



]

# DBMoto

## User's Guide

Version 9.5.0.2

**Software Release Date: 4/9/2018**

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

## Table of Contents

<b>Chapter 1: Getting Started .....</b>	<b>9</b>
Welcome to DBMoto 9.5.....	9
DBMoto 9.5 Highlights.....	9
Tour of the DBMoto Management Center .....	10
Main Toolbar and Menu Bar .....	10
Metadata Explorer .....	11
Tools in the Tabbed Pane .....	13
Properties Pane .....	14
Status Bar .....	14
Upgrading from Earlier Versions of DBMoto to DBMoto 9.....	14
Setting Up a Replication .....	16
Steps for Replicating a Table Using Refresh Mode .....	16
Steps for Replicating with One-Way Mirroring.....	28
Steps for Replicating with Synchronization .....	41
<b>Chapter 2: Setting Up Your Environment.....</b>	<b>59</b>
Requirements .....	59
Managing a DBMoto License.....	59
Installing DBMoto.....	60
Setting Up DBMoto Users.....	62
Adding a DBMoto Server to the Management Center .....	65
DBMoto Client and Server Security Options.....	66
Installing a DBMoto Certificate.....	67
Source and Target Database Setup Guidelines .....	81
Choosing a Log Type for Transactional Replications.....	87
Creating Database Connections .....	89
Copying Database Connections.....	90

Database-Specific Settings .....	90
Replicating To and From a Flat File .....	110
<b>Chapter 3: Designing Replications .....</b>	<b>111</b>
Replication Types .....	111
Planning the Replication Process .....	111
Table Replication Scenarios .....	112
Database Replication Scenarios .....	116
Primary Key Settings for Mirroring and Synchronization.....	118
Running DBMoto on a Wide Area Network .....	120
Trigger-based Transactional Replications.....	120
Resolving Conflicts During Synchronization .....	122
<b>Chapter 4: Managing Replications in the Management Center .....</b>	<b>125</b>
Modifying/Adding Replication Settings.....	125
Copying Replications .....	126
Copying Database Connections.....	126
Using the Management Center while Replications are Running.....	127
Managing and Monitoring Replications on Remote Servers.....	127
<b>Chapter 5: Handling Multiple Replications .....</b>	<b>129</b>
Defining Multiple Table Replications.....	129
Managing Multiple Concurrent Replications.....	130
Setting Priorities for Concurrent Replications .....	131
Creating Replications with Different Connection Settings.....	131
Synchronizing Data Among Multiple Tables .....	131
<b>Chapter 6: Database-Specific Replication Issues .....</b>	<b>137</b>
Replicating from Multi-Member Tables.....	137
Using a Row ID to Identify a Record (Oracle) .....	137
Using a Relative Record Number to Identify a Record (IBM Db2 for i).....	138

Setting the CCSID for Tables and Columns .....	139
Customizing Character Data by Overriding the CCSID Setting.....	139
Synchronization Limitation with IBM Db2 for i.....	140
Writing a Script to Determine Receiver Use.....	141
<b>Chapter 7: Writing Scripts .....</b>	<b>143</b>
Writing Scripts in DBMoto.....	143
Writing a Custom Create Table Rule.....	143
Writing a Field Mapping Function .....	151
Writing a Global Script.....	156
Writing a Replication Script.....	158
Writing an Expression Script.....	162
Writing a Function to use in Scripts and Expressions.....	163
Handling Events for INSERT, UPDATE and DELETE Operations.....	165
Global Script Functions (IGlobalScript Class).....	166
Global Script Events .....	174
VB.NET Global Script Function Examples .....	183
C# Global Script Function Examples .....	186
Replication Script Events.....	190
Replication Script Properties.....	205
IRecord Interface .....	212
<b>Chapter 8: Exploring the Graphical User Interface .....</b>	<b>217</b>
Shortcuts to Creating a Replication.....	217
Wizards .....	218
Scripts.....	263
Add New Server Dialog .....	271
Connection Properties Dialog .....	272
Create DBMoto Login Dialog.....	281

