Retek® Integration Bus™ 10.3.4

Installation Guide

Corporate Headquarters:

Retek Inc.
Retek on the Mall
950 Nicollet Mall
Minneapolis, MN 55403
USA

888.61.RETEK (toll free US)
Switchboard:
+1 612 587 5000

Fax:
+1 612 587 5100

European Headquarters:

Retek
110 Wigmore Street
London
W1U 3RW
United Kingdom

Switchboard:
+44 (0)20 7563 4600

Sales Enquiries:
+44 (0)20 7563 46 46

Fax:
+44 (0)20 7563 46 10

Printed in the United States of America.

Customer Support

Customer Support hours

Customer Support is available 7x24x365 via email, phone, and Web access.

Depending on the Support option chosen by a particular client (Standard, Plus, or Premium), the times that certain services are delivered may be restricted. Severity 1 (Critical) issues are addressed on a 7x24 basis and receive continuous attention until resolved, for all clients on active maintenance. Retek customers on active maintenance agreements may contact a global Customer Support representative in accordance with contract terms in one of the following ways.

Contact Method	Contact Information
E-mail	support@retek.com
Internet (ROCS)	rocs.retek.com Retek's secure client Web site to update and view issues
Phone	+1 612 587 5800

Toll free alternatives are also available in various regions of the world:

Australia	+1 800 555 923 (AU-Telstra) or +1 800 000 562 (AU-Optus)
France	0800 90 91 66
United Kingdom	0800 917 2863
United States	+1 800 61 RETEK or 800 617 3835
Mail	Retek Customer Support Retek on the Mall 950 Nicollet Mall Minneapolis, MN 55403

When contacting Customer Support, please provide:

- Product version and program/module name.
- Functional and technical description of the problem (include business impact).
- Detailed step by step instructions to recreate.
- Exact error message received.
- Screen shots of each step you take.

Contents

Chapter 1 – Introduction

This manual details the installation of the Retek Integration Bus (RIB). An overview of this process is as follows:

1 The SeeBeyond e*Gate Integrator product (version 4.5.3) is installed. This involves installing the registry host and all participating host software, plus Graphical User Interface hosts for development and system monitoring. See Chapter 2 for details.

2 The RIB schema is imported into the e*Gate Integrator product. This is explained in Chapter 3.

3 Update the database connection points, JMS queues, and CLASSPATH configuration values. Also, delete unused adapters. This is explained in Chapter 3.

4 Verify the Error Hospital tables exist; make DTD files available on the network; and install the Hospital GUI components.

5 Application Server specific configurations.

Chapter 2 – Install SeeBeyond e*Gate Integrator

The Retek Integration Bus (RIB) leverages SeeBeyond's e*Gate Integrator for supplying the needed messaging facilities for integrating applications.

The following steps need to be completed successfully to install the e*Gate product:

1 A Registry host that will contain the central database of the message formats, as well as publication, subscription and transformation logic needs to be installed.

2 At least one Participating host, which implements the publishers, subscribers, and transformations, needs to be installed.

3 The required e*Gate add-ons need to be installed.

- Batch e*Way 4.5.4 add-on
- Oracle e*Way 4.5.3 add-on (included in this is the JDBC e*Way)

4 The GUI hosts that are used to monitor the operation of the system and to extend or further develop the system's capabilities need to be installed.

- e*Gate GUI 4.5.3

5 There may be a number of required e*Gate ESR (patches) to be installed. A listing of any of these will be included in the RIB Release Notes. ESRs can be found on Retek's fulfillment center (http://fulfillment.retek.com).

Notes:

- All three types of hosts can be present on the same physical machine. However, GUI hosts must execute on a Microsoft Windows platform.
- e*Gate requires a Java Runtime Environment (JRE) version 1.3.1. This is bundled with the e*Gate install.
- e*Gate Monitor and e*Gate Enterprise Manager applications use the Exceed X-windows application. If a version of Exceed exists on a GUI host, then one must install the e*Gate version into a different directory. The e*Gate version is *not* a full installation of Exceed.

The instructions for installing the SeeBeyond e*Gate Integrator system are documented in the *e*Gate Integrator Installation Guide.* This document is found on Disk 2 of the SeeBeyond installation disk set (docs\eGate_Install_Guide.pdf)

Chapter 3 – RIB schema

Import

If this is a Delta release of the RIB (see Release Notes to identify delta releases), skip to the Install Delta Release section of this document.

The RIB software is distributed in a single messaging schema. This schema contains all of the RIB's publishing and subscribing e*Ways (adapters) and Connection Points. It also contains a single JMS Intelligent Queue Manager.

Once the RIB schema has been imported, a system administrator must configure the connection points. Additional configuration modifications may also be needed, such as e*Way CLASSPATH. These types of changes are detailed in Chapter 3.

The final modifications to the system are due to the site-specific deployment of the system. These changes include distributed components to different hosts, creating fail-over hosts, developing additional event types, adapters, connection points and collaborations for integrating an enterprise's non-Retek applications to the RIB. It also includes creating security roles and privileges. These activities are not considered part of the installation and are not documented in this manual. For more information on these activities, see the SeeBeyond *e*Gate Users Guide.*

Preliminary steps

To create and import the RIB schema, take the following preliminary steps:

1 For security reasons, create an "egate" user that will own the e*Gate files and execute the software.

2 Log onto the Unix system where e*Gate was installed using this account.

3 Copy the RIB tar files from the RIB installation CD(s) to the location where you are planning to install the RIB software. This location will be known as the RETEK_INSTALL_DIR in the remainder of this section. The RIB tar files are named RIBFor<APP><version>.tar (where the application is RDM, RCOM, RMS, ISO, RDC, etc.). An example of the <RETEK_INSTALL_DIR> directory name could be called "INSTALL", located directly under the "egate" user's home directory. Future releases of the RIB should be installed into this directory, as the directories will have new version numbers in their names.

   (eg: /files0/egate/INSTALL/)

4 Once you have copied the RIB tar file(s) to <RETEK_INSTALL_DIR>, extract each file in this directory and change the permissions on the extracted files to make them writable. If you are installing multiple version numbers of the RIB, be sure to copy and extract the tar files of all versions you intend to install.

   - tar xvf 'filename'
   - chmod –R 755 *

5 Change directories to <RETEK_INSTALL_DIR>/RIB<version>.

6 Edit the file egate_profile. If you are installing multiple version numbers, modify the file found in the most recent RIB<version> directory. Make sure the settings for the following variables are correct for your environment.

- EHOME – The directory where SeeBeyond e*Gate was installed.
- RETEK_INSTALL_DIR – The directory created in step 3 above.
- EGATE_SERVER_NAME – The name or IP address of the server e*Gate and the RIB software are installed on.
- EGATE_SERVER_PORT – The port that the e*Gate Registry Host was installed on during the SeeBeyond e*Gate Integrator install.
- RIB_SCHEMA – The name of the eGate Schema into which you wish to import the RIB SeeBeyond components. This value must match the name of the Schema you create in step 3 of Create a new schema below.
- Platform specific section (Sun Solaris, IBM AIX, HP-UX) - Uncomment the section that is applicable to your operating system and ensure that the other two sections are commented out.

7 Edit the "egate" user's .profile located in the "egate" user's home directory. Add an entry at the end of this file that sources the egate_profile modified in step 6 above.

- <RETEK_INSTALL_DIR>/RIB<version>/egate_profile

  (eg: . /files0/egate/INSTALL/RIB<version>/egate_profile)

- Ensure that '.:' (dot colon) is at the beginning of the egate user's PATH variable.

  (eg: PATH=.:${PATH} ; export PATH)

Source the .profile after making these modifications or start a new Unix session before continuing.

8 If there was an earlier attempt at installing the 10.3 version of the RIB, it must be inactivated by renaming it:

a Make sure that all e*Ways, the control broker, and the registry are shut down. On Unix, the following command will show the active processes:

```
> ps -ef | grep stc | grep -v grep
```

If stc processes are still running, be sure to shut down all the stc processes (eg: kill –9).

b Rename the $EHOME/server/registry/RIB103.rdb file to $EHOME/server/registry/RIB103.rdb.bak.

c Rename the $EHOME/server/registry/repository/RIB103 directory to $EHOME/server/registry/repository/RIB103.bak.

d The RIB103.rdb.bak file and RIB103.bak directory can be deleted at a later time once the new version has been successfully installed.

9 Start the e*Gate registry – the following command can be run manually, or the start_egate script can be run; it is located at: <RETEK_INSTALL_DIR>/RIB<version>/Rib_Support/

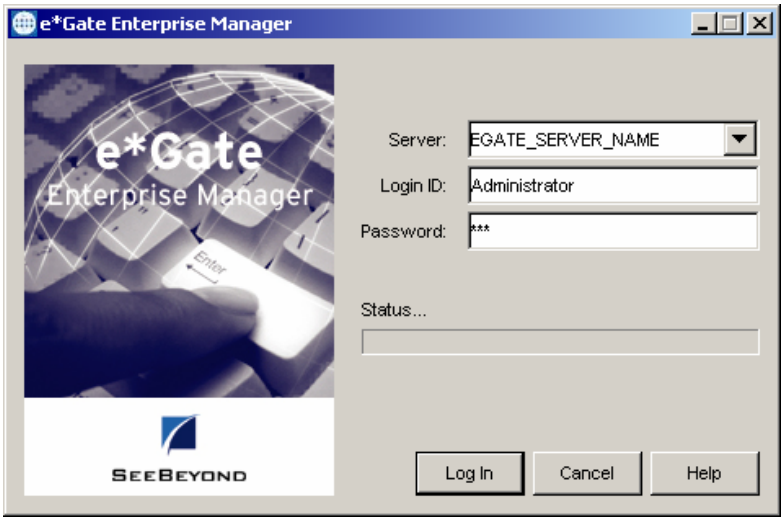
```
> let CB_PORT=$EGATE_SERVER_PORT+1

    > $EHOME/server/bin/stcregd -ss -ln $EGATE_SERVER_NAME -bd
$EHOME/server -pr $EGATE_SERVER_PORT -pc $CB_PORT -mc 1024 !>
/dev/null
```

Create a new schema

The RIB schema is imported through a two-step process. The first step involves creating a new schema. This new schema is empty and does not contain any RIB modules.

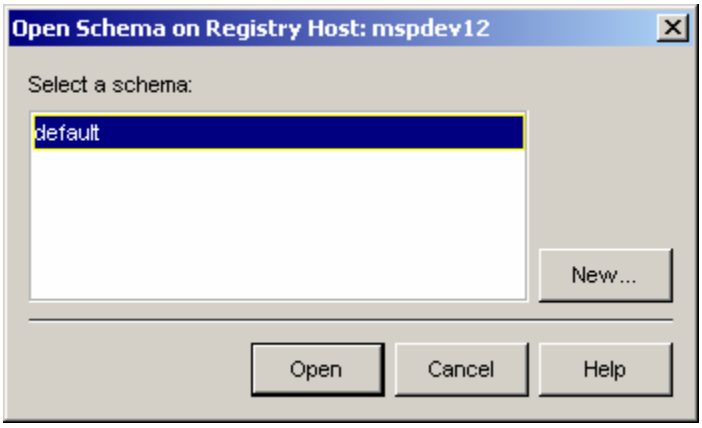
1 Log in to the e*Gate registry using the e*Gate Enterprise Manager GUI tool. Log in as Administrator, using the password that was set during the installation of e*Gate Integrator.

Note: For the Server: field, enter the EGATE_SERVER_NAME or IP address as specified in the egate_profile modified previously. Also, because the e*Gate Registry Host may not be running on the default port (23001), it is good practice to always specify the port along with the hostname. The format is <hostname>:<port>, (e.g. mspdev14:23001).



e*Gate Enterprise Manager dialog box

The Open Schema dialog box is displayed.



Open Schema dialog box

2 Click **New** to create a new schema. The New Schema dialog box is displayed.

3 In the **Enter New Schema Name** field, enter RIB103 (the name entered MUST match the value set as RIB_SCHEMA in your egate_profile referenced above).

4 Click **Open**.

You have now successfully created an empty schema named RIB103.

Import RIB Components

The second step of the RIB Schema import process entails the actual importing of the RIB components into the newly created “RIB103” base schema.

Load RIB Components - Automated Instructions

A script to register the new RIB Schema, “RIB103”, and insert all of the registry modules can be found in the following directory:

```
<RETEK_INSTALL_DIR>/RIB<version>/Migration_Scripts
```

Notes:

- If your RIB103 schema has a password for the Administrator user that is different than the default of “STC” you will need to edit the install script and replace the two occurrences of “STC” with your password (case sensitive).
- The system must be able to locate the unzip and zip utilities for the importeways script to work. If not, update the PATH variable in the egate user’s .profile file.
- Additionally, a working version of perl must be installed on your machine. Be sure the PATH variable contains an entry to where this file resides.
- The DISPLAY variable must be set to the IP address of the machine that the install will run on.
- Unless all of these requirements are met, the manual method to import the RIB modules will have to be used.

1 From this directory, run the “install” script (eg ./install). If you intend to install multiple RIB versions at once (eg 10.3 and 10.3.4) execute “./install” from the $RETEK_INSTALL_DIR/RIB1034/Migration_scripts directory.

   a The ./install script requires input parameters defining which version(s) of the RIB you wish to install (eg ./install 103 1034). Executing the install script with no parameters will provide you with a listing of valid input parameters. This listing is compiled by scanning the RETEK_INSTALL_DIR for RIB<version> directories, which are created when the user extracts the tar files into the RETEK_INSTALL_DIR.

      The user may input multiple version numbers or they may input a single version number. If the user is creating a new RIB103 schema, including version 10.3, a Delta and/or Custom version(s), the user may wish to install multiple versions at once.

   b After providing “./install” with valid input parameters, the user will be prompted to verify the order of install. When installing multiple versions, the user must be certain to install the older RIB versions first.

   c “./install” will then prompt the user to create a backup of the RIB103 schema. Choosing to do so will create a .zip of the schema that is currently installed before proceeding to import new modules.

   d Following this, “./install” will prompt the user for an alternative import control file. Choosing “Yes” is the default installation path and will proceed to import ALL components included in the release. Choosing “No” will require the user to input an alternate control file containing a customized list of components which to import. This can be utilized by clients which have customized one or more of the Retek provided RIB components to ensure that the installation process does not overwrite their customized code.

   e If the user has chosen to install multiple RIB versions, “./install” will import the RIB components associated to the first input parameter first. After successfully completing, “./install” will import the RIB components associated to the second input parameter, etc. If, after successfully importing the RIB components of any version(s) the user chooses to exit the install process, a listing of which versions where successfully completed will be provided to the user. These versions do not need to be re-installed.

2 Once the script has completed, use the e*Gate Enterprise Manager to ensure each module was successfully loaded into the schema.

3 The “install” script creates soft links in $EHOME for the following 6 scripts:

- start_egate → <RETEK_INSTALL_DIR>/RIB<version>/Rib_Support/start_egate
- stop_egate → <RETEK_INSTALL_DIR>/RIB<version>/Rib_Support/stop_egate
- start_rib → <RETEK_INSTALL_DIR>/RIB<version>/Rib_Support/start_rib
- Egate.txt → <RETEK_INSTALL_DIR>/RIB<version>/Egate/Eways/Egate.txt
- start_cb → <RETEK_INSTALL_DIR>/RIB<version>/Rib_Support/start_cb
- plist → <RETEK_INSTALL_DIR>/RIB<version>/Rib_Support/plist

4 Continue the install process: RIB schema configuration.

Load RIB Components - Manual Instructions

These instructions are provided in order to individually load modules into the schema. Skip this section if the automated process was used and completed successfully.

Each RIB module can be loaded into the RIB103 schema manually, if necessary, by running the “Import Definitions from File” feature of the e*Gate Enterprise Manager GUI.

1 From a Windows PC, which has the e*Gate GUI installed, put the <MODULE>.zip file(s) to be loaded on an accessible drive.

2 Start the e*Gate Enterprise Manager.

3 Log in to the RIB103 schema.

4 Select File > Import Definitions from File. The Import Wizard is displayed.

5 Click **Next**. The Step 1 page is displayed.

6 Select the “Module” radio button. Click **Next**. The Step 2 page is displayed.

7 Locate/select the <MODULE>.zip file to be imported into the RIB103 schema. Click **Next**.

8 Click **Finish**. The Import Component dialog box is displayed, asking for confirmation as to which Participating Host/Control Broker pair to import into. Click **OK**.

9 Repeat the above steps for each <MODULE>.zip file that you wish to import.

10 Continue the install process: RIB schema configuration.

Install Delta Release

Preliminary Steps

1 Log into the UNIX system as the “egate “ user.

2 Copy the RIB .tar files from the RIB installation CD to the <RETEK_INSTALL_DIR> directory.

3 Once the RIB .tar files have been copied over, extract each file and change the permissions on the extracted files to make them writable. If you are installing multiple version numbers of the RIB, be sure to copy and extract the tar files of all versions you intend to install.

- tar xvf ‘filename’
- chmod –R 755 *

4 Change directories to <RETEK_INSTALL_DIR>/RIB<version>.

5 Edit the file egate_profile. If you are installing multiple version numbers, modify the file found in the most recent RIB<version> directory. Make sure the settings for the following variables are correct for your environment:

- EHOME – The directory where SeeBeyond e*Gate was installed.
- RETEK_INSTALL_DIR – The install directory referenced in Step 2.
- EGATE_SERVER_NAME – The name or IP address of the server e*Gate and the RIB software are installed on.
- EGATE_SERVER_PORT – The port that the e*Gate Registry Host was installed on during the SeeBeyond e*Gate Integrator install.

6 Edit the “egate” user’s .profile located in the “egate” user’s home directory. Add and entry at the end of this file that sources the egate_profile modified in step 5 above.

- . <RETEK_INSTALL_DIR>/RIB<version>/egate_profile
- (eg: . /files0/egate/INSTALL/RIB<version>/egate_profile)
- Ensure that ‘.:’ (dot colon) is at the beginning of the egate user’s PATH variable.
- (eg: PATH=.:${PATH} ; export PATH)
- Source the “egate” user’s .profile after making these modifications or start a new Unix session before continuing.

