Oracle® Healthcare Master Person Index

Real-time Loader User's Guide

Release 2.0.13

E78168-01

August 2016

This document is intended for users who need to get moderate-sized cleansed data into an in-production MPI database.

This document contains the following sections:

- Overview on page 1-1
- Generating Real-time Loader on page 1-1
- Configuring Real-time Loader on page 1-2
- Running Real-time Loader on page 1-4
- Comparing Initial Bulk Matcher and Loader with Real-time Loader on page 1-4
- Configuring deduplicateSystemObject API on page 1-5
- Configuring Real-time Loader Post Processor on page 1-6
- Running Sample Post Processor on page 1-8

Overview

The Real-time Loader (RTL) is an OHMPI data loader that can be used to load both initial data and in-production data into an MPI system. It is a tool that can be easily generated and used for loading data from an input file without the need to build a custom loader.

RTL uses the same input file format as the Initial Bulk Matcher and Loader (IBML). You have to generate only one type of data files that can be used by both IBML and RTL. The same data cleansing tool used for IBML can also be used for RTL. For information on how to cleanse data before loading by using the Real-time Loader, see *Oracle Healthcare Master Person Index Analyzing and Cleansing User's Guide*.

You can use RTL for loading initial data or later on to add large sets of records (for example, daily batch load, or loading records from new source systems after hospital acquisition). RTL is not meant to replace IBML as RTL is not designed for highly performant initial bulk matching and loading situations. It is meant to provide you with a real-time load tool that can be used for moderate-size data files.

For more information about IBML, see *Oracle Healthcare Master Person Index Loading the Initial Data Set User's Guide*.

Generating Real-time Loader

To generate Real-time Loader, perform the following:

1. Right-click the MPI Project node in NetBeans, and select **Generate Loader Zip**.
2. Unzip the `generated loader.zip` file.

RTL is generated simultaneously with IBML. The Real-time Loader's batch file is `run-realtime-loader.bat` and script file is `run-realtime-loader.sh`.

Configuring Real-time Loader

To configure RTL, edit the following properties in the `run-realtime-loader.bat` (or `run-realtime-loader.sh`) file:

```
set DATA_FILE=InputData.txt
set USER_NAME=ohmpi
set PASSWORD=ohmpi1234
set APP_NAME=Person
set APP_SERVER_HOME=D:/Oracle/Middleware/wlserver_10.3/server
set JNDI_URL=t3://localhost:7001
set MAX_NUM_THREADS=10
set MAX_TASK_SIZE=100
set FILE_ENCODING=
set FAIL_FAST=false
set VERBOSE=true
set EXECUTE_MATCH_API=executeMatchUpdate
set MAX_RETRY_NUMBER=5
set MIN_RETRY_DELAY=5
set MAX_RETRY_DELAY=15
```

Property Name	Property Description
DATA_FILE	Semicolon-delimited names of input data files and/or directories. For example, `DATA_FILE=input1.dat;input2.dat;dir1`. For the format of input data file, see *Oracle Healthcare Master Person Index Loading the Initial Data Set User's Guide*. When multiple input data files and/or directories are specified, they are processed in the specified order sequentially, that is, the next data file does not get started until the previous data file has been finished. This makes it easy for you to keep track of the current loading status. For each data file, it can be loaded in parallel by multiple threads though (see MAX_NUM_ THREADS and MAX_TASK_SIZE). Files in specified directories are processed in alphabetical order.
USER_NAME	Name of the user to load the data. This user account must have already been properly configured in the application server. For information on how to create MPI users, see *Oracle Healthcare Master Person Index User's Guide*.
PASSWORD	The corresponding password for USER_NAME.
APP_NAME	MPI Project's application name. The MPI Project's application name can be different from the MPI Project's project name or the MPI Object Model's primary object name. The loader generator should generate this value properly.
APP_SERVER_HOME	Application server installation directory.

