DBMoto®

Setup Notes for IBM® Db2® for i Transactional Replications

Version 9.5.0.2

Software Release Date: 4/9/18

HiT Software, Inc.
4040 Moorpark Ave
Suite 221
San Jose, CA 95117

T +1 408-345-4001
F +1 408-345-4899
info@hitsw.com
www.hitsw.com

Document History

Version	Author	Date	Reviewer	Date	Approver	Date	Comments
1	JHLorenzin	5/8/18	ALaspertini		VFarruggio		

Table of Contents

These notes provide essential information for setting up replications using **IBM Db2 for i (AS/400 or iSeries)** as a source database for one-way mirroring and synchronization**.** The setup process for a refresh replication can usually be completed using the DBMoto wizards without additional documentation because it does not involve access to the Db2 journals. For complete details on the setup process, check the *DBMoto User Guide* available from the Management Center **Help** menu or the *DBMoto Setup Guide*, available for download in the Help Center.

IBM Db2 for i/iSeries/AS400 User Permissions

When setting up replications that use IBM Db2 for i/iSeries/AS400 as either a source or target database, you need to be sure that the user ID used for making connections to the database has sufficient privileges to complete all the operations required for DBMoto to perform a replication.

This section is organized by the type of replication you want to perform. It describes in detail all the user authorities required during the setup and execution of replications.

Refresh with Db2 as Either Source or Target Database

1. AUTHORITY TO SELECT A CATALOG
   To display a list of tables and show fields in the table in the Management Center (for selecting a source or target table and for setting which fields to replicate), DBMoto runs a SELECT_CATALOG command. If the user ID has insufficient privileges, an error is generated on the Db2 server. Set the *ALL authority for tables that will be involved in replication using the following syntax:

   ```
   GRTOBJAUT OBJ(<LIB>) OBJTYPE(<TYPE>) USER(<UID>) AUT(<AUT>)
   ```

   Example:

   ```
   GRTOBJAUT OBJ(JACKC) OBJTYPE(*LIB) USER(DBMOTO6) AUT(*ALL)
   ```

2. AUTHORITY TO SELECT A TABLE
   DBMoto runs a SELECT statement to identify records to replicate. Therefore, the user ID used to make a connection must have adequate authority to run a SELECT statement for tables involved in replication.

   To verify existing authorities for a user ID, you can run the EDTOBJAUT command, where <TYPE> can be *LIB, *FILE, etc.

   ```
   EDTOBJAUT OBJ(<LIB>/<TABLE>) OBJTYPE(<TYPE>) USER(<UID>)
   ```

   Example:

   ```
   EDTOBJAUT OBJ(MYLIB/TEST) OBJTYPE(*FILE)
   ```

   Then you can change authority or add a new user with the command:

   ```
   GRTOBJAUT OBJ(<LIB>/<TABLE>) OBJTYPE(<TYPE>) USER(<UID>) AUT(<AUT>)
   ```

   Example:

   ```
   GRTOBJAUT OBJ(MYLIB/TEST) OBJTYPE(*FILE) USER(DBMOTO) AUT(*USE)
   ```

3. AUTHORITY TO UPDATE A TABLE, CREATE A TABLE
   To create a target table in the Management Center (as part of the Create Table Wizard), DBMoto uses the UPDATE_TABLE and CREATE_TABLE commands. The authority (<AUT>) needed for modifying a table is

*CHANGE. To verify existing authorities for a user ID, you can run the EDTOBJAUT command, where <TYPE> can be *LIB, *FILE, etc. You can then run the GRTOBJAUT command to set the authority to *CHANGE.
Example:

```
EDTOBJAUT OBJ(MYLIB/TEST) OBJTYPE(*FILE) USER(DBMOTO)
GRTOBJAUT OBJ(MYLIB/TEST) OBJTYPE(*FILE) USER(DBMOTO) AUT(*CHANGE)
```

4. To create a table, the *CHANGE authority needs to be assigned at the library level. If the library is a collection, you also need permissions to define journaling, because creating a table will automatically assign a journal to the new table.
5. For journaling, the privileges held by the authorization ID of the statement must include at least one of the following:

   a. The following system authorities:

