Oracle® Retail Integration Bus

Support Tools Guide

Release 14.1

E57323-01

December 2014

ORACLE®

Oracle Retail Integration Bus Support Tools Guide, Release 14.1

E57323-01

Primary Author: Sanal Parameshwaran

Contributing Author: Anshuman Accanoor

Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

4 RIB Diagnostics and Monitoring Tool (RDMT)

5 Deployment Env Info File Editor

Send Us Your Comments

Oracle Retail Integration Bus Support Tools Guide, Release 14.1

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at http://www.oracle.com.

Preface

The purpose of this document is to introduce you to the “RIB Support Tools “and familiarize you with the installation, configuration and operational steps for listed tools. The document explains the following tools:

- RIB API Simulators
- FileIO
- RIB Diagnostic and Monitoring Tool (RDMT)
- Deployment Env Info File Editor

Audience

This guide is for:

- Systems administration and operations personnel
- Systems analysts
- Integrators and implementation staff

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Retail Integration Bus documentation set:

- *Oracle Retail Integration Bus Implementation Guide*
- *Oracle Retail Integration Bus Installation Guide*
- *Oracle Retail Integration Bus Release Notes*

- *Oracle Retail Integration Bus Hospital Administration Guide*
- *Oracle Retail Integration Bus Security Guide*
- *Oracle Retail Integration Bus Operations Guide*
- *Oracle Retail Integration Bus Java Messaging Service (JMS) Console Guide*
- *Oracle Retail Enterprise Integration Guide*
- *Oracle Retail Integration Bus Integration Gateway Services Guide*
- *Oracle Retail Functional Artifacts Guide*
- *Oracle Retail Functional Artifact Generator Guide*
- *Oracle Retail Service-Oriented Architecture Enabler Tool Guide*
- *Oracle Retail Integration Bus Data Model*
- *Oracle Retail Payload Mapper Guide*

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

`https://support.oracle.com`

When contacting Customer Support, please provide the following:

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

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 14.1) or a later patch release (for example, 14.1.1). If you are installing the base release and additional patch releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times not be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you

have the most recent version of the Oracle Retail documentation set. Oracle Retail documentation is available on the Oracle Technology Network at the following URL:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-02 is an updated version of a document with part number E123456-01.

If a more recent version of a document is available, that version supersedes all previous versions.

Oracle Retail Documentation on the Oracle Technology Network

Documentation is packaged with each Oracle Retail product release. Oracle Retail product documentation is also available on the following Web site:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

(Data Model documents are not available through Oracle Technology Network. These documents are packaged with released code, or you can obtain them through My Oracle Support.)

Documentation should be available on this Web site within a month after a product release.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1
Overview

The RIB is difficult to test as a stand-alone sub-system. It is part infrastructure and part application, and needs to have the integrating application end-points for even a simple installation. To aid in the initial installation, simulation, diagnosing and monitoring of the RIB infrastructure, various support tools have been provided, namely Deployment-env-info file editor, RIB API Simulators, RDMT, and FileIO.

RIB API Simulators aids in testing the integrity of the RIB infrastructure by stubbing out integrating retail applications. The JAVAEE -api-stub, simulates JAVAEE driven retail applications like SIM, RPM, and AIP. PL/SQL-api-stub, simulates PL/SQL driven retail applications like RMS, RFM, and RWMS.

FileIO is a JAVAEE application that allows the user to publish and subscribe messages by using the file system. The user can publish by placing the message to be published in the defined location of the file system. Once the message is subscribed it is written to the file system. This approach of publishing can be majorly used to publish bulk messages.

The RIB Diagnostic and Monitoring Tool Kit (RDMT) is a collection of command line tools, written in UNIX shell script along with supporting Java classes packaged in jar files. The tool supports diagnostics and monitoring of RIB, at various phases namely installation, operation, production, test, support and AQ JMS support

The Oracle Retail Integration Bus (RIB) Deployment Configuration File Editor is an application for configuring the rib-deployment-env-info.xml file. The editor simplifies the alteration to the XML file by hiding the raw text form of XML. It provides an interface for adding, removing, and rearranging XML elements. It also provides an interface for editing the contents of elements.

2

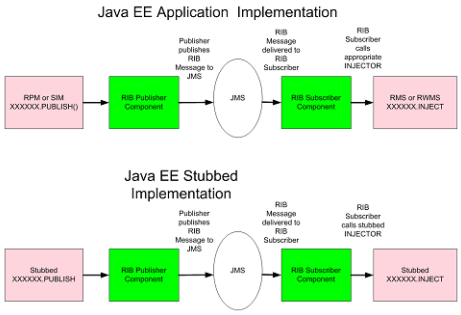
RIB API Simulators

This chapter discusses the RIB API simulators.

Java API Simulator

The javaee-api-stubs is an API simulator designed to act in the same manner as when the RIB is connected to the actual application, but at the same time, have means to process specific status and other parameters from a stubbed application. This set of tools is designed to emulate those applications exposing Java EE APIs to the RIB: SIM, RPM, AIP, and OMS.

Architecture and Design



Installation

Table 2–1 Prerequisite Tasks

Task	Notes
Install WebLogic server 10.3.6 and create an instance.	This instance is used to deploy the javaee-api-stubs.
Install the Oracle12c database.	The installation creates the data source that refers to this database.
Select a location for the javaee-api-stubs to reside.	Recommended location is in the rib-app-builder/rib-home tree structure: rib-app-builder/rib-home/tools-home/ javaee-api-stubs

Table 2–1 (Cont.) Prerequisite Tasks

Task	Notes
Get the latest version of the javaee-api-stubs (JavaEeApiStubs14.1.0ForAll14.x.xApps_eng_ga.tar) from RIB wiki.	The javaee-api-stub is packaged as a stand-alone tar.
Extract the tar file to locate the installable.	javaee-api-stub-<version>.ear resides in the extracted folder.
Create a database user that owns the javaee-api-stubs objects.	The user requires no special permissions. CREATE USER <*javaee stub user*> PROFILE DEFAULT IDENTIFIED BY <*javaee stub password*> DEFAULT TABLESPACE USERS TEMPORARY TABLESPACE TEMP; GRANT CONNECT TO <*javaee stub user*>; GRANT RESOURCE TO <*javaee stub user*>;

Table 2–2 Installation

Task	Notes
Determine the WebLogic instance to which to deploy the javaee-api-stubs-<version>.ear.	It is recommended but not required that an instance separate from the rib-<app> instance is used.
Using the WebLogic console, select the WebLogic instance and then deploy javaee-api-stubs-<version>.ear.	See WebLogic deployment documentation for more details on how to deploy a Java EE application.

Table 2–2 (Cont.) Installation

Task	Notes
Using the WebLogic console, configure the database resources for the javaee-api-stubs JDBC resources. ■ Log in to the WebLogic admin console ■ Navigate to the Data Sources screen using Services > JDBC > Data Sources menu. ■ Click New. Enter the following values in the respective fields. Name: javaee-api-stubs-non-xa-managed-datasource JNDI Name: jdbc/OracleRibDsNonXA Database Type: Oracle Database Driver: Oracle's Driver(Thin) ■ Click Next. Uncheck Supports Global Transactions. ■ Define connection properties for the database user in question. ■ Verify the configuration by clicking Test Configuration. ■ Do not proceed if the test fails. Ensure that the configuration is accurate. ■ Select target as the server that would host javaee-stubby (for example, javaee-stubby-instance). Click Finish.	See WebLogic documentation for details.

Table 2–2 (Cont.) Installation

Task	Notes
Create one more data source named javaee-api-stubs-xa-managed-datasource. Navigate to the Data Sources screen using Services > JDBC > Data Sources menu. ■ Click New. Enter the following values in the respective fields. Name: javaee-api-stubs--xa-managed-datasource JNDI Name: jdbc/OracleRibDs Database Type: Oracle Database Driver: Oracle's Driver(Thin XA) ■ Define connection properties for the database user in question. ■ Verify the configuration by clicking Test Configuration. ■ Do not proceed if the test fails. Ensure that the configuration is accurate. ■ Select target as the server that would host javaee-stubby (for example, javaee-stubby-instance). Click Finish. ■ Verify that both data sources are listed on Services > JDBC > Data Sources screen.	
Install Hospital tables	See the Oracle Retail Integration Bus Installation Guide.

Table 2–3 Configuration of the rib-<app> to use Injection Stubs

Task	Notes
Decide which rib-<app> to configure for.	The stubbed implementation has been written to insert the payload to a database once inject has been called. Injectors.xml has been configured to include all the RPM, SIM subscribing families.

Table 2–3 (Cont.) Configuration of the rib-<app> to use Injection Stubs

Task	Notes
Using RIB App Builder or the RIB Installer configure and deploy the rib-app using the jndi information of the javaee-api-stubs in place of the app.	<app id="sim" type="javaee-app"> <jndi> <url>t3://ribhost.example.com:18022/javaee-api-stubs</url> <factory>weblogic.jndi.WLInitialContextFactory</factory> <user-alias>sim_jndi_user-name-alias</user-alias> </jndi> </app>

Operation

javaee-api-stubs application can be used to publish a message to the RIB integration environment.

Populate the following fields:

Table 2–4 RMI Properties

User Name	Mandatory Field	The user name used to connect to WebLogic server.
Password	Mandatory Field	The password used to connect to WebLogic server.
URL	Mandatory Field	t3://host:port/applicationName

Table 2–5 RIB Message Data

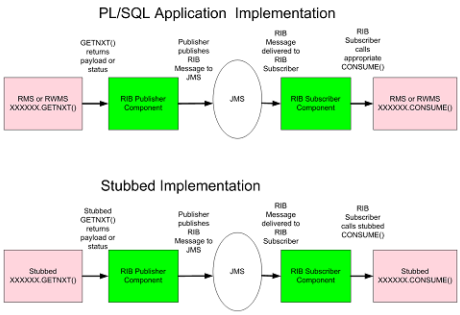
Message Family	Mandatory Field	The family of the message that is to be published.
Message Type	Mandatory Field	The message type of the message that is to be published.
Business Object ID	Optional	Numeric ID.
Message Data	Mandatory Field	Payload data.

The RIB app publishing adapter submits the message to be published.

PL/SQL API Simulator

The plsql-api-stubs is an API simulator designed to acts in the same manner as when the RIB is connected to the actual application, but at the same time, have means to process specific status and other parameters from a "stubbed" application. This set of tools is designed to emulate those applications exposing PL/SQL APIs to the RIB: RMS, RFM, and RWMS.

Architecture and Design



The tool set contains three main subsystems:

- A common set of PL/SQL packages, stored procedures and database tables. These are used by the other subsystems.
- A thin API-specific set of packages and stored procedures that the RIB directly interfaces with. These interfaces map calls to the common subsystem to output parameters or statuses.
- The Stub Admin and Setup Application. A set of simple application functions and a character based menu that allows installation and set up of specific behaviors for a specific API.

The Common Subsystem

The purpose of the common subsystem is to provide a standard means of implementing specific behavior by an API. The stubbed APIs simulate a real application by using the common subsystem which will be loaded during the installation through JDBC calls to the database. It is comprised of a group of tables, sequences and other database objects created for each stubbed API.

There is a set of tables and sequences created for each GETNXT procedure. These tables are generated with the OUT and IN/OUT parameters of the GETNXT procedure as the fields. The user is prompted to enter data into these tables when he is trying to test for a particular API.

For example:

If there is a GETNXT procedure in a package called RMSMFM_ORDER, then the common subsystem for this procedure would be a table RMSMFM_ORDER_GE_TBL and sequence called RMSMFM_ORDER_GE_SEQ created in the database.

For each PUB_RETRY Procedure in the API, a set of tables and sequences are created the same as GETNXT, except that the names of tables and sequences have PU instead of GE

For a CONSUME API, there is a table called RIB_CONSUME created with the O_STATUS_CODE, O_ERROR_MESSAGE and EXCEPTION_TO_THROW as the fields. If the user needs the CONSUME to throw a specific type of exception, then the exception

can be uploaded into the RIB_CONSUME table, so that when the consume procedure is executed it throws the specified exception type.

The Thin API layer

The API subsystem consists of packages and stored procedures that have the exact same signature as those found within the real application. This layer queries the appropriate common subsystem tables, sequences and other database objects to get the appropriate out parameters. These are then mapped to the API specific parameters of the stubbed application API.

The implementation of the stubbed API is written as Java classes and loaded into the database during installation. The PL/SQL stubbed APIs are implemented in a way that these API internally call the Java functions present in the classes and the PL/SQL OUT parameters are then mapped with the Java return types.

So when the RIB calls the GETNXT stubbed API as it normally calls the GETNXT API of a real application, the stubbed API internally calls the Java class which uses the common subsystem tables to get messages as a CLOB, it then converts the CLOB to an Oracle Object and then maps it with the PL/SQL OUT parameters and returns.

The Stub Admin and Setup Functions

These are a set of simple application functions written in Java and wrapped by shell scripts and a character based menu that allow installation and set up of specific behaviors for a specific API.

Table 2–6 Shell Scripts

Shell Script	Description
stubbymenu.sh	Simple character based menu that calls the wrapper scripts.
install.sh	Wrapper script that calls the Java classes to install the RIB Objects and stubby Java classes dynamically created from the metadata into the database (see stubby.properties).
configure_api.sh	Wrapper script that calls the Java classes to set up the behavior and messages of a given CONSUME, PUB_RETRY, or GETNXT API.
read_metadata.sh	Wrapper script to call a Java utility that will read a PL/SQL application (RMS, RFM, RWMS) schema and create a metadata file as input to create the stubbed APIs.

