JCC LogMiner Loader

Release Notes 3.6

February, 2019

From the JCC Toolset for Databases

Contact Information

E-mail	JCC-LMLoader@JCC.com
Phone	+1 (740)587-0157
FAX	+1 (740)587-0163
Post Office	LogMiner Loader JCC Consulting, Inc. Box 381 Granville, OH 43023

For JCC LogMiner Loader licensing and support, contact JCC Consulting.

JCC Consultants are also eager to hear your comments, questions, and examples.

JCC also provides consulting for solution architectures and on-site support for database review and for getting started with new architectures. Training and temporary licenses are available.

Notices

The information in this manual has been carefully checked and is believed to be accurate. However, changes to the product are made periodically. These changes are incorporated in new publication editions. JCC Consulting, Inc. may improve and/or change products described in this publication at any time. Due to continuing system improvements, JCC Consulting, Inc. is not responsible for inaccurate information that may appear in this manual. For the latest product updates, consult the JCC Consulting, Inc. web site at **www.jcc.com** or consult JCC in any of the ways indicated in the **"Contact Information" on page 3**. In no event will JCC Consulting, Inc. be liable for direct, indirect, special, exemplary, incidental, or consequential damages resulting from any defect or omission in this manual, even if advised of the possibility of such damages.

This product should not be used in any industry or fashion for which the underlying database products are not valid.

In the interest of continued product improvement, JCC Consulting, Inc. reserves the right to make improvements in this manual and the products it describes at any time, without notices or obligations.

This file and the LogMiner Loader software are confidential and proprietary to JCC Consulting, Inc. All documentation and the LogMiner Loader software are provided on an AS-IS basis.

Trademark Acknowledgments

JCC Toolset and JCC LogMiner Loader are trademarks of JCC Consulting, Inc. LogMiner, Oracle Rdb, Oracle 12 (and multiple other versions), MySQL, and Tuxedo are trademarks of Oracle Corporation. Windows (in its multiple versions) and SQL Server are trademarks of Microsoft Corporation. MariaDB is a trademark of MariaDB Corporation. Teradata is a trademark of Teradata Corporation. DB2 is a trademark of IBM Corporation. OpenVMS is a trademark of Hewlett Packard Enterprise (HPE). OpenVMS is also available from VSI (VMS Software Inc.) which holds the copyright to versions created by VSI. Apache, Apache Kafka, Kafka, and associated open source project names are trademarks of the Apache Software Foundation.

Uses in this Document

In this document, Oracle Rdb and Oracle's other RDBMS are referred to frequently and need to be distinguished. Consequently, Oracle Rdb is referred to as "Rdb" and Oracle's other database product (whatever its version) is referred to simply as "Oracle." Similarly, Oracle Rdb LogMiner is referred to as "Rdb LogMiner" or as "LogMiner." The JCC LogMiner Loader is sometimes referred to simply as "the Loader".

Disclaimer

CHAPTER

Version 3.6

The notes in this chapter are specific to Version 3.6 of the JCC LogMiner Loader and represent changes made since Version 3.5.1. An updated version of the full documentation will be available after the Version 3.6 release. The blogs - updated between releases - may also be helpful.[1]

Version 3.6 of the JCC LogMiner Loader includes support for publishing Rdb committed transactions to Kafka message servers. The JCC LogMiner Loader Kafka Option is a separately licensed add on. There is also separate documentation for the Kafka option.

Version 3.6 of the JCC LogMiner Loader corrects all known issues, adds new features, and enhances Loader flexibility, functionality, and performance.

1. Find the documentation for download in the upper right corner at http://www.jcc.com/lml and the blogs at http://www.jcc.com/lml-blog.

All corrections and enhancements included in these release notes are available with a standard license to the JCC LogMiner Loader.

Corrected Issues

All known issues are addressed. Some of the issues relate to only one target or to other specifics and would not have been relevant in all cases.

The issues addressed are:

1. A recently introduced bug that caused the empty string to be reflected for the JCCLML_AERCP virtual column. See page 11.
2. Mishandlng of cases where function name is the same as a column name. See page 11.
3. The unintentional disabling of features only needed with versions of Rdb older than 7.0.6.3. See page 11.
4. Mishandling of situations for which the definition of the record key includes more columns than the non-key columns defined for the table. See page 11.
5. An error when a NULL date column was processed with the OCI interface. See page 12.
6. An error of MapResult attaching two bytes of binary data to the beginning of the value for varchar. See page 12.
7. MapResult and Virtual Column incompatibility that resulted in data being over-written in memory and incorrect data being written to the target. See page 13.
8. A failure to write to the target on an initial start that caused issues on restart. See page 13.
9. Partial transaction committed on an exit with an unexpected status. See page 14.
10. Problem that generated FilterMap syntax that did not work on DATE ANSI source column data types. See page 14.
11. A stack dump resulting from a full disk. See page 30.
12. Loader info procedure reported a Loader session as active when it was not. See page 15.

The sections to follow describe each of these corrections.

JCCLML_AERCP and the Empty String

There was a bug introduced in a recent release that caused the virtual column JCCLML_AERCP to output an empty string.

This problem has been fixed.

MapResult with the Same Name as Target Column

The Loader was not distinguishing between a function name and a column name in the SQL expressions used with MapResult and FilterMap processing. This led to unexpected and confusing results.

The Loader session would exit with an SQL exception, while trying to validate the MapResult statement.

This problem has been fixed.

Old Rdb Versions

The JCC_LOGMINER_LOADER_RDB_VERSION logical name establishes features that are implemented for backward compatibility with versions of Rdb older than 7.0.6.3. The features were unintentionally disabled. The disabling caused no issues unless the Loader was run with an output target of Rdb and an Rdb version older than 7.0.6.3.

This bug has been fixed.

Number of Columns Defined for the Key

For a delete record and the JDBC interface, a definition of the record key to include more columns than the non-key columns in the table resulted in

incorrect processing. The Loader session would exit with an ArrayIndex-OutOfBounds exception.

This bug has been fixed.