      *USE to the Create Physical File (CRTPF) command

      *EXECUTE and *ADD to the library into which the table is created

      *OBJOPR and *OBJMGT to the journal

      *CHANGE to the data dictionary if the library into which the table is created is an SQL schema with a data dictionary

   b. Administrative authority

6. AUTHORITY TO DROP A TABLE, OR ALTER A TABLE (optional)
   The use of these commands from within DBMoto is entirely optional (i.e. not necessary for running a refresh replication.) They are used if you choose to remove a table from Db2 or change the table via the Management Center SQL Query tab. The authority (<AUT>) needed for modifying or dropping a table is *CHANGE. To verify existing authorities for a user ID, you can run the EDTOBJAUT command, where <TYPE> can be *LIB, *FILE, etc. You can then run the GRTOBJAUT command to set the authority to *CHANGE.
   Example:

```
EDTOBJAUT OBJ(MYLIB/TEST) OBJTYPE(*FILE) USER(DBMOTO)
GRTOBJAUT OBJ(MYLIB/TEST) OBJTYPE(*FILE) USER(DBMOTO) AUT(*CHANGE)
```

Transactional Replications/Initial Refresh with Db2 as Either Source or Target Database

This section includes information for mirroring where IBM Db2 for i is the data source, and synchronization where IBM Db2 for i can be either the “source” or “target” data source.

1. AUTHORITY TO SELECT A CATALOG
   To display a list of tables and show fields in the table in the Management Center (for selecting a source or target table and for setting which fields to replicate), DBMoto runs a SELECT_CATALOG command. If the user ID has insufficient privileges, an error is generated on the Db2 server. Set the privileges for tables that will be involved in replication using the following syntax:

```
GRTOBJAUT OBJ(<LIB>) OBJTYPE(<TYPE>) USER(<UID>) AUT(<AUT>)
```

Example:

```
GRTOBJAUT OBJ(JACKC) OBJTYPE(*LIB) USER(DBMOTO6) AUT(*ALL)
```

2. AUTHORITY TO SELECT A TABLE
   DBMoto runs a SELECT statement to identify records to replicate. Therefore, the user ID used to make a connection must have adequate authority to run a SELECT statement for tables involved in replication.
   To verify existing authorities for a user ID, you can run the EDTOBJAUT command, where <TYPE> can be *LIB, *FILE, etc.

```
EDTOBJAUT OBJ(<LIB>/<TABLE>) OBJTYPE(<TYPE>) USER(<UID>)
```

   Example:

```
EDTOBJAUT OBJ(MYLIB/TEST) OBJTYPE(*FILE)
```

   Then you can change authority or add a new user with the command:

```
GRTOBJAUT OBJ(<LIB>/<TABLE>) OBJTYPE(<TYPE>) USER(<UID>) AUT(<AUT>)
```

   Example:

```
GRTOBJAUT OBJ(MYLIB/TEST) OBJTYPE(*FILE) USER(DBMOTO) AUT(*USE)
```

3. AUTHORITY TO UPDATE A TABLE, OR CREATE A TABLE
   To create a target table in the Management Center (as part of the Create Table Wizard), DBMoto uses the UPDATE_TABLE and CREATE_TABLE commands. The authority (<AUT>) needed for modifying a table is *CHANGE. To verify existing authorities for a user ID, you can run the EDTOBJAUT command, where <TYPE> can be *LIB, *FILE, etc. You can then run the GRTOBJAUT command to set the authority to *CHANGE.
   Example:

```
EDTOBJAUT OBJ(MYLIB/TEST) OBJTYPE(*FILE) USER(DBMOTO)
GRTOBJAUT OBJ(MYLIB/TEST) OBJTYPE(*FILE) USER(DBMOTO) AUT(*CHANGE)
```

   To create a table, the *CHANGE authority needs to be assigned at the library level. If the library is a collection, you also need permissions to define journaling, because creating a table will automatically assign a journal to the new table.

   For journaling, the privileges held by the authorization ID of the statement must include at least one of the following:

   a. The following system authorities:

      *USE to the Create Physical File (CRTPF) command

      *EXECUTE and *ADD to the library into which the table is created

      *OBJOPR and *OBJMGT to the journal

      *CHANGE to the data dictionary if the library into which the table is created is an SQL schema with a data dictionary

   b. Administrative authority

4. AUTHORITY TO DROP A TABLE, OR ALTER A TABLE (optional)
   The use of these commands from within DBMoto is entirely optional (i.e. not necessary for running a transactional replication.) They are used if you choose to remove a table from Db2 or change the table via the Management Center SQL Query tab. The authority (<AUT>) needed for modifying or dropping a table is *CHANGE. To verify existing authorities for a user ID, you can run the EDTOBJAUT command, where <TYPE> can be *LIB, *FILE, etc. You can then run the GRTOBJAUT command to set the authority to *CHANGE.
   Example:
   ```
   EDTOBJAUT OBJ(MYLIB/TEST) OBJTYPE(*FILE) USER(DBMOTO)
   GRTOBJAUT OBJ(MYLIB/TEST) OBJTYPE(*FILE) USER(DBMOTO) AUT(*CHANGE)
   ```

