Retek® Integration Bus 10.3

Implementation Guide

Retek

The software described in this documentation is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

Information in this documentation is subject to change without notice.

Retek provides product documentation in a read-only-format to ensure content integrity. Retek Customer Support cannot support documentation that has been changed without Retek authorization.

Corporate Headquarters:

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

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

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 e-mail, 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.

Contact Method	Contact Information
Internet (ROCS)	www.retek.com/support Retek's secure client Web site to update and view issues
E-mail	support@retek.com
Phone	US & Canada: 1-800-61-RETEK (1-800-617-3835) World: +1 612-587-5800 EMEA: 011 44 1223 703 444 Asia Pacific: 61 425 792 927
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 – Application implementation 1

Database installation 1

Application installation 1

Chapter 2 – RIB implementation 9

Chapter 1 – Application implementation

Database installation

It is very important that the database is installed correctly before any of the applications are installed. Verify that the following Oracle 9i packages have been installed as "SYS" and privileges have been given to the appropriate user.

- Oracle XA – needed for two phase commits
- Oracle XDK – The XML Development Kit is needed for parsing, creating and/or validating XML. Grants need to be issued to the schema owner of RMS, RDM and RCOM and execute privileges to the RIB user.
- The Oracle Java Virtual Machine (JVM) is installed and initialized. The script initjvm.sql may need to be run if a "manual" install was performed.

Application installation

Install each application according to the installation guide.

Seed data

Make sure that the Seed data is implemented. Verify that you are populating this seed data the same way across all three applications. For example, you need to make sure that all three applications are using either a three digit or a two digit country code. They can not be mixed or the applications will not be able to communicate.

Verify the RIB_DOC_TYPES tables

Verify that the rib_doc_types tables are populated with the data found in the following data files. This table is used by certain publishing Message Family Managers[1]. Loading this table should be part of the RMS/RCOM/RDM/RLM installation.

- RMS - rib_doctypes_rms.ctl, rib_doctypes_rms.dat
- RCOM - rib_doctypes_rcom.ctl, rib_doctypes_rcom.dat
- RDM - rib_doctypes_rdm.ctl, rib_doctypes_rdm.dat
- RLM – rib_doctypes_rlm.ctl, rib_doctypes_rlm.dat

[1] A Publishing Message Family Manager is an Oracle package containing stored procedures used by an application trigger and the SeeBeyond e*Way to publish a message to the RIB. The packages contains at least two stored procedures: ADDTOQ () – called by the trigger, and GETNXT() – called by the e*Way.

The following Rib_doc_types entries should exist for RCOM, RDM, RLM and RMS.

- VendorAddrDesc
- VendorAddrRef
- VendorDesc
- VendorHdrDesc
- VendorRef

Verify triggers

Verify that all the appropriate application triggers are enabled. For more information on the specific triggers, look at the application specific operations guide.

Verify hospital tables

Verify that the Error Hospital tables been created for each Error Hospital instance. The following tables should exist for each instance of the hospital.

- RIB_MESSAGE
- RIB_MESSAGE_FAILURE
- RIB_MESSAGE_ROUTING_INFO

Verify data in publication system

Before we turn on the RIB, make sure that the Publication system is working correctly. Go through the following steps:

- Key data into Publication system through GUI. (In this example, enter a vendor.)
- Verify data is populated in the staging table or queue. (In this example. SUPPLIER_MFQUEUE is the name of the staging table.)

Verify the Publication System Message Family Manager (MFM)

Verify that the Publication system application is able to publish using the GETNXT within the Message Family Manager. You may have to write a quick script via SQL PLUS to accomplish this. For vendor the RMS MFM package name is RMSMFM_SUPPLIER. You should not move beyond this step until the GETNXT returns a status_code of 'S' for success and you are able to look at the XML message. Once this is successful, make sure to ROLLBACK this transaction. Also, save this XML message for testing the destination system.

