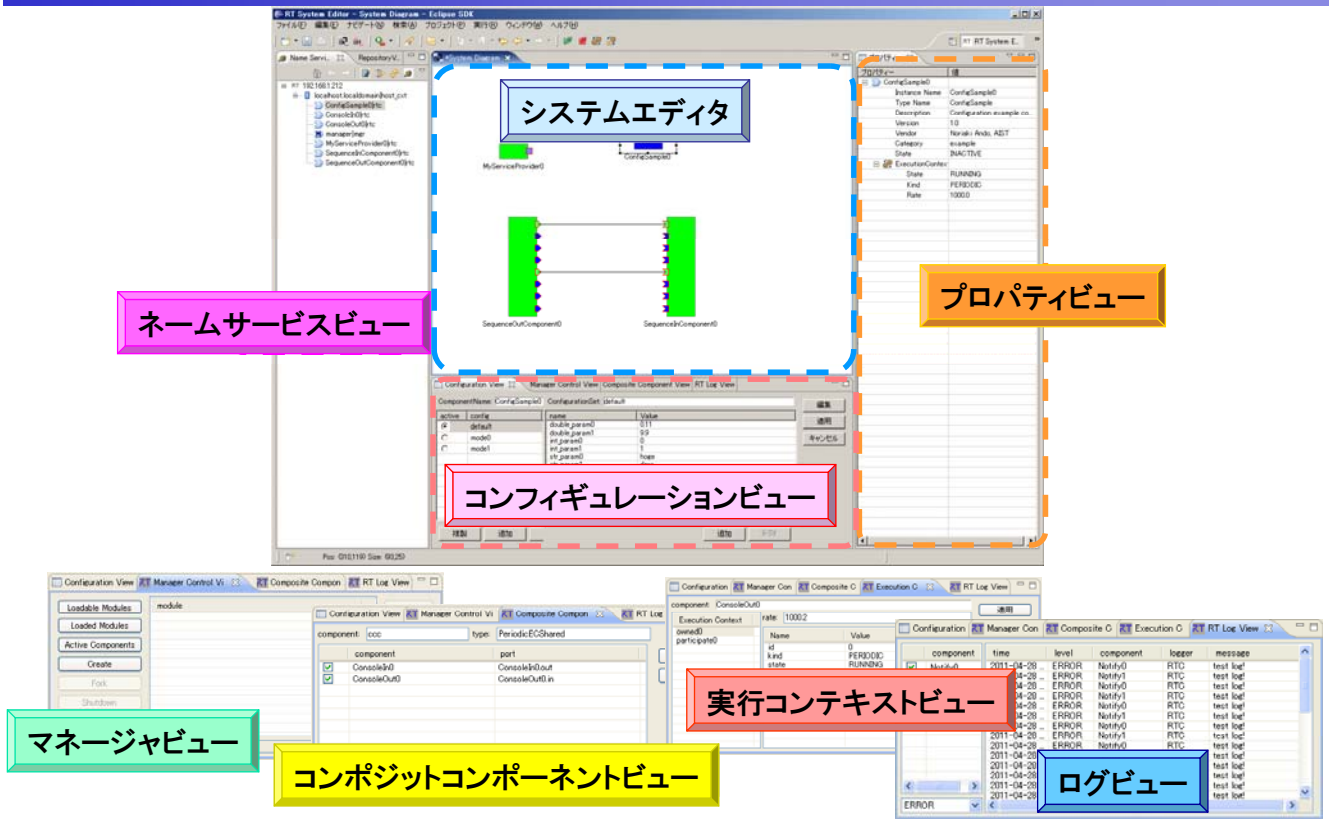
# RTSystemEditorについて


## RTSystemEditor概要


- 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

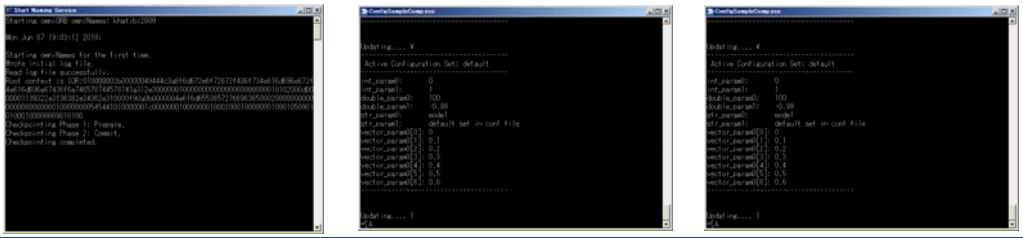





# 準備

- 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]

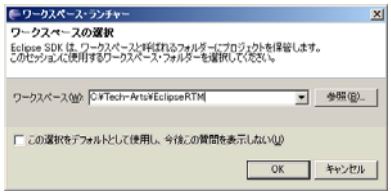
  CosoleOutCompの起動
    [スタート]メニューから起動
    [プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]
    →[examples]→ [ConsoleOutComp.exe]



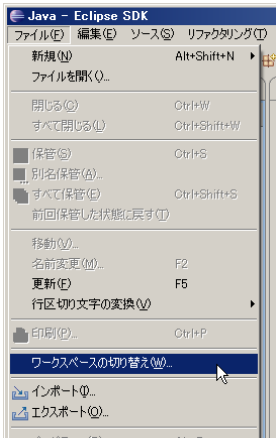
# ツールの起動

- 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)

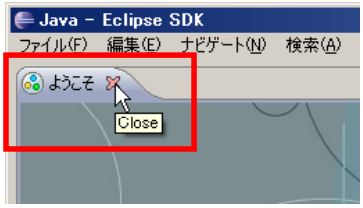



※ワークスペース
Eclipseで開発を行う際の作業領域
Eclipse上でプロジェクトやファイルを作成すると
ワークスペースとして指定したディレクトリ以下に
実際のディレクトリ，ファイルを作成する