5. AUTHORITY TO SET UP A TRANSACTION
   (**Note:** Not needed when IBM Db2 for i is a target in mirroring replications)
   When setting up the connection for transactional replications (where Db2 for i is serving as the source of data, either for mirroring or for synchronization), you need additional privileges that are not required for replication operations. You need a login that has write permissions and QSECOFR privileges. Additionally, you need special authorities: save system authority (*SAVSYS) to restore the save file to the system; and all object authority (*ALLOBJ) to work with system resources and run the RSTLIB command.
   ```
   CHGUSRPRF USRPRF(<UID>) USRCLS(*SECOFR) SPCAUT(*SAVSYS *ALLOBJ)
   ```
   Example:
   ```
   CHGUSRPRF USRPRF(DBMOTOUSR) USRCLS(*SECOFR) SPCAUT(*SAVSYS *ALLOBJ)
   ```

6. AUTHORITY TO SET UP REPLICATIONS
   (**Note:** Not needed when IBM Db2 for i is a target in mirroring replications)
   This command is also used by DBMoto when setting up your environment for transactional replication (where Db2 for i is serving as the source of data, either for mirroring or for synchronization). It requires SELECT_TABLE permissions on the source table. Additionally, it requires *ALL permissions to access the journal:
   ```
   GRTOBJAUT OBJ(<JRNLIB>/<JRNNAME>) OBJTYPE(*JRN) USER(<UID>) AUT(*ALL)
   ```
   Example:
   ```
   GRTOBJAUT OBJ(V5/QSQJRN) OBJTYPE(*JRN) USER(DBMOTOUSR) AUT(*ALL)
   ```

7. AUTHORITY TO ACCESS JOURNALS
   (**Note:** Not needed when IBM Db2 for i is a target in mirroring replications)
   When running transactional replications (where Db2 for i is serving as the source of data, either for mirroring or for synchronization), authorities in 5 and 6 are no longer needed but DBMoto still needs to access the source table (SELECT TABLE permissions on the source table) and *ALL authority for the journal.
   ```
   GRTOBJAUT OBJ(<JRNLIB>/<JRNNAME>) OBJTYPE(*JRN) USER(<UID>) AUT(*ALL)
   ```
   Example:
   ```
   GRTOBJAUT OBJ(V5/QSQJRN) OBJTYPE(*JRN) USER(DBMOTOUSR) AUT(*ALL)
   ```

IBM Db2 for i System Settings

Journals and Receivers

When performing mirroring or synchronization with a Db2 system source table, you need to manage the journals and receivers for your source tables. Typically, your system administrator manages journals and receivers, but it is helpful to know a little about the operations involved.

A journal is a collector of modified data from "journaled" files. The modifications that occurred in the files are detailed and written in a receiver as a log of the operations performed on the physical file.

The terminology is a little misleading: a journal is not, as you would expect, the place where modifications are tracked, but only the reference to write them on a receiver. A receiver is the physical location where traced modifications are written.

When a journal is started, it must be associated with a receiver. Receivers are set to a defined size that can be configured to fit your needs. The receiver is "bound" to the journal.

A physical file cannot be associated with more than one journal at the same time, like a journal cannot be associated with more than one receiver at a time. However, the same journal can track information for many physical files. If you want to change the current journal/receiver setting for a file, you can stop the journaling for that file and associate the file with a different one, but not at the same time.

Logical files (view, indexes, etc.), as well as LOB data types, are not journaled. Because a file must be journaled to be replicated in mirroring or synchronization modes, logical files can only be replicated in refresh mode.

Journal and Receiver Names

Use receiver names in which the last 4 or 5 character of the name are digits, starting with 00001, like

```
DBRSJ00001
```

Then create a new receiver with a command like

```
CRTJRNRCV JRNRCV(DBRSWRK/DBRSJ00001) AUT(*ALL)
```

and the journal with the command

```
CRTJRN JRN(DBRSWRK/DBRSJ) JRNRCV(DBRSWRK/DBRSJ00001) AUT(*ALL)
```

Note the names of journal and receiver: the second the second is the same as the first with a "00001" suffix.