Import RIB Components

Load RIB Components – Automated Instructions

A script to insert the new registry modules can be found in the following directory:

```
<RETEK_INSTALL_DIR>/RIB<version>/Migration_Scripts
```

Notes:

- If your RIB103 schema has a password for the Administrator user that is different than the default of "STC" you will need to edit the install script and replace the two occurrences of "STC" with your password (case sensitive).
- The system must be able to locate the unzip and zip utilities for the importeways script to work. If not, update the PATH variable in the egate user's .profile file.
- Additionally, a working version of perl must be installed on your machine. Be sure the PATH variable contains an entry to where this file resides.
- The DISPLAY variable must be set to the IP address of the machine that the install will run on.

Unless all of these requirements are met, the manual method to import the RIB modules will have to be used.

1 From this directory, run the "install" script (eg ./install). If you intend to install multiple RIB versions at once (eg 10.3.4) execute "./install" from the $RETEK_INSTALL_DIR/RIB1033/Migration_scripts directory.

   a The ./install script requires input parameters defining which version(s) of the RIB you wish to install (eg ./install 1034). Executing the install script with no parameters will provide you with a listing of valid input parameters. This listing is compiled by scanning the RETEK_INSTALL_DIR for RIB<version> directories, which are created when the user extracts the tar files into the RETEK_INSTALL_DIR.

      The user may input multiple version numbers or they may input a single version number. If the user is creating a new RIB103 schema, including version 10.3, a Delta and/or Custom version(s), the user may wish to install multiple versions at once.

   b After providing "./install" with valid input parameters, the user will be prompted to verify the order of install. When installing multiple versions, the user must be certain to install the older RIB versions first.

   c "./install" will then prompt the user to create a backup of the RIB103 schema. Choosing to do so will create a .zip of the schema that is currently installed before proceeding to import new modules.

   d Following this, "./install" will prompt the user for an alternative import control file. Choosing "Yes" is the default installation path and will proceed to import ALL components included in the release. Choosing "No" will require the user to input an alternate control file containing a customized list of components which to import. This can be utilized by clients which have customized one or more of the Retek provided RIB components to ensure that the installation process does not overwrite their customized code.

e If the user has chosen to install multiple RIB versions, "./install" will import the RIB components associated to the first input parameter first. After successfully completing, "./install" will import the RIB components associated to the second input parameter, etc. If, after successfully importing the RIB components of any version(s) the user chooses to exit the install process, a listing of which versions where successfully completed will be provided to the user. These versions do not need to be re-installed.

2 Once the script has completed, use the e*Gate Enterprise Manager to ensure each module was successfully loaded into the schema.

3 The "install" script creates soft links in $EHOME for the following 6 scripts:

- start_egate → <RETEK_INSTALL_DIR>/RIB< Version>/Rib_Support/start_egate
- stop_egate → <RETEK_INSTALL_DIR>/RIB< Version>/Rib_Support/stop_egate
- start_rib → <RETEK_INSTALL_DIR>/RIB<Version>/Rib_Support/start_rib
- Egate.txt → <RETEK_INSTALL_DIR>/RIB<Version>Egate/Eways/Egate.txt
- start_cb → <RETEK_INSTALL_DIR>/RIB<Version>/Rib_Support/start_cb
- plist → <RETEK_INSTALL_DIR>/RIB<Version>/Rib_Support/plist

Change directories to $EHOME and run the "start_cb" script.

3 Continue the install process: RIB schema configuration.

Load RIB Components - Manual Instructions

These instructions are provided in order to individually load modules into the schema. Skip this section if the automated process was used and completed successfully.

Each RIB module can be loaded into the RIB103 schema manually, if necessary, by running the "Import Definitions from File" feature of the e*Gate Enterprise Manager GUI.

1 From a Windows PC, which has the e*Gate GUI installed, copy the <RETEK_INSTALL_DIR>/RIBfor<APP><version>/Egate/eways-in/<MODULE>.zip file(s) to a drive accessible by the Windows PC.

   (e.g. /files0/egate/INSTALL/RIBforISO<version>/Egate/eways-in/ewInvReqToRMS.zip)

   (e.g. /files0/egate/INSTALL/RIBforRMS<version>/Egate/eways-in/ewItemZonePrcFromRMS.zip)

2 Open the e*Gate Enterprise Manager.

3 Log into the RIB103 schema.

4 Select File → Import Definitions from File. The Import Wizard is displayed.

5 Click **Next**. The Step 1 page is displayed.

6 Select the "Module" radio button. Click **Next**. The Step 2 page is displayed.

7 Locate/select the <MODULE>.zip file to be imported into the RIB103 schema. Click **Next**.

8 Click **Finish**. The Import Component dialog box is displayed, asking for confirmation as to which Participating Host/Control Broker pair to import into. Click **OK**.

9 Repeat the above steps for each <MODULE>.zip file that you wish to import.

10 Continue the install process: RIB schema configuration

RIB schema configuration

After the RIB schema has been imported, the schema must be configured for the site-specific environment. This section details the minimum configuration changes needed to get the RIB schema into an operational state. It assumes that all schema components will run on a single host and that all databases referenced are accessible from this host.

This chapter details the minimum changes needed for the RIB to run. It assumes that the RIB is deployed on a single host and that only a single JMS IQ Manager is needed. This deployment configuration is *not* appropriate for all RIB installations. Production environment deployments may choose to distribute different specific e*Ways and JMS queues among multiple hosts. This type of production deployment is not covered in this manual.

The following steps are required to configure the RIB schema:

1 Modify the main Participating Host and Control Broker configuration.

2 Modify the JMS IQ Manager configuration.

3 Modify Connection Point configurations.

4 Delete unused e*Ways.

5 Add/Copy e*Ways for additional components.

6 Edit the rib.properties file to correspond to the system.

7 Create/modify startup scripts.

Step 1: Modify the main Participating Host and Control Broker configuration

The first step in the RIB messaging schema configuration is to modify the main participating host and control broker's configuration. The RIB103 schema includes a single participating host and control broker that contains all of the messaging e*Ways and associated components. If these are not modified, then the configuration will attempt to resolve host names and ports as specified by the supplied/shipped configuration.

Alternatively, you can change the configuration of the participating host and its supplied control broker.

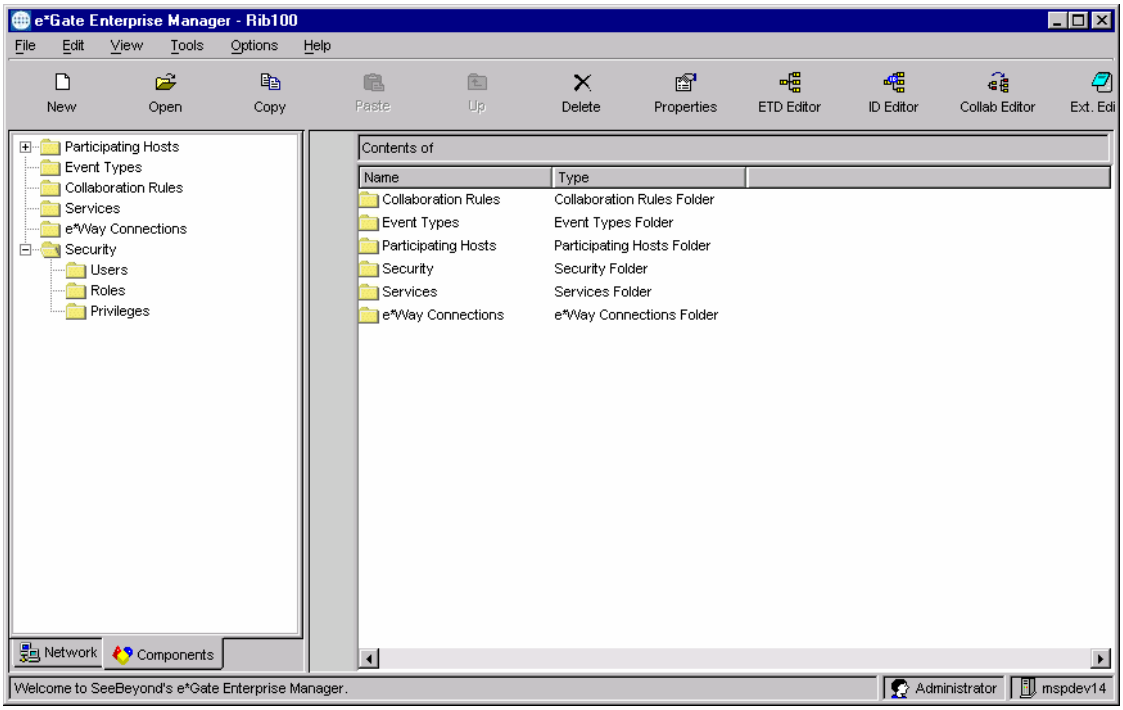
Changing these configurations is a manual process performed with the e*Gate Enterprise Manager application. This application must be installed on a Microsoft Windows 2000 or Microsoft Windows NT platform. Specific platform requirements are detailed in the SeeBeyond *e*Gate Integrator Installation Guide.*

These instructions modify both the names and IP address of the participating host and control broker. The name of the control broker must match any start-up scripts used.

Modify the configuration

1 Open the e*Gate Enterprise Manager.

2 Connect the e*Gate Enterprise Manager to the RIB103 schema. The following window is displayed:

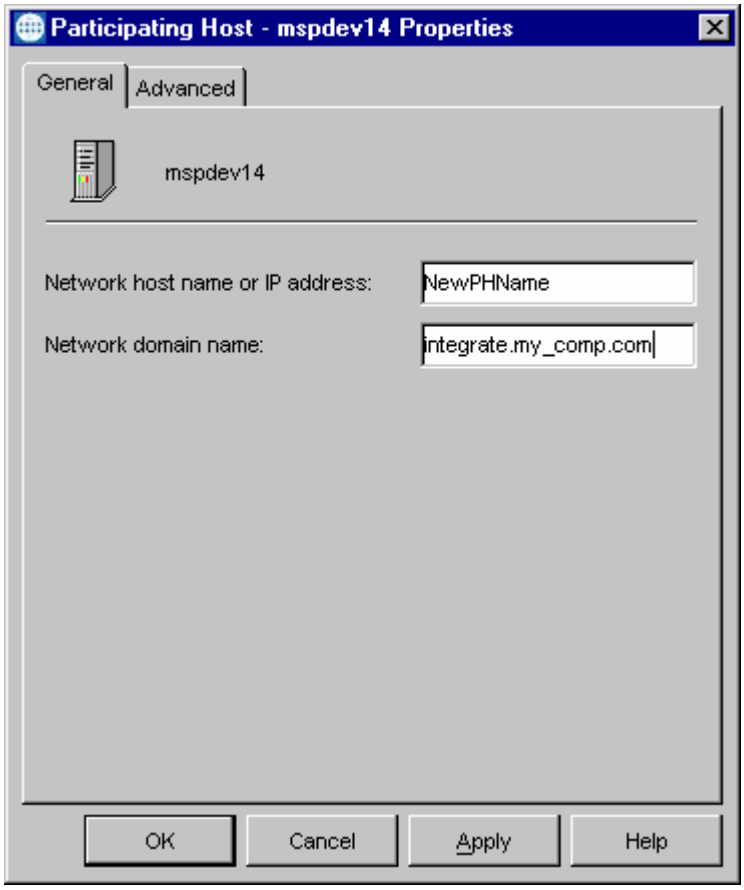


The main e*Gate Enterprise Manager window

3 Right-click on the first active participating host displayed. A command list is displayed.

Note: An active participating host is one *without* the string "(inactive)" as part of its name. If there is *not* a participating host *without* the "(inactive)" string, refer to the SeeBeyond System Administrator's Guide for instructions on how to activate the correct participating host.

4 Select **Properties…**. The Participating Host Properties dialog box is displayed.



Participating Host Properties window

5 In the Network host name or IP address field, enter the e*Gate server name.

6 In the Network domain name field, enter the correct network domain name for your environment.

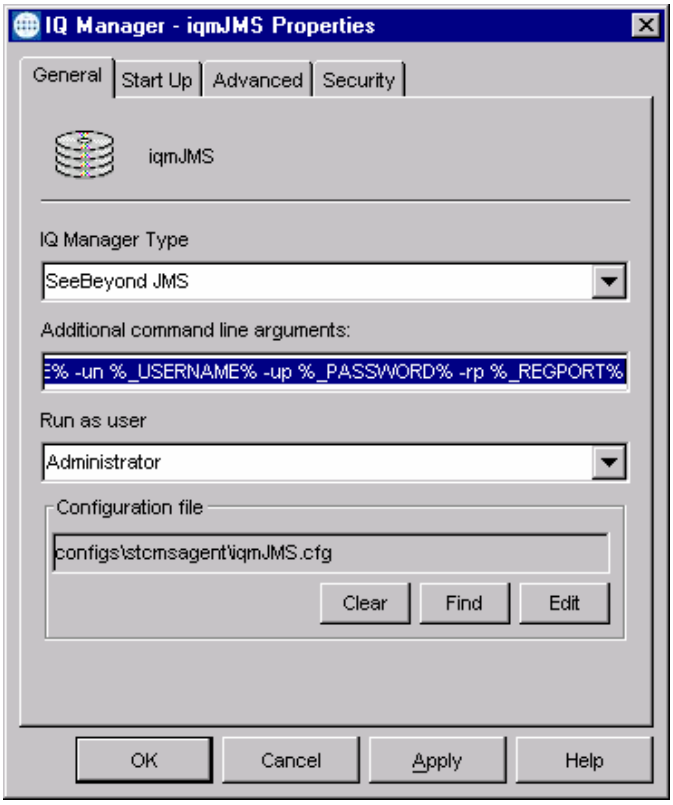
7 Click **OK**.

Step 2: Modify the JMS IQ Manager configuration

The JMS IQ Manager is initially configured to use the TCP/IP port number of **24053** for all e*Gate publishers and subscribers to connect to. If this port is used by other applications, then the JMS IQ Manager will not be able to be started. Complete this step only if port 24053 is **NOT** available.

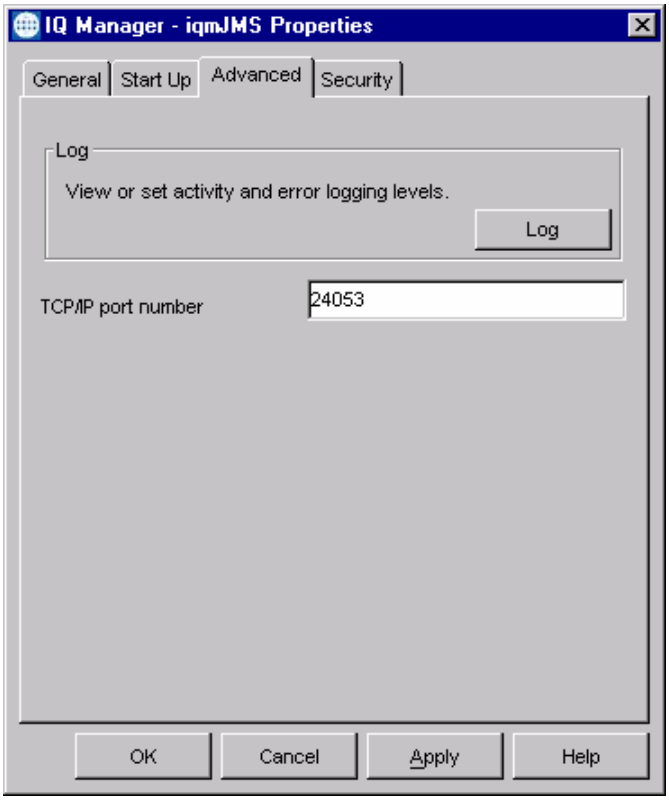
1 In the main e*Gate Enterprise Manager window, right-click on the **iqmJMS** queue manager. (The iqmJMS queue manager is towards the bottom of the Components frame, below all of the e*Ways.)

2 Select **Properties…**. The IQ Manager Properties dialog box is displayed.



IQ Manager Properties dialog box for iqmJMS

3 Click on the Advanced tab at the top of the window.



Advance IQ Manager Properties window

4 In the TCP/IP port number field, change the port number to an available port.

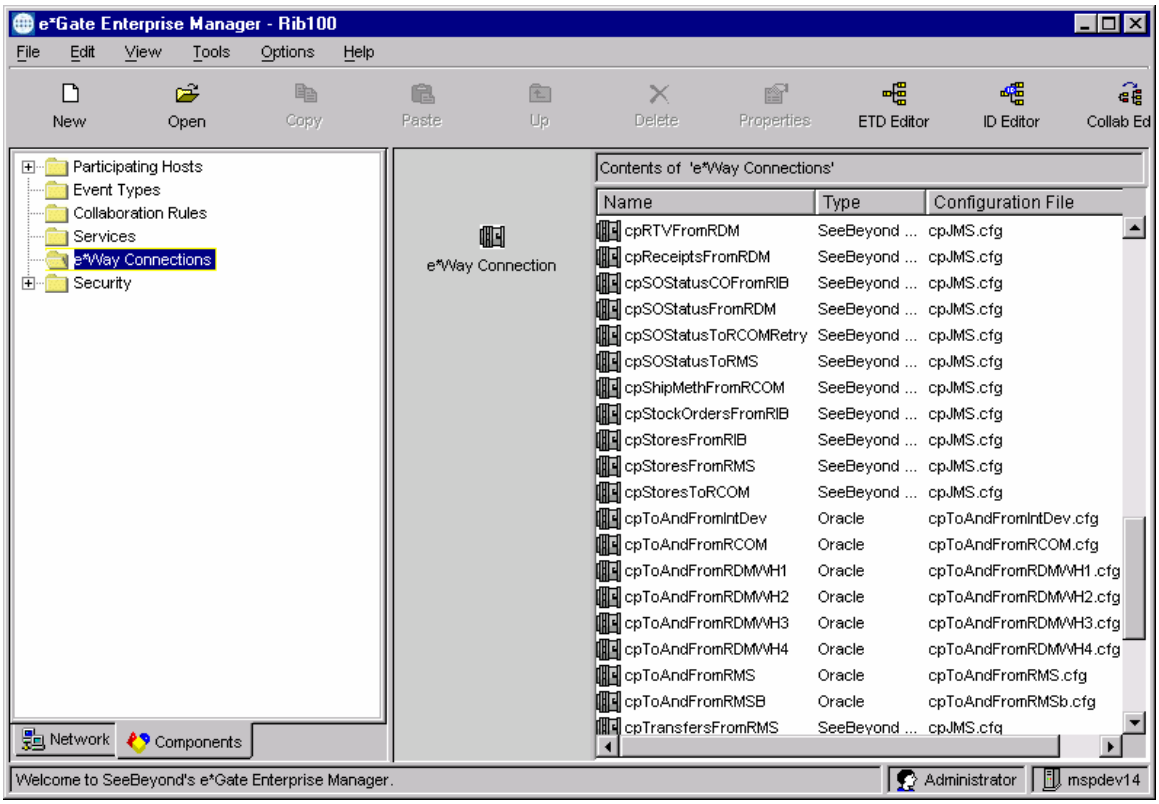
5 Click **OK**. Note the port number for the next step.