Property Name	Property Description
JNDI_URL	Client JNDI URL, or Java Naming Provider URL. This property replaces the two old properties, HOST_NAME and IIOP_PORT, in OHMPI 2.* releases for better clustering support. ■ For a regular non-clustered environment, the JNDI URL must be in the following format: *t3://<HOST_NAME>:<IIOP_PORT>* HOST_NAME is the name or IP address of the host, where the MPI Project is deployed, and IIOP_PORT is the IIOP listener port number on HOST_NAME. For example, *t3://localhost:7001* ■ For a clustered environment, the JNDI URL must be in the following format: *t3://<HOST_NAME_1>:<IIOP_PORT_1>,<HOST_NAME_2>:<IIOP_PORT_2>,...* HOST_NAME_N is the name or IP address of a node in the cluster and IIOP_PORT_N is the corresponding IIOP listener port number on HOST_NAME_N. For example, *t3://slc06lhl:8801,slc06lhl:8802,slc06lhl:8803,slc06lhl:8804*
MAX_NUM_THREADS	Maximum number of concurrent threads for parallel data loading (of a single input file). The actual number of concurrent threads is also determined by input data file size and MAX_TASK_SIZE.
MAX_TASK_SIZE	Maximum number of input records in each parallel loading task. This configuration divides an input file into multiple loading tasks, so that multiple threads can work on the same input file at the same time. If you have an input file of 20000 records, specifying 1000 as MAX_TASK_SIZE will divide the input file into 20 tasks, with 1000 records in each task. These 20 tasks can be performed by a single thread or by up to 20 threads at a time. In this case, specifying a MAX_NUM_THREADS larger than 20 is the same as specifying the MAX_NUM_THREADS to be 20.
FILE_ENCODING	Encoding of the input data file. For example, UTF8, GBK, and so on. To use the default system encoding, leave it empty.
FAIL_FAST	Whether to fail fast when some problem is encountered during data loading (for example, on mal-formatted input record). The valid options are *true* and *false*. For input records that are failed in loading by the Real-time Loader, they are written to a bad record file called `<timestamp>.failed`. It is also written into the `<timestamp>.failed.details` file along with a detailed error message about why the record was not loaded successfully. Failed records can be cleaned up and loaded again using RTL.
VERBOSE	Verbosity of the loading progress. The valid options are *true* and *false*. When this is set to *true*, data loading progress is written to console, as follows: `1: Loaded record System/LID = SysA/1000. Returned EUID = 0000000000. Match Result = NEW_EO`
EXECUTE_MATCH_API	Specify which executeMatch API to load to process incoming record. The valid options are *executeMatchUpdate*, *executeMatch*, and *deduplicateSystemObject*. For more information on these APIs, see *Oracle Healthcare Master Person Index Message Processing Reference*. **Note**: You can configure the deduplicateSystemObject API. For information on how to do so, see Configuring deduplicateSystemObject API on page 1-5.

Property Name	Property Description
MAX_RETRY_NUMBER	Maximum number of retries when concurrent modification happens due to parallel processing.
MIN_RETRY_DELAY	Minimum retry delay (in seconds) when concurrent modification happens due to parallel processing. The actual retry delay is a random number in the range of [MIN_RETRY_DELAY..MAX_RETRY_DELAY].
MAX_RETRY_DELAY	Maximum retry delay (in seconds) when concurrent modification happens due to parallel processing. The actual retry delay is a random number in the range of [MIN_RETRY_DELAY..MAX_RETRY_DELAY].

Running Real-time Loader

The Real-time Loader requires the MPI project to be deployed and running first.

For information on how to create MPI database tables, configure application server resources, and deploy MPI application, see *Oracle Healthcare Master Person Index User's Guide*.

To run the Real-time Loader, execute `run-realtime-loader.bat` or `run-realtime-loader.sh`.

Comparing Initial Bulk Matcher and Loader with Real-time Loader

The following table shows comparison between IBML and RTL:

Feature	Initial Bulk Matcher and Loader	Real-time Loader
Initial Loader	Yes	Yes
In-production Loader, that is, OHMPI already in production, application server deployed, and other OHMPI operations running.	No	Yes
Requires Empty MPI Database	Yes	No
Requires Running MPI Application	No	Yes
Match Configuration	It uses match configuration defined in the loader configuration file (`conf/initial-bulk-loader-config.xml`), by default. A different configuration file can also be specified.	It uses match configuration in the MPI runtime. No client-side match configuration file can be specified.
Bulk Matching	Yes	No Matching is done by the MPI Runtime, not the RTL.
Performance	Very performant on large amount of input data.	Not as performant as IBML on large amount of input data. Each record is loaded separately against all the records in the current MPI database. Therefore, the performance degrades as more unique records are loaded into the MPI database.
Multi-thread Support	Yes	Yes