OCI and NULLs

When a date column is NULL, the OCI interface still processes the data stored in the date which can be invalid. This could lead to a number of errors, depending on the pattern of bytes found in the internal buffer.

Some possible exceptions include:

```
ORA-06502: PL/SQL: numeric or value error: character to number
conversion error
```

```
%LIB-S-STRTRU, string truncated
```

This behavior occurred only with the OCI interface and, then, only if a date value was NULL and the Control File did not define a value to substitute for NULL.

This bug has been fixed.

MapResult and Varchar

MapResult was not properly handling the varchar data type. Instead, two bytes of binary data were attached as the first two bytes of a varchar value.

This problem has been fixed.

MapResult and VirtualColumns

An incompatibility between MapResult and VirtualColumns caused some data to be overwritten in memory. This, then, caused invalid data to be written to the target for the first MapResult column.

This problem has been fixed.

Initial Starts

There was an error in the restart logic of the Loader that was only encountered, when using checkpoint type LML_INTERNAL[2] and after a specific series of events occurred. Those events were:

1. Start a new Loader session
2. The session fails to connect or write to the target on the first checkpoint
3. The issue is corrected and the Loader is restarted

At this point, the CTL process would fail attempting to read the checkpoint data from the failed first attempt and display the message:

```
   Logic failure; return_status == dba_restart_validate && restart_validate
   %jcc_continuous_logminer_loader: Failed to get restart information.
   o
   o
   o
   %DBA-E-UNEXPECT_EXCEPT, Unexpected exception encountered. See previous message
for more information.
```

This bug has been fixed.

2. Checkpoint type LML_INTERNAL is used for all Loader targets except Rdb and OCI (Oracle) which write the checkpoint to the target database.

Partial Transaction Committed on Exit

A case was discovered, for Loader targets of Rdb, that caused the Loader to write a partial transaction when it was exiting due to an unresolved deadlock. The scenario, shown in the exception messages to follow, was that all Loader threads became tangled in a deadlock. When one thread exceeded the "retry count" requested in the Control File, all threads aborted.

The messages received were

```
29-OCT-2018 15:45:16.67 OAAF9E48 ||1 REGTESTRDBS -RDMS-F-DEADLOCK, deadlock on
record 72:1156:0

29-OCT-2018 15:45:16.75 OAAF9E48 %jcc_logminer_loader: ended with an exception:
%DBA-E-MAX_OUT_RETRIES, Maximum message output failures received.
```

The unexpected part was that rows that had been written to a buffer were committed. In this particular case, when the Loader was restarted, processing started far enough back to resend the entire transaction and the results were correct. However, committing partial transactions is not the desired behavior.

The Loader is modified to ensure that a rollback occurs when the Loader is exiting with an unexpected status.

FilterMap and Date ANSI

The FilterMap syntax generated from the JCC_LML_CREATE_CONTROL_FILE utility did not work with DATE ANSI source column data types.

This bug has been fixed.

Stack Dump on Full Disk

When the disk that is used for the LML or CLM log files filled, the control process for the Loader exited with a stack dump. The stack dump is now replaced with more supportive behavior. A full disk is still a problem, but not one that the Loader can resolve.

The fix provides a message similar to the following in the CTL process log file:

```
   20-NOV-2018 11:20:44.77: %RMS-F-FUL, device full (insufficient space for allocation)
   20-NOV-2018 11:20:44.77: Exception; suspending output to jcc_tool_logs:jcc_run_lml-
SB0540FCAE_0.log
```

All data written to the log files during the suspension will be lost.

After correction of the full disk problem, another message is shown.

```
   20-NOV-2018 11:43:21.84: Resuming output to jcc_tool_logs:jcc_run_lml-SB0540F-
CAE_0.log
```

Loader Info Incorrectly Finding Loader Active

The procedure JCC_GET_LOADER_INFO provides information about a running Loader session. It could, previously, be misled when a process name that included the name of the Loader session, caused it to incorrectly declare that the session was active, when it was not.

This bug is fixed.

Enhancements

Enhancements are added for Version 3.6 of the JCC LogMiner Loader. Some of these are developed for the Kafka Option, but also made available for other targets. All enhancements listed here are included in the general release for all targets.

The enhancements are:

1. Wildcarding for VirtualColumn and MapColumn keywords to extend a statement to all tables. See page 17.
2. Extension to the procedure to create a Control File to support creating the originating_DBKey support for all tables, as needed. See page 19.
3. Expansion of the XML keyword to modify the default header or define whether to include NULL values. See page 19.
4. Ability to include a specific JRE (Java Runtime Environment). See page 21.
5. Enhancements for the IVP. See page 23.
6. Improvements to the statistics display. See page 24.
7. Added robustness for the “heartbeat” support. See page 32.
8. Performance improvements in the efficiency of the logging, in database attaches, and in the JDBC interface. See page 32.
9. Expansion of JCC_GET_DB_INFO to check the version of Rdb and set a symbol. See page 34.
10. Removal of the I2 build which is no longer needed. See page 34.
11. Support for a logical name to turn off the Data Pump’s “No rows found” informational message. See page 35.
12. Support for a logical name to set delta dates to NULL. See page 35.
13. A logical name to reduce the buffer count. See page 35.
14. Default logical names for Java. See page 35.
15. Minimization of DCL exceptions. See page 36.
16. Provide additional logging for the Control File. See page 36.
17. Automated moving of the checkpoint with sessions that are defined in such a way that there are few rows written to the Loader target. See page 37.
18. Support for changing the prefix setting for lines in the log. See page 37.

The sections to follow describe these enhancements.

Wildcards for Virtual Column and MapColumn

When the Loader Administrator wants a virtual column, for example the transaction commit datetime stamp, to be included in each target table, it can be very useful to have a way of defining that the inclusion is needed in each and every table, instead of having to make a separate statement for each table.

Wildcarding supports doing that. It is both convenient and a useful assurance of greater accuracy.

Wildcard for VirtualColumns

The Loader supports adding some meaningful additional data to a row written to the target. See Keyword: VirtualColumn in the Control File chapter for the full documentation.