Finally, use the command

```
CHGJRN JRN(DBRSWRK/DBRSJ) JRNRCV(*GEN)
```

The option (*GEN) enables automatic naming for receivers based on last number + 1.

If you need to have "different" names for journal and receivers, you can set the automatic naming management using the following command:

```
CHGJRN JRN(DBRSWRK/DBRSJ) MNGRCV(*SYSTEM)
```

(which is automatically set when using the JRNRCV(*GEN) option)

Using automatic naming management, MNGRCV(*USER), for receivers is useful because when the current receiver ("attached" receiver) becomes full, the system will automatically unbind the attached receiver, then create and bind a new receiver with a name based on last number + 1, i.e.

DBRSJ00001 --> DBRSJ00002

Activating a Journal

STRJRNPF FILE(DBRSWRK/TableFILE) JRN(DBRSWRK/DBRSJ) IMAGES(*BOTH) OMTJRNE(*OPNCLO)

The IMAGES parameter can be set to either *BOTH or *AFTER. HiT Software recommends that you set the IMAGES parameter to *BOTH. This option saves the record's image in the log, before and after the update command, and is requested by DBMoto in order to correctly manage the record's primary key. If you choose to set the IMAGES parameter to *AFTER, you will need to modify the target table by adding a Relative Record Number (RRN) field and mapping it to the source table !RecordID field as follows:

1. Add an Int field on the target table and make it the primary key.
2. Create source and target connections for the replication.
3. Define the replication.
4. When mapping source and target fields, right click on the target table field you want to map the RRN to, and choose Map to Expression...
5. In the upper pane of the Expression Editor, type [!RecordID].
6. Click **OK** to save the expression.

Note: You should not set the IMAGES parameter to *AFTER if you are planning to perform synchronization (bi-directional mirroring). Also if the source table is reorganized, you need to run a refresh replication on the target table (to update RRN changes) before mirroring can proceed again.

Receiver Management

When a receiver is unbound for any reason (because it becomes full, manual management, system management), you can choose what to do with it between:

- Unbind it and let it remain on disk
- Unbind and delete it

This assumes that you have automated the receiver management as described earlier, and therefore excludes a manual intervention: you are giving the system the appropriate instructions to manage receivers by itself.

To allow the receiver to be unbound but not deleted, run the following command on your journal object:

CHGJRN JRN(DBRSWRK/DBRSJ) DLTRCV(*NO)

The default is *YES. Using this setting, the system will unbind old receivers and keep them available on disk. The unbound receivers available on disk are called "online" (their status), while the unique and currently bound receiver is called "attached".

If you need to change the system management to delete old receivers when they're unbound, you need to run

CHGJRN JRN(DBRSWRK/DBRSJ) DLTRCV(*YES)

Sequence Number for Journal Entries

A receiver is the final container which keeps track of every operation (transaction) performed on the physical files with which they are associated via the journal. Every operation tracked is completely described and numbered with a list of details - date, time, kind of operation, etc. - and a numeric progressive value, named "Sequence number".

You can choose to set the sequence number for journal entries to continue among several receivers or to be reset at every receiver change. Run the command

```
CHGJRN JRN(xxx) JRNRCV(*GEN) SEQOPT(*CONT)
```

to allow your journal entry sequence number to continue in the subsequent receivers when the attached receiver is unbound.

Normally, when you change journal receivers, you continue the sequence number for journal entries. When the sequence number becomes very large, you should consider resetting the sequence to start the numbering at 1.

You can reset the sequence number only when all changes are forced to auxiliary storage for all journaled objects and commitment control is not active for the journal. Resetting the sequence number has no effect on how the new journal receiver is named.

If you use system journal-receiver management for a journal, the sequence number for the journal is reset to 1 whenever you restart the system or vary on the independent disk pool containing the journal. When you restart the system or vary on an independent disk pool, the system performs the change journal operation for every journal on the system or disk pool that specifies system journal-receiver management.

The operation that the system performs is equivalent to

```
CHGJRN JRN(xxx) JRNRCV(*GEN) SEQOPT(*RESET)
```

The sequence number is not reset if journal entries exist that are needed for commitment control IPL recovery.

The maximum sequence number is 2147483136. If you specified RCVSIZOPT(*MAXOPT1) or RCVSIZOPT(*MAXOPT2) for the journal that you attached the receiver to, then the maximum sequence number is 9999999999. If this number is reached, journaling stops for that journal.