Define Primary Key Dialog .....	286
Data Replicator Options Dialog.....	287
DBMoto Dashboard (Statistics Tab).....	294
DBMoto Login Dialog.....	298
DBMoto Service Monitor .....	299
DBMoto Verifier.....	300
DBMoto Verifier Options Dialog.....	302
Edit DBMoto Login Dialog .....	304
Event Properties Dialog.....	308
Expression Generator.....	309
Fields Mapping Dialog.....	310
Group Properties Dialog.....	312
History Viewer .....	314
License Information Dialog.....	315
Log Viewer.....	315
Log Database Connection Properties Dialog.....	317
DBMoto Log Settings Dialog .....	318
Manage Transactional Log Settings Dialog.....	319
Metadata Properties Dialog.....	320
Object Browser.....	323
Options Dialog.....	324
Oracle Change Log Settings Dialog.....	327
Predefined Values in the Expression Generator.....	329
Replication Activity Viewer .....	330
Replication Alert Properties Dialog .....	332
Replication Browser .....	334
Replication Monitor .....	334

Replication Properties Dialog .....	336
Replication Script Editor .....	349
Restore Metadata Options Dialog .....	353
Scheduler Dialog.....	354
Select Tables Dialog.....	354
Server Alert Properties Dialog .....	356
Server Connection Properties Dialog .....	359
Service Dependencies Window.....	360
Service Installer Window.....	363
Show Replication Pairs Dialog.....	364
SQL Query Tab .....	364
Table Properties Dialog .....	368
Table Script Dialog .....	371
User Settings Dialog.....	371
Validate Replications Dialog.....	372
<b>Chapter 9: Running Replications.....</b>	<b>375</b>
Running a Replication.....	375
Running the DBMoto Data Replicator as a Service .....	375
Disabling and Stopping Replications .....	377
<b>Chapter 10: Managing Replications Programmatically.....</b>	<b>378</b>
<b>Chapter 11: Verifying Replication Results .....</b>	<b>381</b>
Verifying Replication Results.....	381
Saving and Reviewing Comparisons .....	382
Analyzing Results .....	383
Reconciling Data Differences in Results.....	383
Monitoring and Reviewing Replications .....	384
<b>Chapter 12: Performance and Tuning .....</b>	<b>389</b>

Log Check Frequency in Mirroring and Synchronization Modes .....	389
Performance and Efficiency .....	389
Reviewing Replication Performance over Time .....	390
Improving Performance Using Refresh with SQL Server, MySQL.....	393
Improving Performance Using Mirroring with IBM Db2 for i.....	394
Managing Performance Using Thread Settings.....	394
<b>Chapter 13: Maintenance and Recovery .....</b>	<b>397</b>
Identifying and Recovering From Errors .....	397
Email Alerts .....	397
Understanding Log Files.....	400
Creating and Deleting Text Log Files .....	402
Monitoring Text Log File Size.....	402
Generating Trace Files .....	402
Backing Up and Restoring Metadata .....	404
Moving a Metadata Database.....	406
Monitoring and Reviewing Replications .....	406
Managing Source/Target Table Schema Changes.....	409
Recovering from Interruptions to Refresh Replications .....	410
Running a Refresh Replication with Transactional Replications.....	410
<b>Chapter 14: DBMoto API Reference.....</b>	<b>413</b>
DBMoto API Overview.....	413
<b>Chapter 15: DBMoto Integration .....</b>	<b>418</b>
Installing DBMoto from a Windows Command Line.....	418

## Chapter 1: Getting Started

### Welcome to DBMoto 9.5

What do you want to do?