Configuration Files

The following are /conf directory files:

Table 2–7 Configuration Files

Configuration File	Description
stubby.properties	Primary configuration file. Contains database url info and the metadata scripts to load.
commons-logging.properties	Apache logging conf
simplelog.properties	Apache logging conf
SqlToJavaMapper.java	generated from the storedprocedureметadatxml specified in the Stubby.properties file. Note: Do not edit.

Table 2–7 (Cont.) Configuration Files

Configuration File	Description
StoredProcedureMetaData_RWMS.xml	Note: Do not edit.
StoredProcedureMetaData_RMS.xml	Note: Do not edit.

Installation

Complete the following steps:

Table 2–8 Prerequisite Tasks

Task	Notes
Select a location for the plsql-api-stubs to reside.	Recommended location is in the rib-app-builder/rib-home tree structure: rib-app-builder/rib-home/tools-home
Get the latest version of the plsql-api-stubs.	The plsql-api-stubs is packaged as a stand-alone tar.
Get the latest version of the rib-public-payload-database-object-types.	rib-public-payload-database-object-types-<version>.jar is packaged with the RibFuncArtifacts and should be extracted from there. If this installation is in rib-home, the objects is located in the rib-home/download-home/rib-func-artifacts.
Create a database user that will own the plsql-api-stubs schema and the objects.	The user requires no special permissions. CREATE USER <*plsql stub user*> PROFILE "DEFAULT" IDENTIFIED BY <*plsql stub password*> DEFAULT TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP"; GRANT "CONNECT" TO <*plsql stub user*>; GRANT "RESOURCE" TO <*plsql stub user*>;
This version requires a path to jdk1.7 for compiling Java stored procedures.	Be prepared to specify the path when prompted.

Table 2–9 Installation

Task	Notes
Extract the tar file. cd rib-app-builder/rib-home/tools-home tar xvf PlsqlApiStubs14.1.0ForAll14.x.xApps_eng_ga.tar Place the database objects file in the scripts subdirectory	This will create the file folders and place the executables and config files. In rib-home/tools-home there is a directory already. It is a placeholder and this will over write it.

Table 2–9 (Cont.) Installation

Task	Notes
Extract the rib-public-payload-database-object-types-<version>.jar into the scripts directory. unzip rib-public-payload-database-object-types-<version>.jar	

Edit /conf/stubby.properties to point to the database hostname, port, sid, and alias (see prerequisites).
vi stubby.properties
	# Database details
	hostname=linux1
	port=1521
	sid=ora10g
	username=<*plsql stub user*>
	password=<*plsql stub password*>

Base Script File names	This is where the selection of either RMS or RWMS objects is made. There can be only one per installation.
Execute the installation using install.sh in the stubby base directory cd rib-app-builder/rib-home/tools-home/ plsql-api-stubs ./stubbymenu.sh	The installation performs these actions: Runs a cleanup that will remove any existing RIB related tables, sequences, packages and types installed in the configured user schema. Runs all the script files in the scripts sub-directory. Runs a drop Java utility to remove any existing classes in the configured user schema. Note: The warnings generated by the drop Java utility can be ignored. Runs the load Java utility to load Java classes as objects in the configured user schema. All the RMS or RWMS packages are created in the configured user schema.
Select option [1] Database Credentials SetUp from the Stubby menu. Enter database user and password at the prompts	
Select option [2] Installation	
Install Hospital tables	See the *Oracle Retail Integration Bus Installation Guide*.
Enter the complete path for jdk1.7:	This version of stubby and the RDBMS require jdk1.7 for compiling Java stored procedures.

Operation

The next step in using the tool set is to configure the desired behavior of the APIs under test. Use of the tool requires that the user understand the APIs involved at enough detail to understand and answer several prompts during the configuration process. See the Oracle Retail Integration Bus Integration Guide and the operations guides for the RMS and RWMS applications.

Table 2–10 Prerequisites

Task	Notes
Create a sub-directory for the test messages to configure the API to use. These can be any location on the same host where the tool user has permissions to read.	The RIB ships with sample xml files for each message family. These are packaged with RDMT and are located under the testmsg subdirectory in the rdmt directory. retail-public-payload-xml-samples-<version>.jar. These should be used as a basis for testing and modified to suit the test cases.
Understand and know which API and its type to configure. For example: API Type: GETNXT API Package name: RMSMFM_ITEMS Message Type: ITEMCRE	API Types supported: GETNXT CONSUME PUB_RETRY

Execute the configure_api.sh script or select the menu item (Configure API) and respond to the prompts.

Prompts during configuration of a GETNXT and PUB_RETRY.

Table 2–11 Configuration Prompts

Prompt	Notes
Status Code the GETNXT API should return: S for Success, H for hospital, N for no message, and E for exception	Case sensitive
Enter Error Message to be returned (to be entered only for H or E status codes).	
Enter data for O_MESSAGE	The complete file path of the message to upload.
Enter Business Object ID to be returned.	Optional
Enter the Routing Information, if applicable for your message type.	
Enter the Thread Value for the message.	
Enter the number of times the message must be replicated.	

Prompts during configuration of a CONSUME.

Table 2–12 CONSUME Configuration Prompts

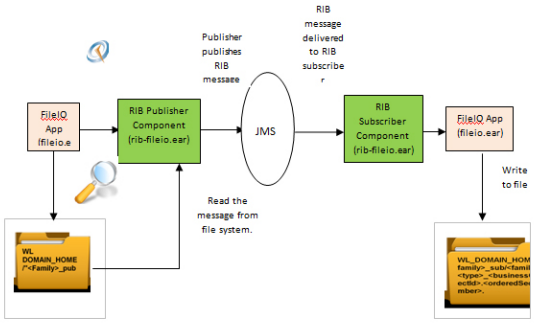
Prompt	Notes
Enter Status Code the Consume should return [S-Success]/[E-Error]	
Enter the Exception to be Thrown eg:nullpointerexception:	The Exception_To_Throw and Error Message are only prompted if the status code is E.
Enter the Exception Message to be Thrown.	
Enter Message Type the Consume should return [CRE,MOD,DEl] eg:ITEMCRE:	

3

FileIO

Until now, the RIB lacked the feature of publishing/subscribing the messages from/to the file system. The FileIO application facilitates RIB with the functionality to publish/subscribe messages from/to files.

FileIO is a Java enterprise application developed using spring framework. The fileio-<version>.ear works in conjunction with rib-fileio.ear in integrated RIB environment to publish and consume the messages. All retry and error handling capability of RIB is automatically lent to this fileio-<version>.ear application.



The FileIO application along with rib-fileio can be used as a publisher as well as a consumer component. FileIO as a publisher scans the Weblogic server DOMAIN_HOME directory for folder names of the format "<Family>_pub". If the directory is found, it adds a scheduler for that folder and starts polling for files inside it, with the filename format <family>_<type>_<businessObjectId>.<orderedSeqNumber >. If the file matching the criteria is found, message is published to the RIB topic through rib-fileio publisher component.

The subscribing component of the rib-fileio application, subscribes to the message on the RIB topics by calling the injector in rib-fileio. The injector unmarshalls the payload and writes the content to a file in the directory WL_DOMAIN_HOME/<family>_sub. The file name will be of the format <family>_<type>_<businessObjectId>.< orderedSeqNumber>.

Installation and Configuration

Take the following steps to install and configure the application:

1. Build a new rib-fileio.ear using rib-app-builder tool. The FileIO application is a javaee application.

> **Note:** Refer to the *Oracle Retail Integration Bus Implementation Guide* Release 14.1 for more detailed steps on adding a new rib-<app>.

2. Compile and deploy rib-fileio.ear using standard rib-app-builder deploy tool.
   - Note down the jndi.providerUrl, for example: t3://localhost:7001
   - Note down the jndi.securityPrincipal, for example: <*weblogic username*>
   - Note down jndi.securityCredentials, for example: <*weblogic password*>

   > **Note:** Refer to the *Oracle Retail Integration Bus Installation Guide* Release 14.1, chapter 4 Run the RIB Application Installer for compilation and deployment steps.

3. Download RibFileIo14.1.0ForAll14.x.xApps_eng_ga.zip to a working folder, example FILEIO_HOME.
4. Extract the zip file. The FILEIO_HOME /fileio directory is created.
5. Traverse to the directory FILEIO_HOME/fileio.
6. Run the command, config-and-secure-fileio-<version>.jar

   > **Note:** This step is required only if FileIO is to be used for publishing. To use FileIO as a consuming application skip to step 8.

7. Provide the following details:
   - JNDI URL of the rib-<app> to publish the message: The JNDI URL of the WLS server where rib-fileio is hosted.
   - User Name: The weblogic user name credential to connect to the above provided URL
   - Password: The password credential to connect to the above provided URL
8. Deploy the fileio-<version>.ear using WebLogic console.
   - The WebLogic domain must not be an OSB domain as there are spring library conflicts when deployed in OSB domain.
   - The recommended option is to create a new WebLogic domain called fileio_ domain and deploy the application there.
   - If the tool is used for publishing, edit the weblogic.policy fie located at "${WL_ HOME}/wlserver_10.3/server/lib" and give the following permissions to read the wallet file.

   Sample permission:

   grant codeBase "file:/home/xyz/install/wls1036/user_projects/domains/base_ domain/-" {

   permission java.security.AllPermission;

   permission oracle.security.jps.service.credstore.CredentialAccessPermission "credstoressp.credstore", "read,write,update,delete";

permission oracle.security.jps.service.credstore.CredentialAccessPermission "credstoressp.credstore.*", "read,write,update,delete";

};

9. Verify that the application is in running/active state using WLS console.
10. Bounce the server.

> **Note:** To see the system output logs in the console, set the following two variables in the console before starting WLS using the startWebLogic.sh script.

export SERVER_NAME=<your server name>

export WLS_REDIRECT_LOG =

servers/${SERVER_NAME}/logs/${SERVER_NAME}.sysout

General Usage

The following sections provide guidelines on using the application.

Publishing Using FileIO

Take the following steps to publish the payload using FileIO application:

1. Configure and install using rib-app-builder tool, the application which uses FileIO for publishing the message.

> **Note:** Refer to the *Oracle Retail Integration Bus Implementation Guide* Release 14.1, Chapter 11, RIB Customization/Extension.

2. Login to rib-admin-gui of the publishing application and make sure that the publishing adapter is up and running.
3. On the WebLogic server where fileio.ear is installed, under DOMAIN_HOME, create a directory with the name that has a format <message_family>_pub. For example, `mkdir Banner_pub`.
4. Copy the payload file to this folder. Rename the file to the format <message family>_<message type>_<businessObjectId>.<orderSeqNumber>. For example, Banner_BannerCre_1.1.
5. The rib-fileio application automatically schedules a publisher to pick the payload file and publish the file to respective RIB topic. After successful publication the file is deleted from the folder.
6. To stop publishing, add a file by name "STOP" to the <message_family_pub> folder. For example, `touch Banner_pub/STOP`.
7. Notice that the message is not published to the topic and the payload file still remains in the same folder.
8. Publishing can be resumed by deleting the "STOP" file. For example, rm Banner_pub/STOP.

Consuming Using FileIO

Take the following steps to publish the payload using FileIO application:

1. Configure and install using rib-app-builder tool, the application which uses FileIO for publishing the message.

   Note: Refer to the *Oracle Retail Integration Bus Implementation Guide* Release 14.1, Chapter 11, RIB Customization/Extension.

2. Publish the message to RIB topic through a publishing adapter or RDMT.

   OR

   For testing purpose, upload the payload.xml file by accessing the link http://<host>:<port>/fileio-web/InvokeInjector.jsp?family=<*message_family*>&type=<*message_type*>&businessObjectId=someid. This guides you to place the payload.xml in the application defined WLS folder.

   The host and port are specific to the WLS instance where the FileIO is hosted.

3. Access the link in the step 2 again.

4. Scan the folder WL DOMAIN_HOME/<message_family>_sub for the file with the name format <message_family>_<message_type>_<businessObjectId>.<orderSeqNo>.

   The content of the message consumed is written to this file.

Limitations of FileIO

- Setting routing information during publication is not supported.
- The read and write operations, associated with the files, do not participate in RIB's XA transaction.

Operational Conditions

In a production environment, it is recommended that you use the Oracle RAC database for hosting RIBAQ topics and RIB related tables. If the database is started in an archive mode, the redo logs are archived into database archive logs. The count of the archive log is proportional to the number of database transactions.

Specific to RIB, every message that flows through RIB participates in database transaction. All the transactions get written to a redo log, which later gets archived. Example transactions involve, queuing the message on to topic, de- queuing from topic, writing error message to hospital tables.

With fileio, publishing of messages in bulk involves creating required number of message files under WLS_DOMAIN_HOME directory, within defined location and file name format that has been already discussed. Since messages are huge in number with bulk messaging scenario, an alarming increase in the database archive logs is expected.

Use Cases

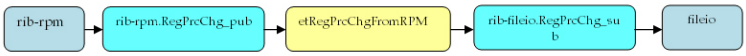
The use cases considered are as follows:

FileIO As a Publisher

FileIO as a publisher, publishes a InvReq message to etInvReq topic and RMS consumes the message from etInvReq topic. The same is depicted in the flow diagram below.