# 準備

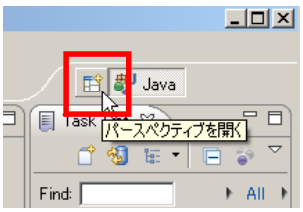
- 初期画面のクローズ
  - 初回起動時のみ

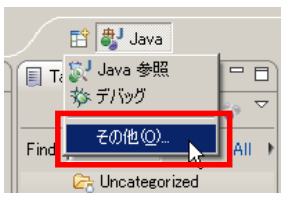



※パースペクティブ
Eclipse上でツールの構成を管理する単位
メニュー，ツールバー，エディタ，ビューなど
使用目的に応じて組み合わせる
独自の構成を登録することも可能

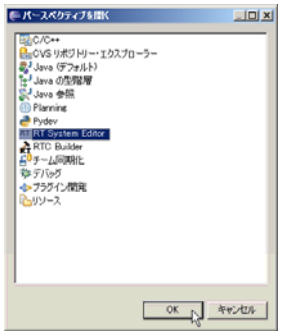
- パースペクティブの切り替え

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



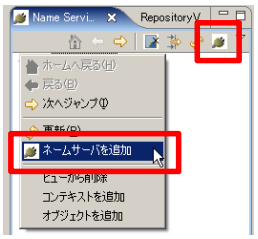



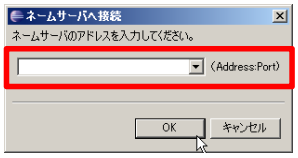

②一覧画面から対象ツールを選択



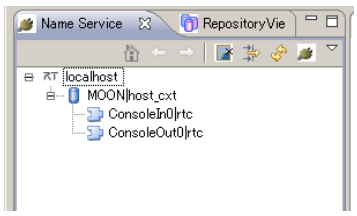
# RTシステム構築の基本操作

## ■ ネームサービスへ接続

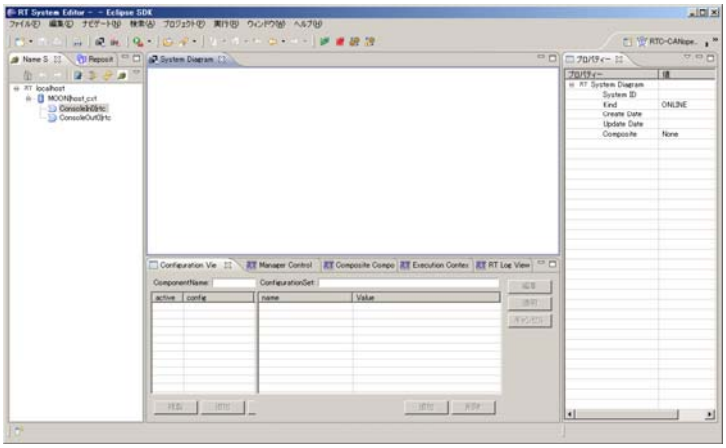



※対象ネームサーバのアドレス，ポートを指定
→ポート省略時のポート番号は
設定画面にて設定可能



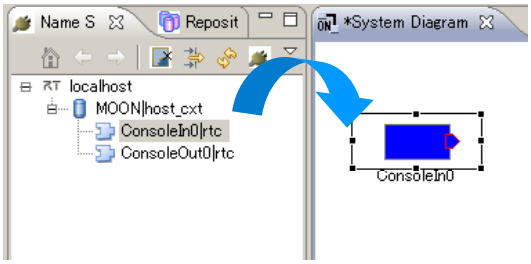

## ■ システムエディタの起動

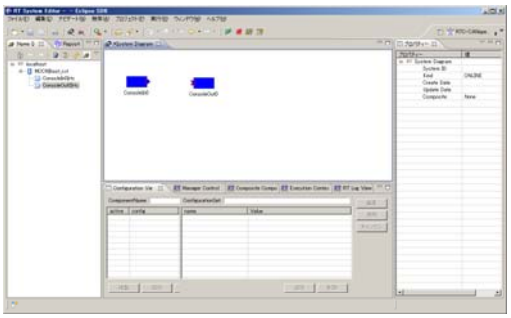


# RTシステム構築の基本操作

## ■ RTコンポーネントの配置



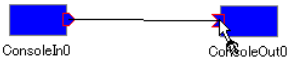

※ネームサービスビューから対象コンポーネントをドラッグアンドドロップ



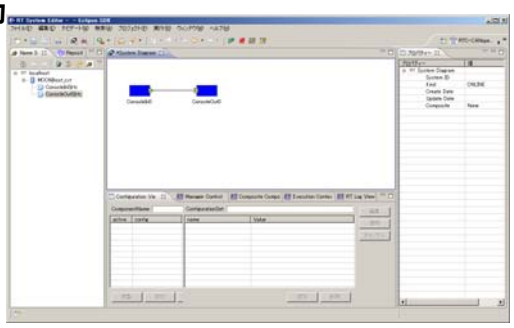
## ■ ポートの接続

①接続元のポートから接続先のポートまでドラッグ

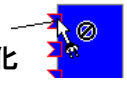
②接続プロファイルを入力







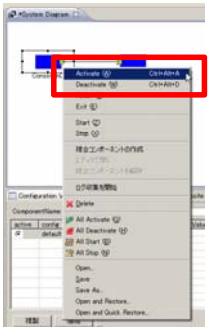
※ポートのプロパティが異なる場合など，
接続不可能なポートの場合にはアイコンが変化