Note: If there are multiple instances of e*Gate running on a single Unix server, close attention must be paid to the registry, control broker and JMS ports. Runtime problems will be encountered if instances share the same ports.

Step 3: Modify Connection Point configurations

The next step is to modify the Connection Point configurations to reflect the JMS IQ Manager and Oracle databases used. This is performed in the e*Gate Enterprise Manager application.

- From the main window, click on the e*Way Connections folder. The window changes to reflect the available connections.



Connection Points

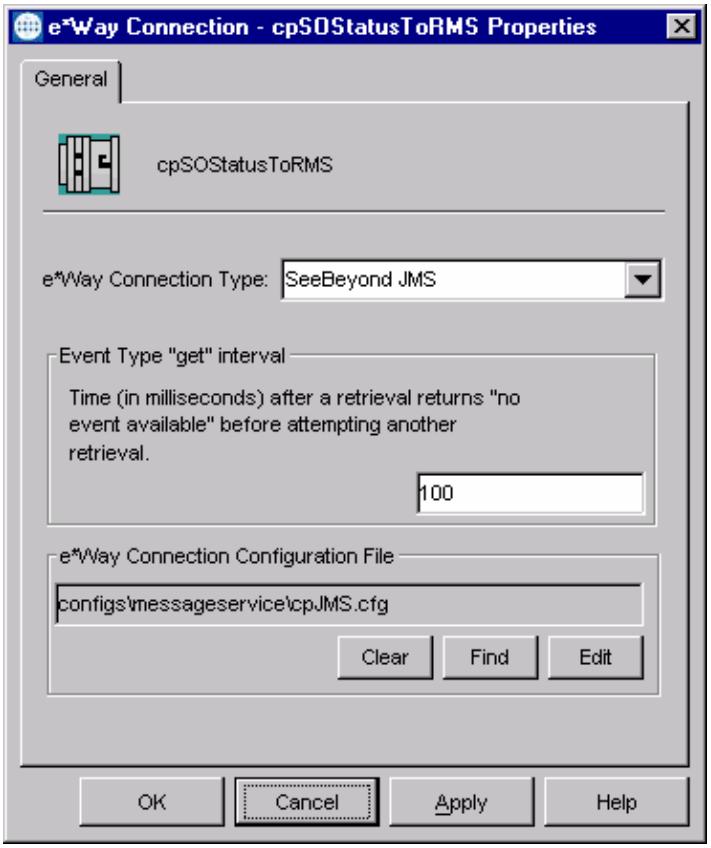
There are two types of connection points supplied with the RIB: SeeBeyond JMS and Oracle.

Change SeeBeyond JMS Connection Points

The SeeBeyond JMS connection points must connect to a known JMS IQ Manager. This requires knowledge of both the port number and host name. The host name is the name of the host used in step 1 "Modify the main Participating Host and Control Broker configuration". The TCP/IP port number is initially set to 24053. Change the TCP/IP port number only if Step 3 changed the port number of the iqmJMS IQ Manager. Otherwise, leave the port number as 24053.

Ensure that the connection point connects to the correct JMS IQ Manager. Note that in the default installation, all SeeBeyond JMS Connection points share the same configuration file. This being the case, only ***one*** of the SeeBeyond JMS connections needs to be modified.

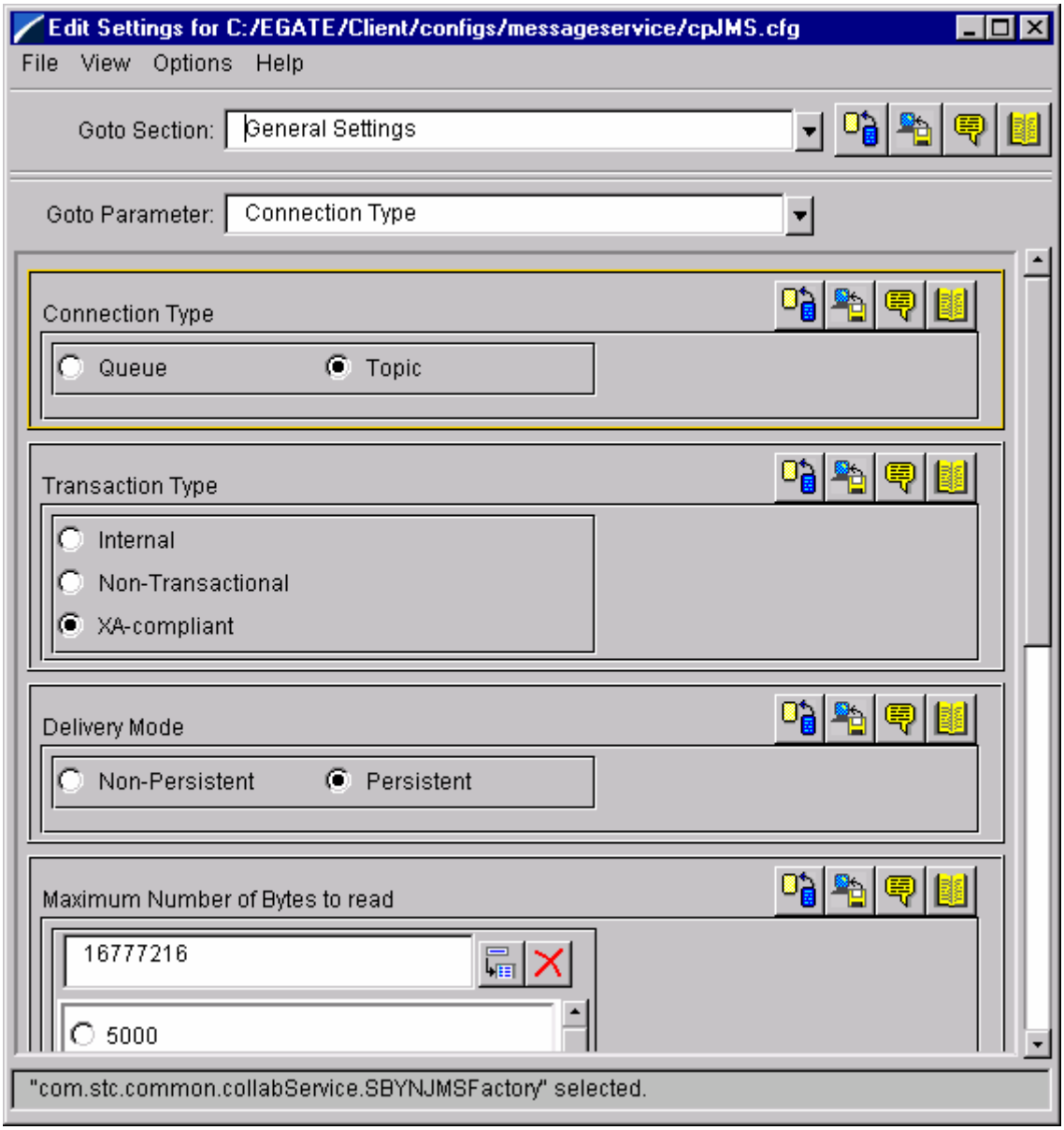
1 Locate any one of the SeeBeyond JMS connection points, right click on it and select **Properties**. The e*Way Connection Properties dialog box is displayed.



e*Way Connection Properties window

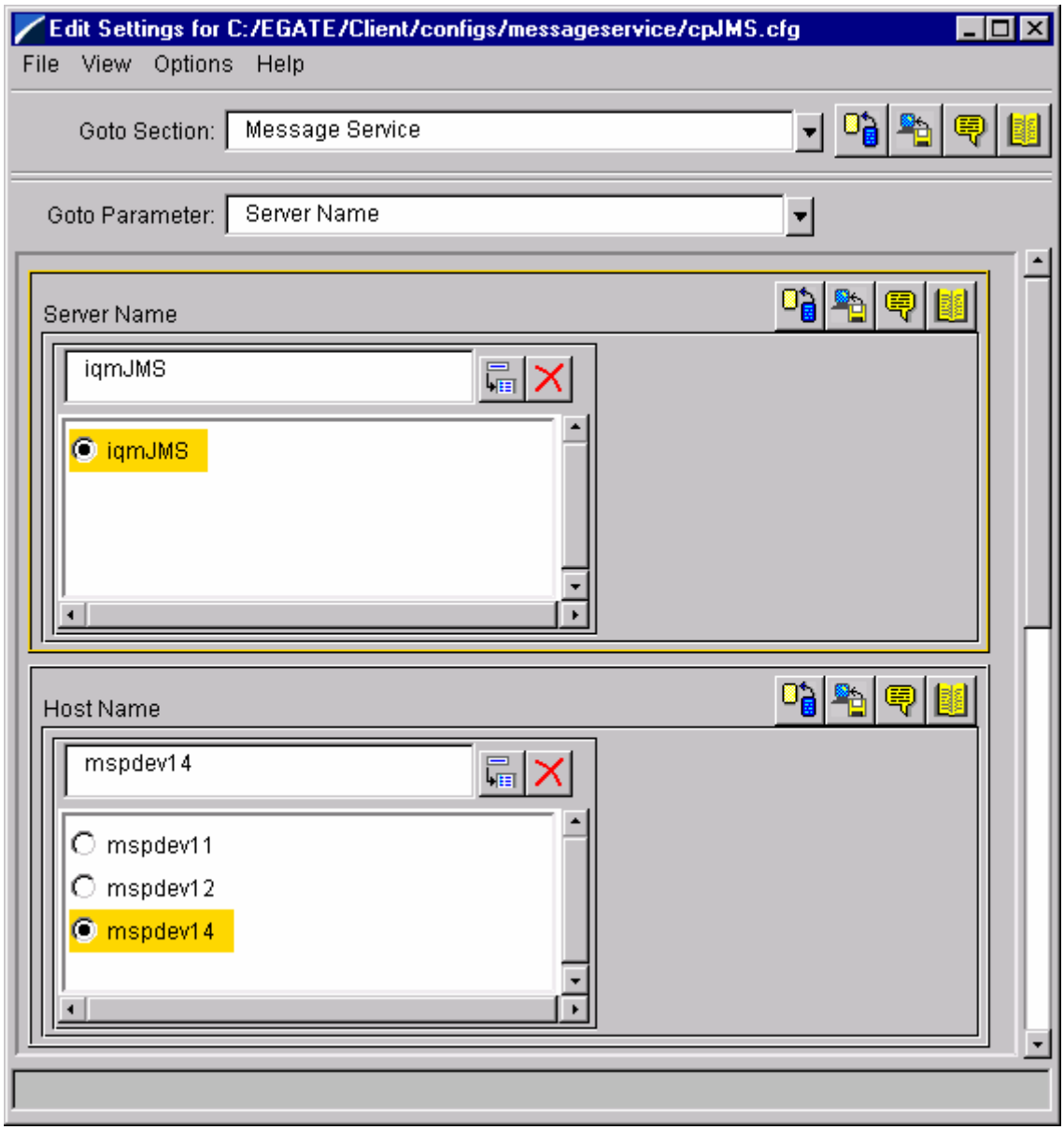
2 To change the address of the JMS IQ Manager the connection point connects to, edit the configuration file from *one* of the connection points using it. Multiple connection points may use the same connection point for sending messages to and from the JMS queue. The RIB schema initially uses only a single JMS queue for all messages.

3 Click **Edit** to change the address of the queue associated with the e*Way Connection Configuration File section of this properties window. The Connection Point configuration file edit dialog box is displayed.



Connection Point Configuration Edit window (General Settings Section)

4 In the GoTo Section field, select the **Message Service** section from the drop-down list.



Connection Point Configuration Edit (Message Service Section)

5 In the Host Name field, enter the name of the host on which the JMS IQ Manager resides. If all components are running on the same host, this is the same name as specified in Step 1.

6 If the JMS IQ Manager's TCP/IP Port number was changed in Step 3, then scroll down to the Port Number field and enter the number used in Step 3.

7 Select File > Save to save the new configuration.

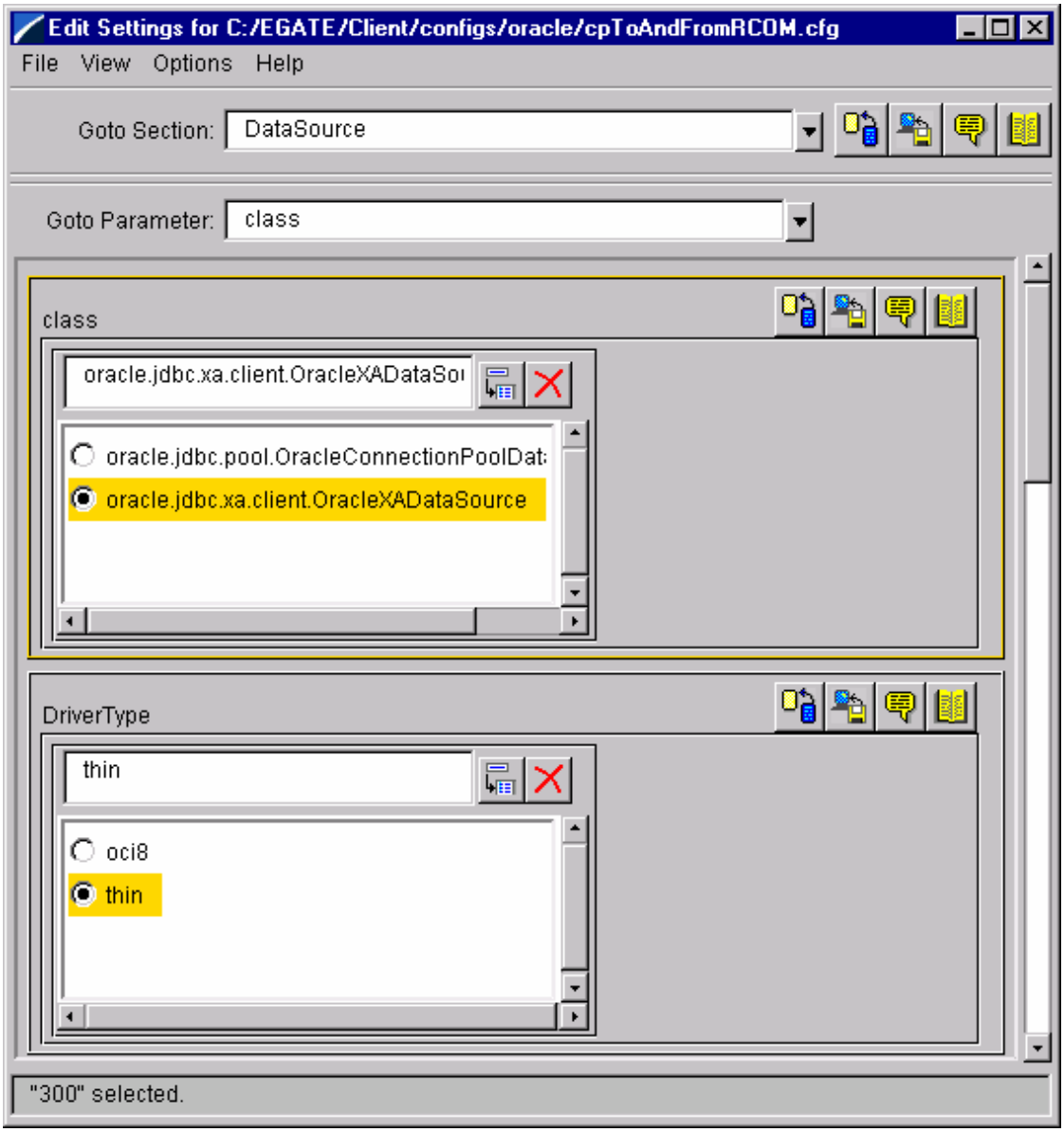
8 Select File > Promote To Runtime to make the configuration change take effect in the schema.

9 Select File > Close to exit the window.

Changing Oracle Database Connection points

All of the Oracle database Connection points must be altered to reflect the database instance and the user-ID/login for each of the applications.

1 Open the Connection Point Configuration window for the Oracle Connection Points, in the same manner as was done in the previous section for the SeeBeyond JMS Connection points.



Oracle Database Configuration Edit window (DataSource Section)

2 All configuration parameters of interest are found in the **DataSource** section.

3 The table below lists which parameters should be changed.

Parameter Name	Description
Class	Specifies the name of the Java class in the JDBC driver (Usually oracle.jdbc.xa.client.OracleXADataSource)
DriverType	This is the JDBC driver type (Usually thin)
ServerName	Name of server to connect to. Must have a valid Oracle listener.
PortNumber	Database connection port number. (Usually 1521.)
DatabaseName	Database System ID (SID).
user name	Login name to use
Password	Login password to use. This is stored in an encrypted form and displayed as a series of asterisks.

4 Change all Oracle database connection point settings.

You can delete the connection points used by publishers and subscribers for applications not installed. However, there is no harm in leaving these connection points as is.

Note: Be sure to save and promote to runtime any changes made in the connection point configuration files.