An ability to add a specific VirtualColumn to *every* table is included with Release 3.6.

The general syntax for VirtualColumns is

```
VIRTUALCOLUMN~<table name>~<virtual column name>              \
[,<output column name>]
```

Some of the VirtualColumns have additional parameters. See the full documentation for details.

For wildcarding an asterisk is substituted for the table name. The general syntax, for Release 3.6 of the Loader, therefore, is

```
VIRTUALCOLUMN~<table name>|*~<virtual column name>            \
[,<output column name>]
```

A wildcard virtual column directive can use any VirtualColumn. The example of wildcarding below shows some of the possibilities.

```
!
! JCC LML V3.6 Wildcard VirtualColumn
! Add virtual columns to all table definitions
!
VirtualColumn~*~action,jcclml_action
```

```
VirtualColumn~*~loader_sequence_number,jcclml_sequence_number
VirtualColumn~*~loadername
VirtualColumn~*~loader_version
VirtualColumn~*~loader_link_date_time
VirtualColumn~*~transaction_commit_time
VirtualColumn~*~transaction_start_time
VirtualColumn~*~jcclml_read_time
VirtualColumn~*~jcclml_aercp
VirtualColumn~*~transmission_date_time
```

Note: VirtualColumn must be defined in the Control File after TABLE definitions and before MAPTABLE definitions.

Wildcard MapColumn

To map the wildcarded virtual columns to target columns is similar.

The syntax for prior versions was

```
MapColumn~<map table name>~<source column name>             \
[,<target column rename>][~<value if null>]
```

or, for VirtualColumns only,

```
MapColumn~<map table name>~                                 \
<source (virtual) column name|output column name>           \
[,<target column rename>][~<value if null>]
```

For version 3.6, the syntax for MapColumn[3] for Virtual Columns is

```
MapColumn~<map table name>|*~                               \
<source (virtual) column name|output column name>           \
[,<target column rename>][~<value if null>]
```

The following shows some examples.

```
!
! JCC LML V3.6 Wildcard MapColum
! Add virtual columns to all MapTable definitions
!
MapColumn~*~jcclml_action
MapColumn~*~jcclml_sequence_number,loader_sequence_number
```

3. If the Control File still uses the old format that creates MapTable from the Table definition, that defines the columns for output and MapColumn is unnecessary.


```
MapColumn~*~loadername
MapColumn~*~loader_version
MapColumn~*~loader_link_date_time
MapColumn~*~transaction_commit_time
MapColumn~*~transaction_start_time
MapColumn~*~jcclml_read_time
MapColumn~*~jcclml_aercp
MapColumn~*~transmission_date_time
```

MapColumn must be defined in the Control File after MapTable definitions and before FilterMap or MapResult definitions.

Expansion in Procedure to Create Control Files

There is a command provided with the Loader kit to create a Control File. Version 3.6 adds a new option for this routine that may be used when needed.

The syntax is

```
$ JCC_LML_CREATE_CONTROL_FILE <db file> ORIGINATING_DBKEY
```

Adding ORIGINATING_DBKEY creates a file that will have the necessary SQL[4] table statements for tables that have no discernible primary key. The file is named <db file>_ODBKEY.SQL (where <db file> is the name of the database root file).

See the full documentation for more on the need for a primary key, the Rdb construct of originating dbkey, the procedure to create the Control File, and this option.

Expansions in Keyword: XML~Header

XML is supported as output of the Loader. It is defined with the output conversion parameter of the Output Keyword. The XML keyword, which is also available, is

4. The utility generates standard SQL which is valid for databases such as Rdb and Oracle that provide standard SQL. Some other databases may require changes to the output syntax in order to be successfully executed.

not generally used because defining XML as output conversion sought sets defaults that are suitable in most instances. See the full documentation for more discussion on this.

Attributes that are new with the JCC LogMiner Loader Version 3.6 include Header and NULL.

Header Syntax

By default, the Loader includes a header for XML output. The header has two parts: one called Prolog and one called DocType.

Prolog consists of "<?xml version='1.0'?>" which specifies the version of the XML standard that is used for this XML document.

DocType consists of "<!DOCTYPE pkt SYSTEM 'packet-commented.dtd'>".[5]

Because some application architects do not want this header, the Loader has been modified to include a keyword for the XML header. The syntax has four options.

TABLE 1. XML Header Options

Syntax	Result
`XML~Header~Prolog,DocType`	The default includes both parts of the header.
`XML~Header~`	Includes neither part of the default header.
`XML~Header~Prolog`	Include the part of the default header called prolog, but not the part called DocType
`XML~Header~DocType`	Include the part of the default header called DocType, but not the part called Prolog

NULL Syntax

There are two alternative values for the XML keyword to indicate whether to show columns with NULL values.

```
XML~NULL~explicit|implicit
```

5. The JCC LogMiner Loader's DTD's can be found in JCC_TOOL_ROOT:[SOURCE].

Explicit is the default and the specification is not required, unless implicit is desired. Explicit lists null columns.

Implicit causes the Loader to not list null columns.

However, there are actually three possible results, depending on how the column itself is defined. Keyword MapColumn includes the option of defining <value if null>. If a value definition for null is specified, that value replaces NULL. In which case, for this column, implicit and explicit have the same effect.

Java Run-Time Environment

The JCC LogMiner Loader uses a Java Virtual Machine to access a Java Environment to support JDBC connections to a variety of targets.

A Java Virtual Machine (JVM) is an emulated computer that provides the ability to run a Java program. The operating system in the JVM is referred to as the Java Environment. The Java Environment comes in two flavors; Java Development Kit (JDK) and Java Runtime Environment (JRE). The Java Development Kit includes compilers and other utilities necessary to create your own Java programs. The Java Runtime Environment is a proper subset of the JDK that includes only the files required to run existing Java programs.