- [Install DBMoto and Set Up Your Environment for Replication](#)
- [Check Supported Data Providers \(in the Help Center\)](#)
- [Create Database Connections](#)
- [Set up a Replication](#)
- Upgrade from [DBMoto 8](#) to DBMoto 9
- View a summary of [new features and changes in DBMoto 9](#)
- [Tour the DBMoto Management Center](#)

### DBMoto 9.5 Highlights

#### 1. Added performance transactional mode: Log Server Agent

[Log Server Agent mode](#) offered for Oracle, Microsoft SQL Server, MySQL, IBM Db2 for i, IBM Informix, PostgreSQL

#### 2. Enhanced Transactional Replication Setup

Mirroring and synchronization now use [Enable Transactional Replication](#) and [Disable Transactional Replication](#) wizards for increased transparency

#### 3. Added SAP HANA as a source database for transactional replications

Mirroring from SAP HANA to all major databases is offered using [trigger-based replication](#)

#### 4. Added a bulk mirroring option for performance

Replications to Oracle, MySQL, SQL Server and Azure, PostgreSQL can now use a bulk mirroring insert option configured in the [Connection Properties](#) as the Default Mirroring Insert Mode.

#### 5. Enhanced scripting events for replication disabling/recovery mode

Replication\_OnError, Record\_onBeforeMapping, Record\_onAfterMapping, Record\_onBeforeExecute and Record\_onAfterExecute allow the user to disable the replication, and the Record\_OnExecuteError event can put a replication into recovery mode

#### 6. Improved, integrated support for Hadoop

Improved support for replicating relational data to Apache Hadoop distributed processing and data storage framework.



## 7. Added support for replication to JSON format files/JSON datatypes

In replication configuration, JSON datatype mapping is supported, and it is possible to set up replications to JSON format files.

## 8. Added the ability to customize a target table during creation

[Writing a Create Table Rule](#) in the Global Script Editor provides a way to customize table creation for multiple tables associated with a connection

## 9. Enhanced support for commitment control in transactional replications

The [Replication Properties dialog](#) offers additional properties to manage transactions.

## 10. Added support to manage schema changes

For Oracle, Microsoft SQL Server, IBM Db2 LUW, IBM i and MySQL, notification and update of schema changes can be managed in the [Replication Properties Preferences](#) and the [Validate Replications dialog](#).

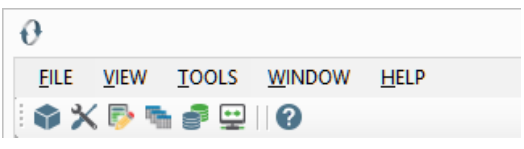




## 11. Script Editor upgrades

The [Global Script Editor](#) and [Replication Script Editor](#) now offer intelligent code completion for global and replication scripts

## Tour of the DBMoto Management Center

The DBMoto Management Center provides all the tools needed for setting up, running and evaluating replications.

### Main Toolbar and Menu Bar

	<p>Use these menus and icons to access tools and options to manage your working environment</p>
<ul style="list-style-type: none"> <li data-bbox="138 1291 625 1375">  <b>Metadata Explorer</b>            Adds a new Metadata Explorer pane to the Management Center window, if one is not already open.         </li> <li data-bbox="138 1396 625 1480">  <b>Property Window</b>            Opens a new Property Window, if one is not already open.         </li> <li data-bbox="138 1501 625 1669">  <b>Object Browser</b>            Adds a new <a href="#">Object Browser tab</a> to the Management Center window, if one is not already open. The Object Browser displays information related to the item selected in the Metadata Explorer. For example, if you select a source table in the database tree, you can see the table column names and column information.         </li> <li data-bbox="138 1690 625 1793">  <b>Replication Browser</b>            Adds a new <a href="#">Replication Browser tab</a> to the Management Center window, if one is not already open. The         </li> </ul>	

Replication Browser displays replications for the selected metadata together with summary information for the replications.