FileIO As a Subscriber

RPM publishes a RegPrcChg message to etRegPrcChgFromRPM topic through rib-rpm and rib-fileio subscribes to the message.



Preparing and Deploying rib-fileio.ear

Take the following steps to prepare and deploy the rib-fileio.ear:

1. Login to the Unix machine and traverse to rib-home directory.
2. Create a rib-fileio directory using the command: mkdir application-assembly-home/rib-fileio
3. Create the rib-fileio adapter specific file.
   a. Create a new file "rib-fileio-adapters.xml" as below, by running touch application-assembly-home/rib-fileio/rib-fileio-adapters.xml. Edit it to add the adapter details as below snippet.

```
<?xml version="1.0" encoding="UTF-8"?>

<rib-adapters
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="rib-adapters.xsd"
appName="rib-fileio">


<subscribers>

<message-driven id="RegPrcChg_sub_1"
initialState="running" />

</subscribers>

 <publishers>

<request-driven id="InvReq_pub_1"
initialState="notConfigurable" />

</publishers>

<hospitals>

<timer-driven id="sub_hosp_0" initialState="running"
timeDelay="10" >

 <timer-task>

   <class
name="com.retek.rib.j2ee.ErrorHospitalRetryTimerTask"/>
```

```
      <property name="reasonCode" value="SUB" />
  </timer-task>
</timer-driven>
<timer-driven id="jms_hosp_0" initialState="running"
timeDelay="10" >
 <timer-task>
  <class
name="com.retek.rib.j2ee.ErrorHospitalRetryTimerTask"/>
      <property name="reasonCode" value="JMS"/>
  </timer-task>
</timer-driven>
</hospitals>
</rib-adapters>
```

b. Create rib-fileio-adapters-resources-properties as below, by running touch application-assembly-home/rib-fileio/rib-fileio-adapters-resources.properties. Edit it to add the adapter details as below snippet.

```
#
# If this changes, ManagedAdaptersResourcesPropertiesTest will need to
# change accordingly.
#


sub_all.name=Subscribers
sub_all.desc=Manages all subscribers at the same time


RegPrcChg_sub_1.name=RegPrcChg Subscriber, channel 1
RegPrcChg_sub_1.desc=Subscriber for the RegPrcChg family
through channel


InvReq_pub_1.name=InvReq Publisher, channel 1
InvReq_pub_1.desc=Publisher for the InvReq family through
channel 1


hosp_all.name=Hospital Retriers
hosp_all.desc=Manages all hospital retriers at the same
time


sub_hosp_0.name=SUB Hospital Retry
sub_hosp_0.desc=Inject messages into from the Error
Hospital
```

jms_hosp_0.name=JMS Hospital Retryjms_hosp_0.desc=Re-publish messages from to JMS after JMS is brought back up.

c. Create rib-fileio.properties as below, by running touch application-assembly-home/rib-fileio/rib-fileio.properties. Edit it to add the adapter details as below snippet.

```
##################################################
# rib-fileio application specific properties go here.  #
#                                #
# All properties have default values, add the      #
# property here only if the default value does not  #
# suit your environment.                  #
##################################################
```

4. Add rib-fileio as an application in application-assembly-home/conf/rib-application-assembly-info.xml.

```
<rib-app id="rib-fileio" type="javaee-app">
<ear>
  <classpath refid="rib-app.global.ear.classpath" />
  <java-ee-module>
  <web-war />
 <ejb-jar>
   <classpath refid="rib-app.global.ejb-jar.classpath" />
 </ejb-jar>
   <jms-jca-connector>
  <classpath
refid="rib-app.global.jms-jca-connector.classpath"/>
 </jms-jca-connector>
 </java-ee-module>
</ear>
<resource>
 <resource-path refid="rib-app.global.resource-path" />
<resource-path>
 <fileset dir=".">
   <include name="retail/remote_service_locator_info_
ribserver.xml" />
   <include name="rib-fileio.properties" />
   <include name="rib-fileio-adapters.xml" />
    <include
name="rib-fileio-adapters-resources.properties" />
 </fileset>
```

```
 </resource-path>
 </resource>
</rib-app>
```

5. Update the various sections of the deployment-home/conf/rib-deployment-env-info.xml file with the following steps:

   a. Add "fileio" in app-in-scope-for-integration.

      <app-in-scope-for-integration>

      <app id="tafr" type="tafr-app"/>

      <app id="sim" type="javaee-app"/>

      <app id="rpm" type="javaee-app"/>

      <app id="fileio" type="javaee-app">

      </app-in-scope-for-integration>

   b. Define the WebLogic Server information for the fileio application. Add the following XML section under the <weblogic> tag:

      ```
      <wls id="rib-fileio-wls1">
      <wls-instance-name>rib-fileio-wls-instance</wls-instance-name>
      <wls-instance-home>ribuser@ribhost.example.com://u00/Oracle/Middleware/user_
      projects/domains/RIBDomain/servers/rib-fileio-wls-instance
      </wls-instance-home>
      <wls-listen-port protocol="http">19103</wls-listen-port>
      <wls-user-alias>rib-fileio-wls-user-alias</wls-user-alias>
      </wls>
      ```

   c. Edit rib-deployment-env-info.xml and point rib-sim to fileio by modifying the <jndi> section for rib-sim as follows:

      ```
      <jndi>
      <url>t3://ribhost.example.com: 19103/fileio</url>
      <factory>weblogic.jndi.WLInitialContextFactory</factory>
      <user-alias>sim_jndi_user-name-alias</user-alias>
      </jndi>
      ```

   d. Add the following XML section under the <rib-applications> (before </rib-applications>) section. In the jndi/url xml tag section, point it to the location where fileio-<version>.ear (not rib-fileio.ear) is deployed. Refer to *Oracle Retail Integration Bus Installation Guide* for details.

      ```
      <rib-app id="rib-fileio" type="javaee-app">
      <deploy-in refid="rib-fileio-wls1"/>
      <rib-admin-gui>
      ```

```
<web-app-url>URL to the rib admin gui app.</web-app-url>
<web-app-user-alias>rib-fileio_rib-admin-gui_
web-app-user-alias</web-app-user-alias>

</rib-admin-gui>

<error-hospital-database>
<hosp-url>jdbc:oracle:thin:@ribhost.example.com:1521:ribde
v01></hosp-url>

<hosp-user-alias>rib-fileio_error-hospital-database_
user-name-alias</hosp-user-alias>

</error-hospital-database>

<app-database-not-applicable/>

<notifications>

<email>
<email-server-host>mail.oracle.com</email-server-host>

<email-server-port>25</email-server-port>
<from-address>admin@example.com</from-address>
<to-address-list>admin@example.com</to-address-list>

</email>

<jmx/>

</notifications>

<app id="fileio" type="javaee-app">

<jndi>

<url>t3://fileiohost.example.com:7002/fileio</url>

<factory>weblogic.jndi.WLInitialContextFactory</factory>

<user-alias>fileio_jndi_user-name-alias</user-alias>

</jndi>

</app>

</rib-app>
```

6. Update the rib-integration-flows.xml of rib-func-artifact-<version>.war file to add the publisher and subscriber flow information for the fileio application. Follow these steps:

   - mkdir temp
   - cd temp
   - jar xf ../application-assembly-home/rib-func-artifacts/rib-func-artifact-<version>.war integration/rib-integration-flows.xml
   - edit integration/rib-integration-flows.xml.

     ```
     <message-flow id="31">

     <node id="rib-fileio.InvReq_pub" app-name="rib-fileio"

     adapter-class-def="InvReq_pub" type="DbToJms">

     <in-db>default</in-db>
     ```

```
<out-topic>etInvReq</out-topic>
</node>
<node id="rib-sim.InvReq_pub" app-name="rib-sim"
        adapter-class-def="InvReq_pub" type="DbToJms">
         <in-db>default</in-db>
         <out-topic>etInvReq</out-topic>
      </node>
<node id="rib-rms.InvReq_sub" app-name="rib-rms"
adapter-class-def="InvReq_sub" type="JmsToDb">
<in-topic>etInvReq</in-topic>
<out-db>default</out-db>
</node>
</message-flow>
<message-flow id="34">
      <node id="rib-rpm.RegPrcChg_pub" app-name="rib-rpm"
         adapter-class-def="RegPrcChg_pub" type="DbToJms">
          <in-db>default</in-db>
          <out-topic>etRegPrcChgFromRPM</out-topic>
       </node>
      <node id="rib-sim.RegPrcChg_sub" app-name="rib-sim"
          adapter-class-def="RegPrcChg_sub" type="JmsToDb">
          <in-topic>etRegPrcChgFromRPM</in-topic>
          <out-db>default</out-db>
      </node>
  <node id="rib-fileio.RegPrcChg_sub"
app-name="rib-fileio"
          adapter-class-def="RegPrcChg_sub" type="JmsToDb">
          <in-topic>etRegPrcChgFromRPM</in-topic>
          <out-db>default</out-db>
      </node>
```

- `jar uvf ../application-assembly-home/rib-func-artifacts/rib-func-artifact-<version>.war integration/rib-integration-flows.xml`
- cd ..
- rm -rf temp

7. Run rib-app-compiler with -setup-security-credential from application-assembly-home/bin.

8. Prepare the JMS using rib-app-builder tool.

   For example, `rib-home/deployment-home/bin ./rib-app-deployer.sh -prepare-jms`

9. Deploy rib-func-artifact-<version>.war by using rib-app-builder tool.

   For example, `rib-home/deployment-home/bin ./rib-app-deployer.sh -deploy-rib-func-artifact-war`

10. Deploy rib-fileio.ear by using rib-app-builder tool.

    For example, `rib-home/deployment/bin ./rib-app-deployer.sh -deploy-rib-app-ear rib-fileio`

11. Bounce all rib-<app> servers.

Implementing the Use Cases

Take the following steps to implement "FileIO as publisher" use cases:

1. Deploy fileio-<version>.ear. Refer to section, Installation and Configuration in this chapter for more details.
2. Prepare and deploy rib-fileio.ear. Refer to the section, Preparing and Deploying rib-fileio.ear in this chapter for more details.
3. Login to rib-admin-gui of rib-fileio application and make sure the publishing adapter "InvReq_pub "is up.
4. Create a directory under WLS_DOMAIN_HOME with name "InvReq _Pub"
5. Copy InvReqDesc.xml from "retail-public-payload-xml-samples.zip" to above created folder and rename it to "InvReq_InvReqCre_1.1".
6. The message should be published successfully by rib-fileio. Check rib publishing adapter logs for successful publication.
7. Check the subscribing adapter logs from rib-rms admin gui for successful subscription.

Take the following steps to implement "FileIO as subscriber" use cases:

1. Deploy fileio-<version>.ear. Refer to section, Installation and Configuration in this chapter for more details.
2. Prepare and deploy rib-fileio.ear. Refer to the section, Preparing and Deploying rib-fileio.ear in this chapter for more details.
3. Login to rib-admin-gui of rib-fileio application and make sure the subscribing adapter "RegPrcChg_sub "is up.
4. Access URL : http://fileiohost.example.com:7002/fileioweb/InvokeInjector.jsp?family=RegPrcChg&type=RegPrcChgCre&businessObjectId=1001.
5. Traverse to directory WLS_DOMAIN_HOME and verify that a new directory by name RegPrcChg_sub got created and has a file with name RegPrcChg_ RegPrcChgCre_1001_1.1 created in it.

4

RIB Diagnostics and Monitoring Tool (RDMT)

The RIB Diagnostic and Monitoring Tool (RDMT) is a collection of command line tools, written in UNIX shell script along with supporting Java classes packaged in jar files. There are various tools to address these areas:

- Installation Verification (reports)
- Operations (scanning and monitoring)
- Production (scanning and quick triage)
- Test and Support (scanning and fine grain control)
- AQ JMS support and tools

RIB is a complex collection of distributed components, and there are a variety of GUI tools. RDMT augments the GUI tools and provides command line control and access to RIB functions at all levels. RDMT is written to be stand-alone and to provide examples and capabilities for integration into enterprise level OSS and management frameworks, such as Oracle Enterprise Manager, Tivoli, or HP OpenView.

Functionality

- Support for Oracle WLS RIB Version.
- Support for local/remote installation.
- Support for Oracle Streams AQ JMS as the JMS Provider.
- Support for RIB Hospital databases.
- Support for RAC Configured Databases.
- Support for JMX control of all RIB Components
- Support for message Pub/Sub.

All of the scripts are written to be examples of specific functionality, but have been integrated into a simple tool kit that is configuration driven and has a very simple character-based menu system provided to allow a single point of integration.

RDMT and User Roles and Responsibilities

The tools are written to provide capabilities and examples of functions for users with various roles and responsibilities.

The primary target role is the RIB administrator, who is responsible for the installation, configuration, and deployment of RIB components. The RIB administrator also performs ongoing RIB Software Life Cycle management and provides production

operation support. This person has full permissions on all of the application server directories and has full read and execute permissions on the Oracle Application Server tools, such as opmnctl and the WLS instance sub directories.

Local or Remote Installations and Capabilities

RDMT can be installed by a user on the system that may or may not have the RIB/WLS environment. RDMT tools support local and remote WLS functions through JMX.

In remote installations, some scripts in the toolkit expect the installing user to have read permissions of the WebLogic home RIB WLS sub-directories or require execute permission of opmnctl. Therefore, these will return file or permissions errors.