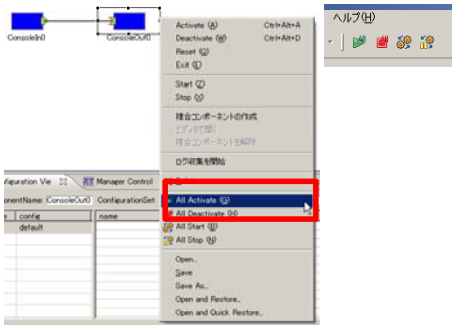


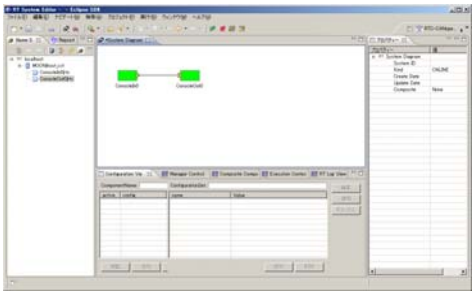
# RTシステム構築の基本操作

## ■ 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

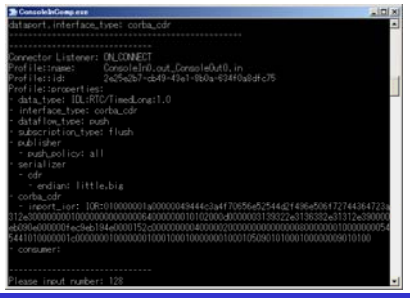




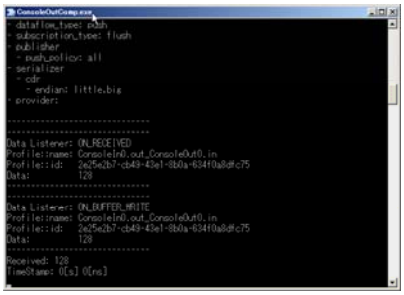


## ■ サンプルの実行

①ConsoleIn側で数字を入力



②ConsoleOut側が表示



※停止はDeactivateを実行

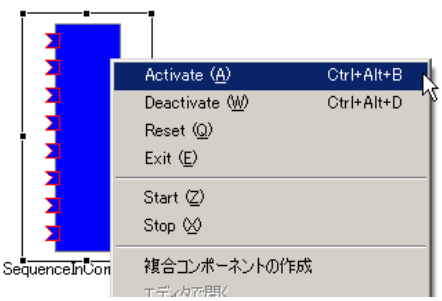
※RTC間の接続を切る場合には
接続線をDelete
もしくは，右クリックメニューから「Delete」を選択



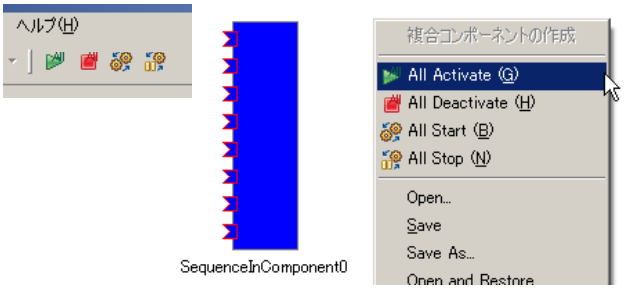
# RTコンポーネントの動作

| 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ー定義を追加

# 接続プロファイル(DataPort)について

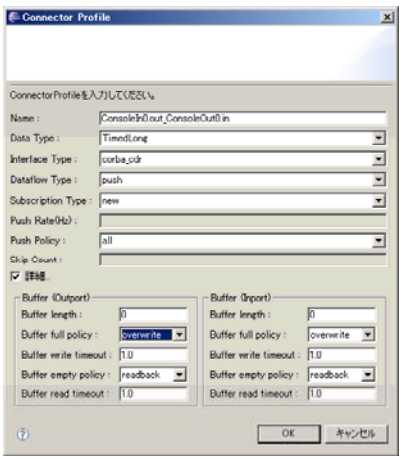
| 項目 | 設定内容 |
|---|---|
| Name | 接続の名称 |
| DataType | ポート間で送受信するデータの型. ex)TimedOctet,TimedShortなど |
| InterfaceType | データを送受信するポートの型. ex)corba_cdrなど |
| DataFlowType | データの送信方法. ex)push, pullなど |
| SubscriptionType | データ送信タイミング. 送信方法がPushの場合有効. New, Periodic, Flushから選択 |
| Push Rate | データ送信周期(単位:Hz).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



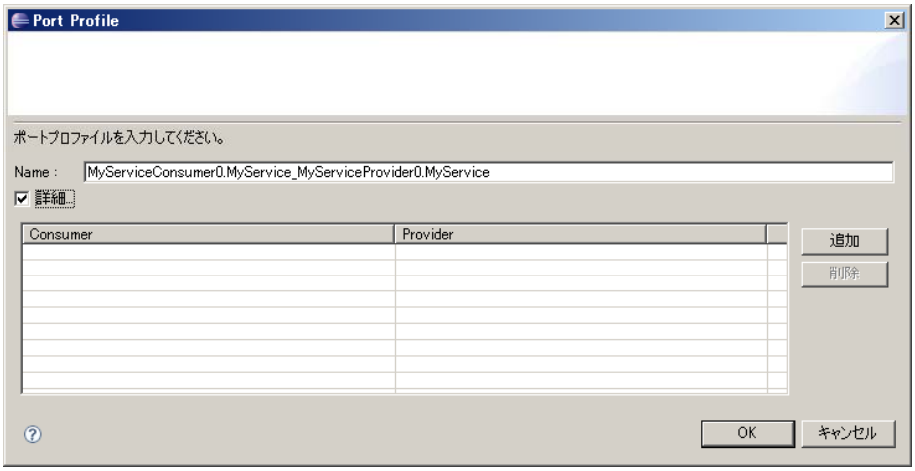

※OutPort側のバッファ, InPort側のバッファそれぞれに設定可能
※timeoutとして「0.0」を設定した場合は, タイムアウトしない

- 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.<br>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 |