Feature	Initial Bulk Matcher and Loader	Real-time Loader
Multi-node Support	Yes The IBML process can utilize multiple nodes (1 master node plus n optional slave nodes) for better performance.	No RTL does not have built-in multi-node support. However, multiple instances of RTL can be fired from different nodes against the same MPI server at the same time loading different data set.
Input Data Format	Same as RTL	Same as IBML
GID	GID is the first field in each record in the input data file. It is the unique global identifier for each record in the input data files for IBML. If there are multiple input files, each GID must be unique across all input files. In other words, there cannot be two input records sharing the same GID even if they are in two different input files. GIDs have to be valid integers (or in the lower range of long numbers). When generating input data files for IBML, it is preferred to generate GIDs starting from 1 and going up.	GID value is completely ignored by RTL. This is because each input record is loaded against the running MPI server independently. There is no need to uniquely identify each input record among each other like in IBML.
Basic Workflow	1. Execute `cluster-synchronizer.sql` or `cluster-truncate.sql`. 2. Execute `run-initial-bulk-loader.bat`. 3. Execute `generate-sql-loader.bat`. 4. Execute `sqlldr/drop.sql`. 5. Execute `bulk_loader.bat`. 6. Execute `sqlldr/create.sql`.	1. Execute `run-realtime-loader.bat`.
Difficulty of Running	Moderate	Easy
Clustering Support	N/A	Yes
Implementation	Standalone Java SE application	Java EE (EJB) Client

Configuring deduplicateSystemObject API

Note: This section is applicable only for OHMPI 2.0.13 patch.

You can configure the options for deduplicateSystemObject API in the rtl.properties file. You must copy this file in the <loader_home>/config folder.

Table 1 lists the properties that you can configure in the rtl.properties file.

Table 1 Configurable Properties in rtl.properties

Name	Description
deduplicate.source.system.search.id	The name of the query to be used. This release supports only BLOCKER-SERACH.
deduplicate.source.system.weighted	It will always be 'weighted' even if you set the value as false.
deduplicate.source.system.match.threshold	Match threshold is used for determining the match. That can be above or lower the assumed match threshold. It provides dynamic flexibility. The default value is 0.0. When you pass the default value, it uses assumed match threshold configured in master.xml.
deduplicate.source.system.transaction.option	Valid options are TRANSACTION_LOG_DELTA_ ENABLED and TRANSACTION_LOG_DELTA_DIS-ABLED. For more information, see *Oracle Healthcare Master Person Index Message Processing Reference* guide.
realtime.loader.post.process.class.impl	The run time loader post processor class name.
deduplicate.source.system.replace	This property decides, if the incoming systemobject has null values, whether it will overwrite the values of corresponding fields in the database. The valid options are *true* and *false*. The value *true* will overwrite the object with null values.
db.url	Connection string to connect to database table for post processor.
db.userName	Username that post processor will use to connect to the database.
db.password	Password that post processor will use to connect to the database.

Configuring Real-time Loader Post Processor

Note: This section is applicable only for OHMPI 2.0.13 patch.