Once the roles and responsibilities of the user have been understood and established, follow the installation instructions available in the *Oracle Integration Bus Installation Guide*.

RDMT Support jars

The following table provides information on the support RDMT jars:

Table 4–1 jar Files

.jar File	Description
rib-jms-api-<version>.jar	Support classes for jms.
rib-jms-admin-aq-impl-<version>.jar	Specific impl for AQ
rib-jms-admin-<version>.jar	Support classes for AQ admin
jmx-cmd-line-ui-<version>.jar	JMX client
rdmt-<version>.jar	Support tools

Sample XML Messages

The RDMT release packages a zip file of example xml messages for each message family and message type payload. The zip file is located in the RDMT subdirectory testmsgs.

Tools Overview

RDMT has been designed as a set of command line tools that can serve generally needed functions with examples for retailer specific uses, and to provide a ready to use, low impact application. In many situations, it is a requirement to have tools that consume low bandwidth to manage and triage the RIB. These tools provide alternatives to the GUI based tools. The other common requirement is for control and monitoring command line scripts that can be incorporated into enterprise operations scheduling frameworks, such as Autosys or Appworkx.

RDMT has been organized around a very simple character-based menu system that can be modified to suit the deployment roles and responsibilities and to provide some structure by functional area.

RDMT as an Application

This section describes RDMT as an application.

SCRIPTDIR

All of the tools have been organized into a simple application and accessible via the character-based menu system. All of the tools have been designed to execute relative to a based directory (rdmt). Within that base directory, all tools expect to find all of the support libraries and other scripts. To execute any tool, all that is needed is to set the base directory as an environmental variable, SCRIPTDIR.

Setup

RDMT can be installed either inside or outside rib-home or in a remote server. To install inside or outside rib-home, the installation script (configbuilder.sh) automatically fetches all necessary configuration parameters from rib-deployment-env.-info.xml from inside the specified rib-home/deployment-home/conf directory.

However, if RDMT is installed in a remote server, the installation script prompts for the RIB deployment environment specific values. All of the scripts have been designed to be configuration driven by property files. The setup process updates these files.

Current Configuration

Because there are multiple configurations possible with the fully distributed RIB, all of the tools are designed to work against a set of property files that provide the values needed to execute. Collectively, these are called "current." In the menu system, there are functions that allow configuration of n-number of configurations. For example, there can be n-number of rib-<app>'s configured. Other functions set runtime configuration files to these "current" configurations. All tools then read these "current" values and perform tasks against them.

RDMTLOGS

All of the tools are designed to produce logs and to use temporary files. The location of these logs is a configuration parameter and defaults to RDMTLOGS within the rdmt base directory.

RDMT RAC Support

RDMT supports RAC configured databases. The user needs to provide the entire database connection URL when prompted during the setup process.

The user needs to provide the same JDBC connection URL for AQ JMS or Hospital Databases as supplied in rib-deployment-env-info.xml during RIB installation. The user can provide either thin JDBC connection URL or long JDBC connection URL format (in case of RAC configured database) depending on the user's environment.

The user needs to provide only the database user name, password, and connection URL to configure for any AQ JMS/ Hospital database. The example below shows the configuration for a hospital database during the RDMT setup process.

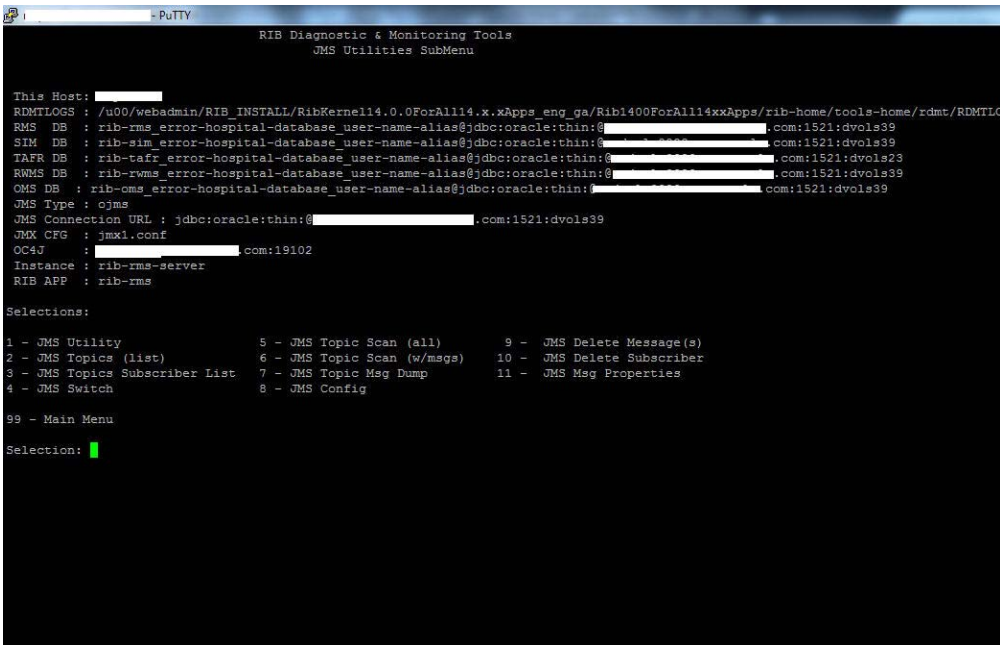
Example 4–1

```
Enter RMS database Connection URL [needs_value]: jdbc:oracle:thin:@(DESCRIPTION
```

```
=(ADDRESS_LIST =(ADDRESS = (PROTOCOL = TCP)(HOST = dbhost1)(PORT = 1521))(ADDRESS
= (PROTOCOL = TCP)(HOST = dbhost2)(PORT = 1521))(LOAD_BALANCE = yes))(CONNECT_DATA
=(SERVICE_NAME = orcl)))
You entered: jdbc:oracle:thin:@(DESCRIPTION =(ADDRESS_LIST =(ADDRESS = (PROTOCOL =
TCP)(HOST = dbhost1)(PORT = 1521))(ADDRESS = (PROTOCOL = TCP)(HOST = dbhost2)(PORT
= 1521))(LOAD_BALANCE = yes))(CONNECT_DATA =(SERVICE_NAME = orcl)))
Enter to Continue? y/n/q [y]:
```

RDMT Main Menu

This is the main entry point into the RDMT tool kit application. Most selections invoke other submenus. But, for convenience, several tools included in other submenus, are directly accessed from this menu.



WLS/JMX Utilities

Script Used:

rdmt_jmx_submenu

Description:

This menu option exposes the various tools that use JMX to interact with the WLS instance and to control or status the current rib-<app> and its components.

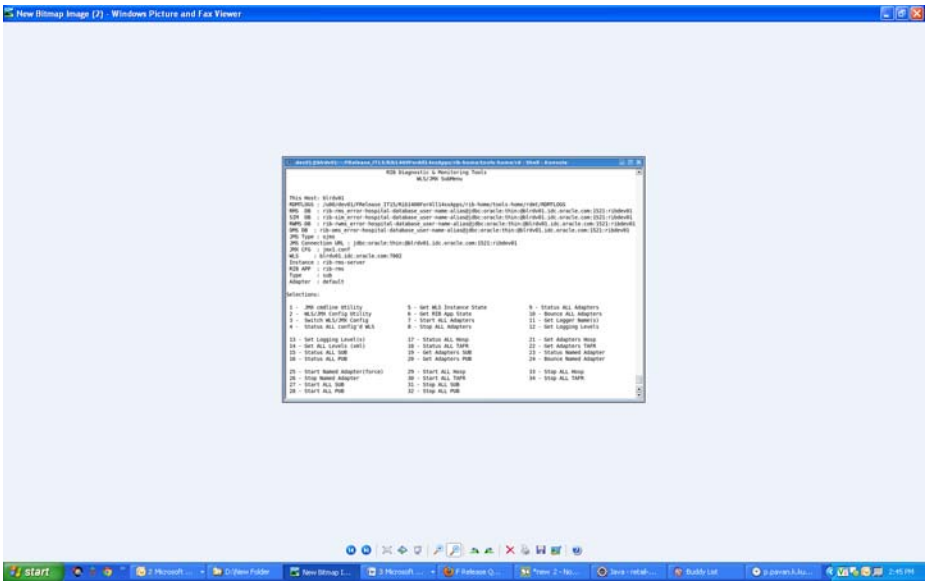


Table 4–2 WLS/JMX Utilities

Script	Description
jmx_app_state.sh	status of the currently active rib application
jmxcmdline_interactive.sh	A wrapper to the jmx client support classes. This script directly invokes the interactive functions.
jmxcmdline.sh	General wrapper for other tools to invoke specific jmx functions.
jmxconfig.sh	There are multiple configurations possible with the fully distributed J2EE RIB. This utility is used to manage the configuration files that allow the rdmt tools to access them. This option can also be used to switch/re-configure the previously configured WLS/JMX configuration.
jmx_get_logger_names.sh	RIB app logging tool
jmx_get_logging_levels_ all.sh	RIB app logging tool
jmx_get_logging_levels.sh	RIB app logging tool
jmx_set_logging_levels.sh	RIB app logging tool
jmx_managed_adpaters.sh	Common script used by all jmx tools to interact with the jmx client jar. Many of the menu selections merely set the calling parameters to this tool.
jmx_OC4Jribstatus.sh	Get the run state of the rib-app WLS instance and application for all configured.
jmx_OC4Jrib_scan.sh	For all configured rib-app scans the state of the instance, app and adapters.
jmx_oc4j_state.sh	Status of the currently active WLS instance.
jmx_switch_config.sh	This utility is used to switch the active configuration file that the rdmt tools use.
jmx_tester.sh	Test script for testing arbitrary jmx commands within the RDMT framework. This is not a menu selection since it requires user editing.

Table 4–3 WLS/JMX Utilities

Utility	Description
start all adapters (jmx)	This utility option starts all adapters of a rib-<app>, where app refers to rms, rwms, sim, rpm, rfm, oms, or tafr. The adapters start only when the initial state is specified as running for the adapters in the rib-<app>-adapters.xml. If the initial state=stopped, an error is thrown: "Cannot start; initial state is set to stopped." If an adapter already is running, executing this option keeps the adapter in the previous state.
start adapter (jmx)	This utility option starts a single adapter of a rib-<app>, where app refers to rms, rwms, sim, rpm, rfm, oms, or tafr. It starts the adapter only when the initial state is specified as running for the adapter in the rib-<app>-adapters.xml. If the initial state=stopped, an error is thrown: "Cannot start; initial state is set to stopped." If the adapter already is running, executing this option keeps the adapter in the previous state itself.
startForced adapters (jmx)	This utility starts all adapters of a rib-<app>, where app refers to rms, rwms, sim, rpm, rfm, oms, or tafr. It starts all adapters irrespective of their initial state in rib-<app>-adapters.xml.
startForced adapter (jmx)	This utility option starts a single adapter of a rib-<app>, where app refers to rms, rwms, sim, rpm, rfm, oms, or tafr. It starts the adapter irrespective of its initial state in rib-<app>-adapters.xml.

JMS Tools

Script Used:

rdmt_jmsutil_AQ_submenu

Description:

This menu option exposes the various JMS functionalities available in the tool kit. For convenience, some tools from other submenus are presented here as well.

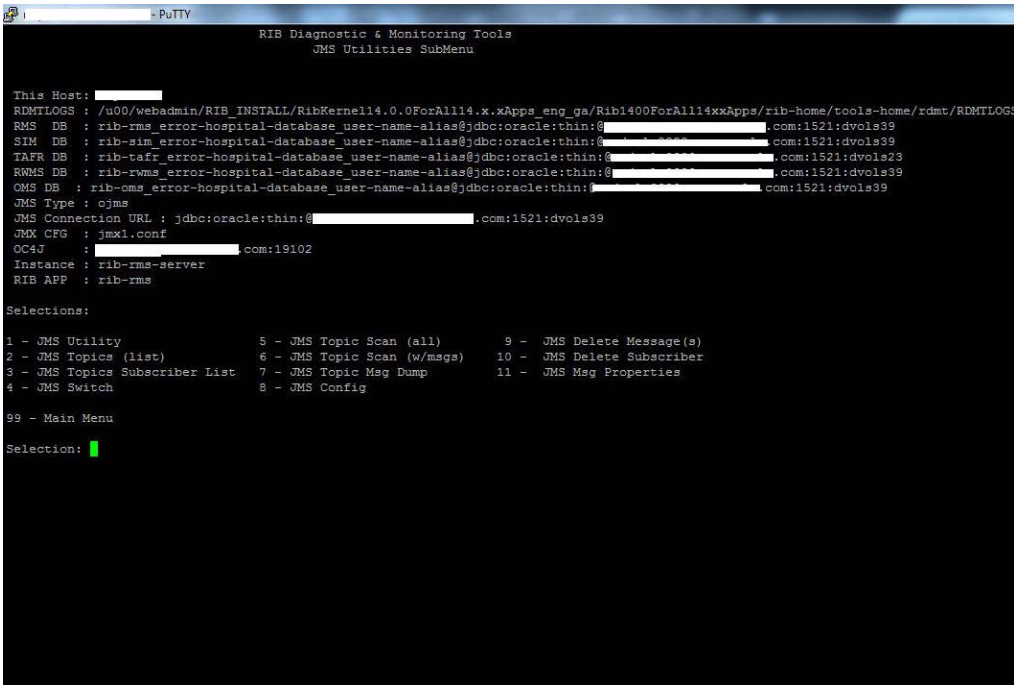


Table 4–4 JMS Tools

Script	Description
aqjmscmdline.sh	Common script used by all JMS tools to interact with the AQ JMS client jar. Many of the menu selections merely set the calling parameters to this tool.
deletemsgAQ.sh	Delete message(s) on a specified JMS topic for a specified subscriber.
dmpmsgAQ.sh	Get a dump of message(s) on an AQ JMS topic for a specified subscriber.
dmp_msg_statsAQ.sh	Get a dump of properties of message(s) on an AQ JMS topic for a specified subscriber.
jmsconfig.sh	RDMT supports configuration of n-number of JMS Providers. This utility configures the values to support. rdmt_jms_submenu is used to make one the current configuration.
jmstopicsAQ.sh	Query the AQ JMS for all of the topics and the message count on each topic.
jmstopicsAQ_scan.sh	Query the AQ JMS for just the topics with message counts.
jmsutil.sh	This utility provides direct access to the AQ JMS Java API utilities.