System administrators often install the JDK to provide a single version on the system for all Java users. This practice can, at times, conflict with requirements for some applications. The JCC Logminer Loader has encountered bugs in various versions of the Java Environment and so requires customers to use a version that does not contain those bugs. See the blog aritcle “JCC LogMiner Loader Replication to JDBC Targets [Java Versions]” for information about supported Java versions and kit availability. The following example shows the installed JDK is version 1.5.0-5.

```
$ @TOM$DKA300:[JAVA.JAVA$150.COM]JAVA$150_SETUP.COM
$ java -version
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition
Fast VM (build 1.5.0-5, build J2SDK.v.1.5.0:12/20/2008-03:46, native threads, jit_150)
```

Included in the Java distribution are the files necessary to install and use a private JRE for a given application. The JRE files can be found in the same kit downloaded from HPE as the JDK and are encapsulated in a self-extracting ZIP file which includes "JRE" in the name. Details about how to install it in a user-specified directory are in the Java release notes included in the same kit, usually as an HTML document.

The following is an example of installing the JRE for private use of the JCC LogMiner Loader. Note that saveset "B" ("JRE-V150-9_B.SAV" in the example below) is optional, not needed by the JCC LogMiner Loader and so not used in the example. The goal of this example is to install a version of JAVA in the Loader's own directory tree. The JRE can also be placed in a more globally accessible location; the only requirements are that the disk has sufficient space and can be set to support an ODS5 file system.

In the example, the required saveset is unpacked from the HPE distribution.[6] The proper Loader context is set and then the saveset is restored to the appropriate point in the Loader's directory tree. The example is lengthly. There are additional comments at the end.

```
Directory TOM$DKA300:[S_D.JAVA.150-9]

DEC-AXPVMS-JRE-V0105-9-1.ZIPEXE;1
                        74535/74544     1-MAR-2018 10:07:15.20

Total of 1 file, 74535/74544 blocks.

$ run DEC-AXPVMS-JRE-V0105-9-1.ZIPEXE
UnZipSFX 5.40 of 28 November 1998, by Info-ZIP (Zip-Bugs@lists.wku.edu).
  inflating: JRE-V150-9_A.SAV
  inflating: JRE-V150-9_B.SAV
$ dir/size=all/date=create JRE-V150-9_%.SAV;

Directory TOM$DKA300:[S_D.JAVA.150-9]

JRE-V150-9_A.SAV;1    134253/134256   15-MAR-2016 13:34:36.37
JRE-V150-9_B.SAV;1     19719/19728    15-MAR-2016 13:37:18.60

Total of 2 files, 153972/153984 blocks.

$ @jcc_tool_com:jcc_lml_user
Setting JCC LogMiner Loader version Standard
$ backup/log JRE-V150-9_A.SAV/save jcc_tool_root:[java...]*.*;

o
o
o
$ @jcc_tool_java_lib:JAVA$150_JRE_SETUP.COM
$ java -version
java version "1.5.0"
```

6. Alpha distributions still say “DEC”.


```
Java(TM) 2 Runtime Environment, Standard Edition
Classic VM (build 1.5.0-9, 03/14/2016-10:50, native threads, jit)
```

Once this is completed, the processes used to run the JCC LogMiner Loader or any of its associated utilities should be modified to use the JRE instead of the system version. To do this, change all of your Loader-related procedures that set the Java version using the "jcc_lml_jdbc_user" command to specify the new JRE setup procedure instead of the system-wide JDK setup procedure.

```
$! jcc_lml_jdbc_user 1.5.0 TOM$DKA300:[JAVA.JAVA$150.COM]JAVA$150_SETUP.COM
$ jcc_lml_jdbc_user 1.5.0 jcc_tool_java_lib:JAVA$150_JRE_SETUP.COM
```

You will need to logout and login again for the changes to take effect. Any Loader processes will need to be restarted as well.

IVP

Release 3.6 of the Loader adds two checks to the operation of the IVP.

Run Local

The Loader checks to ensure that the IVP is running from a locally mounted disk to provide support of creating and altering the databases required for the IVP. If the IVP is not running from a locally mounted disk the following message will be displayed.

```
This procedure must be run on the same node as that
on which the Loader software is installed.
Local node: <current computer>
Installed node: <remote mounted disk node>
```

Run User Procedure

The Loader verifies that the JCC_LML_USER procedure has been executed prior to running the IVP. This ensures the appropriate environment to perform the requested verification. If the procedure has not been run, the following message will be displayed.

```
This procedure requires the JCC LogMiner Loader
environment be set. Please execute jcc_lml_user
procedure for this process before running the
the IVP.
Use either:
$ @jcc_tool_com:jcc_lml_user
or, for multiversion support:
$ @jcc_tool_com:jcc_lml_user <LML version>
```

Improvements to the Statistics Display

For a full discussion of the Statistics display, see the chapter Monitoring an Ongoing Loader Operation in the full documentation.

The DCL command that invokes the monitor includes parameters for the Loader family, the report interval, the report type, the threshold for reporting something as tardy, and the operator class for reporting.

Beginning with release 3.6, there are enhancements to all report types and an additional report type to reveal more about the state of each thread.

Interactive Support Added to Statistics

While running JCC_LML_STATISTICS interactively, there are keys that can be pressed to get additional information or change the behavior of the Monitor. The keys that are defined and their impact are shown in the table and are also shown online by pressing '?' or 'h'.[7]

TABLE 2. Control Keys for Interactive Statistics Display

Key	Description
?	Display help message with all of the keys defined.
h	
b	Switch to the 'b'rief display.
f	Switch to the 'f'ull display (the default).
d	Switch to the 'd'etail display.

TABLE 2. Control Keys for Interactive Statistics Display

Key	Description
s	Switch to the ‘s’tates display.
r	Show the rates, rather than totals. Available only for the full display.
t	Show the totals, rather than rates. Available only for the full display.
ctrl t	Print runtime information.
ctrl m	Switch between display mode and scroll mode.
ctrl w	If in display mode, clear the screen.
ctrl c	Exit the jcc_lml_statistics utility.
ctrl y	
ctrl z	

7. Lower case keys are shown in the table. The uppercase values have the same impact.

Time Scale Conversion

When the statistics display has what amounts to an overflow value for the length of time, the time units are changed to reflect a more meaningful result or greater precision, depending on the circumstances. (Previously, the change in units happened when the field length was maxed.)