A sample SQLPLUS script is seen below:

```
set serveroutput on

declare

    l_status_code  varchar2(1);

    l_error_text   varchar2(32767);

    l_message_clob clob;

    l_message_type varchar2(255);

    l_supplier     number;

    l_addr_seq_no  number;

    l_addr_type    varchar2(2);

begin

    rmsmfm_supplier.getnxt( l_status_code, l_error_text,
l_message_type, l_message_clob

                      , l_supplier, l_addr_seq_no,
l_addr_type);

    dbms_output.put_line('l_status_code is '
||l_status_code);

    dbms_output.put_line('l_error_text is '
||l_error_text);

    dbms_output.put_line('l_message type is
'||l_message_type);

   rollback;

end;

              /
```

Verify the Subscription System APIs.

Verify that the Subscription system can subscribe to the message that was published in the above example. Once again a script will have to be written to accomplish this using the consume procedure. For this example we will only use the create APIs for RCOM, RDM and RLM. The API package names are as follows: RCOMSUB_VENDORCRE and RDMSUB_VENDORCRE. You should not move beyond this step until the CONSUME returns a status_code of 'S' for success. Once this is successful, make sure to ROLLBACK this transaction.

A sample PL/SQL script is seen below that uses the RDMSUB_VENDORCRE.CONSUME() stored procedure. Errors running this script may arise from:

- the script assumes that a Facility ID of 'PROD' has been created (used on the call to RDMSUB_CRE.CONSUME())
- the script uses seed data with the country id of USA and state of WI exist.

```
set serveroutput on size 20000

declare

l_x1 varchar2(200) := '<!DOCTYPE VendorDesc SYSTEM
"http://www.retek.com/dtd/rib/VendorDesc.dtd">';

l_x2 varchar2(200) := '<VendorDesc> <VendorHdrDesc>
<supplier>3333333333</supplier> <sup_name>FIF Supplier
TC01</sup_name>';

l_x3 varchar2(200) := '<contact_name>Joe Blow</contact_name>
<contact_phone>111-111-1111</contact_phone> <contact_fax>222-
222-2222</contact_fax>';

l_x4 varchar2(200) := '<contact_pager>333-333-
3333</contact_pager> <sup_status>A</sup_status>
<qc_ind>Y</qc_ind>';

l_x5 varchar2(200) := '<qc_pct>10</qc_pct> <qc_freq>1</qc_freq>
<vc_ind>Y</vc_ind> <vc_pct>80</vc_pct> <vc_freq>5</vc_freq>
<currency_code>CAD</currency_code>';

l_x6 varchar2(200) := '<lang>1</lang> <terms>02</terms>
<freight_terms>03</freight_terms>
<ret_allow_ind>N</ret_allow_ind>
<ret_auth_req>N</ret_auth_req>';

l_x7 varchar2(200) := '<ret_min_dol_amt/>
<ret_courier>FedX</ret_courier> <handling_pct>10</handling_pct>
<edi_po_ind>N</edi_po_ind> <edi_po_chg>N</edi_po_chg>';

l_x8 varchar2(200) := '<edi_po_confirm>N</edi_po_confirm>
<edi_asn>N</edi_asn> <edi_sales_rpt_freq>D</edi_sales_rpt_freq>
<edi_supp_available_ind>N</edi_supp_available_ind>';

l_x9 varchar2(200) := '<edi_contract_ind>N</edi_contract_ind>
<edi_invc_ind>N</edi_invc_ind>
<cost_chg_pct_var>2</cost_chg_pct_var> <cost_chg_amt_var/>';

l_x10 varchar2(200) :=
'<replen_approval_ind>N</replen_approval_ind> <ship_method/>
<payment_method/> <contact_telex/>
<contact_email>testemail.net</contact_email>';

l_x11 varchar2(200) := '<settlement_code>E</settlement_code>
<pre_mark_ind>N</pre_mark_ind>
<auto_appr_invc_ind>N</auto_appr_invc_ind> <dbt_memo_code/>';

l_x12 varchar2(200) :=
'<freight_charge_ind>N</freight_charge_ind>
<auto_appr_dbt_memo_ind>N</auto_appr_dbt_memo_ind>
<prepay_invc_ind>N</prepay_invc_ind>';
```

```
l_x13 varchar2(200) := '<backorder_ind>N</backorder_ind>
<vat_region/> <inv_mgmt_lvl>D</inv_mgmt_lvl>
<service_perf_req_ind>N</service_perf_req_ind>';

l_x14 varchar2(200) := '<invc_pay_loc/> <invc_receive_loc/>
<addinvc_gross_net>N</addinvc_gross_net>
<delivery_policy>NEXT</delivery_policy>';

l_x15 varchar2(200) := '<comment_desc>this is the comment
field</comment_desc> <default_item_lead_time/> <duns_number/>
<duns_loc/>';

l_x16 varchar2(200) :=
'<bracket_costing_ind>N</bracket_costing_ind>
<vmi_order_status/> </VendorHdrDesc> <VendorAddrDesc>
<module>SUPP</module>';

l_x17 varchar2(200) := '<key_value_1>3333333333</key_value_1>
<key_value_2/> <seq_no>1</seq_no> <addr_type>01</addr_type>
<primary_addr_ind>Y</primary_addr_ind>';

l_x18 varchar2(200) := '<add_1>123 Business Forest Lane</add_1>
<add_2>Suite 8-6</add_2> <add_3>Line 3</add_3> <city>New
City</city> <state>WI</state>';

l_x19 varchar2(200) := '<country_id>USA</country_id>
<post>55555</post> <contact_name>Jane Doe</contact_name>
<contact_phone>8888888888</contact_phone>';

l_x20 varchar2(200) :=
'<contact_telex>4444444444</contact_telex> <contact_fax>666-666-
6666</contact_fax>
<contact_email>jdoe@headhoncho.net</contact_email>';

l_x21 varchar2(200) :=
'<oracle_vendor_site_id>123456789</oracle_vendor_site_id>
</VendorAddrDesc> <VendorAddrDesc> <module>SUPP</module>';

l_x22 varchar2(200) := '<key_value_1>3333333333</key_value_1>
<key_value_2/> <seq_no>1</seq_no> <addr_type>03</addr_type>
<primary_addr_ind>Y</primary_addr_ind>';

l_x23 varchar2(200) := '<add_1>444 Return Pile Alley</add_1>
<add_2>60 Alley Cat Tower</add_2> <add_3/> <city>Hudson</city>
<state>WI</state> <country_id>USA</country_id>';

l_x24 varchar2(200) := '<post>33333</post>
<contact_name>KittyKat</contact_name> <contact_phone>444-444-
4444</contact_phone> <contact_telex>555-555-
5555</contact_telex>';

l_x25 varchar2(200) := '<contact_fax>333-333-3333</contact_fax>
<contact_email>kk@junkyard.net</contact_email>
<oracle_vendor_site_id>222233334444555</oracle_vendor_site_id>';

l_x26 varchar2(200) := '</VendorAddrDesc> <VendorAddrDesc>
<module>SUPP</module> <key_value_1>3333333333</key_value_1>
<key_value_2/> <seq_no>1</seq_no>';

l_x27 varchar2(200) := '<addr_type>04</addr_type>
<primary_addr_ind>Y</primary_addr_ind> <add_1>444 Order Pile
Alley</add_1> <add_2>60 Alley Cat Tower</add_2> <add_3/>';
```

```
l_x28 varchar2(200) := '<city>Hudson</city> <state>WI</state>
<country_id>USA</country_id> <post>33333</post>
<contact_name>Kitty Kat</contact_name>';

l_x29 varchar2(200) := '<contact_phone>444-444-
4444</contact_phone> <contact_telex>555-555-5555</contact_telex>
<contact_fax>333-333-3333</contact_fax>';

l_x30 varchar2(200) :=
'<contact_email>kk@junkyard.net</contact_email>
<oracle_vendor_site_id>222233334444555</oracle_vendor_site_id>
</VendorAddrDesc> </VendorDesc>';


l_clob clob;


l_status_code   varchar2(2);

l_error_message varchar2(32767);

begin


        dbms_lob.createtemporary(l_clob, false);

        dbms_lob.writeappend(l_clob, length(l_x1), l_x1);

        dbms_lob.writeappend(l_clob, length(l_x2), l_x2);

        dbms_lob.writeappend(l_clob, length(l_x3), l_x3);

        dbms_lob.writeappend(l_clob, length(l_x4), l_x4);

        dbms_lob.writeappend(l_clob, length(l_x5), l_x5);

        dbms_lob.writeappend(l_clob, length(l_x6), l_x6);

        dbms_lob.writeappend(l_clob, length(l_x7), l_x7);

        dbms_lob.writeappend(l_clob, length(l_x8), l_x8);

        dbms_lob.writeappend(l_clob, length(l_x9), l_x9);

        dbms_lob.writeappend(l_clob, length(l_x10), l_x10);

        dbms_lob.writeappend(l_clob, length(l_x11), l_x11);

        dbms_lob.writeappend(l_clob, length(l_x12), l_x12);

        dbms_lob.writeappend(l_clob, length(l_x13), l_x13);

        dbms_lob.writeappend(l_clob, length(l_x14), l_x14);

        dbms_lob.writeappend(l_clob, length(l_x15), l_x15);

        dbms_lob.writeappend(l_clob, length(l_x16), l_x16);

        dbms_lob.writeappend(l_clob, length(l_x17), l_x17);

        dbms_lob.writeappend(l_clob, length(l_x18), l_x18);

        dbms_lob.writeappend(l_clob, length(l_x19), l_x19);

        dbms_lob.writeappend(l_clob, length(l_x20), l_x20);

        dbms_lob.writeappend(l_clob, length(l_x21), l_x21);

        dbms_lob.writeappend(l_clob, length(l_x22), l_x22);
```

```
        dbms_lob.writeappend(l_clob, length(l_x23), l_x23);
        dbms_lob.writeappend(l_clob, length(l_x24), l_x24);
        dbms_lob.writeappend(l_clob, length(l_x25), l_x25);
        dbms_lob.writeappend(l_clob, length(l_x26), l_x26);
        dbms_lob.writeappend(l_clob, length(l_x27), l_x27);
        dbms_lob.writeappend(l_clob, length(l_x28), l_x28);
        dbms_lob.writeappend(l_clob, length(l_x29), l_x29);
        dbms_lob.writeappend(l_clob, length(l_x30), l_x30);


        rdmsub_vendorcre.consume( l_status_code,
l_error_message, l_clob, 'PROD');


        dbms_output.put_line('rdmsub_vendorcre.consume status is
: '||l_status_code);
        dbms_output.put_line('rdmsub_vendorcre.consume
l_error_message is : '||l_error_message);
        dbms_lob.freetemporary(l_clob);
        rollback;


end;
/
```