PUB/SUB Msg Tools

Script Used:

rdmt_msgutil_submenu

Description:

All of the tools is this menu are wrappers that expose functions in the Java utilities rib-jms-api.jar included in the tool kit library. These are general purpose pub/sub functions that are written to support the various JMS Providers for the RIB, such as AQ JMS.

In addition to these, we have a utility to test any TAFR's business implementation. By providing the necessary parameters which are prompted, the user can see the output of a particular TAFR either on the console or in the specified file.

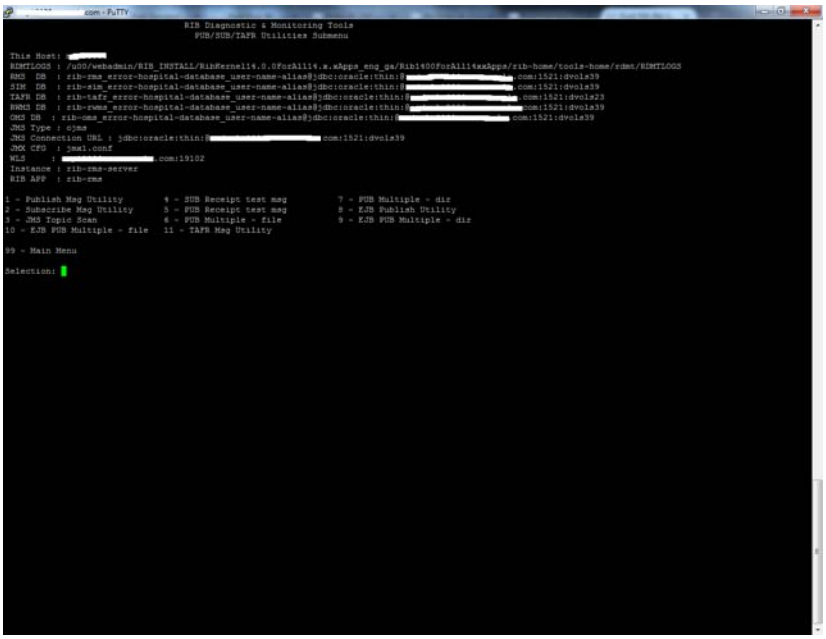


Table 4–5 PUB/SUB Msg Tools

Script	Description
pubmsgutil.sh	Provides direct access to the Java API by prompting for all of the expected command line values it expects.
submsgutil.sh	Provides direct access to the Java API by prompting for all of the expected command line values it expects.
submsg.sh	Higher level wrapper that uses the configuration values to shortcut the values needed to call the JavaAPIs.
pubmsg.sh	Higher level wrapper that uses the configuration values to shortcut the values needed to call the Java APIs.
pubmsgutil_multiple.sh	Script that supports multi-message publication.
pubmsgutil_directory.sh	Script that supports multi-message publication.
ejbpub_util.sh	Utility to wrapper the EJB Message Publish Java API.
ejbpubutil_multiple.sh	Script that supports multi-message publication through EJB Message Publish Java API.
ejbpubutil_directory.sh	Script that supports multi-message publication through EJB Message Publish Java API.
tafrmsgutil.sh	Script that can be used to test any TAFR Business implementation.

RIB Health Tools

Script Used:

rdmt_ribhealth_submenu

Description:

This option leads the user to the submenu through which the user can get the current RIB health status.

Table 4–6 RIB Health Tools

Script	Description
cron_ribhealth.sh	See "RIB Health" in this chapter.
ribejbping.sh	See "EJB Ping (RIB)" in this chapter.
appejbping.sh	See "EJB Ping (RIB)" in this chapter.
loglookoc4j.sh	See "Scan RIB Logs / Scan RIB Logs (Delta)" in this chapter.
loglookoc4j_delta.sh	See "Scan RIB Logs / Scan RIB Logs (Delta)" in this chapter.
timingsutil.sh	See "RIB Timings Utility" in this chapter.
ttestrms.sh	This script scans a list of RMS MFQ tables using a JDBC connection. (see mfqtables.conf).
ttestrdm.sh	This script scans a list of RWMS Upload tables using a JDBC connection. (see uploadtables.conf).
OC4JConfigReport.sh	See "RIB Configuration Report" in this chapter.

Hospital Scan Tools

Script Used:

rdmt_hosp_submenu

Description:

This option leads the user to the RIB hospital for various applications submenu through which the user can get the current RIB hospital status.

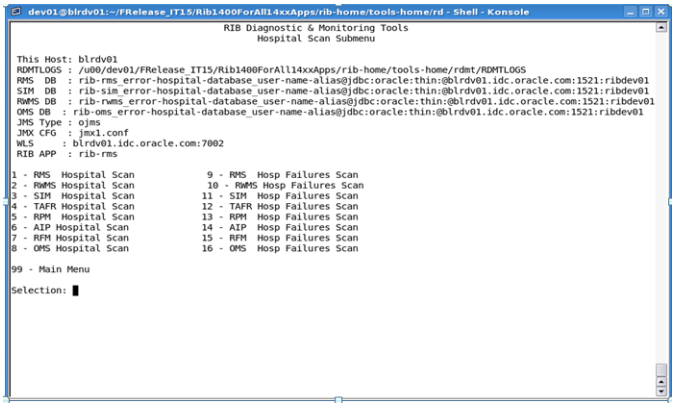


Table 4–7 Hospital Scan Tools

Script	Description
htest.sh	This script calls a Java class that uses JDBC to access the database(s) containing the Hospital tables. It scans the Hospital RIB_MESSAGES table and reports the following: ▪ how many messages (row count) ▪ how many messages exceed the retry count ▪ how many messages of a topic are present
htest_failures.sh	This script calls a Java class to scan the RIB Hospital Message Failure table using a JDBC connection.

RIB Admin Tools

Script Used:

rdmt_ribadmin_submenu

Description:

The ribadmin script was stand-alone in previous RIB releases. Those functions have since been moved into this menu item. The ribadmin.sh script is sourced to make the existing functions available to the menu items and the variables that the scripts expected have been mapped to rdmt configuration files.

Since many of the functions expect execute permissions on opmnctl as well as read/write permissions on the WLS directory tree, this menu and the tools are designed for the ribadmin role.

If RDMT is installed in the RIB App Builder rib-home and that is accessible and configured, then this menu exposes the rib-app-builder menu selection. A test is performed to verify the rib-home is configured, if not, then the selection will not appear.

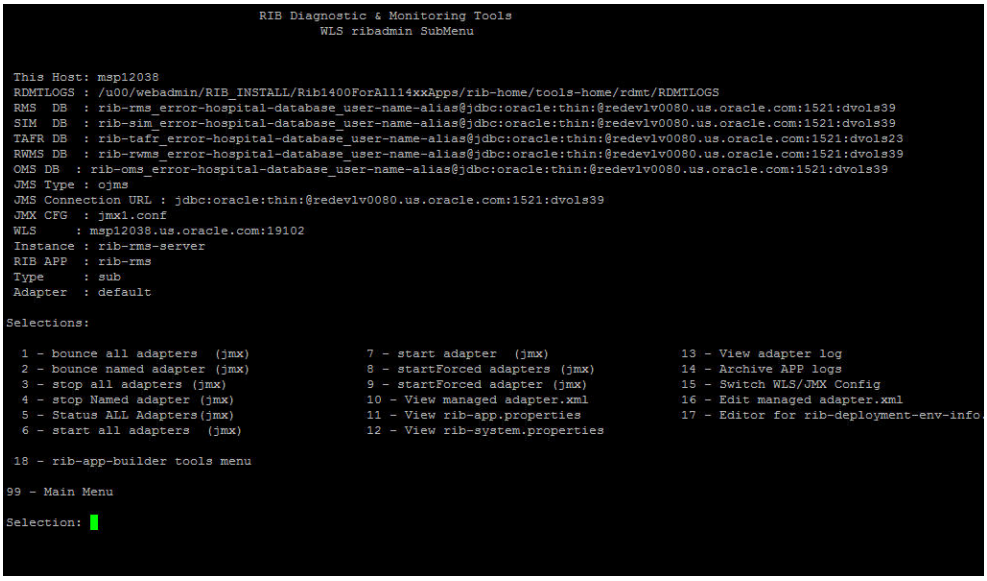


Table 4–8 RIB Admin Tools

Script	Description
ribadmin.sh	This script contains most all of the functions that are exposed by this menu.

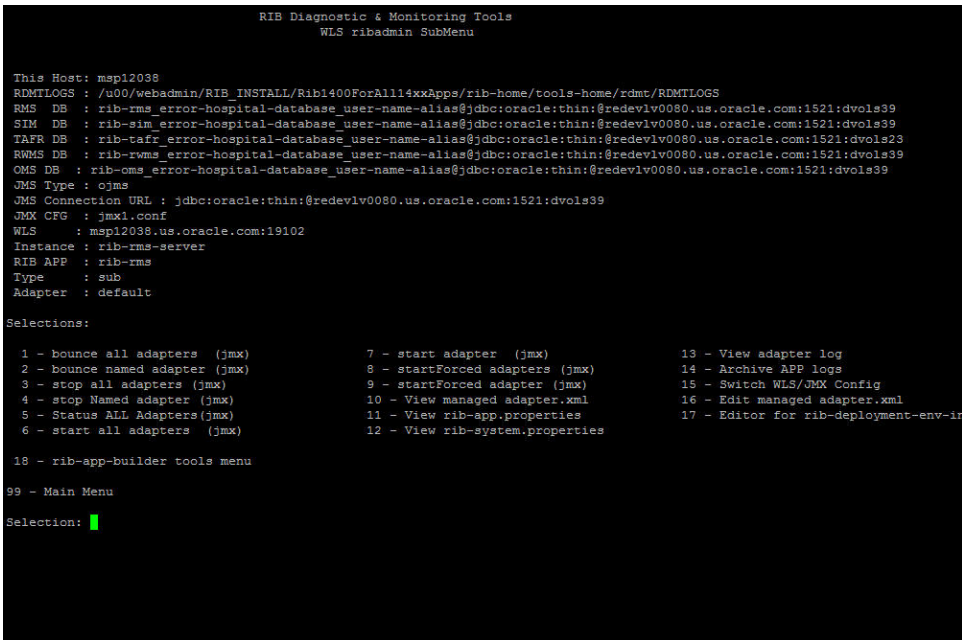
RIB App Builder Tools

Script Used:

rdmt_ribappbuilder_submenu

Description:

This option leads the user to the RIB App Builder tools installed in the rib-home. For a description of the tools and usage, see Chapter 2, "Application Builder" in the *Oracle Retail Integration Bus Operations Guide*.



Scan RIB Logs / Scan RIB Logs (Delta)

Scripts Used:

loglookoc4j.sh, scan_logs.sh, loglookoc4j_delta.sh, scan_logs_delta.sh

Description:

These scripts perform a log scan to look for a pattern ("Exception") in all of the log files in a directory of the currently active WLS instance. Since they perform file system scans, the RDMT tools must be installed on the host that contains these logs and must have read permissions on the directories and the files.

As the tool scans all of the logs it writes the matches to a single log file. This becomes the base file. A second script (delta) looks for the same pattern, but compares the matches against the base file, and outputs only new ones. The primary scripts are the scan_logs.sh and the scan_logs_delta.sh. The files created and used by these scripts are controlled by the rdmt.conf entries.

The location of these files should be sized to handle large text files, since it is possible for there to be many exceptions and these will contain the consolidated entries from potentially hundreds of logs. The location is the tmp files parameter set during RDMT installation and is defaulted to RDMTLOGS/tmp.

RIB Health

Script Used:

cron_ribhealth.sh

Description:

This utility is a general purpose script that invokes other tools and functions in the tool kit to take a snapshot in time of the run-time state of all of the configured rib-apps. Because this script uses specific jar files as well as other tools in the tool kit, it requires that SCRIPTDIR be set to the rdmt base directory.

It produces a rib_health report on the console as well as a time-stamped log written to the RDMTLOGS/tmp directory. Each execution of the script produces one of the logs, and then over-writes a log called lastrun as well. There is a menu selection that views the lastrun report.