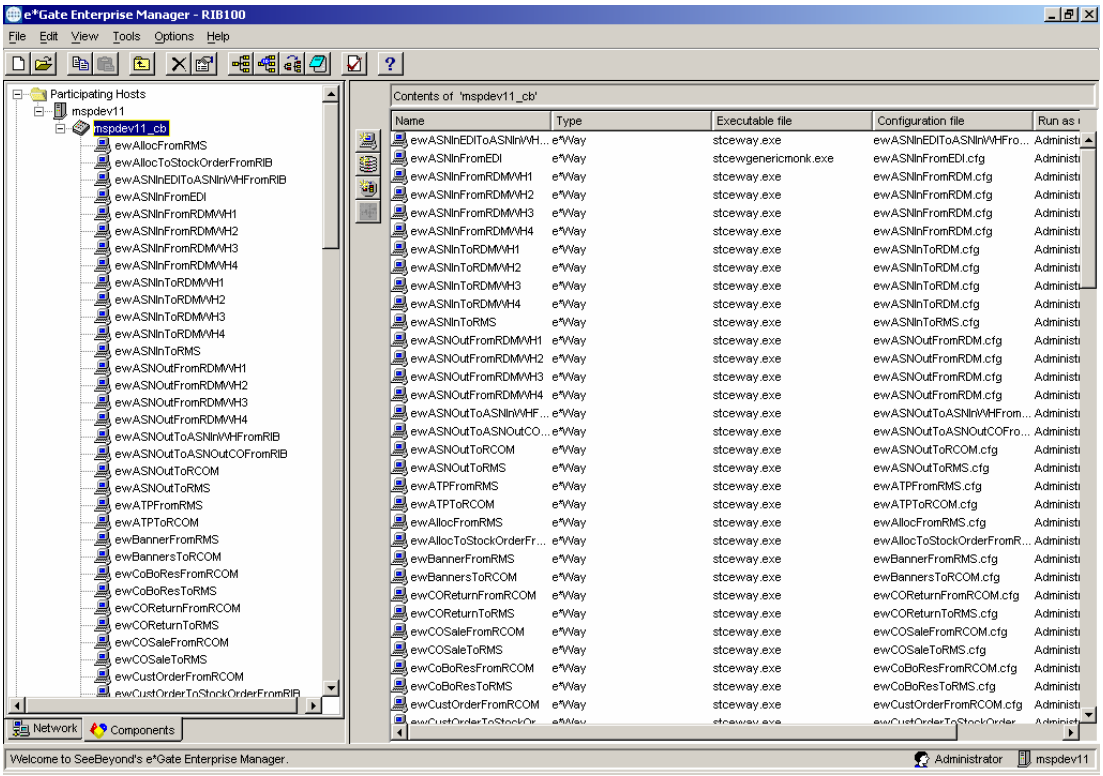
Step 4: Delete unused e*Ways

If the entire suite of RIB supported applications are not available or are not used, then delete the e*Ways associated with these applications. For messages that are directed to multiple applications or application instances, the presence of an e*Way will cause the JMS queue to store messages until all subscribers have received them. If a subscriber exists and never starts nor successfully consumes a message, then the JMS queue will never delete its copy of the message. Eventually, the JMS queue will exceed its configured message storage limits and message publication will halt.

1 From the main e*Gate Enterprise Manager window, click on the Components tab in the lower left corner of the screen.

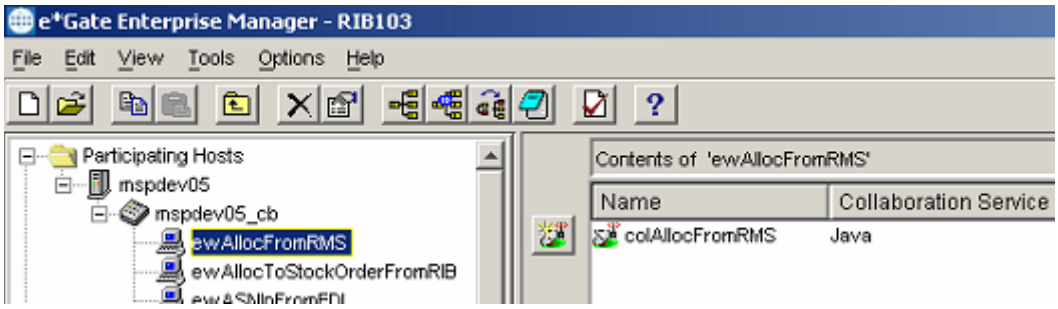
2 Expand the Participating Hosts folder in the left hand side frame, if not already expanded.

3 Expand the control broker containing all of the RIB e*Ways so that the list of e*Ways is presented.



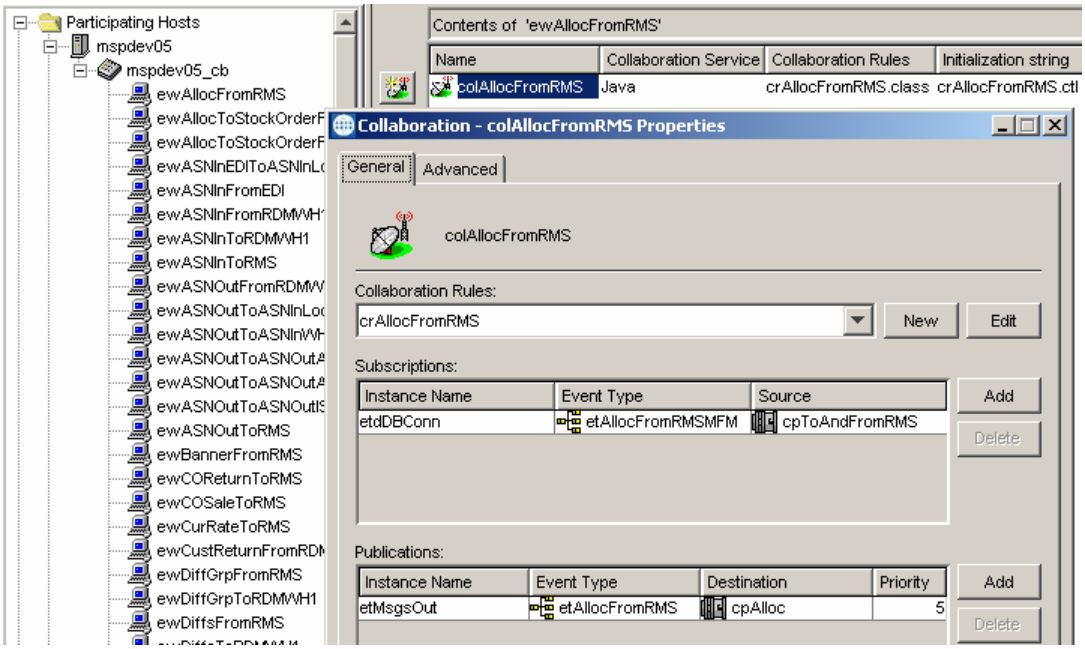
Expanded control broker

4 Highlight the e*Way you wish to delete by clicking on it in the left hand frame. This will give you visibility in the right hand window to the collaboration associated with this e*Way.



Viewing a collaboration

5 Open the collaboration associated with the e*Way you wish to delete by highlighting and double-clicking on it.



Collaboration properties window

6 At this point you must determine if the e*Way you wish to delete is a publishing e*Way or a subscribing e*Way. You can identify a publishing e*Way by determining its source of subscription. This is identified in the Subscriptions section of the Collaboration –properties window (opened in step 5 above). This column indicates which connection settings the e*Way will use. If this connection is to an outside application (i.e. RMS, RDM, ISO, RCOM, etc) this e*Ways is considered a publishing e*Way. The naming convention employed by Retek makes these easily identifiable. Examples include cpToAndFromRMS, cpToAndFromRDMWH1, and cpToAndFromISO.

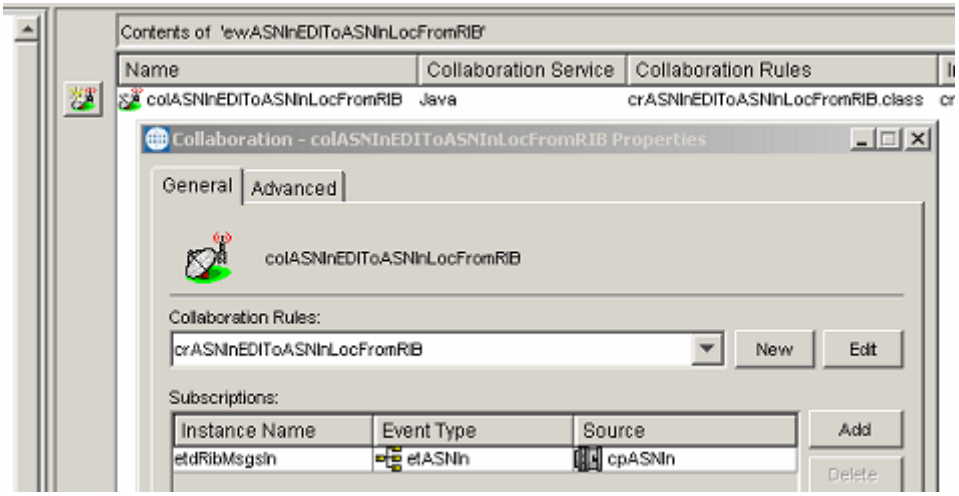
If you have determined that the e*Way you wish to delete is a publishing e*Way you may now skip ahead to step 8. If the e*Way you wish to delete is a subscribing e*Way, please proceed with the next step.

7 You have determined that your e*Way is a subscribing e*Way. Because this e*Way is a subscribing e*Way, there are additional pieces to remove that if left in a SeeBeyond schema could ultimately cause the performance issues alluded to in the opening paragraph.

When a registry is started in a SeeBeyond schema, it creates a unique subscriber for each subscribing e*Way/collaboration in that schema on the JMSQueue. SeeBeyond then ensures that each subscriber receives a message once. Until all subscribers for a particular message indicate to SeeBeyond that they have handled that message, a copy of the message is kept in the JMSQueue. Because of this "journaling" of messages, we must be sure that if we remove a subscribing e*Way from a SeeBeyond schema, we remove its subscriber as well. If we fail to do this, a copy of the message will be stored in the JMSQueue taking up memory resources.

Removing a Subscriber:

a To remove a subscriber from a JMSQueue you must first identify the Topic which the e*Way is subscribing to. The Topic can be found by opening the collaboration properties window and looking under the Event Type column in Subscriptions section of the window. Topics will always begin with the two letters "et" (I.E. etASNIn).

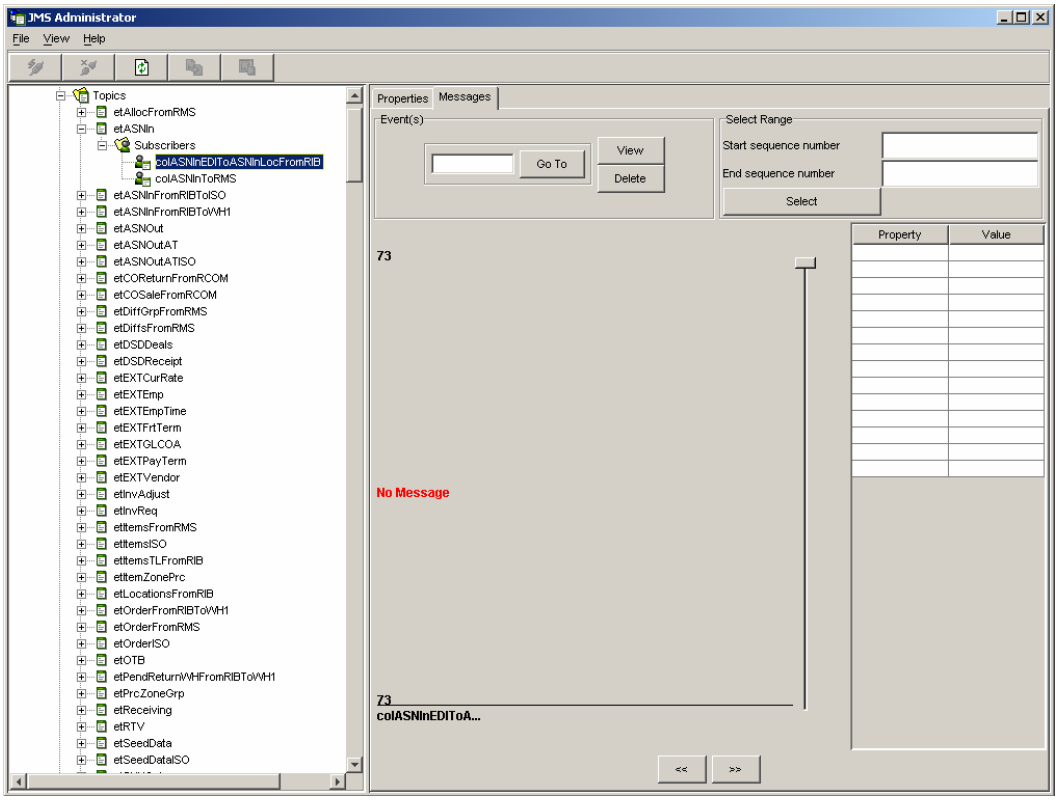


Identify the Topic in the Event Type column

b Be sure the JMS IQ is running. Open the JMS Administrator tool. Instructions on use of the JMS Administrator can be found in the SeeBeyond document – JMS Intelligent Queue User's Guide – Chapter 5: Managing Events and JMS IQ Managers.

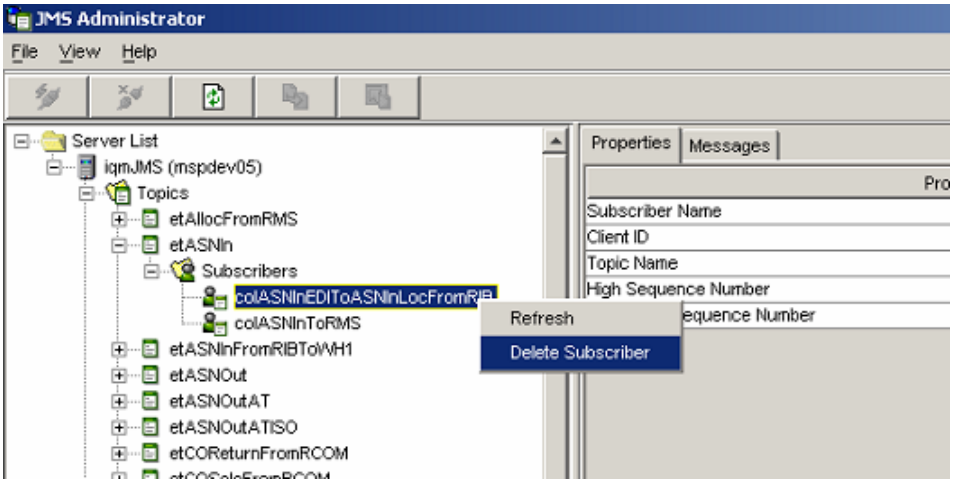
c From the list of topics given in the JMS Administrator, find the topic that matches the Topic you found in step A above. Once you find your topic, expand the Topic and Subscriber nodes in the GUI. Here you see a list of all collaborations that contain subscribers to this topic. Highlight the collaboration which matches the collaboration whose properties you looked at in step A.

d Before proceeding to the next step it is essential the user understand the significance of deleting a subscriber. The deletion of a subscriber also deletes any data from the JMSQueue for which that subscriber may be waiting to process. When deleting a subscriber that was not desired as part of a schema, this is of no concern. However, when deleting a subscriber because the functionality of its e*Way has been replaced, special care must be taken to insure no loss of data occurs. Before deleting subscriber which is being replaced by another, be sure that all data in the JMSQueue for the old subscriber has been processed. This can be done by shutting down the publishing e*Ways/process, clicking on the Messages tab in the right hand pane of the JMS Administrator and looking to see that No Message exists for the subscriber you wish to delete. If a message does exist, wait for the subscribing application to finish processing the message before proceeding.



Check to see no messages exist for subscriber

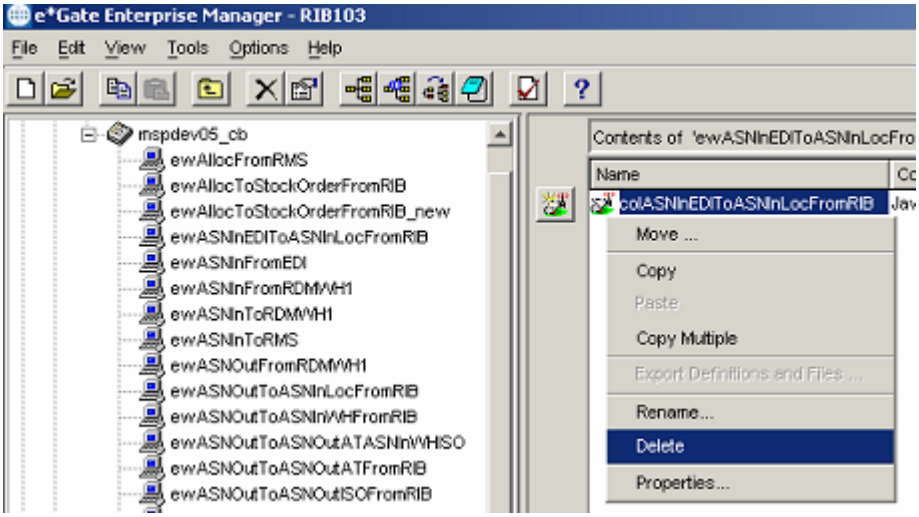
e Right click on the appropriate subscriber/collaboration and choose “Delete Subscriber” from the dropdown menu (note: the e*Way which corresponds to this subscriber must not be running in order to delete).



Deleting a subscriber

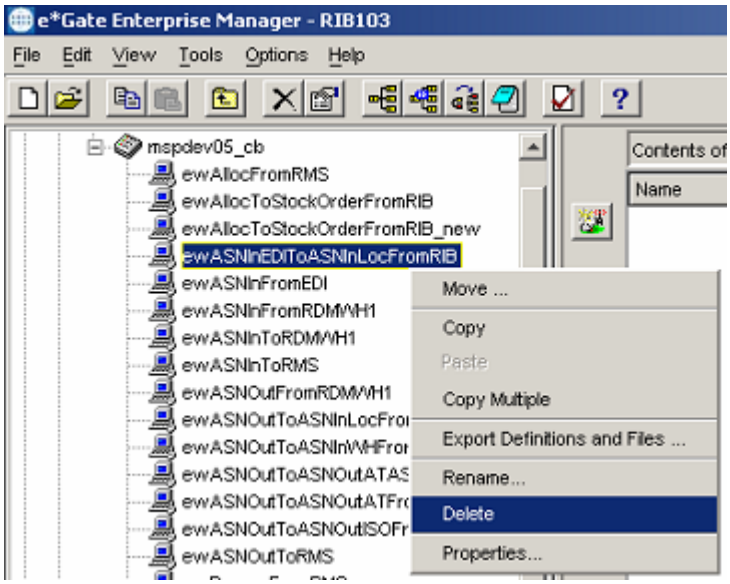
f Close the JMS Administrator and return to the e*Gate Enterprise Manager.