When the size limit is reached (or a manual intervention occurs), a system-managed receiver is unbound. If the delete option is disabled, it becomes an "online" receiver, meaning it is not attached but still available.

Sequence Numbers and DBMoto Mirroring

DBMoto keeps all the information about the current status of the replication, including the journal entry number and corresponding receiver, in the metadata.

It is important to have the sequence number for journal entries to continue from receiver to receiver because DBMoto, when replicating in mirroring mode from the i/iSeries/AS400, performs comparisons between the last used (mirrored) journal entry and the current one. If the replicator detects a current value lower than the last managed value (stored in the metadata), it stops replicating and reports a message saying that a potential problem has occurred.

For this reason, it is important to keep old receivers on disk without deleting them (keep them "online".) DBMoto stores journal entry numbers so that if a change receiver (or more than one) occurs and the last managed entry is

now in an unbound receiver but the receiver is still online, the data are retrieved and the mirroring process continue without problems. If, instead, the last managed journal entry is in a deleted receiver, a potential problem is signalled in the log because something unexpected has occurred, and transactions could be missing. DBMoto also stops replicating the source files associated with that journal,

The information stored in an online receiver is still retrievable using simple i/iSeries/AS400 CL commands. DBMoto uses these commands. Deleted receivers cannot be used at all, so if they are not yet processed, the information they hold is lost; and, when mirroring, even a potential loss of information is a problem.

Even if a new receiver is created and bound, it does not help. If, for instance, the Data Replicator was stopped BEFORE it had processed all the journal entries in the old receiver, data may already be (potentially) missing. As a general rule, it is appropriate for the Data Replicator to signal an error and stop.

A contextual stop for DBMoto when an IPL is performed or backup operations are in place on the i/iSeries/AS400 is a very good rule. Journaling and backup at the same time can often lead to problems.

Special Cases

In some real-life deployment cases, DBMoto administrators have decided to keep the receiver deletion when the receiver is changed to manage the whole process in a different manner.

- Stopping the Data Replicator at night
- Creating and scheduling a script to change the Replication status from "Mirroring" to "Refresh to run". This impacts metadata contents.
- Creating and scheduling a script to re-synch the journal reading the current status (the process that you usually need to do manually via Management Center), so no need to search for old receivers and no loss of data because the Data Replicator starts processing only journal entries for "today".
- Restarting DBMoto in the morning: it will start to operate with a brand new refresh

Of course, this kind of management can be safely done when the amount of data to refresh is not huge.

Add a Source Connection Wizard

Your DBMoto installation includes Ritmo/i, a .NET Data Provider to connect to IBM Db2 for i (AS/400 or iSeries). Configure your connection to Db2 from within the DBMoto Management Center using the information below. More information on the steps to configure a connection can be found in the DBMoto User Guide (available from the Management Center) or the DBMoto Setup Guide. The following field(s) require specific information for IBM Db2 for i.

User ID

The **User ID** field should contain a user ID with the following settings:

- Authority to run the DSPOBJD command on the schema used for replication
- Authority to access the following system tables:

  QSYS2.SYSTABLES

  QSYS2.SYSCOLUMNS

  QSYS2.SYSCST

QSYS2.SYSKEYCST

- Additionally, when the system is used as source in mirroring or synchronization mode, the user ID needs access to:
    - DSPJRN (on the journals/receivers involved in replication)
    - DSPFD (on the tables involved in replication)
    - DSPFFD (on the tables involved in replication)

For Synchronization Replications:
The login/user ID that you provide must be unique to DBMoto. It should not be used for any transactions occurring in either database involved in the synchronization. DBMoto does not replicate transactions by the user you specify in this connection. This user ID is used by DBMoto during synchronization to read the database logs and perform the synchronization operations. Therefore, any transactions found in the logs with this user ID are not replicated as part of the synchronization data.

Enable Transactional Replication Wizard

For transactional replications (mirroring and synchronization), use the Enable Transactional Replication wizard after setting up a source connection. The following field(s) require specific information for IBM Db2 for i.

Log Type

Select whether you plan to perform replications using the Log Reader (default) Log Server Agent or Log Reader API (required when replicating tables with LOB values. Available for Db2 for i V6R1 and above.)

Library

For mirroring or synchronization replications, you first need to set up a library on your Db2 system. This screen helps you to set up the library by transferring a savf file to the Db2 server and creating the DBMOTOLIB library (or a library name of your choosing). You can also perform this procedure manually for more complete control over operations or in case the automatic process does not work.