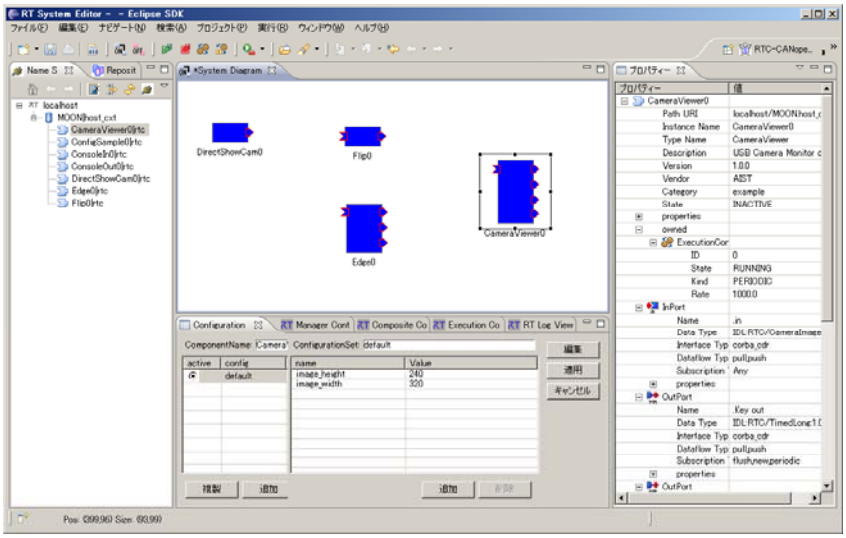



# 準備

- 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]
- DirectShowCamComp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

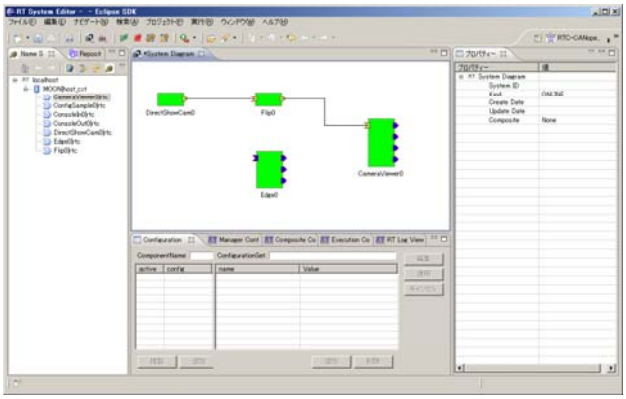
- 以下のコンポーネントをエディタ上に配置
  - DirectShowCam
  - Flip
  - Edge
  - CameraViewer



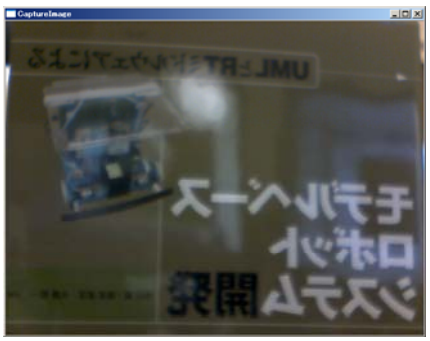


# システム構成の変更

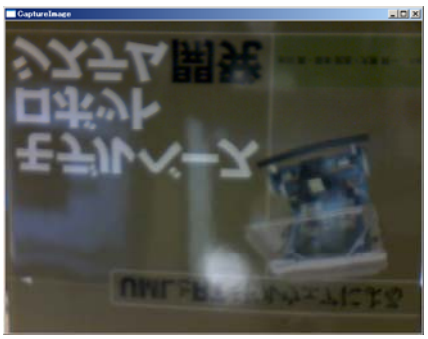
- Flip側との接続
  - DirectShowCam → Flip
    → CameraViewerと接続
    (接続プロファイルはデフォルト設定)
  - AllActivateを実行



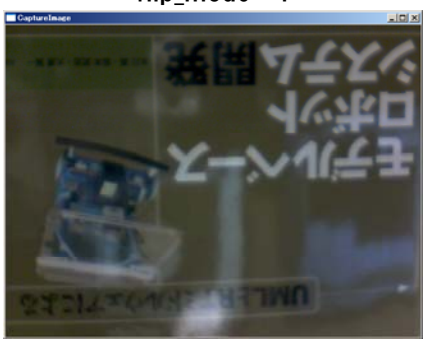
flip_mode=1



flip_mode=0

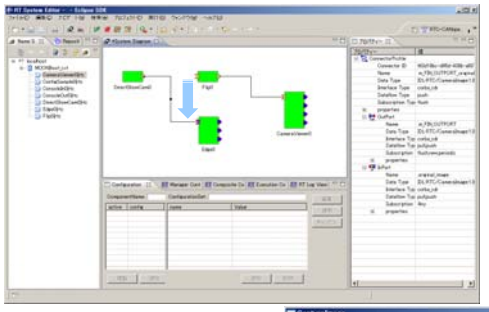
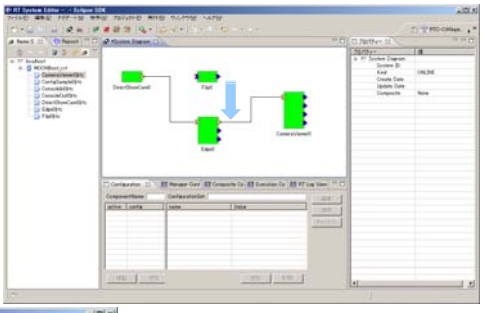
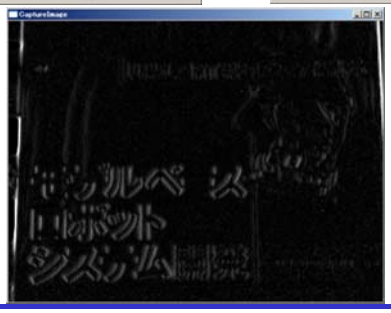


flip_mode=-1



# 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



# 既存コンポーネントの再利用

- プロジェクトとは
  - ユーザが作成した様々なコンポーネントやツールの公開場所
  - ユーザ登録すれば、誰でも自分の成果物の紹介ページを作成可能
  - 他のユーザに自分のコンポーネント等を紹介することができる