1. Write a class which implements PostProcess interface

   The following is a sample implementation of PostProcess. This post processor stores EUID, SystemCode, and LocalId of the processed record in the custom database table.

```
package oracle.hsgbu.ohmpi.loader.realtime.custom;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Collection;
import java.util.Iterator;
import java.util.logging.Level;
import java.util.logging.Logger;
import oracle.hsgbu.ohmpi.loader.realtime.PostProcess;
import oracle.hsgbu.ohmpi.loader.realtime.RealtimeLoaderConfig;
import oracle.hsgbu.ohmpi.loader.realtime.RealtimeLoaderException;
import com.sun.mdm.index.objects.SBR;
import com.sun.mdm.index.objects.EnterpriseObject;
import com.sun.mdm.index.objects.SystemObject;
import com.sun.mdm.index.objects.exception.ObjectException;
```

```
/**
 * a sample implementation of RTL PostProcess
 *
 * @author ohmpi
 */
public class PostProcessImpl implements PostProcess {
    private static Logger logger =
Logger.getLogger(PostProcessImpl.class.getName());
    private Connection conn;
    private PreparedStatement pStmt;
    private String sqlStmt = "insert into mpi_rtl_data (\"EUID\",
\"SYSTEMCODE\" ,\"LID\") values (?,?,?)";

    public PostProcessImpl() {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        } catch (ClassNotFoundException ex) {
            logger.log(Level.SEVERE, null, ex);
        }
    }

    public void execute(SystemObject so, EnterpriseObject eo,
RealtimeLoaderConfig rlc) throws RealtimeLoaderException {
        try {
            SBR sbr = eo.getSBR();
            String euid = eo.getEUID();
            String systemCode = so.getSystemCode();
            String LID = null;
            logger.info("post processing  euid = " + euid);
            if (so.getLID() != null && !so.getLID().equals("")) {
                LID = so.getLID();
            } else {
                Collection systemObjects = eo.getSystemObjects();
                Iterator iter = systemObjects.iterator();
                while (iter.hasNext()) {
                    SystemObject object = (SystemObject) iter.next();
                    if (object.getSystemCode().equals(so.getSystemCode()) &&
object.getLID().startsWith("U")) {
                        LID = object.getLID();
                        break;
                    }
                }
            }
            conn = DriverManager.getConnection(rlc.getProperty("db.url"),
rlc.getProperty("db.userName"), rlc.getProperty("db.password"));
            pStmt = conn.prepareStatement(sqlStmt);
            pStmt.setString(1, euid);
            pStmt.setString(2, systemCode);
            pStmt.setString(3, LID);
            pStmt.execute();

        } catch (SQLException ex) {
            logger.log(Level.SEVERE, null, ex);
        } catch (ObjectException ex) {
            logger.log(Level.SEVERE, null, ex);
        } finally {
            try {
                if (pStmt != null) {
                    pStmt.close();
```

```
                }
                if (null != conn) {
                    conn.close();
                }
            } catch (SQLException ex) {
                logger.log(Level.SEVERE, null, ex);
            }
        }
    }
}
```

2. Add the implementation class of PostProcess interface and required jars in the <loader_home>/lib folder.
3. Update the classpath property.
   - For Windows, update in the run-realtime-loader.bat file.
   - For Unix, update in the run-realtime-loader.sh file.
4. Configure the realtime.loader.post.process.impl property in the rtl.properties file.

Running Sample Post Processor

Note: This section is applicable only for OHMPI 2.0.13 patch.

The sample post-processor and required files are bundled with the OHMPI 2.0.13 patch.

1. Execute the postprocess-tables.sql script from samples directory. This creates a custom table named MPI_RTL_DATA.
2. Copy the jdbc driver and sample/postprocess-impl.jar files to the <loader-home>/lib folder.
3. In the <loader-home>/config/rtl.properties file, configure the database details for the MPI_RTL_DATA custom table. You can find the properties under the comment #custom in the rtl.properties file.
4. Configure the value of the "realtime.loader.post.process.class.impl" property as:

   ```
   realtime.loader.post.process.class.impl =
   oracle.hsgbu.ohmpi.loader.realtime.custom. PostProcessImpl
   ```

5. Update the classpath property to include jdbc driver jar file and postprocess-impl.jar.
   - For Windows, update classpath in the run-realtime-loader.bat file.
   - For Unix, update classpath in the run-realtime-loader.sh file.

Related Documents

For more information and instructions for implementing and using a master person index application, see the following documents in the Oracle Healthcare Master Person Index documentation set:

- *Oracle Healthcare Master Person Index Real-time Loader User's Guide* [This document]

- *Oracle Healthcare Master Person Index Analyzing and Cleansing Data User's Guide*
- *Oracle Healthcare Master Person Index Australia Patient Solution User's Guide*
- *Oracle Healthcare Master Person Index Command Line Reports and Database Management User's Guide*
- *Oracle Healthcare Master Person Index Configuration Guide*
- *Oracle Healthcare Master Person Index Configuration Reference*
- *Oracle Healthcare Master Person Index Data Manager User's Guide*
- *Oracle Healthcare Master Person Index Installation Guide*
- *Oracle Healthcare Master Person Index Loading the Initial Data Set User's Guide*
- *Oracle Healthcare Master Person Index Match Engine Reference*
- *Oracle Healthcare Master Person Index Message Processing Reference*
- *Oracle Healthcare Master Person Index Provider Index User's Guide*
- *Oracle Healthcare Master Person Index Release Notes*
- *Oracle Healthcare Master Person Index Security Guide*
- *Oracle Healthcare Master Person Index Standardization Engine Reference*
- *Oracle Healthcare Master Person Index United Kingdom Patient Solution User's Guide*
- *Oracle Healthcare Master Person Index United States Patient Solution User's Guide*
- *Oracle Healthcare Master Person Index User's Guide*
- *Oracle Healthcare Master Person Index Working With HPD Profile Application User's Guide*
- *Oracle Healthcare Master Person Index Working With IHE Profiles User's Guide*

Note: These documents are designed to be used together when implementing a master index application.