Library Name/Version
The library name, DBMOTOLIB, is supplied. This field can be modified to supply a different library name. This is the name of the library that will be created on the Db2 system. It can be useful to change the default library name when, for example, you have two DBMOTO installations using the same database server, and you wish to keep separate libraries for each installation. The Version field displays the DBMOTOLIB version number—it is not editable

Savf File
The location of the savf file in the DBMoto installation directory. This file is run to create a stored procedure on the Db2 system, and the version of the file should match the operating system that you are using. Modify this path to point to a different file version if needed, or if you move the savf file to a different location.

Create Replication Wizard

After creating a connection, and setting up for transactional replications, create the replication. The following fields require specific information for IBM Db2 for i.

Source Log Info

Journal
This field displays the journal that has been set for the source table on the IBM Db2 for i (iSeries/AS400) system. Typically, you do not need to edit this field.

Any journal specified in this field should usually have the IMAGES parameter set to *BOTH. This option saves the record's image before and after the transaction and makes available the primary key information needed by DBMoto. However, it is possible to set the IMAGES parameter to *AFTER, but you will need to modify the target table by adding a Relative Record Number (RRN) field and mapping it to the source table !RecordID field. For more information, see i/iSeries/AS400 System Journals and Receivers.

This information is available and can be changed in the Table Properties dialog after the wizard is completed.

Receiver
This field displays the receiver that has been set for the source table on the Db2 system. Typically, you do not need to edit this field. However, if you know that you want to begin replication from a specific receiver and transaction ID, you can enter values in this field and the Transaction ID field.

Transaction ID
Enter the sequence number, retrieved from the receiver, from which you want to start monitoring database transactions for replication. If you do not know the sequence number, click **Read** to open the Journal Read Point dialog. In this dialog, you can either retrieve the current sequence number or the sequence number for a specified date and time. If you enter a date and time, DBMoto retrieves the first sequence number after the time entered. This information is available and can be changed in the Replication Properties dialog after the wizard is completed.

Transaction Timestamp
This field is automatically filled in and reports the timestamp for the sequence identified in the Transaction ID field.

Read Interval (sec)
The frequency (in seconds) with which you want to check the log during replication. For example, if the setting is 90 seconds, DBMoto will check the log every 90 seconds to see if any transactions have occurred that need to be replicated to the target table. This information is available and can be changed in the Replication Properties dialog after the wizard is completed.

Create the Db2 Library Manually

If you are unable to create the library for your Db2 system automatically, you can access the appropriate savefile in the DBMoto ServerFiles folder and restore it manually as described below. Use the name of the restored library when configuring your System i connection in the Management Center. The default name provided in the Management Center is DBMOTOLIB, so either use this name in the instructions below, or be sure to change the name.

If more than one DBMOTO installation is sharing the same IBM i server, be sure to set up a different library for each installation.

Note that operating system version V3R2 or higher is required for DBMoto 6.0.0 or higher.

1. Create a temporary folder on your PC (e.g., C:\DBMLib)

2. Copy the appropriate savefile for your i operating system version from DBMoto's ServerFiles folder to C:\DBMLib. The savefiles are:

i/iSeries/AS400 Operating System Version	DBMoto Savefile
V3R2 up to and including V4R2	DBMLIB32.SAVF
V4R3 up to and including V5R0	DBMLIB43.SAVF
V5R1 up to and including V5R3	DBMLIB51.SAVF
V5R4 or above. For use with Log Reader transaction mode.	DBMLIB54.SAVF
V6R1 and above. For use with Log Reader API transaction mode.	DBMLIBAPI61.SAVF

3. Run the DOS command prompt and change the working directory to C:\DBMLib.
   C:>cd C:\DBMLib
4. Run an FTP session followed by the Db2 system IP address
   C:\DBMLib> ftp 111.111.111.111
5. Insert your username and password when prompted. Make sure that your user ID has write permissions and QSECOFR privileges.
6. Make QGPL the current directory.
   ftp> quote cwd QGPL
7. Create an empty Savefile on the Db2 system.
   ftp> quote rcmd CRTSAVF FILE(QGPL/DBMLIBSAVF) AUT(*ALL)
8. Switch to BINARY mode.
   ftp> bin
9. Transfer the savefile. In the command below, replace DBMLIB.SAVF with the name of the savefile you are using.
   ftp> put DBMLIB.SAVF DBMLIBSAVF
10. Restore the savefile, for example in a library called MYDBMOTOLIB:
    ftp> quote rcmd RSTLIB SAVLIB(DBMOTOLIB) DEV(*SAVF) SAVF(QGPL/DBMLIBSAVF) MBROPT(*ALL) ALWOBJDIF(*ALL) RSTLIB(MYDBMOTOLIB)
    Note: If using the default library name, DBMOTOLIB, be sure to replace only the library name in the command **RSTLIB(MYDBMOTOLIB). SAVLIB(DBMOTOLIB)** indicates the name of the library as saved in the SAVF file. The `RSTLIB` parameter instead indicates the name of the library where you want to restore the SAVF file which by default is the name of the library saved in the SAVF.