- プロジェクトのカテゴリ
  - RTコンポーネント: 1つのコンポーネントまたは複数のコンポーネント群などが登録されています。
  - RTミドルウエア:OpenRTM-aistや他のミドルウエア、ミドルウエア拡張モジュール等が登録されています。
  - ツール:各種ツール(RTSystemEditorやrtshellを含む)ツールはこのカテゴリになります。
  - 関連ドキュメント:関連ドキュメントとは、各種インターフェースの仕様書やマニュアル等を含みます。

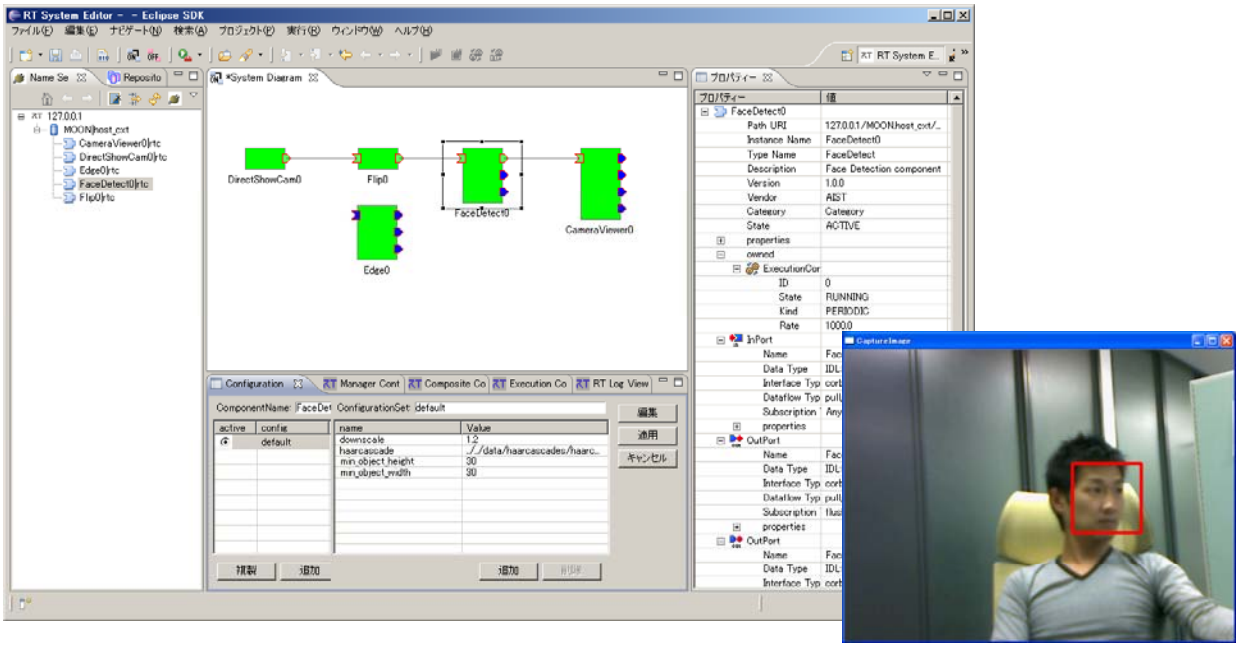
# 既存コンポーネントの再利用

- プロジェクトから対象コンポーネントを取得
  - 「顔検出コンポーネント」
    http://www.openrtm.org/openrtm/ja/project/facedetect
    対象コンポーネントをダウンロード



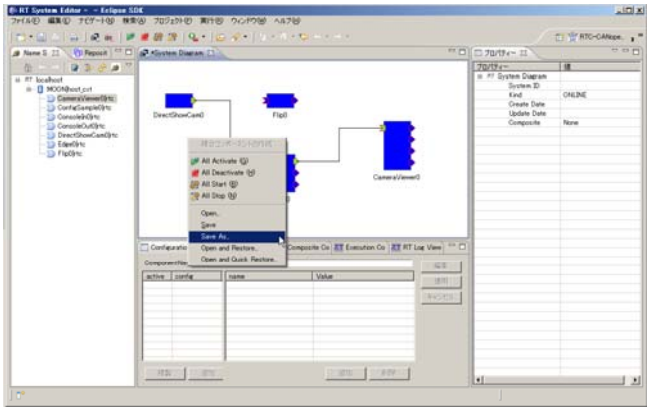

# 既存コンポーネントの再利用

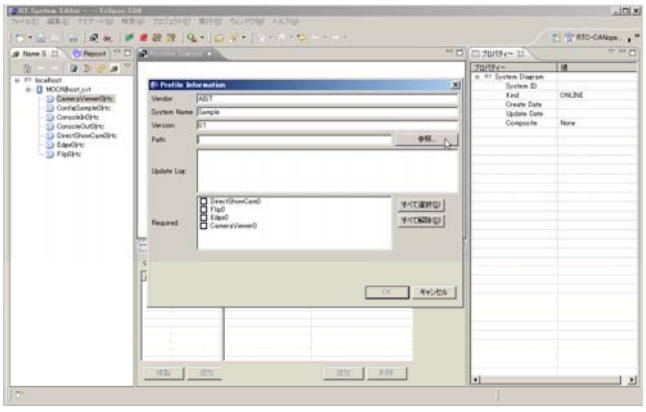
- ダウンロードしたファイル(FaceDetect.zip)を解凍
- 解凍したディレクトリ内の以下のファイルを実行し，システムエディタ上に配置
  $(FaceDetect_Root)/build/Release/FaceDetectComp.exe