Seconds is the default unit and seconds will be displayed without the ‘s’ to define the units as seconds.

Example conversions are shown in the table.

TABLE 3. Example Time Conversions for Statistics Display

Calculated (in seconds, if not noted as otherwise)	Displayed for Readability
28.1h	1.2d
36.4h	1.5d
90.02	1.5m
90.25m	1.5h
0.001	1.0ms
0.0000001	1.0us

See “Detail Report Example” on page 28 for an example. The trailing times are given in minutes rather than the default seconds.

Again, the example provided is the ‘D’etail report type. Time scale conversions are also reflected in other report types.

Loader Target Type Displayed with Statistics

Beginning with release 3.6, the statistics display will include the Loader target type - JDBC, OCI, FILE, Tuxedo, ... - defined in the Output keyword.

Again, see “Detail Report Example” on page 28 for an example.

In the example, the left side is labeled “Input:” and the right side is labeled “JDBC:” For this example, JDBC is the Loader target.

Detail Report Type

The detail report type is not new. It is changed with Version 3.6 to reflect

- Interactive support (page 24)
- Time scale conversion (page 26)
- Loader target type (page 27)

These new features are also available with other report types.

Detail Report Example

```
Rate:   6.00                      REGTESTJDBR           20-JAN-2019 16:43:29.00
================================================================================
   Input: 20-JAN-2019 16:32:39.71                JDBC: 20-JAN-2019 16:32:39.63
--[Trail:   10.8m]---------------            ---[Trail:   10.8m]---------------
Transactions               125072            Checkpoints                   9312
Records                    503489               Timeout                      13
   Modify                  376404               BufferLimit(   120)           0
   Delete                    2013               NoWork                        0
   Commit                  125072            Records(   3)               489738
Discarded                                       Messages(  N/A )            N/A
   Filtered                     0               Filtered                      0
   Excluded                     0            Failure                          0
   Unknown                      0            Timeout                          0
   Restart                      0         - Current ---------------- Ave/Second -
   NoWork                   13688         Checkpoints            36         6.00
   Heartbeat                    0         Records              1456       242.67
Timeout                      1013         Rate                 7.18%
--- Restart Context ------                - Latency(sec) ------ LML detail ------
M|AIJ#                 10 |               CLM   10.8m |  Inpt   0.9%  Cnvt  49.0%
Q|VBN             349609 |               ------------   Sort   0.0%  Trgt  38.6%
P|TSN             163596 |               LML    2.21    Sync  11.4%  Ckpt   0.1%
CTSN              163596 |         - Loaders - 0123456789abcdefghij ------------
LSN               111300 |         - States  - RRRRR RR>R>RRRRRRRRR
```

Statistics Display and the State Report Type

Version 3.6 introduces report type ‘S’ (State). The State report type provides expanded information about the current state of each of the Loader threads.[8] The example shown first includes 3 threads. See “State Report Example with Three Threads” on page 29.

8. The “state” could alternately be called the “status” or the “current operation”.

State Report Example with Three Threads

```
Rate:   6.00                       REGTESTORAJ            20-JAN-2019 16:41:22.46
================================================================================
 In: 20-JAN-2019 16:41:22.46 [5.000ms]    OCI: 20-JAN-2019 16:41:22.33 [   0.13]
                                         Chkpt             0  Recs             1
                                         CLM    0.00 |  Inpt   0.0%  Cvnt   0.0%
012 ----------------------------        ------------   Sort   0.0%  Trgt   0.0%
<>R                                      LML    0.00    Sync   0.0%  Ckpt   0.0%
[  0.00] 0 VA->Append
[  0.00] 1 Write->OCI
[  0.01] 2 Input->MBX wait
```

The first lines of the display are the same as the Detail report type.

Rate. The rate shown (first line, left) is 6.00. The refresh rate for the statistics is every 6 seconds. This is specified in starting the monitor.

Title. REGTESTJDBR is the name of the thread family, the Loader name. (In the example, the title indicates that this is a run of the regression testing that is writing to Rdb.)

Dates. The date on the title line is date/time for the statistics generation. The date for input is the timestamp of the last source transaction read from the Continuous LogMiner mailbox or input file. The date for output (on the right) is the timestamp of the last source transaction written to the target. The trailing times are shown just after the dates, 5ms in the case of input and .13 in the case of output. (Seconds is the default units and 0.13 should be read as .13 seconds or .13s.)

Target Type. The target type of 'OCI' is shown just before the date for output. OCI is the transport used for an Oracle database target. This is new with Version 3.6, but is reflected similarly on other Report Types.

The next lines give familiar information, but in a compact format.

Latency. The CLM and LML items show the percentages that represent the latency for the Continuous LogMiner and for the Loader. That the values are both 0.00, meaning there is no latency.

Latency details are also available from "Inpt", "Sort", "Sync", "Cnvt", "Trgt", and "Ckpt" which reflect the latencies for Inptu, Sort, Lock Syn-

chronization, Conversion, Target, and Checkpoint. See the full documentation for more information.

Loader States Summary. The two lines on the left that are 012 followed by hyphens and <>R give the same summary of thread states that is shown in the Detail report type. The top line identifies the threads and the next line shows their state. That is, the first thread < is currently reading from the input mailbox, the next > is writing to the target, and the last R is waiting to get the read lock. The chart for interpreting these symbols is included in the full documentation and at the end of this chapter. See “Decoding the Thread States” on page 37.

Detail for Each Thread. The list that comes next is the reason for the State report type. Each line shows the latency for the thread, the thread number, and the state. These descriptions of the state of the thread are wordier and, perhaps, more readily interpreted than the single character descriptions. These options and what they mean are also reflected in the appendix to this document.

There are 3 lines because there are 3 threads. There are 16 lines available for the threads which means that the bottom of the screen is blank in this case. If there are more than 16 threads, the State report type uses 2 columns. See “State Report Example with More Than Sixteen Threads” on page 31.