.

Chapter 2 – RIB implementation

Install SeeBeyond

Make sure to install SeeBeyond under a unique user name such as ‘egate’.

- Verify that you have installed all the appropriate ADD-ONS:
  - Batch e*Way
  - Oracle e*Way
- Verify that you have installed all ESRs associated with this release:
  - Esr54082 – note this patch $TEMP will need to be exported in unix to run the UpdateEsrLog.sh script.
  - Esr54082

Install the RIB

Using the RIB Installation Guide provided, alter the egate_profile and run the install. After the install is complete, verify the registry is running and that you can logon to the RIB schema with the e*Gate Enterprise Manager using the user name: Administrator Password: STC.

Set up SeeBeyond user accounts

Once the e*Gate Enterprise Manager application is running, make sure that you have set up user accounts for all staff that will be using the SeeBeyond products. Once these users have been created, signoff and use your user account.

Verify the RIB configuration

If you followed the RIB Installation Guide, at this point mostly everything should be configured. However, verify that the following have been completed.

- Verify that the configuration files for the eways (ewayname.cfg) has right JNI DLL absolute path for example Solaris OS should have libjvm.so and AIX libjvm.a, etc. If the configuration files for the eways are edited, make sure that these are promoted to run time.
- Verify that each Connection Point has been configured correctly. Remember to hit <enter> after each entry made to these configuration files. If this does not occur, the change will not happen.

- The following are the list that needs to be updated
  - RMS – cpToAndFromRMS

    cpHospitalRMS
  - RCOM – cpToAndFromRCOM

    cpHospitalRCOM
  - RDM – cpToAndFromRDM

    CpHospitalRDM
  - RLM – cpToAndFromRLM

    cpHospitalRLM
- Verify that you JMS connection Points are pointing at the same port as your JMS queue manager.
- Verify that you only have one JMS Queue Manager. If multiple exist, delete them. The VALID one should be named ***'iqmJMS'***.
- Verify the Rib.properties file is configured according to Installation and Operations Guides.