# 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



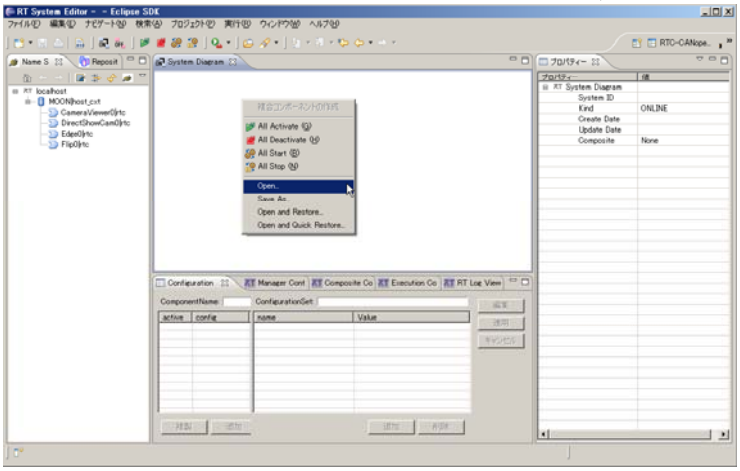


※指定したファイルにXML形式(RtsProfile)で保存



# 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

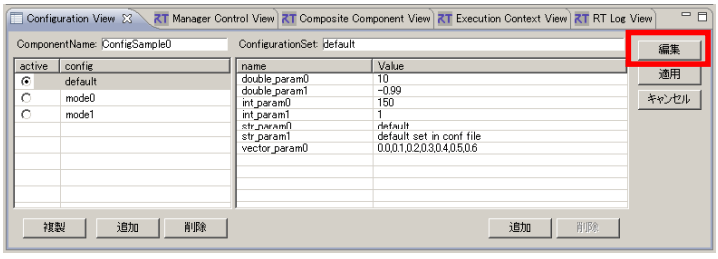


※Open：使用していたRTCのみを読み込み
Open and Restore：使用していたRTCを読み込むと同時に，接続，コンフィギュレーションセットの内容も復帰
Open and Quick Restore：読み込み内容はOpen and Restoreと同様．該当RTCを検索する際にIORのみを使用

# 補足説明

## コンフィギュレーションビュー

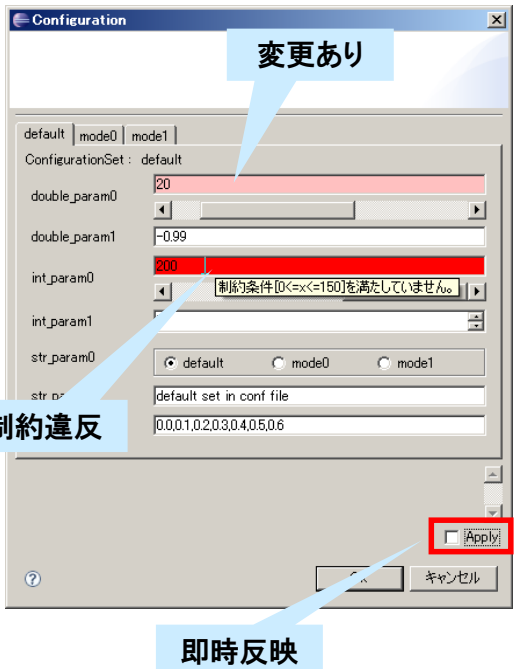
- RTコンポーネントのコンフィギュレーション情報の確認/編集




※「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，
スライドバーを操作する，
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)

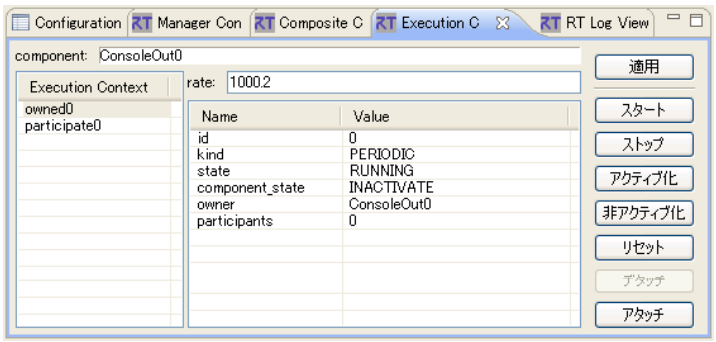
RTCの利用者が設定するのではなく，RTC開発者，RTC管理者が設定することを想定．

RTCBuilderを使用することで設定可能



# 実行コンテキストビュー

- 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

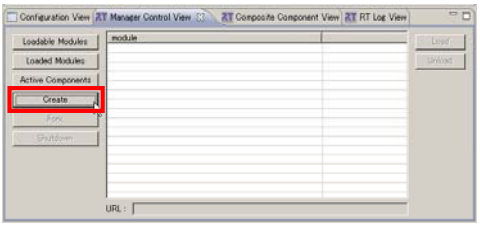



| 属性名 | 説明 |
|---|---|
| id | ECのID．オンラインの場合には，context_handleを表示 |
| kind | ECの種別(PERIODIC/EVENT_DRIVEN/OTHER) |
| state | ECの状態(RUNNING/STOPPING) |
| component state | 対象RTCの状態(ACTIVE/INACTIVE/ERROR) |
| owner | 対象ECを所有しているオーナーRTCのインスタンス名 |
| participants | 対象ECに参加中のRTCの数 |