To interpret the thread states shown, note that DETAILS and PEOPLE are table names and MATERIALIZED_TABLE also identifies a table. Also see “Decoding the Thread States” on page 37.

State Report Example with More Than Sixteen Threads

```
Rate:   6.00                        REGTESTJDBR            20-JAN-2019 16:34:36.00
================================================================================
 In: 20-JAN-2019 16:32:03.67 [   2.5m]    JDBC: 20-JAN-2019 16:32:02.94 [   2.6m]
                                          Chkpt             45  Recs          2177
                                          CLM    2.5m |  Inpt   1.5%  Cnvt  39.1%
 0123456789abcdefghij ------------        ------------   Sort   0.0%  Trgt  41.6%
 W>W>>>>>>>>>>WW>>>>>                     LML    2.71    Sync  15.7%  Ckpt   2.1%
 [  0.84] 0 Output->Synch wait            [  0.38] g Write->JDBC
 [  0.19] 1 DETAILS[batch:19]             [  0.11] h Output->Process buffer
 [  1.82] 2 Output->Synch wait            [  0.20] i PEOPLE[batch:10]
 [  0.05] 3 MATERIALIZED_TABLE[batch:12] [  0.77] j DETAILS[batch:20]
 [  0.30] 4 Output->Process buffer
 [  0.07] 5 DETAILS[batch:14]
 [  0.20] 6 DETAILS[batch:16]
 [  0.53] 7 PEOPLE[batch:10]
 [  1.18] 8 Output->Process buffer
 [  0.18] 9 Write->JDBC
 [  0.94] a Output->Process buffer
 [  1.03] b Output->Process buffer
 [  0.31] c PEOPLE[batch:10]
 [  1.77] d Output->Synch wait
 [  5.70] e Output->Synch wait
 [  0.47] f Checkpoint
```

Heartbeat Support Enhancement

Heartbeat is a Loader supplied mechanism to ensure that Rdb LogMiner does not block Rdb backup. See the full Loader documentation for more detail.

Heartbeat relies on a single row in the JCCLML$HEARTBEAT table in the source database. Previously, if the row that heartbeat relies on was deleted, while the Loader was running, a message was given and the heartbeat feature disabled itself. When the session was restarted, a new heartbeat row was created and the feature was enabled again.

Beginning with Version 3.6 of the Loader, the heartbeat feature is more resilient. When the logical names are set to request the heartbeat function, if the Loader does not find the expected row, the heartbeat code will insert a new row.

Performance Adjustments

Enhancements to existing features improve performance.

Logging

Release 3.6 ensures that the priority of the Control (CTL) process is set higher than the subprocesses. This setting enables the CTL logging facility to process logging requests more efficiently.

Attaches to the FilterMap Database

A database is maintained as a work space for both FilterMap and MapResult. There was a scenario that caused unnecessary attaches to this database. That performance issue has been eliminated.

Making the JDBC Interface Writes More Efficient

The JDBC interface for Version 3.6 provides better handling of batches with both updates and inserts.

When the Loader reads data from the LogMiner, there is an indication that the record is either a delete or a modify. A modify may be either an insert of a new row or an update to an existing row. The Loader tries an update and, if the row is not found, does an insert.

The Loader may be set to include multiple transactions in what is written to the target in a single transaction. (See Keyword: Checkpoint in the full documentation.) When using the JDBC target, it is also possible to set the JDBC batch size.

When a batch includes an insert for a row and, later, includes an update for that row, the JDBC code will first send the batch to the UPDATE statement. Since the row does not yet exist, both the insert and the update rows will have no affect. Both rows will next be sent to the INSERT statement. The first record will succeed, but the second will fail due to a duplicate value.

When this happens, in the previous release, the Loader temporarily disabled batch mode and sent each record, one at a time, to the update and failing that the insert. The data was eventually correct. However, each row in the batch was processed, including rows that were already updated.

That approach meant unnecessary work and obscuring of temporal markers that are important for some applications.

For Version 3.6 of the Loader, the solution is to keep track of which records in the buffer have been successfully processed so that if the Loader needs to fall back and process rows individually, the Loader will only process ones that have not yet been successful.

Expansion of DB Info

Beginning with Version 3.6, the JCC_GET_DB_INFO will set the symbol RDB_VERSION_SET as shown in the table.

TABLE 4. Values for the RDB_VERSION_SET

Value	Meaning
“”	The database was not found.
“True”	The process Rdb version context is the same as the database version.
“False”	The process Rdb version context is different from the database version.

Remove I2 Build

Beginning with Version 3.6, the useless I2 build variant is no longer included in the Loader. To protect backward compatibility, the start up procedure (JCC_DBA_STARTUP.COM) still accepts specification of I2, but uses the standard IPF images.

Logical Names

The keywords in the Control File provide one way of controlling the operation of the Loader. Logical Names provide another.

The full documentation includes descriptions of many logical names and a summary of them in an appendix.

There are a variety of logical name enhancements in Version 3.6.

Turn Off Data Pump Message

The Data Pump is a widely used portion of the JCC LogMiner Loader kit. See the Data Pump chapter in the full documentation for more complete information.

The Data Pump can be too informative. Administrators have, sometimes, wanted a way to turn off the informational message "No rows found matching the selection criteria." Beginning with Version 3.6, the logical name JCC_LML_DP_TRACE_NO_ROWS can be set to '1' or 'E' or 'T' (case sensitive) to continue the default behavior, while setting it to any other value will disable the message.

Delta Dates

By default, delta dates will appear as negative dates. Setting the logical name JCC_LML_NULL_DELTA_DATE to '1' or 't' or 'T' will set any date column that is negative to NULL.

Reducing the Buffer Count

The logical name RDM$BIND_BUFFERS can be used to set the Oracle Rdb database buffer count. The JCC LogMiner Loader now sets this logical name to limit the number of buffers allocated to five for the Continuous LogMiner (CLM) source database attach. Reducing the buffer count to five is appropriate because the CLM process does not require many database buffers to perform its work.