RIB Configuration Report

Script Used:

OC4JConfigReport.sh

Description:

After the RIB has been installed and configured on WLS, the user can verify installations and configurations using RDMT. A script is linked in the RDMT menu that scans the installations and configurations of rib applications deployed using the configuration settings in the RDMT configuration files.

It is recommended that after the installation is complete, you run the RIB Config Report utility from the RIB Health Menu option. This outputs the results of the scan on the console as well as in an output file under the specified TEMP FILES DIRECTORY. Each run produces a time-stamped log and updates a log called lastrun-config that is viewable from a menu selection.

This script was written to take a snapshot of the RIB environment and test for basic configuration issues. This utility does the following:

- Displays all RIB apps and shows the status for each RIB WLS instance and application.
- Performs JMX related functions such as scanning configurations for each jmxX.conf file and displays the status of the adapters, exceptions from scanning the logs, and so on.
- Performs checks using JMS configuration.
- Performs checks using Hospital configurations.

RIB Timings Utility

Script Used:

timingsutil.sh

Description:

RIB can log a set of timing entries whenever it creates, transform, routes, filters, or subscribes to messages on the RIB. This utility only works when the RDMT is installed on the host system where the logs are generated and the RDMT user has permissions to read the log directories.

The timingsutil.sh script is a wrapper to the RIBTimings Java class. This script runs the RibTimings post processor on an adapter's timing file. It prompts for the adapter name then it analyzes the timings logs for that particular adapter. The output is to the screen as well as a file of CSV format in the RDMT temp files directory; RDMTLOGS/tmp/<adapter>.csv. which contains the detailed analysis of timings logs.

This csv file can be directly viewed by Excel. To use this function, the adapter timing log parameters must be set to DEBUG.

JMS Publish Utility

Script Used:

pubmsgutil.sh

Class:

RibJmsPublisherTester

Description:

This utility was developed to publish a message to a JMS topic. The pubmsgutil.sh is a wrapper script to RibJmsPublisherTester. It calls the JMS API and publishes the message on to the topic.

Usage:

```
"java com.retek.rib.jms.RibJmsPublisherTester
-j <JMS provider>
-x <JMS URL>
-u <JMS username>
-p <JMS password>
-t <topicName>
-n <xmlFileName>
-wm <should messages be wrapped in RIBMessage envelop>
```

```
-f <messageFamily>
-m <messageType>
-ri <routingInfo>
-tv <threadValue>
-nt   <<number of times - optional (default value is 1)>>
```

EJB Publish Utility

Script Used:

ejbpub_util.sh

Class:

RibMessagePublisherClient

Description:

This utility was developed to wrapper the EJB Message Publish Java API. The ejbpub_util.sh is a wrapper script to RibMessagePublisherClient. It calls the specified EJB service and publishes the message on to the AQ JMS. It uses the platform service to publish the message. The user needs to specify the necessary parameters.

Usage:

```
java com.oracle.rib.rdmt.util.RibMessagePublisherClient
 -host <<host|1>>
 -port <<RMI port -- required>
 -app  <<App name -- required>>
 -fa   <<family -- required>>
 -ty   <<type -- required>>
 -us   <<user | optional>>
 -pw   <<password | optional>>
 -fi   <<file -- required>>
```

TAFR Msg Utility

Script Used:

tafrmsgutil.sh

Class:

TestAnyTafrBOImpl

Description:

This utility is developed to test any TAFR Business implementation. This helps to check the output of a particular TAFR by providing the necessary prompted arguments. It prompts the user for the TAFR name, the location of the input sample file, adapter id, and the output file name to direct the output (optional). If the output file name is not specified, the default output is routed to stdout. Once provided the valid arguments, the user can see the output of that particular TAFR.

Sample Output:

```
------------------------------------------------
    TAFR TEST UTILITY
------------------------------------------------
 This utility was developed to test any TAFR Business implementation.
 These are designed to help diagnose/validate the TAFR implementation.

 Usage: java TestAnyTafrBOImpl
```

```
        <tafr name -- required>
        <file path -- required>
        <true|false print message data -- required>
        <adapter-id -- required>
        <file name -- optional | default output routed to stdout>

Do you wish to continue ?
Enter to Continue? y/n/q [y]:

Existing parameter values:
StoresToStoresPhys /u00/rib-home/tools-home/rdmt/testmsgs/PODesc.xml true
 StoresToStoresPhys_tafr_1

Do you wish to execute using existing values?

Enter to Continue? y/n/q [y]: n

Enter TAFR Name [StoresToStoresPhys]:
You entered: StoresToStoresPhys
Enter to Continue? y/n/q [y]:

Enter the Input message file and path
 /u00/rib-home/tools-home/rdmt/testmsgs/PODesc.xml]:
 /u00/rib-home/tools-home/rdmt/testmsgs/storedel.xml
You entered: /u00/rib-home/tools-home/rdmt/testmsgs/storedel.xml
Enter to Continue? y/n/q [y]:

Print Message Data? [ true | false]  [true]:
You entered: true
Enter to Continue? y/n/q [y]:

Enter Adapter ID [StoresToStoresPhys_tafr_1]:
You entered: StoresToStoresPhys_tafr_1
Enter to Continue? y/n/q [y]:
Enter the Output file name and path if required (default output routed to stdout)
 []:
You entered:
Enter to Continue? y/n/q [y]:
log4j:WARN No appenders could be found for logger
 (com.retek.rib.domain.ribmessage.bo.RibMessagesFactory).
log4j:WARN Please initialize the log4j system properly.
Trying to load rib-system.properties from
class.path=lib/rib-private-tafr-business-impl.jar:../../application-assembly-home/
rib-tafr/
rib-system.properties loaded from
 file:/u00/rib-home/application-assembly-home/rib-tafr/rib-system.properties
Trying to load rib-tafr.properties from
class.path=lib/rib-private-tafr-business-impl.jar:../../application-assembly-home/
rib-tafr/
rib-tafr.properties loaded from
 file:/u00/rib-home/application-assembly-home/rib-tafr/rib-tafr.properties

**********RIBMESSAGES TRANSFORMED DATA**********
<?xml version="1.0" encoding="UTF-8"?>
<RibMessages xmlns="http://www.oracle.com/retail/integration/rib/RibMessages"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.oracle.com/retail/integration/rib/RibMessages
file:///C:/rib-home/application-assembly-home/rib-tafr/integration/xsd/RibMessages
.xsd">
  <ribMessage>
```

```
    <family>Stores</family>
    <type>storedel</type>
    <ribmessageID>11.0.x|RIBMessagePublisherEjb|null|2006.01.23
 11:45:46.052|510</ribmessageID>
    <routingInfo>
      <name>loc_type</name>
      <value>p</value>
    </routingInfo>
    <routingInfo>
      <name>dest_id</name>
      <value/>
      <detail>
        <dtl_name>loc_type</dtl_name>
        <dtl_value>p</dtl_value>
      </detail>
    </routingInfo>
    <publishTime>2006-01-23 11:45:46.052 CST</publishTime>
  <messageData>&lt;AllocDesc&gt;&lt;store&gt;5&lt;/store&gt;
   &lt;/AllocDesc&gt;
</messageData>
    <customData/>
    <customFlag>F</customFlag>
  </ribMessage>
</RibMessages>
**********************PAYLOAD DATA************************
<?xml version="1.0" encoding="UTF-8"?>
<AllocDesc>
<store>5</store>
   </AllocDesc>
*****************************************************************
***OUT MESSAGE WILL BE ROUTED TO THE FOLLOWING TOPIC(S)***********
*****************************************************************
        1) etStoresPhys
*****************************************************************
```

Configure Multi Channels

Script Used:

configMultiChannel.sh

Description:

This utility is developed to make multi channel configuration easy. Using this utility user can configure multichannel for the desired flow id, also configuration changes can be reverted back in case needed.

Script reads rdmt.conf file to get rib-home and log file location. After rib-home is located, script reads rib-deployment-env-info.xml to know the applications in scope. Also script reads rib-integration-flow.xml present in rib-home/application-assembly-home/rib-func-artifacts/rib-func-artifact.war.Script makes list of adapters to be multi channel configured. Script also takes into consideration adapters from flows with ref-id to one or more adapter of given flow.

After the script gets all the desired inputs from user and confirmation to proceed, script will update xml and properties files for the applications which are in scope.

Usage:

This utility is available in rdmt main menu option 17. On running this utility, you are prompted to enter the message-flow-id. Second input to the utility will be number of

channels to configure. After the utility is run with all the desired inputs, a user friendly message is displayed on screen. In case this configuration already exists for the given flow and number of channels, the following message appears on the screen - multi channel configuration is already existing. Else, the following message appears - configuration completed. The following is a sample output:

Selection: 17

```
**********************************************************************
           CONFIGURING MULTI CHANNELS OPTION IS CHOSEN.
**********************************************************************

RIB-HOME = /u00/dev01/FRelease_IT15/Rib1410ForAll14xxApps/rib-home

Applications in scope are : [rib-oms, rib-rms, rib-sim, rib-tafr]
Log file location: /u00/dev01/FRelease_
IT15/Rib1410ForAll14xxApps/rib-home/tools-home/rdmt/RDMTLOGS/MultiChannelConfig.lo
g
Enter "return" to terminate this process and return to main menu.
Enabling multi channels needs two inputs from the user. First one is,
"message-flow id" number from "rib-integration-flows" file. This file will be
available at deployment of "rib-func-artifact-<version>.war".
Enter "message-flow id" number:1

You are about to enable multi channels to the following adapters of applications
in scope:
rib-rms.Alloc_pub
rib-rms.Transfers_pub
rib-rwms.StockOrder_sub
rib-sim.StockOrder_sub
rib-tafr.Alloc_tafr
rib-tafr.CustOrder_tafr
rib-tafr.Transfers_tafr

The second input,the user to enter is number of the channels required for the
above entered "message-flow id" number.
Enter count of multi channels:2
The adapter files named "rib-<app>-adapters.xml" and property files named
"rib-<app>-adapters-resources.properties" of
applications in scope will be edited to enable multi channels. Please take a
backup copy of those files.
Do you want to continue? [Y/(N)]:Y
Multi channels configuration completed. Time Taken = 1.516 Seconds.
```

Tool Usage Examples

The following are sets of steps for resolving tool usage concerns.

Ensure RIB is Correctly Installed

Complete the following steps:

1. Using the RDMT Menu system, select the RIB Health SubMenu.
2. Execute RIB Config Report option. This produces the basic report on installation.

   This step scans the installations and configurations of rib applications deployed in WLS. It finally produces a RIB WLS configuration report on the console as well as writes it into a file under the RDMT Temp directory, which contain the status of all the RIB configurations necessary to detect/diagnose any RIB related issues.

3. If you find any discrepancies, refer to the Oracle Retail Integration Bus Installation Guide and follow the steps mentioned there.

Determine Whether the Local WLS is Running

Complete the following steps:

1. Using the RDMT Menu system, select the WLS/JMX Utilities SubMenu.
2. Execute Get WLS Instance State option. It displays the current WLS status.
3. If it is not running, start the local WLS instance. See the Oracle Retail Integration Bus Installation Guide for how to start it.

Determine Where an Issue is Occurring

Complete the following steps:

1. Select RDMT Main Menu.
2. Execute the Scan RIB Logs option. It performs a log scan to look for a /pattern/ ("Exception") in all of the log files in a directory of the currently active WLS instance.
3. Select JMS Topic Scan. Look for topics with messages stuck.

Determine Whether the Adapter Status is Up or Down

Complete the following steps:

1. Select WLS/JMX Utilities Menu.
2. Execute Status ALL Adapters option. It displays the status of all the adapters, namely the publishers, subscribers, hospital and TAFR for the currently active WLS instance.
3. If anything is down, use the Start ALL adapters option and start the same.

Perform a Config/Switch for a New WLS Instance

Complete the following steps:

1. Select WLS/JMX Utilities Menu.
2. Execute WLS/JMX Config Utility option.
3. Provide the desired parameters and configure an instance.
4. You can switch to the desired instance using the same option.

Determine the Subscriber for a Particular JMS Topic

Complete the following steps:

1. Select RDMT Main Menu.
2. Select JMS Utilities Menu.
3. Execute the JMS Topics Subscribe List option.
4. Provide the topic name for which the subscriber name is needed. It provides the same.

5

Deployment Env Info File Editor

The Oracle Retail Integration Bus (RIB) Deployment Configuration File Editor is an application for configuring the rib-deployment-env-info.xml file. The editor simplifies the alteration of the XML file by hiding the raw text form of XML. It provides an interface for adding, removing, and rearranging XML elements. It also provides an interface for editing the contents of elements.

Installation

The tool is located in the RDMT package and installed with RDMT in the <rib-home>/tools-home/RDMT directory. It is available as a menu selection from the ribadmin sub menu.

> **Note:** The editor is a GUI application. To execute it on a host other than the one on which RDMT is installed, use an X server, such as Exceed, and set the DISPLAY environment variable.

Important Installation Warning

All rib-app-builder tools use the rib-deployment-env-info.xml as the single source of truth about the deployment configuration.

All tools use the values in this file. Editing the file directly affects the compilation, configuration, and deployment of the rib-apps. Use extreme caution and understand the ramifications of the values being manipulated.

> **Note:** Fore more information, see the *Oracle Retail Integration Bus Implementation Guide*.

Before editing the source file in rib-home, make a backup of the file and place it securely outside of rib-home. Do not create a backup in the rib-home.