Finding Information and Patches on My Oracle Support

Your source for the latest information about OHMPI is Oracle Support's self-service Web site My Oracle Support (formerly MetaLink).

Before you install and use OHMPI, always visit the My Oracle Support Web site for the latest information, including alerts, White Papers, installation verification (smoke) tests, bulletins, and patches.

Creating My Oracle Support Account

You must register at My Oracle Support to obtain a user name and password account before you can enter the website.

To register for My Oracle Support:

1. Open a Web browser to `https://support.oracle.com`.
2. Click the **Register here** link to create a My Oracle Support account. The registration page opens.

3. Follow the instructions on the registration page.

Signing In to My Oracle Support

To sign in to My Oracle Support:

1. Open a Web browser to https://support.oracle.com.
2. Click **Sign In.**
3. Enter your user name and password.
4. Click **Go** to open the My Oracle Support home page.

Finding Information on My Oracle Support

There are many ways to find information on My Oracle Support.

Searching by Article ID

The fastest way to search for information, including alerts, White Papers, installation verification (smoke) tests, and bulletins is by the article ID number, if you know it.

To search by article ID:

1. Sign in to My Oracle Support at https://support.oracle.com.
2. Locate the Search box in the upper right corner of the My Oracle Support page.
3. Click the sources icon to the left of the search box, and then select **Article ID** from the list.
4. Enter the article ID number in the text box.
5. Click the magnifying glass icon to the right of the search box (or press the Enter key) to execute your search.

   The Knowledge page displays the results of your search. If the article is found, click the link to view the abstract, text, attachments, and related products.

Searching by Product and Topic

You can use the following My Oracle Support tools to browse and search the knowledge base:

- Product Focus — On the Knowledge page under Select Product, type part of the product name and the system immediately filters the product list by the letters you have typed. (You do not need to type "Oracle.") Select the product you want from the filtered list and then use other search or browse tools to find the information you need.
- Advanced Search — You can specify one or more search criteria, such as source, exact phrase, and related product, to find information. This option is available from the **Advanced** link on almost all pages.

Finding Patches on My Oracle Support

Be sure to check My Oracle Support for the latest patches, if any, for your product. You can search for patches by patch ID or number, or by product or family.

To locate and download a patch:

1. Sign in to My Oracle Support at https://support.oracle.com.

2. Click the **Patches & Updates** tab. The Patches & Updates page opens and displays the Patch Search region. You have the following options:
   - In the **Patch ID or Number is** field, enter the number of the patch you want. (This number is the same as the primary bug number fixed by the patch.) This option is useful if you already know the patch number.
   - To find a patch by product name, release, and platform, click the **Product or Family** link to enter one or more search criteria.
3. Click **Search** to execute your query. The Patch Search Results page opens.
4. Click the patch ID number. The system displays details about the patch. In addition, you can view the Read Me file before downloading the patch.
5. Click **Download.** Follow the instructions in the patch Read Me to install the patch.

Finding Oracle Documentation

The Oracle Web site contains links to all Oracle user and reference documentation. You can view or download a single document or an entire product library.

Finding Oracle Health Sciences Documentation

To get user documentation for Oracle Health Sciences applications, go to the Oracle Health Sciences documentation page at:

http://www.oracle.com/technetwork/documentation/hsgbu-154445.html

Always check the Oracle Health Sciences Documentation page to ensure you have the latest updates to the documentation.

Finding Other Oracle Documentation

To get user documentation for other Oracle products:

1. Go to the following Web page:

   http://www.oracle.com/technology/documentation/index.html

   Alternatively, you can go to http://www.oracle.com, point to the Support tab, and then click **Documentation**.
2. Scroll to the product you need and click the link.
3. Click the link for the documentation you need.

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.

Oracle Healthcare Master Person Index Real-time Loader User's Guide, Release 2.0.13
E78168-01