11. Delete the save file.
    ftp> quote dele DBMLIBSAVF
12. Close the ftp session.
    ftp> quit
13. Finally, create the stored procedure DBMOTOLIB.JRNSQNM on the Db2 system. This needs to be executed as a SQL command.
    Note: If using the IBM i console to perform this operation, use "/" instead of "." below in "DBMOTOLIB.JRNSQNM" to give you "DBMOTOLIB/JRNSQNM"

Operating System Versions up to and including V4R1

Use with DBMLIB32.SAVF

```
CREATE PROCEDURE DBMOTOLIB.JRNSQNM
(IN JOUR CHAR(10) ,
IN JLIB CHAR(10) ,
IN FNMS CHAR(900) ,
IN JDAT CHAR(8) ,
IN JTIM CHAR(6) ,
INOUT NUMSEQ DECIMAL(10, 0) ,
INOUT RECVR CHAR(10) ,
INOUT LIBRCV CHAR(10) ,
OUT LSTSQN DECIMAL (10, 0) ,
OUT LSTRECVR CHAR(10) ,
OUT LSTLIBREC CHAR(10) ,
OUT FLAG CHAR(1) ,
OUT CODC CHAR(7) ,
OUT MSGG CHAR(100) )
EXTERNAL NAME 'DBMOTOLIB/JRNSQNM'
LANGUAGE CL
PARAMETER STYLE GENERAL
```

Operating System VersionV4R2

Use with DBMLIB32.SAVF

```
CREATE PROCEDURE DBMOTOLIB.JRNSQNM
(IN JOUR CHAR(10),
 IN JLIB CHAR(10),
 IN FNMS CHAR(900),
 IN JDAT CHAR(8),
 IN JTIM CHAR(6),
 INOUT NUMSEQ DECIMAL(10,0),
 INOUT RECVR CHAR(10),
 INOUT LIBRCV CHAR(10),
```

```
 OUT LSTSQN DECIMAL(10,0),
 OUT LSTRECVR CHAR(10),
 OUT LSTLIBREC CHAR(10),
 OUT FLAG CHAR(1),
 OUT CODC CHAR(7),
 OUT MSGG CHAR(100))
LANGUAGE CL SPECIFIC DBMOTOLIB.JRNSQNM
NOT DETERMINISTIC
NO SQL
EXTERNAL NAME 'DBMOTOLIB/JRNSQNM'
PARAMETER STYLE GENERAL
```

Operating System Versions V4R3 up to and including V5R0M0
Use with DBMLIB43.SAVF

```
CREATE PROCEDURE DBMOTOLIB.JRNSQNM
(IN JOUR CHAR(10),
 IN JLIB CHAR(10),
 IN FNMS CHAR(900),
 IN JDAT CHAR(8),
 IN JTIM CHAR(6),
 INOUT NUMSEQ DECIMAL(10,0),
 INOUT RECVR CHAR(10),
 INOUT LIBRCV CHAR(10),
 OUT LSTSQN DECIMAL(10,0),
 OUT LSTRECVR CHAR(10),
 OUT LSTLIBREC CHAR(10),
 OUT FLAG CHAR(1),
 OUT CODC CHAR(7),
 OUT MSGG CHAR(100))
LANGUAGE CL SPECIFIC DBMOTOLIB.JRNSQNM
NOT DETERMINISTIC
NO SQL
EXTERNAL NAME 'DBMOTOLIB/JRNSQNM'
PARAMETER STYLE GENERAL
```