8 Right-click on the collaboration. Choose **Delete** to delete the collaboration. A confirmation dialog box is displayed.



Deleting the collaboration

9 Right-click on the e*Way. Choose **Delete** to delete the e*Way. A confirmation dialog box is displayed.



Deleting the e*Way

10 Log onto the Unix system using the "egate" user that owns the e*Gate files and executes the software.

11 Navigate to the Unix $EHOME directory. The Egate.txt file contains a listing of all of the e*Ways contained the RIB 10.3.4 release. Modify this file to no longer contain a reference to the e*Way you have just removed from your schema.

12 Repeat this process to delete all e*Ways which are not desired.

Step 5: Add/Copy e*Ways for additional components

Occasionally, there will be a need to add additional e*Ways to the imported schema. Often times, it is possible to copy an existing e*Way, reconfigure the various pieces that make up the e*Way, and continue from there. If it is necessary to add/copy e*Ways, please refer to the SeeBeyond e*Gate Users Guide for the correct procedures.

Step 6: Edit the rib.properties file to correspond to the system

Next, edit the rib.properties file to reflect the site-specific mappings and properties. The RIB import process copied a rib.properties file from <RETEK_INSTALL_DIR>/RIB<version>/Rib_Support/src to the $EHOME/client/classes directory. If a previous version of this file existed, it was renamed to rib.properties.bak. Some of the types of properties in this file are:

- Error Hospital specifics such as the max number of attempts to try for a failing message and the delay between each attempt.
- Multi-channel indication for Purchase Orders.
- Multi-threading settings.
- Facility ID mappings. These must correspond to codes in the RMS application for the correct routing of message to RDM instances.
- Logging specific settings.
- "no event" sleep duration settings.
- Directory location of RIBLOGS logging files. The RIBLOGS directory, by default, exists in the $EHOME/ directory. The pathname of the RIBLOGS directory should be modified before starting the e*ways.

  If the rib.properties file in $EHOME/client/classes is not to be used, then append the full path of the directory containing the file that will be used to the beginning of the CLASSPATH variable setting.

See the Retek Integration Bus Operations Guide for more information on the values for the rib.properties file.

Step 7: Create/modify startup scripts

The final installation step is to create RIB startup scripts for Unix systems. These scripts start up the SeeBeyond e*Gate registry and the control broker for the RIB103 Schema. Default scripts are provided for starting the registry service and the control broker in the $EHOME directory. Refer to the "start_egate" and a "start_cb" scripts.

The commands that implement this functionality are stcregd and stccb. When executed, they run as daemons. Depending on logging and other parameters, they may log items to their stderr or stdout files. These commands are detailed further in the following manuals:

- Retek Integration Bus Operations Guide
- SeeBeyond e*Gate Integrator User's Guide
- SeeBeyond e*Gate Integrator System Administration and Operations Guide

Step 8: Starting the RIB components for a new/modified schema

Start_rib is intended for use in starting the e*Way components of a new schema or when new e*Way components have been added to an existing schema. The RIB 10.3.3 architecture requires that the first time an e*Way is started in a schema, it is started in a particular order (subscriber, TAFR, publisher). The start_rib script uses an input file called Egate.txt to identify an e*Way as a subscriber, a TAFR, or a publisher. Egate.txt contains a listing of all of the e*Way components packaged with RIB 10.3.4. Start_rib is executed with one or several of the parameters listed within its usage. To view the start_rib usage, change Unix directories to $EHOME and run start_rib with no parameters.

It is necessary that start_rib is run to start the e*Way components immediately after installation and configuration of the RIB 10.3.4 components. The following command should be executed while the registry and control broker are running: start_rib JMS SUB TAFR. After these processes have successfully started (check to be certain connection points were accurately set and DB connection was successful), if the client is integrating with ISO, start the ISO services following SIM instruction. Run start_rib once more using the following command: start_rib PUB HOSP. All of the e*Way process listed in the Egate.txt file should now be running.

If desired, new e*Way components may be started using the start_rib command by adding the new components to the start_rib input file - Egate.txt. This file can be located in the $EHOME directory. Unused e*Way components may also be removed from Egate.txt.

If a full install is not being preformed, and integrating with ISO, proceed to Chapter 7 – ISO Integration.

Chapter 4 – Database triggers and Oracle dependencies

Database triggers

Once the RIB has been installed and configured, the publishing applications need to be told to begin to publish data. There are multiple ways to initiate the publishing process. Each Retek product's operation guide contains this information.

Oracle dependencies

In order for the Retek 10.3 RIB to function correctly, *you must install Oracle's XML Developer's Kit for PL/SQL on your database server*. This can be downloaded from Oracle Technology Network. The version of XML Developer's Kit for PL/SQL *must be dated 3/28/2002 or later* – there is a bug in prior version that will prevent the RIB from working correctly.

Verify RIB Error Hospital database tables

One feature of the RIB is the Error Hospital subsystem used to store and retry messages that have processing problems by a subscribing application. This facility allows for non-dependent messages to continue to be processed by the application until the failure has been resolved and the message successfully consumed.

These tables were created during the database portion of the RCOM, MDM, RDM, or RMS installations. The DDL to create these tables can be found on these products' installation CDs.

Note: For more detailed information on the RIB Error Hospital, refer to the <RETEK_INSTALL_DIR>/RIB<version>/Rib_Support/doc/index.html file.

For each Error Hospital, verify that the three hospital tables exist and that they have the correct columns. The three tables necessary are:

RIB_MESSAGE

```
DROP TABLE RIB_MESSAGE CASCADE CONSTRAINTS ;
            CREATE TABLE RIB_MESSAGE (
              MESSAGE_NUM        NUMBER(8)     NOT NULL,
              LOCATION           VARCHAR2(60)  NOT NULL,
              FAMILY             VARCHAR2(25)  NOT NULL,
              TYPE               VARCHAR2(30)  NOT NULL,
              ID                 VARCHAR2(255),
              RIB_MESSAGE_ID     VARCHAR2(255),
              PUBLISH_TIME       DATE,
              IN_QUEUE           VARCHAR2(1)   NOT NULL,
              MESSAGE_DATA       CLOB          NOT NULL,
              ATTEMPT_COUNT      NUMBER(4)     NOT NULL,
              MAX_ATTEMPTS       NUMBER(4)     NOT NULL,
              NEXT_ATTEMPT_TIME  DATE,
              DELETE_PENDING     VARCHAR2(1)   NOT NULL,
              TOPIC_NAME         VARCHAR2(255),
              THREAD_VALUE       NUMBER(22,8),
              JMS_QUEUE_ID       VARCHAR2(255),
              CUSTOM_FLAG        VARCHAR2(1)   DEFAULT 'F' NOT NULL,
              CUSTOM_DATA        CLOB,
              REASON_CODE        VARCHAR2(10)  NOT NULL,
              CONSTRAINT PK_RIB_MESSAGE
              PRIMARY KEY        (MESSAGE_NUM));
```

Note: The field MESSAGE_NUM in the table RIB_MESSAGE has an associated script to create the sequence. It is required to run this script – **rib_message_seq.sql**. This script can be found in the <RETEK_INSTALL_DIR>/RIBfor<APP><version>/XML_Utilities directory.

RIB_MESSAGE_FAILURE

```
DROP TABLE RIB_MESSAGE_FAILURE CASCADE CONSTRAINTS;
CREATE TABLE RIB_MESSAGE_FAILURE (
  MESSAGE_NUM  NUMBER(8)      NOT NULL,
  SEQ_NUMBER   NUMBER(2)      NOT NULL,
  TIME      DATE        NOT NULL,
  LOCATION  VARCHAR2(60)   NOT NULL,
  DESCRIPTION  VARCHAR2(4000) NOT NULL,
  ERROR_TYPE   VARCHAR2(2)    DEFAULT 'SY' NOT NULL,
  ERROR_CODE   VARCHAR2(25)   NULL,
  CONSTRAINT PK_RIB_MESSAGE_FAILURE
  PRIMARY KEY (MESSAGE_NUM, SEQ_NUMBER));
```

RIB_MESSAGE_ROUTING_INFO

```
DROP TABLE RIB_MESSAGE_ROUTING_INFO CASCADE CONSTRAINTS;

CREATE TABLE RIB_MESSAGE_ROUTING_INFO (
              MESSAGE_NUM         NUMBER(8)         NOT NULL,
              SEQ_NUMBER          NUMBER(2)         NOT NULL,
              NAME                VARCHAR2(25)      NOT NULL,
              VALUE               VARCHAR2(25)      NOT NULL,
              DETAIL1_NAME        VARCHAR2(25),
              DETAIL1_VALUE       VARCHAR2(300),
              DETAIL2_NAME        VARCHAR2(25),
              DETAIL2_VALUE       VARCHAR2(300),
              CONSTRAINT PK_RIB_MESSAGE_ROUTING_INFO
              PRIMARY KEY (MESSAGE_NUM, SEQ_NUMBER));
```

RIB_MESSAGE_HOSPITAL_REF

```
DROP TABLE RIB_MESSAGE_HOSPITAL_REF CASCADE CONSTRAINTS;

CREATE TABLE RIB_MESSAGE_HOSPITAL_REF (
  MESSAGE_NUM     NUMBER(8)      NOT NULL,
  SEQ_NUMBER      NUMBER(2)      NOT NULL,
  HOSPITAL_REF    VARCHAR2(8)    NOT NULL,
  LOCATION     VARCHAR2(60),
  MESSAGE_FAMILY  VARCHAR2(25),
```

```
NEW_REASON_CODE VARCHAR2(10),
OLD_REASON_CODE VARCHAR2(10),
            CONSTRAINT PK_RIB_MESSAGE_HOSPITAL_REF
            PRIMARY KEY (MESSAGE_NUM, SEQ_NUMBER));
```

Note: If these tables need to be created outside of a Retek Application's installation, scripts called 'rib_ddl.sql' and 'rib_message_seq.sql' can be found in the <RETEK_INSTALL_DIR>/RIBfor<*APP*><version>/XML_Utilities directory.

- If your database already has these tables in place from a RIB installation prior to version 10.3, the rib_message table may need to be updated with new table changes. Executing the 0001_rib_message.sql script, which can be found in the <RETEK_INSTALL_DIR>/RIB<version>/DBC directory, applies these changes to the table. If the rib_message table contains data, the data should be exported before the script is run, and imported back into the table.
- Check the structure of the rib_message table, prior to running this script, to determine if this script needs to be applied or not.

Note: This script contains table changes that include the addition of a new NON-nullable column, 'custom_flag' (position 17), that should be defaulted to 'F' when importing data back into the table. All other new columns can be null.

RIB_DOCTYPES table and DTD files

The RIB_DOCTYPES table should also have been created and populated by the Retek Application installation(s) (e.g. RMS). The integration with RCOM now requires that the DTD files themselves be network accessible.

To accomplish this, all of the DTD files should be deployed to a web server running at the client. Once this is done, and the URL to the DTDs is known, the value in the DOC_TYPE_URL column will need to reflect this location.

The rib_doctypes_rms.dat file which is found in the <RETEK_INSTALL_DIR>/RIBforRMS/XML_Utilities directory is the script that RMS uses to populate the table with data. The URLs can be globally replaced with the new one and the script re-run.

Note: If custom RMS, a custom version of this script will exist. It should be applied instead.

RMS 9 Integration

If integrating with RMS 9.0.x, the client has the option of publishing/subscribing messages to/from RMS without the use of SeeBeyond. The RmsBatchMsg class will export RMS publishing data into XML files, and import XML file data into RMS subscribers.

In order to use the RmsBatchMsg functionality, make sure to install the rmsbatch components from RIBforRMS, including the properties files. Along with the rmsbatch.jar included in the rmsbatch components, the following jars need to be installed and added to the CLASSPATH:

- retek-rib-support.jar
- retek-pub-trans.jar
- log4j.jar
- xmlParserAPIs.jar
- classes12.jar
- xerces_1_2_3.jar

For details on editing the properties files, refer to the RIB 10.3.4 Operations Guide.

Chapter 5 – RIB Administration Tool installation

The RIB Administration Tool is a web-based tool provided to perform RIB administration functions, which include the Hospital Administration GUI, the Message Statistics GUI and the RIB Properties Editor GUI.

There are two versions of the Hospital Administration GUI that can be used to administer problem messages that have been put in the hospital database tables. One version is a java executable/application and one is web/browser-based. Retek recommends you use the web/browser-based version, if possible, because it does not require anything other than java runtime to be installed on individual workstations.

This section explains the installation of the web-based version; see the next section for the executable/application-based version of the Hospital Administration GUI.

Web-based version install

Prerequisites

1 A J2EE-compliant web server from which to deploy the gui.war file. Jakarta Tomcat is the web server that the RIB Administration Tool was developed on, and is the recommended web server to use.

2 Java Runtime Engine (JRE) version 1.4 installed on all workstations/PCs that will be accessing the GUI via their web browser.

Note: If these two prerequisites cannot be met, install the java executable/application version of the Hospital UI (see next section).

Install RIB Administration Tool

1 Copy gui.war from the <RETEK_INSTALL_DIR>/RIB<version>/Rib_Hospital_Gui/build directory to the web server and deploy it. If deploying on Tomcat, place the gui.war file in the Tomcat /webapps directory and reload the server.

2 After deployment, locate and modify the gui.properties configuration file in the web application directory (for Tomcat, this would be in $CATALINA_HOME/webapps/gui/). The entries in this file that must be changed are listed below:

```
##############################################
# GUI Project Variables
GUI.ProjectHost=
GUI.ProjectPort=
GUI.ProjectName=gui
GUI.TimingsLogFile.Path=
GUI.TimingsLogFile.Name=timings_rib.log
GUI.rib.properties.default.FilePath=
GUI.rib.properties.default.BackupFileExt=.bak
```

Where:

- GUI.ProjectHost is the name or IP address of the server that the J2EE web server is running on. GUI.ProjectPort is the http listener port of the J2EE web server. These are the values that will be set for all of the GUI applets, and will override the applet's baseurl.getContext lookups to find the URL to the servlets. If for any reason this lookup does not find the correct host and port, or if a servlet residing on a different host or port is preferred, set these values appropriately.
- GUI.ProjectName should be set in the properties file to contain the name of the project installation (installed application name) on the web server. The applets will use this name to build the URL to the servlets. The default installation name is "gui".
- GUI.TimingsLogFile.Path and the GUI.TimingsLogFile.Name should be set to contain the default path to the timings log file and the default name for the log file for the Message Statistics GUI Applet. When this applet is loaded, it will display a window where the user can enter the path to the log file and the parameters to pass into the RibTimings class to gather the statistics. If no value is entered, the log file path text field on this window will initially be blank. The TimingsLogFile.Name is defaulted to "timings_rib.log".
- GUI.rib.properties.default.FilePath should be set to the default file path of the rib.properties file. This will be displayed in the RIB Properties Editor's connection window as the default File Name, which the user can modify before retrieving the file from the server.
- GUI.rib.properties.default.BackupFileExt should be set to contain the default file extension the RIB Properties Editor will use when creating a backup copy of the rib.properties on the server. This will be displayed in a dialog that appears on saving the file. The user can modify the extension of the backup file to whatever they choose before the file is saved.

Example:

```
###############################################
# GUI Project Variables
GUI.ProjectHost=localhost
GUI.ProjectPort=8080
GUI.ProjectName=gui
GUI.TimingsLogFile.Path=/files0/egate/timings/
GUI.TimingsLogFile.Name=timings_rib.log
GUI.rib.properties.default.FilePath=/files0/egate/egate/client/classes/
GUI.rib.properties.default.BackupFileExt=rib.properties.bak
```

Note: All File Path entries in this properties file should end with a file separator character, since the file name will be appended to the end of the path (ie "/" or "\\").

2 Locate and modify the gui.servlet.properties file in the web application directory under WEB-INF/classes (for Tomcat, this would be in $CATALINA_HOME/webapps/gui/WEB-INF/classes). The entries in this file that can be changed are listed below:

Where:

```
##########################################
# GUI Project Variables
GUI.jdbc.driver=oracle.jdbc.driver.OracleDriver
GUI.rib.properties.SessionTimeout=900
GUI.rib.properties.local.FilePath=
```

- GUI.jdbc.driver should be set to the driver used to log in to the database for the main Portal login. The default driver that is contained the gui.war is an Oracle database driver.
- GUI.rib.properties.SessionTimeout should be set to the amount of time in which a session is timed out after being idle. The index.jsp will set the HttpSession.setMaxInactiveInterval(); The default is 900 seconds (15 minutes).
- GUI.rib.properties.local.FilePath should be set to the directory where the Rib Properties should locally save the file while editing it. The default is to set this to <appserver-installation-directory>/<installed-application-name>/temp/, but can be changed to any directory on the application server.

Example:

```
##########################################
# GUI Project Variables
GUI.jdbc.driver=oracle.jdbc.driver.OracleDriver
GUI.rib.properties.SessionTimeout=900
GUI.rib.properties.local.FilePath=/files0/jakarta-
tomcat/webapps/gui/temp/
```

Note: All File Path entries in this properties file should end with a file separator character, since the file name will be appended to the end of the path (ie "/" or "\\").

Internationalization

The RIB Administration Tool is initially configured to run in English. To set the RIB Administration Tool to the language of your choice:

1 Edit two files in the web application directory: gui.properties and gui.servlet.properties. Set the GUI.language and GUI.country values to the language you wish to use. If no language or country code is specified for these properties, the applets will default to the locale in which they are running.