**Replication Monitor**

Adds a new [Replication Monitor tab](#) to the Management Center window, if one is not already open. The Replication Monitor provides current status information for all replications defined in the metadata that is selected in the Metadata Explorer.



**Toolbox**

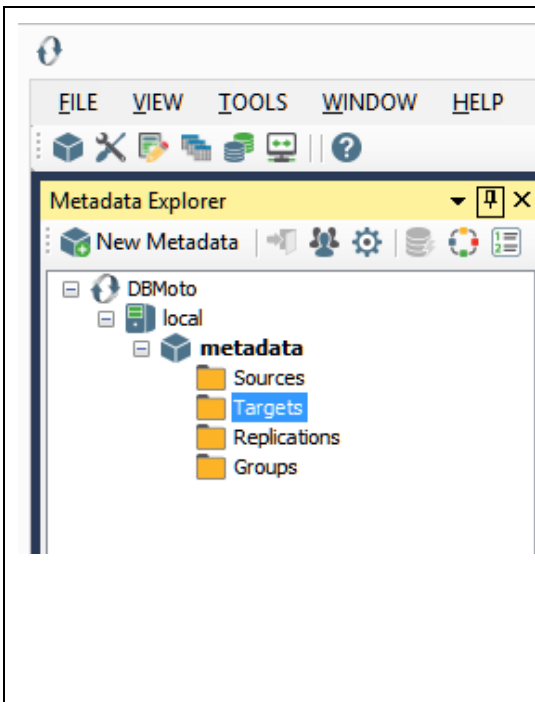
Adds an expandable tab to the Management Center, for easy access to wizards to create source and target connections, tables, replications and replication groups.



**Help**

Opens a separate Help window with access to help topics in the DBMoto User's Guide.

## Metadata Explorer



Use this area of the DBMoto Management Center to:

- Set up user access to DBMoto
- Add and remove views of remote DBMoto installations (servers)
- Manage properties of each server listed in the Metadata Explorer
- Create DBMoto Metadata as a way of managing replication projects
- Add connections for source and target tables
- Create replications and groups

In a standard installation of DBMoto, you will always see a tree structure that looks like the one here, with an entry for the "local" installation of DBMoto. In addition, DBMoto creates a default metadata for you using the Microsoft SQL Server CE DBMS that is distributed with DBMoto. In most cases, you will start adding source and target connections to this metadata, then create and run replications. However, you can choose to create metadata in an alternative RDBMS.

## Toolbar



### New Metadata

Opens the [Metadata Connection wizard](#), allowing you to create a connection for a new metadata database.



### Change Login

Opens the [DBMoto Login dialog](#), allowing you to enter login information to connect to a DBMoto server.



### Manage Users

Opens the [User Settings dialog](#), allowing you to establish access control for DBMoto users.



### Data Replicator Options

Opens the Data Replicator Options dialog, allowing you to set options that affect how the Data Replicator runs, including options for logs, traces and email notifications.



### New SQL Command

Opens a [SQL Query tab](#) with a connection to the item that is selected in the Metadata Explorer. For example, if you have selected a source connection, the tab opens ready to execute a query for that connection.



### Global Script

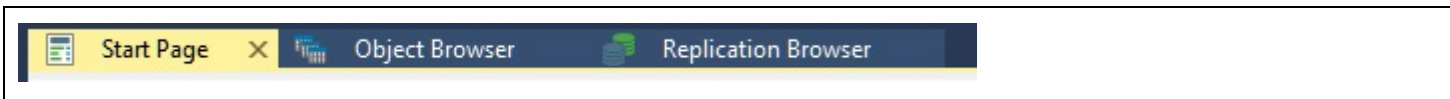
Opens the [Global Script Editor](#), allowing you to write a script that is available to all replications in the metadata.



### View Log

Opens the [Log Viewer tab](#), allowing you to review the contents of the DBMoto log. The DBMoto log contains useful information regarding replication progress, results and issues. Note that it is not possible to view a file-based log if the selected metadata is on a remote installation of DBMoto. The View Log option only opens the log file on the local file system.