Note: The facility_type.default property must equal the facility type within the **RDM transshipment_setup table (i.e. PROD).**

Note: The facility_id.<facility_type>.<location_id> properties exist for every DC_Dest_ID within RDM that is going to be communicating with the RIB. (location_id = dc_dest_id)

Start up the RIB

The following steps will walk you through starting the RIB. If at any point you are unable to accomplish one of the steps, restart the e*Way after turning on the following logging levels to look for errors. TRACE Level with the following checked DB, DBV, COL, COLV, MSG, MSGV, EWY, EWYV. Also remember to check the Use Log File box. Once the error has been identified, restart the e*Ways with logging turned off again.

To start the SeeBeyond registry and control broker, use the appropriate script (example: start_egate) and plist to verify that the process is running. If there is an error message like "Unable to open/create default schema" make sure that the Myschema.rdb, default.rdb, RIB100.rdb file has read and write permissions.

Start up the SeeBeyond registry

- Verify that the registry is running. If it is not, use the Operations Guide to assist you in starting it. This must be started using the Administrator username and password. A script is also provided to help you start the registry (i.e. start_egate).

Start up the SeeBeyond control broker

- Verify that the Control Broker is running. If it is not, use the Operations Guide to assist you in starting it. This must be started using the Administrator username and password. A script is also provided to help you start the registry (i.e. start_cb).

Note: If there is an error message like "Unable to open/create default schema" make sure that the Myschema.rdb, default.rdb, RIB100.rdb file has read and write permissions.

Start up the JMS Manager

- You are now ready to start the JMS Queue Manager. Logon to the SeeBeyond Monitor using your user account. Only start the JMS manager labeled ***'iqmJMS'***. If this component starts successfully, it will no longer be RED.

Start up the publishing e*Way

- You are now ready to start the appropriate publishing e*Way from the SeeBeyond Monitor. For this example we will be starting the publishing e*Way named, *ewVendorFromRMS.* If this component starts successfully, it will no longer be RED.

Verify data has been published

- You can verify this in two ways. One is to verify that the data no longer exists in the publishing queue mentioned in a previous step. The second is to look in the JMS Administrator GUI, which can be accessed from the SeeBeyond Monitor. At this point, it should show the topic messages (i.e. In this example, etVendorFromRMS) as 'uncommitted' or white.

Start up the RCOM Subscribing e*Way

- You are now ready to start the appropriate subscribing e*Way from the SeeBeyond Monitor. For this example, we will be starting the publishing e*Way named, *ewVendorToRCOM.* If this component starts successfully, it will no longer be RED.

Verify data has been subscribed to by RCOM

- You can verify this in two ways. One is to look in the JMS Administrator GUI, which can be accessed from the SeeBeyond Monitor. At this point, it should show the topic messages (i.e. In this example, etVendorFromRMS) as 'committed' or blue under the collaboration that you are testing. Or you can verify that the data has been committed in the application.

Start up the RDMWH1 Subscribing e*Way

- You are now ready to start the appropriate subscribing e*Way from the SeeBeyond Monitor. For this example we will be starting the publishing e*Way named, *ewVendorToRDMWH1*. If this component starts successfully, it will no longer be RED.

Verify data has been subscribed to by RDMWH1

- You can verify this in two ways. One is to look in the JMS Administrator GUI, which can be accessed from the SeeBeyond Monitor. At this point, it should show the topic messages (In this example, etVendorFromRMS) as 'committed' or blue under the collaboration that you are testing. Or you can verify that the data has been committed in the application.

Start up the RLMEmployee Subscribing e*Way

- You are now ready to start the appropriate subscribing e*Way from the SeeBeyond Monitor. For this example we will be starting the publishing e*Way named, *ewEmployeeToRLM*. If this component starts successfully, it will no longer be RED.

Verify data has been subscribed to by RLMEmployee

- You can verify this in two ways. One is two look in the JMS Administrator GUI, which can be accessed from the SeeBeyond Monitor. At this point, it should show the topic messages (i.e. In this example, etEXTEmp) as 'committed' or blue under the collaboration that you are testing. Or you can verify that the data has been committed in the application.

Startup of remaining interfaces

At this point Vendor has been successful and each e*Way should be ready to go. If any problems arise with any specific e*Way verify the following things:

- Verify that the connection points with the Collaboration Publication and Subscription metadata is set up and is pointing at the correct application.
- Verify that the rib_doc_types have been loaded for the message type that is creating the error.