The following settings are available:

For English: GUI.language=en
GUI.country=US

For French: GUI.language=fr
GUI.country=FR

For German: GUI.language=de
GUI.country=DE

For Spanish: GUI.language=es
GUI.country=ES

For Korean: GUI.language=ko
GUI.country=KO

For Japanese: GUI.language=ja
GUI.country=JP

Default locale: GUI.language=
GUI.country=

2 In the web application directory, there is a folder containing specific files for each language (based on the language code: DE, EN, ES, FR, JA, KO). Copy the files from this directory into the web application directory (overwriting the existing files).

3 If Jakarta Tomcat is the web server being used, it must be reloaded for the changes in steps 1 and 2 above to take effect. Other web servers may require reloading as well.

Install JRE

1 Java Runtime Engine (JRE) 1.4 must be installed on all workstations/PC's that will be accessing the GUI via their web browser.

Note: The 1.4 JRE can be downloaded at no charge from Sun's Java web site (http://java.sun.com/j2se/downloads.html).

Test Error Hospital GUI Applet

- To test the GUI, enter the following URL in a browser:
- http://<server>:<port>/<ProjectName>/index.jsp
  - Server = name or IP address of the server that the J2EE web server is running on (should be the same value as GUI.ProjectHost in gui.properties)
  - Port = http listener port of the J2EE web server (should be the same value as GUI.ProjectPort in gui.properties)
  - ProjectName = value of GUI.ProjectName in gui.properties

  Example: http://localhost:8080/gui/index.jsp

Files and classes contained in the war file

Classes

com.retek.rib.gui.AppletCoder: used for encoding and decoding information sent from applets to servlets

com.retek.rib.gui.AppletDialog: used for error message dialogs for all applets

com.retek.rib.gui.DBConnection: used by index.jsp to test authentication with main RIB Administration login

com.retek.rib.gui.GUIHelp: used to display help dialog for RIB Administration Tool

com.retek.rib.gui.HospitalUIApplet: main Hospital Administration class, contains all applet GUI code

com.retek.rib.gui.HospitalUIDBHelper: Hospital Administration class, contains TableModel implementation and command calls

com.retek.rib.gui.HospitalUIHelper: Hospital Administration class, contains calls to servlet

com.retek.rib.gui.HospitalUIQueries: Hospital Administration class, contains queries to retrieve and update the message data

com.retek.rib.gui.HospitalUIServlet: Hospital Administration servlet class

com.retek.rib.gui.PropertiesUI: main RIB Properties Editor class, contains all applet GUI Code

com.retek.rib.gui.PropertiesServlet: RIB Properties Editor servlet class

com.retek.rib.gui.PropsHelper: RIB Properties Editor class, contains calls to servlet

com.retek.rib.gui.ServletHelper: used for generic servlet methods

com.retek.rib.gui.StatisticsUI: main Message Statistics class, contains all applet GUI code

com.retek.rib.gui.StatsDBHelper: Message Statistics class, contains TableModel implementation

com.retek.rib.gui.StatsHelper: Message Statistics class, contains calls to servlet

com.retek.rib.gui.TableMap and **com.retek.rib.gui.TableSorter:** classes used for TableModel implementation for both applets

com.retek.rib.gui.TimingsServlet: Message Statistics servlet class

Jars and other files

js/apps.js: javascript file for RIB Administration index page

taglibs/gui.tld: tag library for RIB Administration index page

WEB-INF/lib/classes12.jar: contains Oracle Database Driver

WEB-INF/lib/retek-rib-support.jar: contains base code for Hospital Administration and Message Statistics functionality

WEB-INF/lib/retek-sbyn.jar: contains base code for Hospital Administration

WEB-INF/lib/etdRibMessages.jar: contains base code for Hospital Administration

WEB-INF/lib/stcjs.jar: contains base code for Hospital Administration

WEB-INF/web.XML: contains servlet mappings and session defaults

WEB-INF/classes/gui.servlet.properties and **gui.properties**: properties files used by RIB Administration Tool and applets

WEB-INF/classes/rib.properties: properties file used for Hospital Administration

Language-specifc files

lang/*.properties: Language-specific properties files for the java resource bundles to provide displayed text in the RIB Administration Tool in the proper language

- The language-specific properties files consist of:

  GUIHelp_*_*.properties: Resource bundle for the GUIHelp applet

  HospitalUI_*_*.properties: Resource bundle for the HospitalUIApplet

  RibPropertiesUI_*_*.properties: Resource bundle for the RibPropertiesUI applet

  Servlet_*_*.properties: Resource bundle for the ServletHelper

  StatisticsUI_*_*.properties: Resource bundle for the StatisticsUI applet

  DE/*: German HTML and JSP files

  EN/*: English HTML and JSP files

  ES/*: Spanish HTML and JSP files

  FR/*: French HTML and JSP files

  JA/*: Japanese HTML and JSP files

  KO/*: Korean HTML and JSP files

- The language-specific JSP files consist of:

  errorpage.jsp: error page for RIB Adminstration index and login pages

  index.jsp: main index page for RIB Administration

  login.jsp: main login page for RIB Administration

- The language-specific HTML files consist of:

  HospitalUIHelp.html: Help file for the HospitalUIApplet

  GUIHelp.html: Help file for the GUIHelp applet

  StatisticsHelp.html: Help file for the StatisticsUI applet

  PropertiesUIHelp.html: Help file for the RibPropertiesUI applet

Chapter 6 – J2EE Integration

This chapter will briefly review the configuration required for integrating with a J2EE application running in a WebSphere version 5 application server. Currently, the RIB integrates with two J2EE applications: MDM and RCOM.

RIBfor<*App*> Installation and Configuration

Introduction

An application-specific version of the RIB in a J2EE environment is referred to as RIBfor<*App*> (ie RIBfor*RCOM*, RIBfor*MDM*, etc). This application is packaged as an EAR file that can be deployed to the WebSphere server.

Note: The naming convention may differ based on applications released. This includes the Retek Service Layer (RSL) and application's related business logic.

The RSL differs from most applications in that it's a framework used by other applications in synchronous communications. Applications must however install and configure themselves for use on the J2EE environment. The commands, utilities, and basic process are the same regardless. Any unused subsystems will not be configured if not needed.

In general for RSL and it's related application installations, where *rib* is mentioned below it may be replaced with *rsl* or <*App*>.

After the installation of the WebSphere application server, the WebSphere instance can be configured using the RIB configuration scripts and then the RIBfor<*App*> application can be installed on the server instance.

Note: For the rest of this document, $WAS_HOME will be used to represent your WebSphere installation directory

(e.g. /files1/rcomj/WebSphere/AppServer).

The instance of WebSphere used will be referred to as $INSTANCE_HOME in this document. This is the path to the specific application server instance the RIBfor<*App*> application was be deployed on

(e.g. /files1/rcomj/WebSphere/ribdev).

If the WebSphere server installation uses only one instance, then the $WAS_HOME will be the same as the $INSTANCE_HOME.

Setup

FTP the rib-<*app*>-was-install.zip file to a temporary directory on the server where WebSphere is installed. This location will be known as $RIB_INSTALL_HOME. Unzip the contents of this file into this directory.

Note: The rest of this document will assume a UNIX install. For a Windows installation, use the equivalent .bat script in place of any .sh script.

WebSphere Configuration

The configure scripts will create and configure the listener ports, the Generic JMS provider and the JDBC DataSources, all in WebSphere, using jacl scripts.

Edit the following parameters in the rib-config.properties file to configure the instance of WebSphere for the RIBfor<*App*> application.

This file can be found in the $RIB_INSTALL_HOME/rib/config directory.

- **WAS_HOME**: The WebSphere installation directory.

  (e.g. /files1/rcomj/WebSphere/AppServer).

- **cell**: The cell name of the WebSphere instance where the RIBfor<*App*> application will be installed.

  (e.g. ribserver01)

- **node**: The node name of the WebSphere instance where the RIBfor<*App*> application will be installed.

  (e.g. server1 or ribserver01_ribdev)

- **hostName**: The host name of the computer that WebSphere is installed on.

- **serverName**: The server name within the WebSphere instance where the RIBfor<*App*> application will be installed.

  (e.g. server1)

- **was_soap_port**: The SOAP port which the above serverName utilizes.

  (e.g. 8880)

- **oracleJarPath**: The path to the oracle jar file.

  ($INSTANCE_HOME/rib/oracle/classes12.jar)

- **<*app*>Alias**: The alias name used for the RIB db connections.

  (e.g. RIBAlias)

- **ribDsName**: The datasource name used for the RIB db connections.

  (e.g. Oracle Rib Datasource)

- **oracleJndiName**: The oracle JNDI name of the datasource used for RIB db connections.

  (e.g. jdbc/OracleRibDs)

- **<*app*>JdbcUrl**: The JDBC URL to the <*App*> Hospital database.

  (e.g. jdbc:oracle:thn:@dbserver01:1521:dbname)

- **<*app*>UserId**: The Oracle username for the <*App*> Hospital database.

  (e.g. dbuser01)

- **<*app*>Password**: The Oracle password for the <*App*> Hospital database.

  (e.g. pwd1)

- **externalProviderURL**: The file URL to the sbynjndi directory.

  (e.g. file:////$INSTANCE_HOME/rib/sbynjndi)

- **externalJMSClasspath**: The classpath to the SeeBeyond JMS jars (stcjms.jar, providerutil.jar and fscontext.jar) in the sbynjndi directory. Note: the path needs to contain a "\n" line break after each jar, but the path should be entered as one continuous string.

  (e.g. $INSTANCE_HOME/rib/sbynjndi/stcjms.jar\n$INSTANCE_HOME/rib/sbynjndi/providerutil.jar\n$INSTANCE_HOME/rib/sbynjndi/fscontext.jar\n)

- **JMS_HOST, JMS_PORT**: The hostname and port where the JMS provider is running.

- **APP_EAR**: The fully qualified path to the rib-<app>.ear file to be installed.

- (e.g. /$RIB_INSTALL_DIR/rib/ear/rib-mdm.ear)

- **destinationName[n], listenerPortNameForDestinationName[0] and numberOfDestinations**: Used to create MDB listener ports and JMS Topics. A set of entries is required for each subscribing interface (to <*App*>).

```
#########################################################
#Subscribing Message Families (MDB Listener port and JMS # Topic
configuration)
#
#destinationName<number>=etVendorFromPB
#listenerPortNameForDestinationName<number>=VendorToSupp#
lierNodesPort
#
# Note: also update the "numberOfDestination" value when # you add
new topic
#########################################################
numberOfDestination=2
destinationName0=etStoresFromRMS
listenerPortNameForDestinationName0=StoresToLocNodesPort
destinationName1=etWHFromRMS
listenerPortNameForDestinationName1=WHToLocNodesPort
```

- **mdb**[**n**]: RIBforRCOM only…The message families used to create MDB listener ports and JMS Topics. An entry is required for each family used for message subscription. Each family should be entered as a separate property, each on a different line with [n] starting at 1, and replaced with [n+1] for each successive family entry.

  (e.g. mdb1=ASNOut

  mdb2=Banner

  mdb3=CustReturn ….)

- **scriptRunPath**: RIBforRCOM only…The path to the config directory.

  (e.g. $INSTANCE_HOME/rib/config)

  From the $RIB_INSTALL_HOME/rib directory, run the ribinstall.sh script. This will perform the following:

  - Creates a new directory under $WAS_HOME called rib. This will be referred to as RIB_LIB.
  - Copies the jar file from the $RIB_INSTALL_HOME/rib/oracle directory to this new RIB_LIB directory.
  - Copies the files from the $RIB_INSTALL_HOME/rib/properties directory to this new RIB_LIB directory.
  - Copies the jar file from the $RIB_INSTALL_HOME/rib/sbynjndi directory to this new RIB_LIB directory.
  - Runs the configure.sh script from the $RIB_INSTALL_HOME/rib/config directory.
  - Runs the ribadmin.sh script to create and configure the JNDI .bindings file. This file is used by WebSphere to connect to SeeBeyond as a Generic JMS Provider.
  - Runs the ribadmin.sh script to install the rib-<app>.ear file.
  - Restart the WebSphere server, which the application was installed into.

  After the script has successfully ran, verify that the Generic JMS Provider, Message Listener Ports, and the Oracle DataSource were configured by looking at the following screens in the WebSphere Admin Console:

- **Generic JMS Provider**: From the WebSphere Admin Console, click Resources -> Generic JMS Providers. You will see “SeeBeyond JMS Provider” as the available resource. The JMS Connection Factory as well as all of the JMS Destinations is defined here.
- **Message Listener Ports**: From the WebSphere Admin Console, click Servers -> Application Servers -> server1 -> Message Listener Service -> Listener Ports. You will see all of the WebSphere Listener Ports defined here.
- **Oracle DataSources**: From the WebSphere Admin Console, click Resources -> JDBC Providers. You will see “Oracle JDBC Thin Driver (XA)” as the available resource. All of the <*App*> DataSources are defined here. The “Oracle Rib Datasource” is the DataSource that the RIB utilizes.

Edit properties files

Copies of the rib.properties, log4j.properties and the hibernate.cfg.xml files should now be in the $INSTANCE_HOME/rib directory. They need to be edited before installation is complete. The log4j.properties file may need to be merged with the J2EE application's log4j.properties file (if one exists) because only one is allowed per JVM. Look at the J2EE application's installation guide for more details on the log4j.properties file.

See the following sections on editing the rib.properties and log4j.properties files for more information.

Bindings

The .bindings file connects WebSphere to the generic JMS provider, which in this case is a SeeBeyond JMS provider. The .bindings file is specific to an instance of a SeeBeyond JMS, based on the JMS host name and JMS port number. This file must be present in order for message subscription and publication to function correctly. This file is created and configured during the installation.

To create a new .bindings file:

1 Delete any existing .bindings file in the following application server directory: $INSTANCE_HOME/rib/sbynjndi.

2 In the $INSTANCE_HOME/rib directory, edit the ribadmin.sh script for the instance of WebSphere the application will be installed on. Make sure to set the SBYNJNDI_DIR, JAVA_HOME, JMS_HOST and JMS_PORT variables in order for this script to execute properly. See the following section on the ribadmin script for more details on editing this file.

3 Run the following command to generate the bindings file:

```
ribadmin.sh binding
```

4 Verify that the .bindings file was created in the $INSTANCE_HOME/rib/sbynjndi directory by running the following command:

```
ls –las .bindings
```

Installation of RIBfor<*App*> EAR

The command line utility $INSTANCE_HOME/rib ribadmin.sh can be used for installation and server administration. See the RIB command-line utility section for information on this utility.

Using the ribadmin script:

If the RIBFor<App> application had been previously installed, run:

```
ribadmin.sh uninstall
```

1 Edit the properties in the ribadmin.sh file for the instance of WebSphere the application will be installed on. To generate the deployed ear file, run:

```
ribadmin.sh install
```

2 Bounce the WebSphere server to accept the changes and start the RIBfor<*App*> application and listener ports by running:

```
ribadmin.sh bounce
```

After installation, verify that the RIBfor<*App*> application is running by looking at the Applications screen in the WebSphere Admin Console. If the application started it will show a green arrow next to the application name. If errors occurred, a red x will be shown next to the application name. Check the WebSphere log files (in the INSTANCE_HOME/logs/<server-name>/ directory) in order to troubleshoot the error.

Note: (For RIBforRCOM only) You must use a single classloader in WAS to avoid ClassCast exceptions between shared RIB objects such as Payload, RoutingInfo, etc. In the WebSphere Admin Console, select Servers -> Application Servers -> and select your server instance. The Application classloader policy variable should be set to “SINGLE”.

RIB deployment command-line utility

ribadmin

The command-line utility performs server-side functions for the RIBfor<*App*> application.

Commands

The following commands are available for this utility:

- **stop**: Stops the server for INSTANCE_HOME.
- **start**: Starts the server for INSTANCE_HOME.
- **dellogs**: Deletes *.log from the LOG_PATH.
- **bounce**: Bounces the server for INSTANCE_HOME.
- **gen**: Generates the deployed application ear file for the RIBfor<*App*> app.
- **install**: Installs the RIBfor<*App*> application on INSTANCE_HOME.
- **uninstall**: Uninstalls the RIBfor<*App*> application on INSTANCE_HOME.
- **reinstall**: Reinstalls the RIBfor<*App*> application on INSTANCE_HOME.
- **binding**: Generates a new .bindings file in the SBYNJNDI directory.

- **props**: Displays the values for all the properties in the script.

Usage

Run this utility by using the following command:

```
ribadmin.sh <command>
```

Configuration

To use this script, first edit the following variables to the specific instance of WebSphere used for running the RIBfor<*App*> application:

- **WAS_HOME**: The WebSphere installation home directory. (e.g. /u00/ibm/WebSphere/AppServer)
- **INSTANCE_HOME**: The instance of WebSphere where the application is installed. (e.g. /u00/ibm/WebSphere/ribdev)
- **NODE**: The node name of the WebSphere instance where the application is installed. (e.g. ribserver01_ribdev)
- **JAVA_HOME**: The directory where java is installed. (e.g. WAS_HOME/java)
- **APP_EAR**: The path to the RIBfor<*App*> ear (rib-<*app*>.ear). (e.g. /u00/ibm/WebSphere/ribdev/rib/ear/rib-<*app*>.ear)
- **DEP_APP_EAR**: The path to create the deployed RIBfor<*App*> ear (Deployed_rib-<*app*>.ear). (e.g. /u00/ibm/WebSphere/ribdev/rib/ear/Deployed_rib-<*app*>.ear)
- **APP_NAME**: The name of the RIBfor<*App*> application. (e.g. RIBforRCOM)
- **LOG_PATH**: The path to the WebSphere and rib log files. (e.g. /u00/ibm/WebSphere/ribdev/logs/server1)
- **SBYNJNDI_DIR**: The location of the sbynjndi directory. (e.g. /u00/ibm/WebSphere/ribdev/rib/sbynjndi)
- **JMS_HOST**: The host name of the SeeBeyond JMS Provider to use. (e.g. ribserver01)
- **JMS_PORT**: The TCP/IP port number of the SeeBeyond JMS Provider to use. (e.g. 24053)
- **APPLICATION_LIBPATH**: The paths to the jars needed to create the. bindings file. The required jars in the sbynjndi directory are: stcjms.jar, providerutil.jar, fscontext.jar and jms.jar. The required jars in the WAS_HOME/lib directory are: naming.jar, lmproxy.jar, namingclient.jar and ecutils.jar. This property will not need to be changed unless any of these jars have been moved from their default directories.