## Tools in the Tabbed Pane



The tabs in the DBMoto Management Center include the main tools and viewer that you will use as you create and monitor replications. The default tabs are:

### Start Page

Provides access to help topics and useful information you will need to get started with DBMoto.

### Object Browser

Displays information related to the item selected in the Metadata Explorer. For example, if you select a source table in the database tree, you can see the table column names and column information.


### Replication Browser

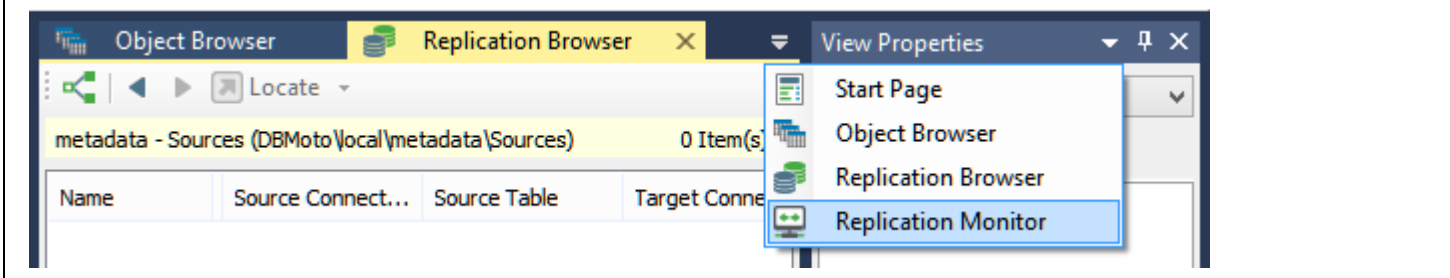
Displays replications for the selected metadata together with summary information for the replications.

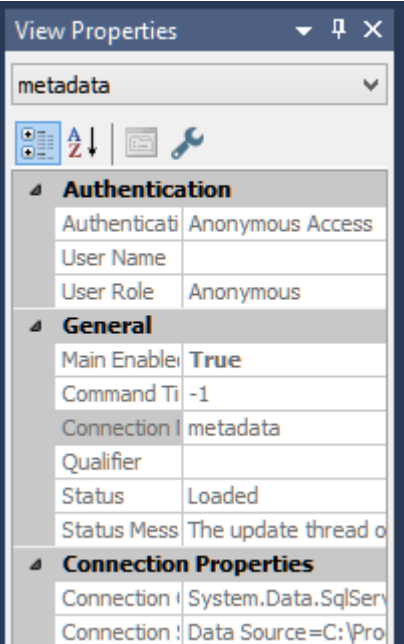

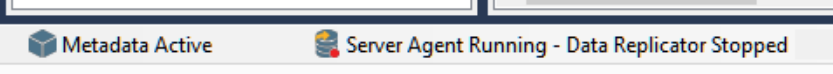
### Replication Monitor

Provides current status information for all replications defined in the metadata that is selected in the Metadata Explorer.

## Managing Tabs

The tab headings are fixed width, so if you open several additional tabs, you need to use the  View All Tabs icon in the tab area to display all tabs.



	<h3>Properties Pane</h3> <p>The Properties pane displays properties for whichever item is selected in the Metadata Explorer. If you select a property in the list, a description of the property is displayed below the list. Note that you cannot edit properties in this pane. Click  <b>Show Properties</b> in the toolbar to open a separate, editable Properties Window.</p>
	<h3>Status Bar</h3> <p>The Status Bar displays the current status of the Metadata, the Server Agent and the Data Replicator.</p>

## Upgrading from Earlier Versions of DBMoto to DBMoto 9

If you already have DBMoto V8 or V9 installed, the upgrade process involves running a new setup and applying a new license key. For earlier versions of DBMoto, contact [HiT Software Technical Support](#).