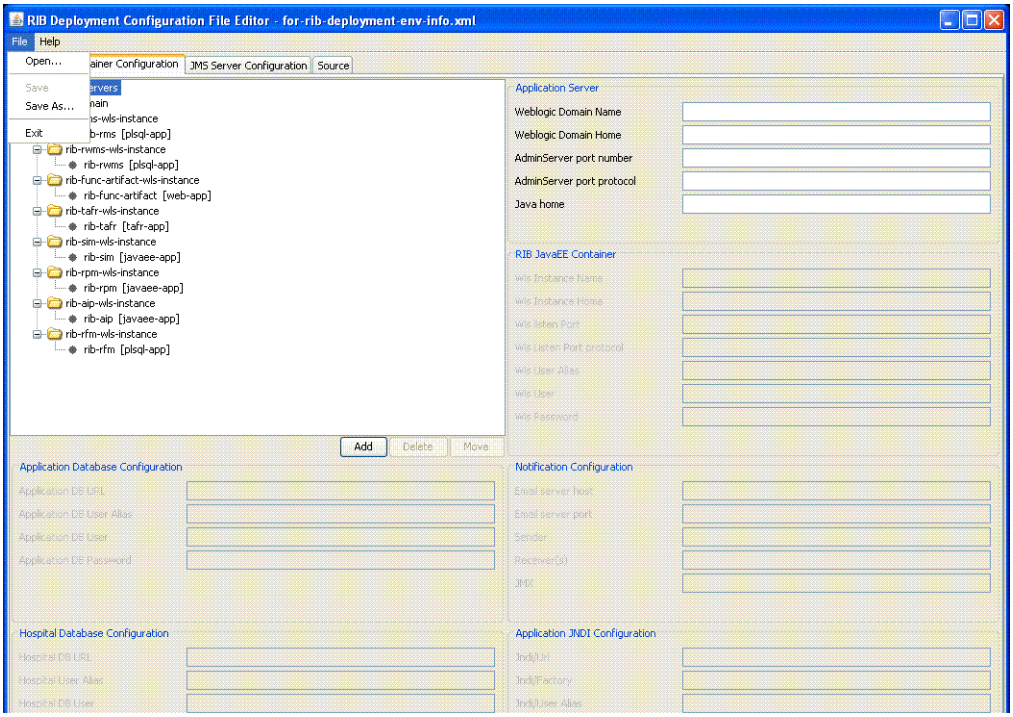
Key Rule

The rib-app-builder tools scan and check versions of all files within rib-home (except for tools-home). The processes do not allow files to have the same name with only an additional extension.

Operation

This section describes the steps you should take to edit the rib-deployment-env-info.xml.

Editing a File

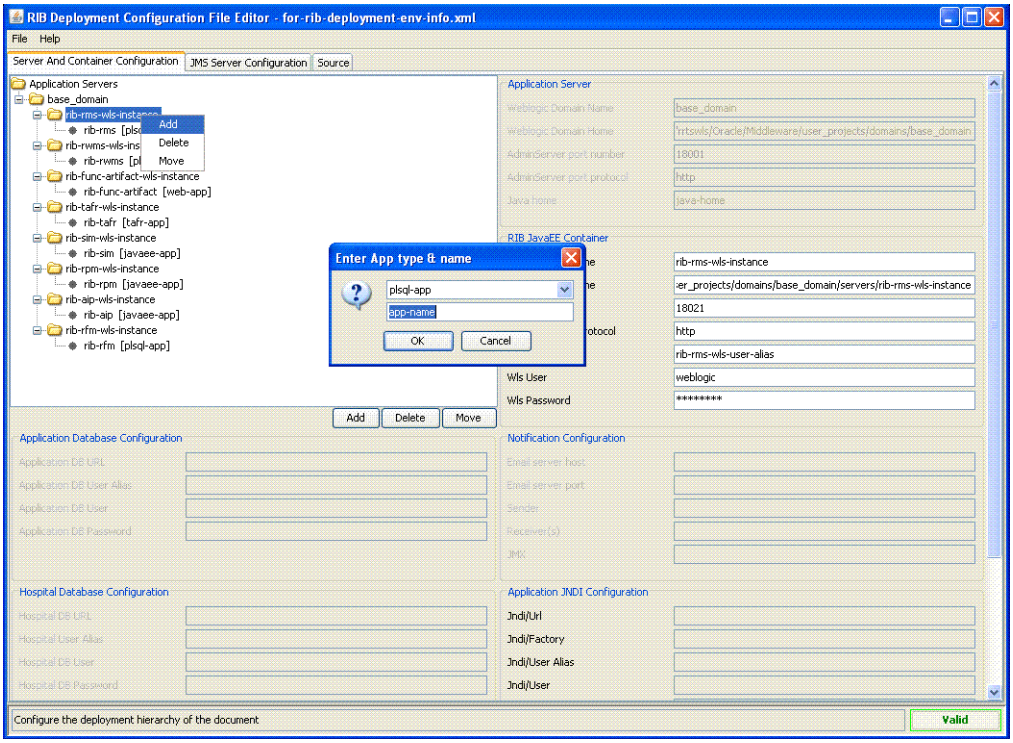


1. From the menu bar, select File.
2. Click **Open**.
3. Navigate to the directory containing rib-deployment-env-info.xml.
4. Select the rib-deployment-env-info.xml file to open it.

Adding an Application

Take the following step to add an application:

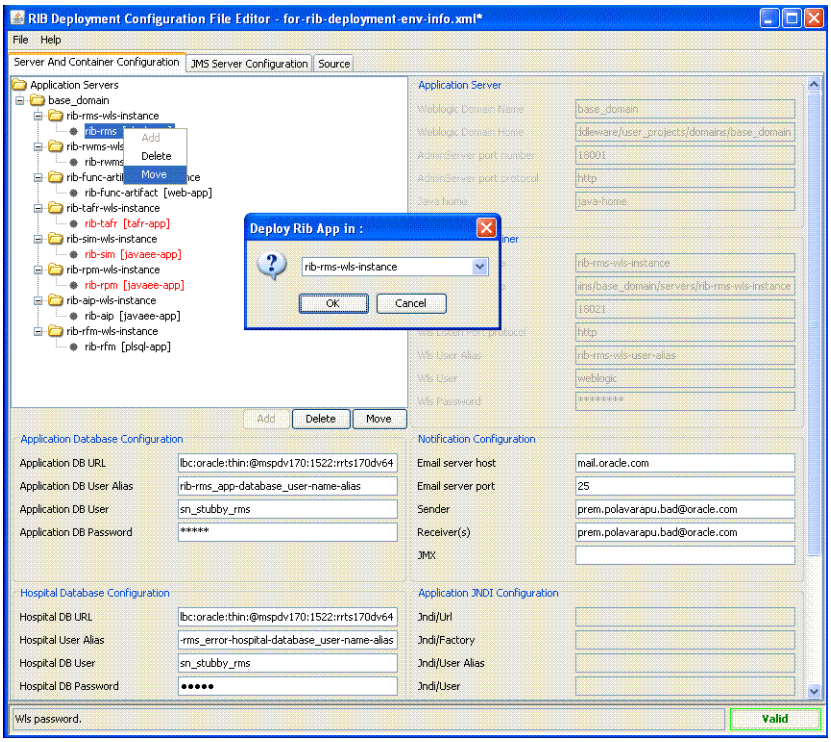
1. Right-click the WLS instance to which you want to add the application. Select **Add**. Alternatively, select the WLS instance to which you want to add the application and click **Add** in the Document window.



Moving an Application

Take the following step to move an application:

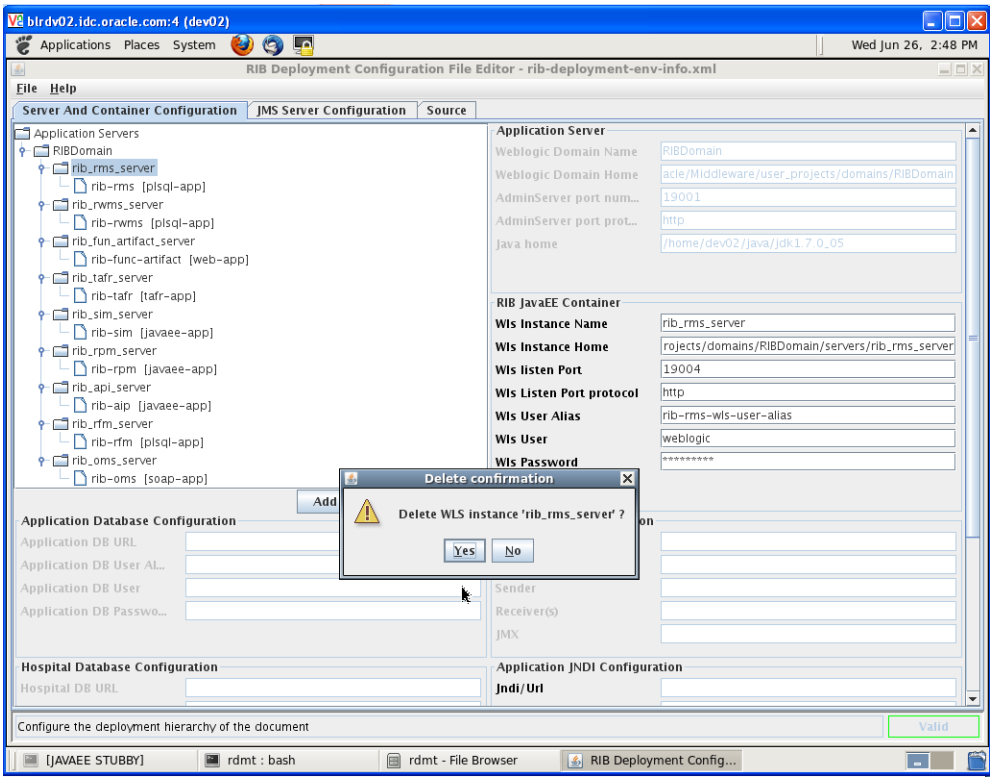
1. Right-click the WLS instance that you want to move. Select **Move**. Alternatively, select the WLS instance you want to move and click **Move** in the Document window.



Deleting an Application

Take the following step to delete an application:

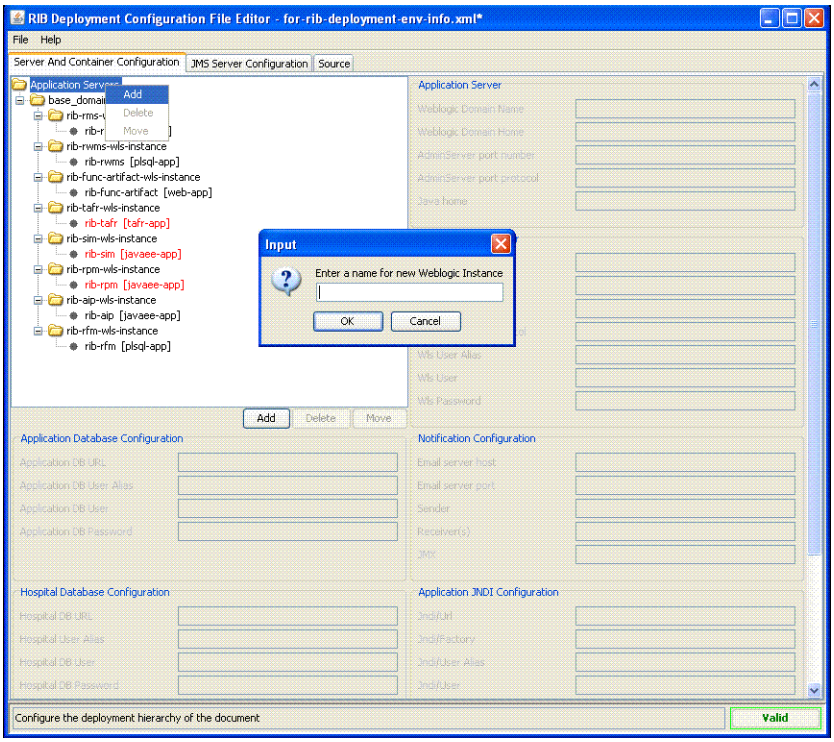
1. Right-click the application you want to delete. Select **Delete**. Click **Yes** to confirm. Alternatively, select the application you want to delete and click Delete in the Document window. Click Yes to confirm.