Troubleshooting

Where to look for help

- Look at the log files generated by WebSphere and/or Log4J to find information on the specific errors that are occurring within the WebSphere container. See the preceding log4j section for more information on Log4J logging. See the WebSphere Admin Console's Troubleshooting section to find the location of the WebSphere log files.

- Look at the RIB Hospital tables for more information on message-specific errors through publishing or subscribing.

Checking the WebSphere configuration

Log in to the WebSphere Admin Console for your instance of WebSphere.

- Check that the SeeBeyond JMS Provider configuration is correct

  Click the Resources menu, and then click on Generic JMS Provider in the WebSphere Admin Console. There should be a provider set up for the SeeBeyond JMS Provider. The classpath on that entry should contain the filepath on the server to three jars: stcjms.jar, providerutil.jar, and fscontext.jar. The external provider URL should contain the filepath to the sbynjndi directory on the server. Click on JMS Destinations, and check that there is an entry for each topic that needs an MDB subscription. Go back one screen, and click on JMS Connection Factories. There should be one entry for and XA TopicConnection Factory.

- Check that the Oracle JDBC Provider configuration is correct

  Click the Resources menu, and then click on JDBC Providers in the WebSphere Admin Console. There should be a JDBC Provider set up for the Oracle JDBC XA Provider.

- Check that the Message Listener configuration is correct

  Click the Server menu, and then click on Application Servers in the WebSphere Admin Console. Select your server instance from the list of servers. Select Message Listener Service from the Additional Properties section, and then select Listener Ports. This list should show all of the available listener ports and their status. They should all have a status of “Started”, which displays a green arrow image.

- Check that the Application classloader policy property is set to “SINGLE” in the WebSphere Admin Console. See the note in the RIBfor<*App*> ear installation section for more information on this property.

Note: For the RIBforMDM application, this should be set to “MULTIPLE”.

Properties Files

RIB Properties

The rib.properties file contains the RIB specific properties used by the RIB subscribing Message-Driven Beans and publishing Stateless Session Beans that are deployed on the Application Server.

In order for integration to function properly, edit the application-specific properties and the JNDI/JMS properties. See the descriptions below for more information:

- **Error Hospital entries**

This section details the entries used for retrying messages from the Error Hospital.

hospital.attempt.max – This is the maximum number of attempts to try to push this record through the RIB automatically, once this retry count is exceeded the message remains the Error Hospital DB but is no longer retried automatically.

hospital.attempt.delay – value (in seconds) used to calculate the next attempt time

hospital.attempt.delayIncrement – value (in seconds) used to calculate the next attempt time.

The next attempt time is calculated as:

hospitalAttemptDelay + (hospitalAttemptDelyIncrement * attempt count)

This is done so that the delay between each attempt is longer than the previous delay.

Example:

```
##############################################
# These are the RIB hospital properties.
hospital.attempt.max=5
hospital.attempt.delay=10
hospital.attempt.delayIncrement=10
```

- **Logging entries**

The RIB has its own logging capabilities. The RIB support Java classes contain logging logic which write to RIB log files which are written to a user specified directory.

log.default.file_path - Path where RIB and Timings log files will be written. It must end with a directory separator / or \.

log.default.verbose – [Y or N] Specifies whether or not the RIB code should be logged at a verbose level.

Example:

```
##############################################
# Default logging level verbose? [Y or N]
log.default.verbose=N
##############################################
# Path where RIB and Timings log files will be written. It must end with
# a directory separator / or \.
log.default.file_path=/files2/websph/WebSphere/AppServer/logs/server1/
```

- **JNDI/JMS Configuration**

These configurations specify the locations of the JNDI naming service and the JMS Queue.

rib.jndi.context.factory – This is the JNDI context factory to use for JNDI lookups to a RIB EJB. This is defaulted to the WebSphere Initial Context Factory.

rib.jndi.url – This is the URL to the RIB JNDI service. If the J2EE application and the RIB are on the same WebSphere instance, this URL will be the same as the app.jndi.url.

rib.jms.hostname – This is the JMS hostname (servername) of the JMS Queue.

rib.jms.port – This is the JMS port of the JMS Queue.

rib.jms.write_file – This property specifies whether or not to write each JMS XML message out to a file.

Example:

```
##############################################
# This is the hostname and port of the eGate JMS provider.
rib.jms.hostname=<servername> e.g. mspdev14.retek.int
rib.jms.port=<portname> e.g. 24053
# Write each JMS message (XML) out to a file? [Y, N, True or False]
rib.jms.write_file=False
##############################################
```

These are JNDI properties for the RIB.

rib.jndi.context.factory=com.ibm.websphere.naming.WsnInitialContextFactory

rib.jndi.url=iiop://localhost:2809

- **J2EE application-specific properties**

app.jndi.context.factory – This is the JNDI context factory to use for JNDI lookups to an application EJB. This is defaulted to the WebSphere Initial Context Factory.

app.jndi.url – This is the URL to the JNDI service. If the J2EE application and the RIB are on the same WebSphere instance, this URL will be the same as the rib.jndi.url.

app.jndi.db – This is the JNDI name for the data source.

app.jndi.cf – This is the JNDI name for the InjectorEJB used to send subscribe messages from JMS to the application.

app.jndi.injector– This is the JNDI name for the J2EE Application Injector used to inject messages into the application.

subscriber.call_injector – Set this to false to stop the RIB from calling the injector in order to test or debug the RIB code. Default is true.

Example:

##

These properties are used to interface with (J2EE). Only applicable

if RIB is not deployed in same AppServer Container.

app.jndi.context.factory=com.ibm.websphere.naming.WsnInitialContextFactory

app.jndi.url=iiop://mspdev33.retek.int:2809

app.jndi.db=jdbc/OracleRibDs

app.jndi.cf=java:comp/env/jms/Topic/XAConnectionFactory

JNDI Name for InjectorEJB

Use

java:comp/env/com/retek/component/rcominjector/impl/remote/RcomInjector

for RCOM

Use java:comp/env/ejb/InjectorTestEJB for internal RIBTest InjectorEJB (for

RIB testing)

# Use com/retek/mdm/rib/service/MDMRibMessageInjector for MDM
app.jndi.injector=com/retek/mdm/rib/service/MDMRibMessageInjector

##

#Flag to stop call to injector (to test subscribe code on RIB side only)

subscriber.call_injector=true

- **Implementation classes**

In order to promote pluggable, platform specific implementations, the RIB allows the specification of platform-specific classes for a variety of functions. These functions include the actual creation of a RibMessages XML message and the interface to an alert mechanism. The following entries are used to specify what Java classes should be used for these functions:

alertPublisherImpl -- Interface to the Alerting mechanism

Values: com.retek.rib.sbyn.alert.EgateAlertPublisher (SeeBeyond)

ribMessageImpl – Class used to create a ribMessage node within a RibMessages container.

Values: com.retek.rib.sbyn.RibMessageWrapper (SeeBeyond)

ribMessagesImpl – Class used to create a RibMessages container.

Values: com.retek.rib.sbyn.RibMessagesWrapper (SeeBeyond)

routingInfoImpl – Class used to create the Routing Information Section within a ribMessage node.

Values: com.retek.rib.sbyn.RoutingInfoWrapper (SeeBeyond)

failureImpl – Class used to create, store and copy message failure information

Values: com.retek.rib.sbyn.FailureWrapper (SeeBeyond)

ribInjectorImpl – Class used for injecting messages into an application.

Values: com.retek.rib.j2ee.RibInjectorJ2EE (RCOM, MDM)

com.retek.binding.rib.RibInjectorImpl (non-J2EE apps, default)

ribPublisherImpl – Class used to publish messages (to JMS or other).

Values: com.retek.rib.j2ee.J2eeJMSPublisher

Example:

```
##########################################
# Version of AlertPublisher, RibMessage, etc. the RIB is using.
alertPublisherImpl=com.retek.rib.alert.NullAlertPublisher
ribMessageImpl=com.retek.rib.sbyn.RibMessageWrapper
ribMessagesImpl=com.retek.rib.sbyn.RibMessagesWrapper
routingInfoImpl=com.retek.rib.sbyn.RoutingInfoWrapper
failureImpl=com.retek.rib.sbyn.FailureWrapper
ribInjectorImpl=com.retek.rib.j2ee.J2eeJMSPublisher

##########################################
# The RibInjector class that the RIB should use to inject messages into the
# application
```

Use com.retek.rib.j2ee.RibInjectorJ2EE for RCOM

Use com.retek.binding.rib.RibInjectorImpl for non-J2EE apps (default)

ribInjectorImpl=com.retek.rib.j2ee.RibInjectorJ2EE

- **Publishing configuration**

These properties are used to configure publishing from WebSphere to the JMS Queue.

<family-name>=<topic-name> - These properties map a publishing family name to the correct topic name on the JMS Queue (eg. COBORES = etCoBoResFromRCOM).

publisher.force_hospital – This property forces all messages published from WebSphere directly to the hospital table in the database instead of the JMS Queue.

dtd.url.default – The location of DTD files. Used when building message payloads.

Example:

##

#List of publishing topics for each message family

COBORES=etCoBoResFromRCOM

CODSRCPT=etCODSRcptFromRCOM

CORETURN=etCOReturnFromRCOM

CORRESPONDENCE=etCorrespondenceFromRCOM

COSALE=etCOSaleFromRCOM

CUSTORDER=etCustOrderFromRCOM

CUSTRETURN=etCustReturnFromRCOM

DSPO=etDSPOFromRCOM

GIFTREG=etGiftRegFromRCOM

INVADJUST=etInvAdjust

PAYMENTS=etPaymentsFromRCOM

PENDRETURN=etPendReturnFromRCOM

WOSTATUS=etWOStatusFromRCOM

WOINT=etWOIntFromRCOM

##

#Flag to force all published messages into the hospital instead of JMS

publisher.force_hospital=false

- **MDM-specific entries**

These properties are used to configure integration with the MDM application. The RIB's TAFR Ejb's perform lookups during processing to obtain the appropriate values in transformed messages.

rib.locale – The location (country) of the RIB. Used to identify the language of localized strings in MDM messages.

mdb.tafr.ident_type.<family> – These contain the exact identifier type values for the various messages going into MDM.

mdb.tafr.level_code.<family> - These contain the exact level code values for the various messages going into MDM.

mdb.tafr.element_code.<family> - These contain the exact element code values for the various messages going into MDM.

```
Example:
##############################################
# locale
rib.locale=en_US

##############################################
# MDM message identifier codes/types
#mdb.tafr.ident_type.<key>=<value>
mdb.tafr.ident_type.vendor=SUPPLIER_ID
mdb.tafr.level_code.vendor=SUPPLIER_LEVEL
mdb.tafr.element_code.item=ITEM
```

Example rib.properties file:

```
###############################################
# These are the RIB hospital properties.
hospital.attempt.max=5
hospital.attempt.delay=10
hospital.attempt.delayIncrement=10

###############################################
# Default logging level verbose? [Y or N]
log.default.verbose=N

###############################################
# Path where RIB and Timings log files will be written.  It must end with
# a directory separator / or \.
log.default.file_path=/u00/rcomj/ibm/WebSphere/AppServer/logs/server1/

###############################################
# Location of DTD files.  Used when building message payload.
dtd_url.default=http://mspdev14:8100/dtd/

###############################################
# These are JNDI properties for the RIB.
rib.jndi.context.factory=com.ibm.websphere.naming.WsnInitialContextFactory
rib.jndi.url=iiop://localhost:2809

###############################################
# This is the hostname and port of the eGate JMS provider.
rib.jms.hostname=mspdev05.retek.int
rib.jms.port=24053

# Write each JMS message (XML) out to a file? [Y, N, True or False]
rib.jms.write_file=False
```

```
##########################################
# Version of AlertPublisher, RibMessage, etc. the RIB is using.
alertPublisherImpl=com.retek.rib.alert.NullAlertPublisher
ribMessageImpl=com.retek.rib.sbyn.RibMessageWrapper
ribMessagesImpl=com.retek.rib.sbyn.RibMessagesWrapper
routingInfoImpl=com.retek.rib.sbyn.RoutingInfoWrapper
failureImpl=com.retek.rib.sbyn.FailureWrapper
ribPublisherImpl=com.retek.rib.j2ee.J2eeJMSPublisher

##########################################
# The RibInjector class that the RIB should use to inject messages into the application
# Use com.retek.rib.j2ee.RibInjectorJ2EE for J2EE applications
# Use com.retek.binding.rib.RibInjectorImpl for non-J2EE apps (default)
ribInjectorImpl=com.retek.rib.j2ee.RibInjectorJ2EE

##########################################
# These properties are used to interface with a J2EE application.  Only applicable
# if RIB is not deployed in same AppServer Container.
app.jndi.context.factory=com.ibm.websphere.naming.WsnInitialContextFactory
app.jndi.url=iiop://mspdev33.retek.int:2809
app.jndi.db=jdbc/OracleRibDs
app.jndi.cf=java:comp/env/jms/Topic/XAConnectionFactory
# JNDI Name for InjectorEJB
# Use java:comp/env/com/retek/component/rcominjector/impl/remote/RcomInjector for RCOM
# Use java:comp/env/ejb/InjectorTestEJB for internal RIBTest InjectorEJB (for RIB testing)
# Use com/retek/mdm/rib/service/MDMRibMessageInjector for MDM
app.jndi.injector=com/retek/mdm/rib/service/MDMRibMessageInjector

##########################################
#List of publishing topics for each message family
# (From RCOM)
COBORES=etCoBoResFromRCOM
COCOGS=etCOCogsFromRCOM
CODSRCPT=etCODSRcptFromRCOM
CORETURN=etCOReturnFromRCOM
CORRESPONDENCE=etCorrespondenceFromRCOM
COSALE=etCOSaleFromRCOM
```

```
CUSTORDER=etCustOrderFromRCOM
CUSTRETURN=etCustReturnFromRCOM
DSPO=etDSPOFromRCOM
GIFTREG=etGiftRegFromRCOM
INVADJUST=etInvAdjust
PAYMENTS=etPaymentsFromRCOM
PENDRETURN=etPendReturnFromRCOM
WOSTATUS=etWOStatusFromRCOM
WOINT=etWOIntFromRCOM
# (From MDM)
ATTRIBUTES=etAttributesFromMDM
ENUMOPTIONSET=etEnumOptionSetFromMDM
GROUPS=etGroupsFromMDM
ITEMNODES=etItemNodesFromMDM
INDUCTITEMRESPONSE=etItemsToUCCNet

##############################################
#Flag to force all published messages into the hospital instead of JMS
publisher.force_hospital=False

##############################################
#Flag to stop call to injector (to test subscribe code on RIB side only)
subscriber.call_injector=true

##############################################
# locale
rib.locale=en_US

##############################################
# MDM message identifier codes/types
#mdb.tafr.ident_type.<key>=<value>
mdb.tafr.ident_type.vendor=SUPPLIER_ID
mdb.tafr.level_code.vendor=SUPPLIER_LEVEL

mdb.tafr.ident_type.store=STORE_ID
mdb.tafr.ident_type.wh=WAREHOUSE_ID
```

mdb.tafr.level_code.store=LOCATION_LEVEL
mdb.tafr.level_code.wh=LOCATION_LEVEL

mdb.tafr.ident_type.ch=CHAIN_ID
mdb.tafr.ident_type.ar=AREA_ID
mdb.tafr.ident_type.re=REGION_ID
mdb.tafr.ident_type.di=DISTRICT_ID

mdb.tafr.level_code.ch=ORG_CHAIN_LEVEL
mdb.tafr.level_code.ar=ORG_AREA_LEVEL
mdb.tafr.level_code.re=ORG_REGION_LEVEL
mdb.tafr.level_code.di=ORG_DISTRICT_LEVEL

mdb.tafr.ident_type.company=CO_ID
mdb.tafr.ident_type.division=DIV_ID
mdb.tafr.ident_type.group=GRP_ID
mdb.tafr.ident_type.department=DEPT_ID
mdb.tafr.ident_type.class=CLASS_ID
mdb.tafr.ident_type.subclass=SUBCLASS_ID

mdb.tafr.level_code.company=MERCH_COMPANY_LEVEL
mdb.tafr.level_code.division=MERCH_DIVISION_LEVEL
mdb.tafr.level_code.group=MERCH_GROUP_LEVEL
mdb.tafr.level_code.department=MERCH_DEPARTMENT_LEVEL
mdb.tafr.level_code.class=MERCH_DEPTCLASS_LEVEL
mdb.tafr.level_code.subclass=MERCH_DEPTSUBCLASS_LEVEL

mdb.tafr.level_code.season=SEASON_LEVEL
mdb.tafr.level_code.season.phase=PHASE_LEVEL

mdb.tafr.diff_type1.color=COLOR
mdb.tafr.diff_type2.size1=SIZE1
mdb.tafr.diff_type3.size2=SIZE2
mdb.tafr.diff_type4.xyz=XYZ

mdb.tafr.ident_type.hier_level1=hier_level2
mdb.tafr.ident_type.parent_level1=parent_level2

```
##########################################
# MDM element code lookups
mdb.tafr.element_code.item=ITEM
mdb.tafr.element_code.seasons=SEASONS
mdb.tafr.element_code.size1=SIZE1
mdb.tafr.element_code.size2=SIZE2
mdb.tafr.element_code.brand=BRAND
mdb.tafr.element_code.color=COLOR
mdb.tafr.element_code.location=LOCATION
mdb.tafr.element_code.supplier=SUPPLIER
mdb.tafr.element_code.wholesalers=WHOLESALERS
mdb.tafr.element_code.distributors=DISTRIBUTORS
mdb.tafr.element_code.manufacturers=MANUFACTURERS
mdb.tafr.element_code.cost_zones=COST_ZONES
mdb.tafr.element_code.country=COUNTRY
```

Log4J Properties

Log4j is used for logging and performance measurements in the RIB Publishing and Subscribing code. The logging is defined in the log4j.properties file. In order for timings log files to be created and written to, the file paths of each entry must be changed to a valid file path on the server. Log files are typically written to the $INSTANCE_HOME/logs/<server-name>/ directory.