Operating System Versions V5R1 up to and including V5R3
Use with DBMLIB51.SAVF

```
CREATE PROCEDURE DBMOTOLIB.JRNSQNM
(IN JOUR CHAR(10),
 IN JLIB CHAR(10),
 IN FNMS CHAR(900),
```

```
 IN JDAT CHAR(8),
 IN JTIM CHAR(6),
 IN JCDE CHAR(100),
 INOUT NUMSEQ DECIMAL(10,0),
 INOUT RECVR CHAR(10),
 INOUT LIBRCV CHAR(10),
 OUT LSTSQN DECIMAL(10,0),
 OUT LSTTMSP CHAR(26),
 OUT LSTRECVR CHAR(10),
 OUT LSTLIBREC CHAR(10),
 OUT FLAG CHAR(1),
 OUT CODC CHAR(7),
 OUT MSGG CHAR(100))
LANGUAGE CL SPECIFIC DBMOTOLIB.JRNSQNM
NOT DETERMINISTIC
NO SQL
CALLED ON NULL INPUT
EXTERNAL NAME 'DBMOTOLIB/JRNSQNM'
PARAMETER STYLE GENERAL
```

Operating System Version V5R4 or greater:
Use with DBMLIB54.SAVF

```
CREATE PROCEDURE DBMOTOLIB.JRNSQNM
(IN JOUR CHAR(10),
IN JLIB CHAR(10),
IN FNMS CHAR(9000),
IN JDAT CHAR(8),
IN JTIM CHAR(6),
IN JCDE CHAR(100),
INOUT NUMSEQ CHAR(20),
INOUT RECVR CHAR(10),
INOUT LIBRCV CHAR(10),
OUT LSTSQN CHAR(20),
OUT LSTTMSP CHAR(26),
OUT LSTRECVR CHAR(10),
OUT LSTLIBREC CHAR(10),
OUT FLAG CHAR(1),
OUT CODC CHAR(7),
OUT MSGG CHAR(100))
LANGUAGE CL SPECIFIC DBMOTOLIB.JRNSQNM
NOT DETERMINISTIC
```

```
NO SQL
CALLED ON NULL INPUT
EXTERNAL NAME 'DBMOTOLIB/JRNSQNM'
PARAMETER STYLE GENERAL
```

Operating System Versions V6R1 and above

Required for transactional replication mode Log Reader API. Use with DBMLIBAPI61.SAVF

```
CREATE PROCEDURE DBMOTOLIB.JRNSQNM
(IN JOUR CHAR(10),
IN JLIB CHAR(10),
IN FNMS CHAR(9000),
IN JDAT CHAR(8),
IN JTIM CHAR(6),
IN JCDE CHAR(100),
INOUT NUMSEQ CHAR(20),
INOUT RECVR CHAR(10),
INOUT LIBRCV CHAR(10),
OUT LSTSQN CHAR(20),
OUT LSTTMSP CHAR(26),
OUT LSTRECVR CHAR(10),
OUT LSTLIBREC CHAR(10),
OUT FLAG CHAR(1),
OUT CODC CHAR(7),
OUT MSGG CHAR(100))
LANGUAGE CL SPECIFIC DBMOTOLIB.JRNSQNM
NOT DETERMINISTIC
NO SQL
CALLED ON NULL INPUT
EXTERNAL NAME 'DBMOTOLIB/JRNSQNM'
PARAMETER STYLE GENERAL
```

The Ritmo/i .NET Provider

The Ritmo/i .NET provider is included with your DBMoto release and installed as part of the DBMoto setup. The version of Ritmo/i that is provided with DBMoto does not include the Ritmo Toolbox or the developer tools. For more information about the different Ritmo versions available, check the BackOffice Associates web site.

The Ritmo/i files can be found in the DBMoto installation folder, in a separate folder called **Ritmo_i**.

The following topic may be useful when using Ritmo/i with DBMoto.

Enabling a trace

The file Ritmo_i.xml contains configuration settings for Ritmo and can be found in the folder where Ritmo was installed. You can set a trace file name and enable the trace from this file. Note that the trace will run whenever Ritmo is in use, and you should set <traceflag> to False immediately after completing the operations that you wanted to trace. If you leave the trace running, it can affect performance and build up large trace files.

1. In the Windows Explorer, go to the DBMoto installation folder, then to the Ritmo_i folder.
2. Open the file Ritmo_i.xml in a text editor.
3. Modify the trace entry (in bold below) as needed:

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<configuration>
<configSections>
<section name="trace" />
</configSections>
<trace>
<tracefile>C:\Program Files\HiT Software\DBMoto V6\Ritmo_i\logs\Ritmo_i.trc</tracefile>
<traceflag>True</traceflag>
</trace>
</configuration>
```

4. Save the file and exit the editor.

Last Updated on 9/13/2018