※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

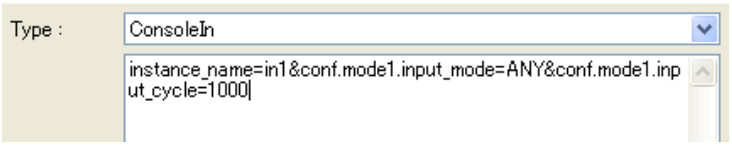
# マネージャビュー

## RTコンポーネントの新規インスタンスの生成





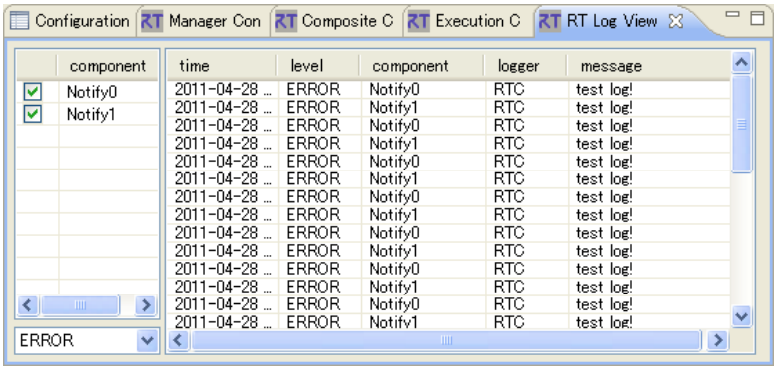
- コンフィギュレーション指定パラメータ
  - conf. [ConfigSet名]. [Configパラメータ名]=[設定値]の形式にてConfigurationSetの値も設定可能





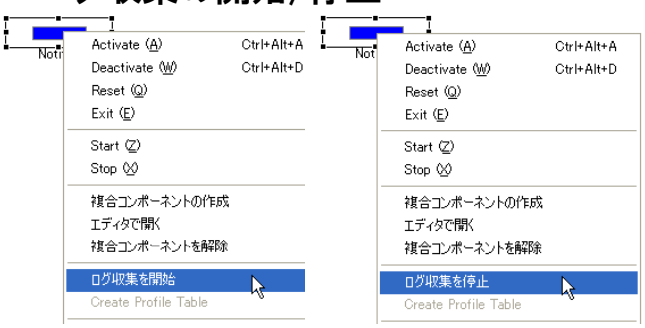

# ログビュー

## 選択したRTCから収集したログ情報を一覧表示

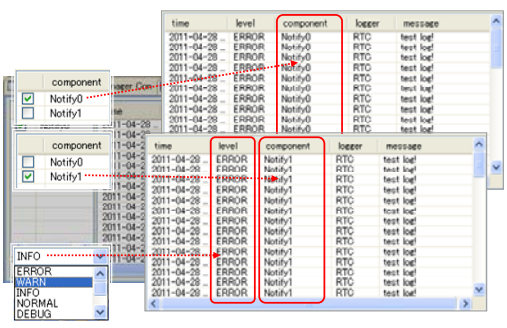



※近日機能追加予定

- ログ収集の開始/停止



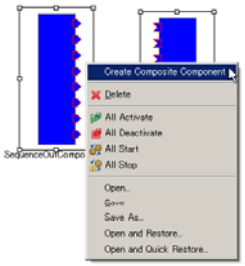

- ログ情報のフィルタリング



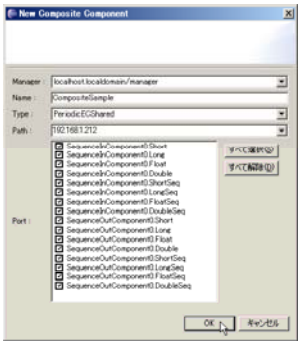
# 複合コンポーネント

- 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



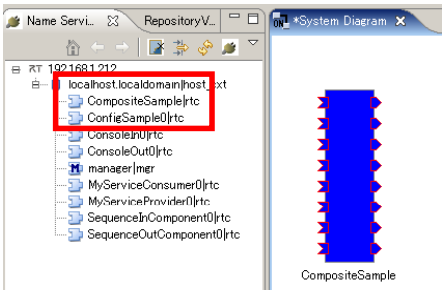

②複合コンポーネントのプロパティを設定




| 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

③複合コンポーネントを生成



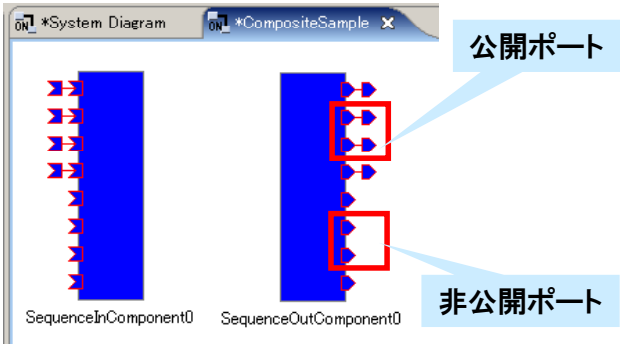



# 複合コンポーネント

- 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．<br>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

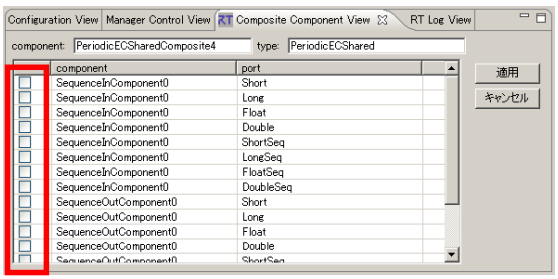
※エディタ内のRTCを削除することで，子コンポーネントの削除が可能
→削除されたRTCは，親エディタに表示