Note: Only one log4j.properties file can be configured for each JVM. If the J2EE application also uses log4j logging and is located on the same instance of WebSphere, the J2EE application's log4j.properties file must be merged with the RIBfor<*App*> log4j.properties file. The location of that log4j.properties file would be defined in the J2EE application's Installation Guide.

Each publishing and subscribing family has its two separate logging entries, one for timings output and one for logging RIB output. The timings logging prints out RIB timestamps to use in running statistics with the RIB Administration GUI Statistics tool (Chapter 5).

- **log4j.logger.rib.<logger-definition>.timings.<family-name>** – defines the logging level (DEFAULT, INFO, ERROR, …), and the appender name associated with the logger.
- **log4j.appender.<family-name>** - the type of appender to use. FileAppender writes the output to a specified file. ConsoleAppender writes the output to the console.
- **log4j.appender. <family-name>.File** - used with the FileAppender. Specifies the file used for output.
- **log4j.appender. <family-name>.layout** – the layout type to use for formatting output. The default is PatternLayout.

- **log4j.appender. <family-name>.layout.ConversionPattern** – the conversion pattern to use for formatting output. This is defined to fit with the RIB format used in generating statistics. If this pattern is changed, the statistics may not generate properly in the RIB Admin Tool's Message Statistics GUI.
- **log4j.appender. <family-name>.MaxFileSize** – (optional) defines the maximum file size of the log file used for output.
- **log4j.appender. <family-name>.MaxBackupIndex** - (optional) defines the number of backup files log4j will generate if the MaxFileSize is reached.

Note: The <logger-definition> depends on whether the logger is a timings logger or a normal logger. The timings logger will have a definition of either "sub.timings" or "pub.timings" depending on whether the message is a publish or a subscribe from the J2EE application. The normal logger will have a definition of either "publisher" or "subscriber" depending on whether the message is a publish or a subscribe from the J2EE application.

Each family in the RIB has a different logger attached to it to isolate timings and logging to a specific message family. Each logger is defined by its message family, and whether that family is a subscribe or publish from the J2EE application. Each message family will also by default have its own log file for timings and for logging. The timings log file is "timings_<message-family>.log" and the normal logging file is "rib_<message-family>.log".

Example of timings (subscribe) entry:

#aASNOut

log4j.logger.rib.sub.timings.asnout=DEBUG, aASNOut

log4j.appender.aASNOut=org.apache.log4j.FileAppender

log4j.appender.aASNOut.File=/u00/rcomj/ibm/WebSphere/rcomdev/logs/server1/timings_asnout.log

log4j.appender.aASNOut.layout=org.apache.log4j.PatternLayout

log4j.appender.aASNOut.layout.ConversionPattern=%d{yyyy.MM.dd HH:mm:ss.SSS}|ASNOutToRCOMJ|%t||%m%n

log4j.appender.aASNOut.MaxFileSize=100KB

log4j.appender.aASNOut.MaxBackupIndex=1

Example of logging (subscribe) entry:

log4j.logger.subscriber.asnout=DEBUG, ASNOutSubscriber

log4j.appender.ASNOutSubscriber=org.apache.log4j.RollingFileAppender

log4j.appender.ASNOutSubscriber.File=/u00/rcomj/ibm/WebSphere/rcomdev/logs/server1/rib_asnout.log

log4j.appender.ASNOutSubscriber.layout=org.apache.log4j.PatternLayout

log4j.appender.ASNOutSubscriber.layout.ConversionPattern=%d [%t] %-5p %c - %m%n

log4j.appender.ASNOutSubscriber.MaxFileSize=1024KB

log4j.appender.ASNOutSubscriber.MaxBackupIndex=1

Log4J Logging Levels

Each logger is defaulted to the ERROR logging level. This will only produce output if an error occurs. To generate timings output, set the logging level in the log4j.logger property to "INFO". To generate debug output, change the logging level to "DEBUG" in the log4j.logger.

Creating Log4J entries for each family

There is a script in the $INSTANCE_HOME/rib/properties directory called BuildLog4JProps that can be used to create the output that will configure loggers for each message family. This output can then be copied into the J2EE application's log4j.properties file.

The following parameters are used to run this script:

- -filename CONFIGURATION-FILENAME

  The fully qualified filename to the configuration file. This file should contain the name of all of the message families that should be used to create Log4J entries. Each family should be on a separate line. Also on the line for each subscriber should be a key to determine whether it is a SUBSCRIBER, PUBLISHER, or PUBANDSUB, separated from the family name by a comma. The file should look as follows:

  ASNOut,SUBSCRIBER

  Banner,SUBSCRIBER

  CoBoRes,PUBLISHER

  COCogs,PUBLISHER

  CODSRcpt,PUBLISHER

  COReturn,PUBLISHER

  Correspondence,PUBLISHER

  COSale,PUBLISHER

  CustOrder,PUBLISHER

  CustReturn,PUBANDSUB

  …

- -outputfile OUTPUT-FILENAME

  The fully qualified filename for the output file that will be generated containing the logging entries.

- -logdir WEBSPHERE-LOG-DIRECTORY

  The fully qualified filepath to the WebSphere log directory where the log4j files will be created at runtime. This path will be part of the generated output to determine the filepath of each logging entry.

- [-timings TRUE]

This Boolean value will determine whether or not to generate output for timings logging. This is an optional parameter, and will default to true (generates timings log entries for each family).

Example log4j.properties file:

```
##Timings Log for SUBSCRIBER: aASNOut

log4j.logger.rib.sub.timings.asnout=DEBUG, aASNOut

log4j.appender.aASNOut=org.apache.log4j.FileAppender

log4j.appender.aASNOut.File=/u00/rcomj/ibm/WebSphere/rcomdev/logs/server1/timings_asnout.log

log4j.appender.aASNOut.layout=org.apache.log4j.PatternLayout

log4j.appender.aASNOut.layout.ConversionPattern=%d{yyyy.MM.dd HH:mm:ss.SSS}|ASNOutToRCOMJ|%t||%m%n

#Optional

#log4j.appender.aASNOut.MaxFileSize=100KB

#Keep one backup file

#log4j.appender.aASNOut.MaxBackupIndex=1


##RIB Log for SUBSCRIBER: aASNOut

log4j.logger.subscriber.asnout=DEBUG, ASNOutSubscriber

log4j.appender.ASNOutSubscriber=org.apache.log4j.RollingFileAppender

log4j.appender.ASNOutSubscriber.File=/u00/rcomj/ibm/WebSphere/rcomdev/logs/server1/rib_asnout.log

log4j.appender.ASNOutSubscriber.layout=org.apache.log4j.PatternLayout

log4j.appender.ASNOutSubscriber.layout.ConversionPattern=%d [%t] %-5p %c - %m%n

log4j.appender.ASNOutSubscriber.MaxFileSize=1024KB

log4j.appender.ASNOutSubscriber.MaxBackupIndex=1


##Timings Log for PUBLISHER: aCOCogs

log4j.logger.rib.pub.timings.cocogs=DEBUG, aCOCogs

log4j.appender.aCOCogs=org.apache.log4j.FileAppender

log4j.appender.aCOCogs.File=/u00/rcomj/ibm/WebSphere/rcomdev/logs/server1/timings_cocogs.log

log4j.appender.aCOCogs.layout=org.apache.log4j.PatternLayout

log4j.appender.aCOCogs.layout.ConversionPattern=%d{yyyy.MM.dd HH:mm:ss.SSS}|COCogsFromRCOM|%t||%m%n

#Optional

#log4j.appender.aCOCogs.MaxFileSize=100KB

#Keep one backup file
```

```
#log4j.appender.aCOCogs.MaxBackupIndex=1);


##RIB Log for PUBLISHER: aCOCogs
log4j.logger.publisher.cocogs=DEBUG, COCogsPublisher
log4j.appender.COCogsPublisher=org.apache.log4j.RollingFileAppender
log4j.appender.COCogsPublisher.File=/u00/rcomj/ibm/WebSphere/rcomdev
/logs/server1/rib_cocogs.log
log4j.appender.COCogsPublisher.layout=org.apache.log4j.PatternLayout
log4j.appender.COCogsPublisher.layout.ConversionPattern=%d [%t] %-5p
%c - %m%n
log4j.appender.COCogsPublisher.MaxFileSize=1024KB
log4j.appender.COCogsPublisher.MaxBackupIndex=1
```

Hibernate Properties

Hibernate is used for Object Relational Mapping and Database interaction within the RIB's hospital code. It is defined in the hibernate.cfg.xml configuration file which is located in the $WAS_HOME/rib directory. The following properties in this file will need to be edited before the RIBfor<*App*> application will be able to run:

"hibernate.connection.datasource" – the JNDI name of the DataSource to use.

"hibernate.connection.username" – the username to connect to that database.

"hibernate.connectionpassword" – the password to connect to that database.

Chapter 7 – ISO Integration

This chapter will review the steps required for integrating RIB 10.3.x with SIM.

Note: RIB 10.3 and RIB 10.3.3 need to be installed before installing the RIB 10.3.4 release.

Manual Steps

Log into the SIM application server as the user who performed the SIM install.

1 Create a temporary install directory for integrating RIB with SIM. This temporary directory will be known as <ISO_TMPINSTALL_DIR> for the remainder of this section:

- > mkdir INSTALL

2 Copy the file ribiso_<version>.tar and ribcomm_<version>.tar to <ISO_TMPINSTALL_DIR>.

3 Once you have copied the files, extract the contents.

- > tar xvf 'ribiso_<version>.tar'
- > tar xvf 'ribcomm_<version>.tar'

4 Ensure all files located in <ISO_TMPINSTALL_DIR> have the correct permissions by issuing the following command in <ISO_TMPINSTALL_DIR>:

- > chmod –R 755 *

5 Edit the file iso_profile. This file is located at <ISO_TMPINSTALL_DIR> /RIB<version>. Make sure the settings for the following variables are correct for your environment.

- ISO_TMPINSTALL_DIR - Same directory created in step 1 above
- ISO_INSTALL_DIR - The directory where ISO was installed.

Note: The install script expects ISO_TMPINSTALL_DIR to include the temporary install directory created in step 1 above (eg: /files0/jadmin/INSTALL) and ISO_INSTALL_DIR to include the full directory path leading up to server<platform> (eg: /files0/jadmin/iso)

6 In <ISO_TMPINSTALL_DIR>/RIB<version> run iso_profile:

- > . ./iso_profile
- 7 In <ISO_TMPINSTALL_DIR>/RIB<version>, run installisoconfig.
- > ./installisoconfig

8 Steps for the RIB Error Hospital:

- Check the SIM schema for the existence of the following RIB error hospital tables: RIB_MESSAGE, RIB_MESSAGE_FAILURE, and RIB_MESSAGE_ROUTING_INFO.
- If the RIB error hospital tables do not exist, ensure that the following tablespaces exist: LOB_DATA, RETEK_DATA, and INDEX_DATA

Note: Do not run the following sql scripts if the rib error hospital tables above already exist. These scripts will drop and re-create the tables, thus deleting all data in the tables.

- run rib_ddl.sql as the SIM schema to create the required RIB tables. This script is located at <ISO_TMPINSTALL_DIR>/RIBforISO<version>/XML_Utilities.
- run rib_message_seq.sql as the SIM schema to create the sequence number for the table RIB_MESSAGE. This script is located at: <ISO_TMPINSTALL_DIR>/RIBforISO<version>/XML_Utilities.

Note: Since the database scripts drop and re-create the tables, ignore any errors resulting from trying to drop objects that don't exist. These errors are expected.

9 Edit the file ribmessaging.cfg. This file is located at <ISO_INSTALL_DIR>/server<platform>/retek/sim/files/prod/config. Make sure the setting for the following variable is correct for your environment. This value must match exactly values listed in the SeeBeyond configuration file cpJMS.cfg of HostName+Port Number (I.E cpJMS.cfg-HostName: mspdev05.retek.int, cpJMS.cfg-Port Number: 37053, ribmessaging.cfg-BROKER: mspdev05.retek.int:37053).

- BROKER=<EGATE_SERVER_NAME>:<EGATE_SERVER_PORT>
- (eg: BROKER=10.1.1.164:25053)

10 Edit the file messaging.cfg. This file is located at <ISO_INSTALL_DIR>/server<platform>/retek/sim/files/prod/config. Make sure the setting for the following variable is correct for your environment. This value must match exactly values listed in the SeeBeyond configuration file cpJMS.cfg of HostName+Port Number (I.E cpJMS.cfg-HostName: mspdev05.retek.int, cpJMS.cfg-Port Number: 37053, ribmessaging.cfg-BROKER: mspdev05.retek.int:37053).

- BROKER=<EGATE_SERVER_NAME>:<EGATE_SERVER_PORT>
- (eg: BROKER=10.1.1.164:25053)

11 Edit the file riblog4j.properties. This file is located at <ISO_INSTALL_DIR>/server<platform>/retek/sim/files/prod/config. Replace all occurrences of <ISO_INSTALL_DIR> in this file with the full directory path leading up to the /server<platform> directory.

- Ex: log4j.appender.ItemsSubscriber.File=<ISO_INSTALL_DIR >/serverUnix/retek/sim/log/itemsmessagingcomponent.log
- (eg: log4j.appender.ItemsSubscriber.File=/files0/jadmin/sim10.1/
- serverUnix/retek/sim/log/itemsmessagingcomponent.log)

12 Edit the file rib.properties. This file is located at <ISO_INSTALL_DIR>/server<platform>/retek/sim/files/prod/config. Replace the occurrence of <ISO_INSTALL_DIR> in this file with the full directory path leading up to the /server<platform> directory.

- Ex: log.default.file_path=<ISO_INSTALL_DIR>/serverUnix
- /retek/sim/log/
- (eg: log.default.file_path =/files0/jadmin/sim10.1/serverUnix
- /retek/sim/log/)

13 After successful completion to installisoconfig, the temporary directory ISO_TMPINSTALL_DIR may be removed

ISO Reference

The following sections are noted here for reference.

rib.properties file

In the rib-redsky.jar file, you will find a file called, “rib.properties”. This file contains the RIB specific properties used by the RIB subscribing messaging components under ISO. These messaging components will be deployed in an ISO container, one for each subscribing API. Some of the important sections of this file are illustrated below:

```
##########################################
# These are the RIB hospital properties.
hospital.attempt.max=5
hospital.attempt.delay=10
hospital.attempt.delayIncrement=10
##########################################
# These are properties that are also used in the process
# of putting a message in the hospital.  The difference here
# is that these properties control some of the concrete classes
# that are used in this process.
failureImpl=com.retek.rib.sbyn.collab.FailureWrapper
routingInfoImpl=com.retek.rib.sbyn.collab.RoutingInfoWrapper
routingInfoDetailImpl=com.retek.rib.sbyn.collab.RoutingInfoDetailWra
pper
ribMessageImpl=com.retek.rib.sbyn.collab.RibMessageWrapper
ribMessagesImpl=com.retek.rib.sbyn.collab.RibMessagesWrapper
```

Example of a messaging component configuration file

We will use the ASNIn API for an example of a configuration file. These configuration files can be found in the <ISO_INSTALL_DIR>/server<platform>/retek/sim/files/prod/config directory.

The topic name from which to accept messages.

TOPIC_NAME=etASNOutISO

Makes the subscription durable (see JMS specification).

DURABLE_SUBSCRIBER=true

The type of component – Publisher or Subscriber.

JMS_COMPONENT_TYPE=Subscriber

The messaging group to which to listen.

MESSAGING_GROUP=

Module name to be used for the Rib's context object.

MODULE_NAME=RibMessagingComponent

Sub-module name to be used for the Rib's context object.

This will be the same as the message family name.

SUB_MODULE_NAME=ASNOut

If TRUE, only a single thread will be used to call the

processMessages(ArrayList) method. If FALSE, multiple

threads may call this method. Default is TRUE.

SINGLE_THREADED=TRUE

```
# Disconnect from the server for the specified number of minutes
between checks

# for messages. Note that this does not make sense for a non-durable
topic-based

# subscription. Therefore, a value of 0 will cause the component to
stay

# connected. The default is 0.

CONNECTION_INTERVAL=0
```

The config file to use for setting up messaging. The default is

"messaging.cfg".

MESSAGING_CONFIG=ribmessaging.cfg

Remote Object Lookup Name

REMOTE_NAME=ASNOutMessagingComponent

Collect Performance Statistics

PERFORMANCE=true

Chelsea Logging Properties

LOGGING_IMPL=com.chelseasystems.cr.logging.LoggingFileServices

LOGGING_FILE_NAME=../log/asnoutmessagingcomponent.log

LOGGING_LEVEL=4

LOGGING_PAUSE=5000

LOGGING_SYSTEM_OUT=true

LOGGING_SYSTEM_ERR=true

LOG4J_CONFIG=riblog4j.properties

ribmessaging.cfg file

This configuration file is used to configure the JMS messaging parameters across all of the RIB's APIs, publishing and subscribing.

The client impl is the class that implements the MessagingServices contract.

CLIENT_IMPL=com.retek.rib.redsky.RibSeeBeyondJmsServices

The time-to-live for messages sent to the server.

MESSAGE_LIFETIME=1800000

This property has to do with who controls the transaction. When set to "true",

the Chelsea framework is in control of a global transaction. When "false",

commits or rollbacks must be done explicitly in the application code. For the

#RIB, this should always be "true".

USE_SESSION_TRANSACTION=true

POS device-specific class uses Store and Register in global repository.

Currently, the RIB does not use grouping, so this entry is irrelevant.

GROUPING_UTIL=com.chelseasystems.cr.messaging.grouping.POSMessagingGroupService

Default group name. Again, the RIB does not currently use grouping, so this

entry is irrelevant.

DEFAULT_GROUP=RTK

The Broker is the JMS server address and port.

For the RIB, this will be the server name and port of the SeeBeyond JMS

queue.

BROKER=mspdev05.retek.int:24053

Username and password are set via administration of the JMS server. For the

RIB, SeeBeyond does not make use of username and password, so these should

be blank.

USERNAME=

PASSWORD=

Number of times to try getting a connection to JMS server

MAX_CONNECTION_TRIES=2

Number of seconds to pause between connection attempts

PAUSE_BETWEEN_TRIES=2