This logical name can be defined by the Loader Administrator as is appropriate to the Loader session. The use is specific to Rdb Loader targets that are local to the machine on which the Loader session is running.

Set Defaults for Logical Names Used with Java

There are logical names that can be set to control how Java is used. These have some convenient defaults that are automatically set beginning with

Version 3.6. If any of these are already defined, it is assumed that the Administrator wishes to override the defaults and the default is not set.

TABLE 5. Logical Name Defaults for Java

Logical Name	Default	Effect
JAVA$FORCE_IEEE_FPSR	1	Meets the requirement for Java garbage collector to work on IPF Java 1.6.0-5 and above.
DECC$STDIO_CTX_EOL	1	Removes the extraneous linefeed from Java logging.
DECC$FILENAME_UNIX-_NO_VERSION	1	Makes changes in how C CRTL routines format VMS files as Unix files.

Minimize DCL Exceptions

In some computing environments, DCL symbols have been defined in ways that generated exceptions in the JCC LogMiner Loader command procedures. The Loader runs on OpenVMS and using DCL in its command procedures.

Therefore, with this release changes were made to the Loader command procedures to minimize conflict with symbol definitions in the computing environments where the Loader is utilized.

Increased Logging of Control File

When an include_file is encountered in the Control File, the name of the file being opened is echoed in the log file.

Checkpoint and Inactive Sessions

A session may be active, but have few rows that are intended for the Loader target, as it is defined in the Control File. Such a session still needs to have the Loader checkpoint moved. This need is addressed by checkpointing the current information whenever an AIJ switch is encountered on a “no-work” transaction. “No work” transactions are those that contain no rows that cause the Loader - with the definitions in the Control File - to publish anything to the target. See the full documentation for more.

Prefix for the Log File Lines

In the batch file that is the Loader’s log, it is useful to have timing information as a prefix to each line in the file. For benefit of Loader support, the Loader automatically sets the DCL prefix of a batch to “!%T” which provides the time. In regression testing of the Loader, a prefix was set to provide the entire timestamp with both time and the date.

Since Loader Administrators might have reasons to set a particular or expanded prefix, the Loader has been modified to set the prefix only if it is not already set to include either time (“!%T”) or timestamp (“!%D”).

Decoding the Thread States

This section is included here so that it provides information without interrupting the description of enhancements to the JCC LogMiner Loader with Version 3.6.

The Statistics monitor has two versions of displaying information on the status of each thread. The single character code is used on most report types. A longer description is used on the State report type.

Single Character Codes for Thread Status

State	Meaning	Used for
R	thread is waiting to get the 'read' lock (the one that controls access to the input mailbox)	Read Phase
z	thread has read lock, but is waiting for input	
<	thread is currently reading from the input mailbox	
*	there is data in-flight and the thread is not currently waiting for any Loader resource (may also occur during write phase)	
W	thread is waiting for dbkey locks to be granted prior to the write phase	Write Phase
s	thread is currently sorting the buffered input data	
>	thread is currently writing to the target	
*	there is data in-flight and the thread is not currently waiting for any Loader resource (may also occur during reads)	
T	thread is waiting as part of a realtime throttle	
t	thread is waiting as part of a fixed throttle	
d	thread is waiting as part of a retry delay	
|	thread has reached lock threshold and is waiting for WriteLock in protected write mode	Stall states related to processing extremely large transactions
]	thread has requested WriteLock in concurrent write mode and is blocked by a process waiting for WriteLock in protected write mode	
.	(period) Loader is active, but not in any of the above states	
	(space) the specified thread is not running. (This will only happen if there is a thread that is displayed to the right of the space and if the Loader is configured with dynamic startup/ shutdown of threads. This is a characteristic of threads starting or stopping.)	

This chart has not changed with Version 3.6.

The full documentation includes information on setting a throttle, retry delay, and other items that will explain the chart further.

Descriptions of the Thread States

The Report Type State displays a longer and more complete description of the thread state.

Some of these descriptions are specific to a Loader target type. The chart begins with the ones that are appropriate to all Loader target types and continues for specific target types. (The chart includes displays that will only be seen with the separately licensed Kafka Option.)

Variables. Note that the items to be displayed include information in angle brackets that will be replaced on the screen with the appropriate variable information. For example, <target type> means that the display will show the target type, <checkpoint type> means that the display will show the checkpoint type, <#of records> means that the display will show the number of records, etc.

Latency. Each line also includes latency information as the first column. The chart to follow includes some aids to understanding the latency data.

Color Coding. The chart includes colors that provide additional information.

Light red in the Displayed column identifies items that are likely to be transient because that portion of the operation should be brief. An example is Open Input.

Light blue in the meaning column identifies items that are part of the Input phase.

Light yellow in the meaning column identifies items that are part of the Output phase.

Displayed	Meaning
LML	Transient state between input and output. Latency shown is since the thread is started.
Initialize	Executing initialization; reading Control Files, mapping global sections, etc.

Displayed	Meaning
Open Input	Opening the data input stream - either the CLM mailbox or the file containing data that was previously processed by the LogMiner.
Input	In the input phase and transitioning from one sub-phase to another. Latency shown is since the time the thread entered the input state.
Input->Read MBX	Has the mailbox (MBX) lock and is either in overhead processing or actively processing data from the LogMiner.
Input->MBX wait	Waiting for another thread to release the mailbox (MBX) lock.
Input->MBX release	Releasing the MBX (mailbox) lock
Input->CLM wait	Waiting for data to read from the CLM mailbox.
Input->Verify record	Verifying a record read from the CLM mailbox.
Input->Buffer data	Adding the verified record to the data buffers.
Output	In the output phase and transitioning from one sub-phase to another. Latency shown is since the time the thread entered the output state.
Create Output	Creating the output stream to the target.
Output->Synch wait	Waiting to synchronize locks. Options are waiting for a lock on one or more originating dbkey (or row) locks or waiting for a lock at the locking level declared by JCC_LML_LOCKING_LEVEL logical name.
Output->Write wait	Queueing a request for the write lock.
Output->Write release	Releasing the write lock.
Output->Process buffer	Processing the buffered records. Latency shown is frrom the time that all necessary locks were received and processing started.
Output-.>Protected write wait	Holding a protected write lock and waiting to write until all other threads complete their writes.
Output->Concurrent write wait	Waiting for other threads seeking protected access to the target.
Output->Realtime throttle wait	Waiting as instructed by the JCC_LOGMINER_LOADER_THROTTLE and JCC_LOGMINER_LOADER_THROTTLE_REALTIME_PERCENTAGE which can be set by the Administrator.
Output->Record	Preparing a record to be written, including FilterMap and MapResult processing and target specific formatting.
Output->Message	Preparing to write to the target. Whether what is written is a record or a message depends on the definition of the <message contents> parameter of the Output keyword.
Output->Send message	Finding the target routines to write.