# 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

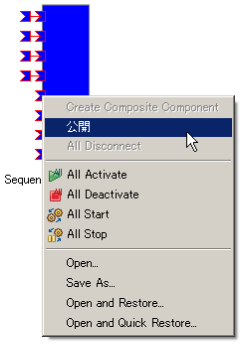
ポート公開情報



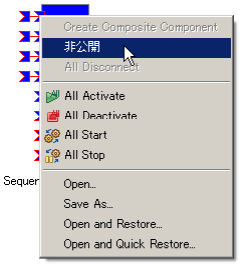

※ポート公開情報を変更し，「適用」をクリック

- 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

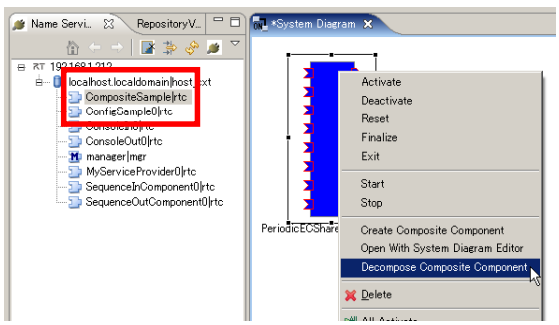


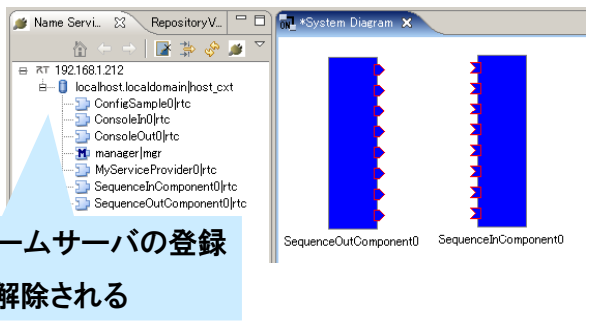
# 複合コンポーネント

## ■ 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



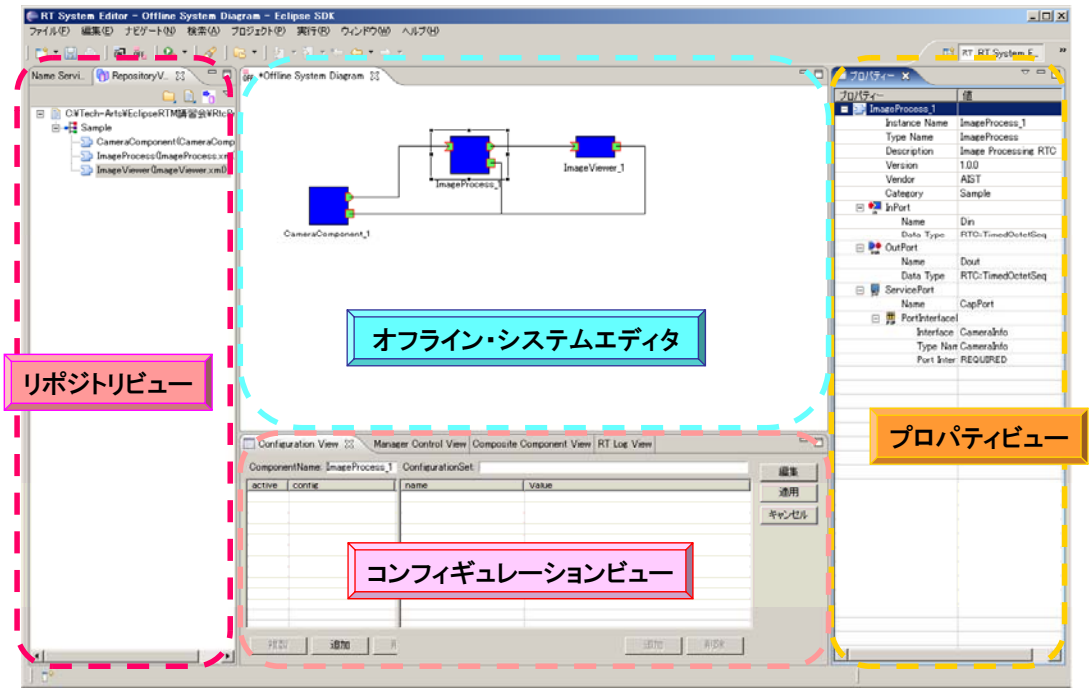




ネームサーバの登録も解除される

※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

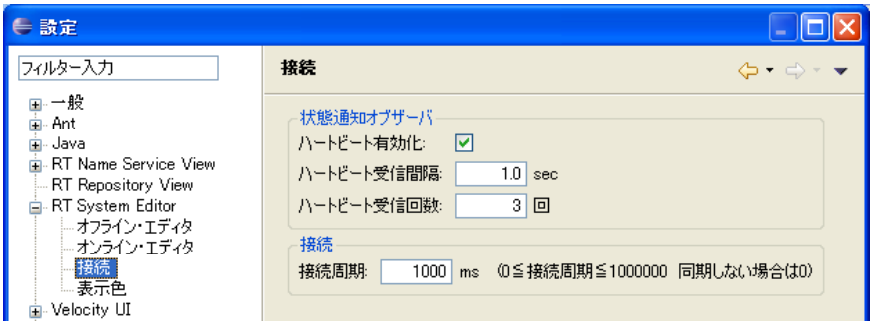
- RTコンポーネントの仕様を用いてRTシステムを構築
  - 実際のRTコンポーネントが動作している必要はない






# 設定画面

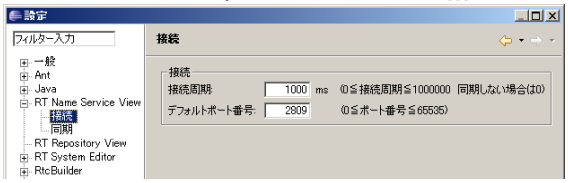
- 接続ー状態通知オブザーバ
  - RTCの生存確認用オブザーバに関する設定
    - RTSE側から生存確認を行うのではなく，RTC側から通知(ハートビート)を行う形
    - OpenRTM-aist-1.1以降で対応




- 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と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

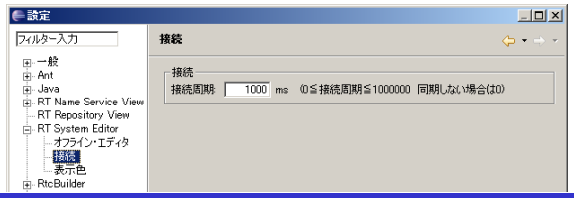
# 設定画面

- 「RT Name Service View」ー「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」ー「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」ー「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



# 熊本県産業技術センター
# 第7回技術普及講習会(組み込み技術)
# 「RTミドルウェアの概要と実習」