Adding a WLS Instance

Take the following step to add a WLS instance:

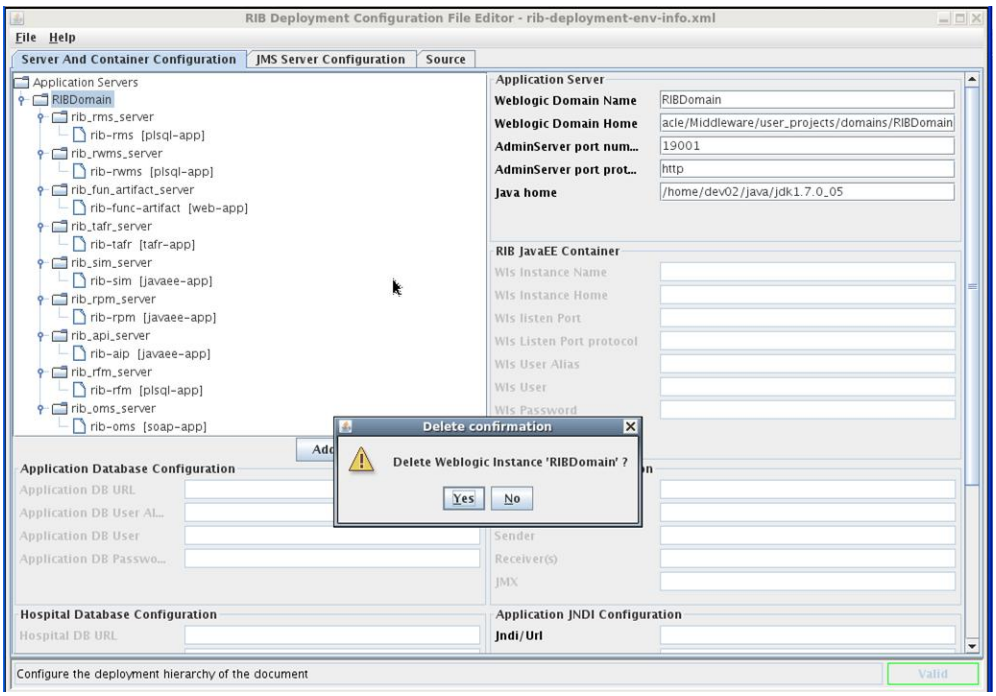
1. Right-click the WLS domain to which you want to add a WLS instance. Select **Add**. Alternatively, select the WLS domain to which you want to add a WLS instance and click **Add** in the Document window.



Deleting a WLS Instance

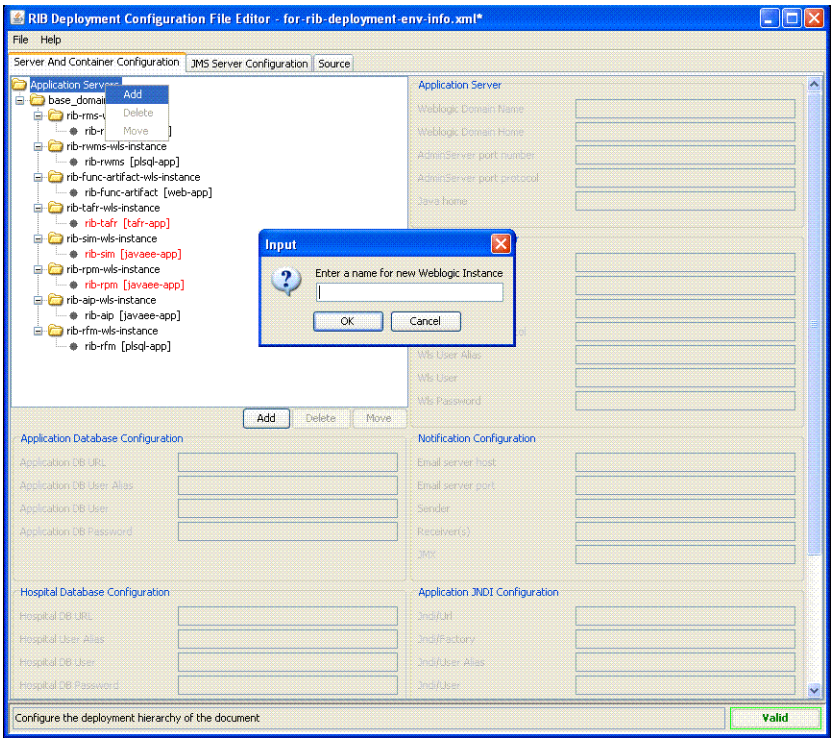
Take the following step to delete a WLS instance:

1. Right-click the WLS instance you want to delete. Select Delete and click Yes to confirm. Alterbatively, select the WLS instance you want to delete and click Delete in the Document window. Click on Yes to confirm.



Adding an Application Server Instance

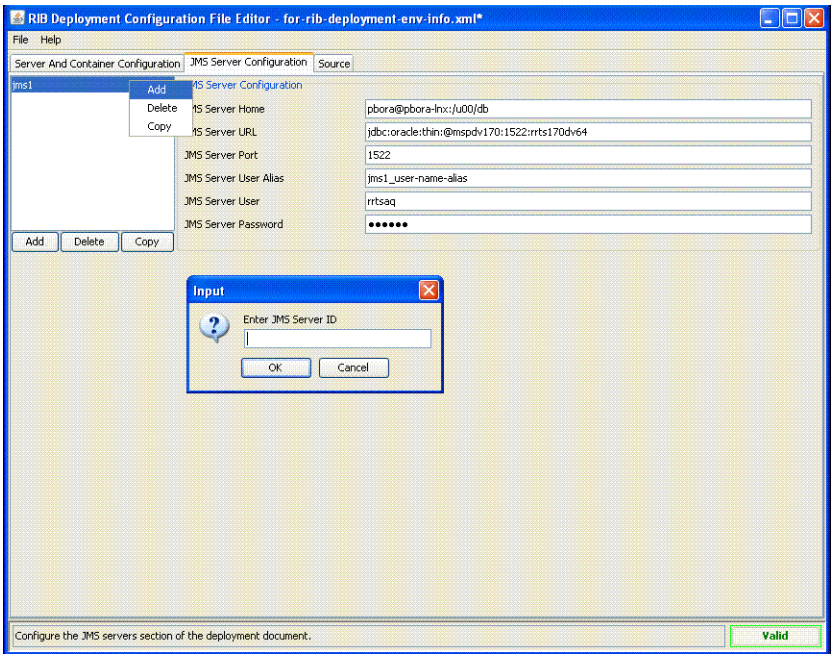
Take the following step to add an application server instance:



1. Right-click on the Application Servers node. Select **Add**. Alternatively, select the application servers node and click **Add** in the Document window.

Adding a JMS Server Instance

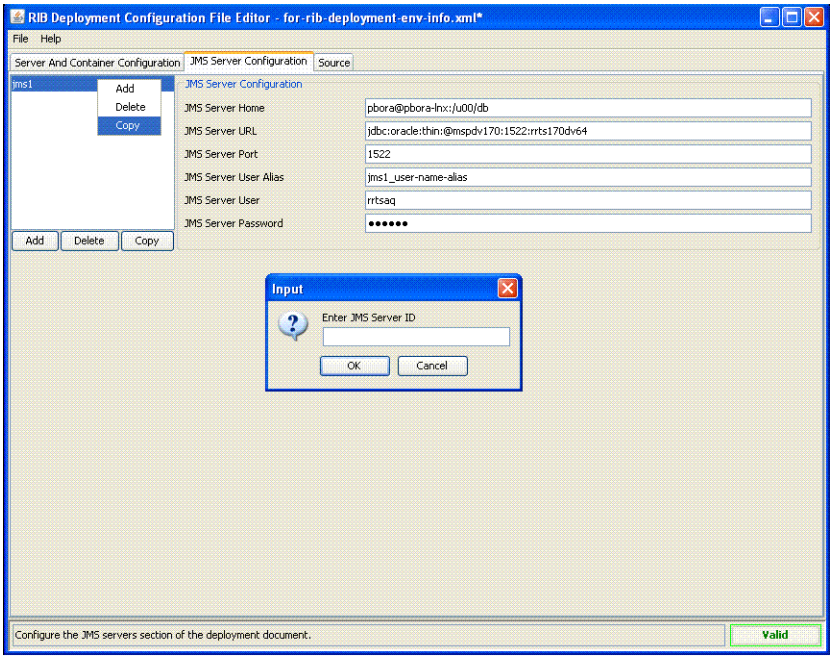
Take the following step to add a JMS server instance:



1. Right-click in the JMS Servers Configuration Panel. Select **Add**. Alternatively, click the **Add** button in the Document window.

Copying a JMS Server Instance

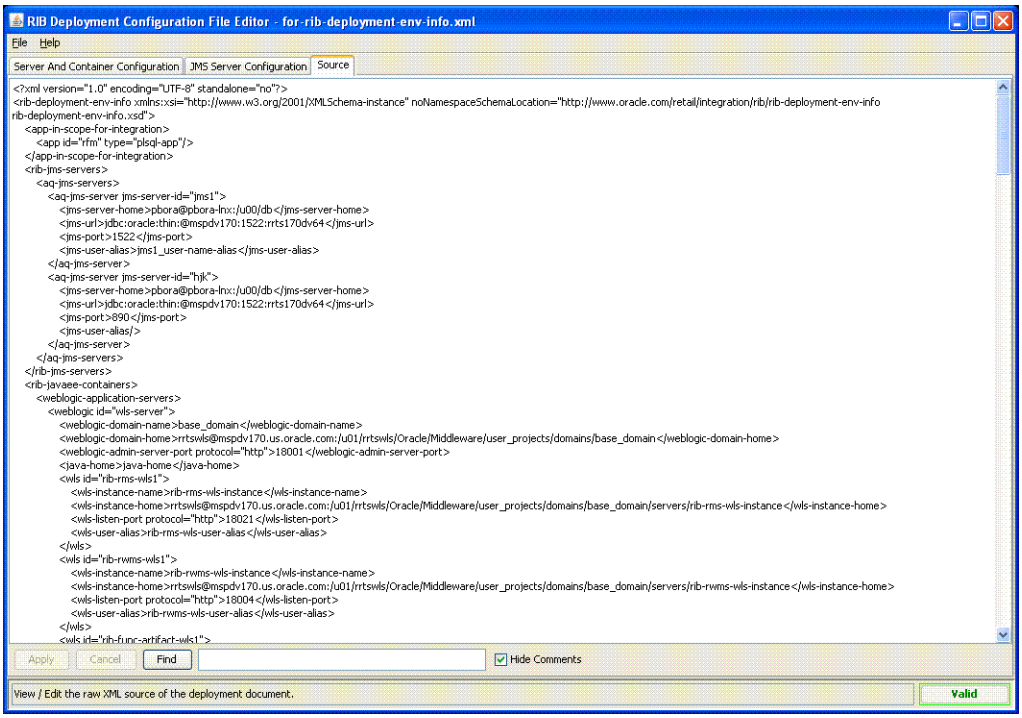
Take the following step to copy a JMS server instance:



1. Right-click in the JMS Servers Configuration Panel. Select **Copy**. Alternatively, click **Copy** in the Document window.

Viewing the XML Source File

The source file containing the raw text of the XML document also can be viewed and edited from the Source tab. The following is an illustration of the source file editor interface:


RIB Deployment Configuration File Editor - for-rib-deployment-env-info.xml
File Help
Server And Container Configuration
JMS Server Configuration
Source
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<rib-deployment-env-info xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" noNamespaceSchemaLocation="http://www.oracle.com/retail/integration/rib/rib-deployment-env-info
rib-deployment-env-info.xsd">
<app-in-scope-for-integration>
<app id="rfm" type="plsql-app"/>
</app-in-scope-for-integration>
<rib-jms-servers>
<aq-jms-servers>
<aq-jms-server jms-server-id="jms1">
<jms-server-home>pbora@pbora-lnx:/u00/db</jms-server-home>
<jms-url>jdbc:oracle:thin:@mspdv170:1522:rrts170dv64</jms-url>
<jms-port>1522</jms-port>
<jms-user-alias>jms1_user-name-alias</jms-user-alias>
</aq-jms-server>
<aq-jms-server jms-server-id="hjk">
<jms-server-home>pbora@pbora-lnx:/u00/db</jms-server-home>
<jms-url>jdbc:oracle:thin:@mspdv170:1522:rrts170dv64</jms-url>
<jms-port>890</jms-port>
<jms-user-alias/>
</aq-jms-server>
</aq-jms-servers>
</rib-jms-servers>
<rib-javaee-containers>
<weblogic-application-servers>
<weblogic id="wls-server">
<weblogic-domain-name>base_domain</weblogic-domain-name>
<weblogic-domain-home>rrtswls@mspdv170.us.oracle.com:/u01/rrtswls/Oracle/Middleware/user_projects/domains/base_domain</weblogic-domain-home>
<weblogic-admin-server-port protocol="http">18001</weblogic-admin-server-port>
<java-home>java-home</java-home>
<wls id="rib-rms-wls1">
<wls-instance-name>rib-rms-wls-instance</wls-instance-name>
<wls-instance-home>rrtswls@mspdv170.us.oracle.com:/u01/rrtswls/Oracle/Middleware/user_projects/domains/base_domain/servers/rib-rms-wls-instance</wls-instance-home>
<wls-listen-port protocol="http">18021</wls-listen-port>
<wls-user-alias>rib-rms-wls-user-alias</wls-user-alias>
</wls>
<wls id="rib-rwms-wls1">
<wls-instance-name>rib-rwms-wls-instance</wls-instance-name>
<wls-instance-home>rrtswls@mspdv170.us.oracle.com:/u01/rrtswls/Oracle/Middleware/user_projects/domains/base_domain/servers/rib-rwms-wls-instance</wls-instance-home>
<wls-listen-port protocol="http">18004</wls-listen-port>
<wls-user-alias>rib-rwms-wls-user-alias</wls-user-alias>
</wls>
Apply
Cancel
Find
Hide Comments
View / Edit the raw XML source of the deployment document.
Valid

Glossary

RIB

Retail Integration Bus

RDMT

RIB Diagnostic and Monitoring Tool

JAVAEE

Java Enterprise Edition

XML

Extended Markup Language

JMS

Java Messaging Service

WLS

Weblogic Server

EJB

Enterprise Java Bean

TAFR

Transforming Addressing Filtering Routing