Displayed	Meaning
Filter&Result	Applying a filter or modifying the output as directed by MapResult keyword.
Checkpoint	Determining the routine appropriate to write checkpoint data.
VA->Create	Writing a virtual array.
VA->Sort	Sorting a virtual array.
VA->Append	Appending one virtual array to another.
VA->Position	Re-positioning the current record in a virtual array.
VA->Read	Reading from a virtual array.
VA->Write	Writing to a virtual array.
Open-><target type>	Calling into the target specific shareable image to open the output stream.
Write-><target type>	Calling into the target specific shareable image to write to the output stream.
SetTrans-><target type>	Calling into the target specific shareable image to set a transaction on the output stream.
Rollback-><target type>	Calling into the target specific shareable image to rollback.
Commit-><target type>	Calling into the target specific shareable image to commit.
Close-><target type>	Calling into the target specific shareable image to close the output stream.
WriteChkpt-><checkpoint type>	Calling into the target specific shareable image to write checkpoint data to the checkpoint stream.
ReadChkpt-><checkpoint type>	Calling into the target specific shareable image to read checkpoint data from the checkpoint stream.
Java (for JDBC or Kafka)	
CreateJavaVM	Creating the Java Virtual Machine (JVM).
GetJavaEnv	Retrieving the Java Native Interface (JNI) to handle the created JVM.
AttachCurrentThread	Connecting the threads JNI to the JVM.
JDBC	
<jdbc_connect_string>	Using the connect string specified in keyword JDBC~connect~<string name> to connect via the JVM to the target.
<table>[batch:<#of records>]	Outputing the number of records indicated to the table indicated.

Displayed	Meaning
Commit[rows:<#of records>]	Commiting the number of records indicated.
Rollback[rows:<#of records>]	Rolling back a transaction with the number of records indicated.
closeConnect	Closing the connection to the target.
Rdb	
Rdb->Connect	Executing the Rdb-specific SQL to attach to the database.
Rdb->Disconnect	Executing the Rdb-specific SQL to disconnect from the database.
Rdb->Prepare	Preparing an SQL statement for use with the target.
<table>	Executing Rdb-specific SQL statement to replicate the specified table to the target.
Rdb->Commit	Committing the transaction to the database.
Rdb->Rollback	Rolling back the transaction.
Rdb->SetTransaction	Setting a transaction.
Rdb->WriteCheckpoint	Writing a checkpoint to the target.
Rdb->ReadCheckpoint	Reading a checkpoint.
OCI (Oracle)	
OCI->Connect	Executing the Oracle-specific SQL to attach to the database.
OCI->Disconnect	Executing the Oracle-specific SQL to disconnect from the database.
OCI->ValidateMetadata	Validating the metadata within the Oracle target.
<table>	Executing Oracle-specific SQL statement to replicate the specified table to the target.
OCI->commit	Committing the transaction to the database.
OCI->Rollback	Rolling back the transaction.
OCI->SetTransaction	Setting a transaction.
OCI->ReadCheckpoint	Reading a checkpoint.
OCI->DeleteCheckpoint	Removing unnecessary checkpoint data.
OCI->InsertCheckpoint	Inserting checkpoint data.
OCI->UpdateCheckpoint	Updating checkpoint data.
Tuxedo	

Displayed	Meaning
tpalloc(TPINIT)	Allocating memory for the Tuxedo connection.
tpalloc(FML32)	Allocating memory for a data buffer.
tpfree	Freeing memory that is no longer needed.
tpchkauth	Checking authorization requirements.
tpenqueue:<qspace>:<table>:[buffer:<#of records>]	Enqueueing a data buffer with the number of rows, qspace, and table shown. Display is tpenqueue:<qspace>:<table>:[buffer:<#of records>] with no wrap.
tpcall:<table>:[buffer:<#of records>]	Calling a Tuxedo application routine for the table, data buffer, and number of rows shown.
tpcall:<table>[<buffer#>][buffer:<#of records>]	Asynchronously calling a Tuxedo application routine for the specified table with a data buffer of the specified number of rows. The buffer# is the count of buffers previously sent within the context of the current commit interval.
tpgetrply:<table>[<buffer#>][buffer:<#of records>]	Asking the Tuxedo application for the status of the buffer for the table, buffer, and number of rows shown.
tpcancel	Cancelling an asynchronous call to a Tuxedo application.
tpbegin	Starting a transaction.
tpcommit	Committing a transaction.
tpabort	Aborting a transaction.
tpterm	Disconnecting from a Tuxedo application.
Kafka (only available with a license for the JCC LML Kafka Option)	
newKafkaProducer	Creating a new Kafka Producer connection.
initTransactions	Initializing the Kafka transaction manager.
beginTransaction	Beginning a transaction.
send(<topic>(<recordID>))	Sending the record shown to the topic shown.
flushBuffers	Flushing buffers.
<topic>(<recprdID>).get()	Asking for acknowledgement that the buffer has been received by the topic in order to free some memory when memory resources are low for the JVM.
abortTransaction	Requesting an abort.
commitTransaction[recs:<#of records>]	Requesting a commit. Number of records shows how many rows were written to the target in this commit interval.

Displayed	Meaning
closeConnect	Requesting closing the connection.