1. Download DBMoto V9.5 as directed by HiT Software support staff.
2. Make sure that you obtain your new license key before starting the installation process. Contact your HiT Software account manager if you encounter issues with your license key.
3. Run the setup.exe.
4. If you are installing DBMoto V.9.5 over an existing installation of DBMoto, a dialog is displayed with the following text.  
 “An older version of DBMoto has been detected on this computer. Click YES to Upgrade to the latest version.”
5. Click **Yes** to proceed with the installation. Click **No** to exit the installation.

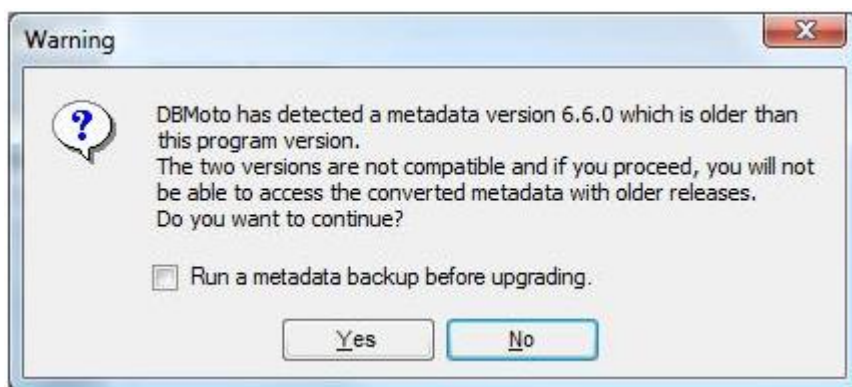
6. There are two installation options:

**Standard Installation:** Installs DBMoto Management Center , DBMoto Server Agent and DBMoto Replicator

**Management Center Only:** Installs only the Management Center component, used for managing and monitoring remote installations of DBMoto.

Select the **Standard Installation** option.

7. When the following dialog is displayed, check the option to **Run a metadata backup before upgrading**, unless you have already made an up-to-date backup of your metadata.



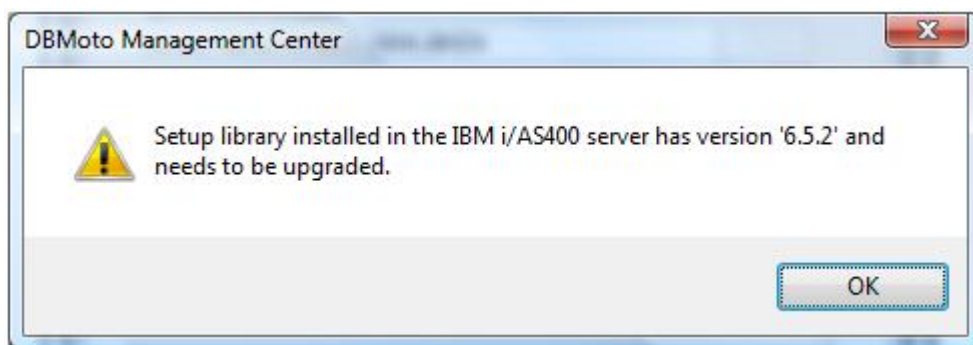
8. Click **Yes** to continue with installation.

9. Enter the pathname to your DBMoto V9 license file when requested.

10. Complete the installation.

11. You are now ready to run the DBMoto Management Center and [explore features in the new release](#).

12. If you are using IBM Db2 for i as one of your source connections, you may also see the following dialog.



- Start the Management Center, then view the Connection Properties for your IBM Db2 for i connection.
- Click in the Transaction Log Type field to open the Setup Info dialog.
- Install a new library on the Db2 server.

## Setting Up a Replication

With DBMoto, there are just a few major steps involved in setting up a quick test replication. When you have tried everything out, you can explore the different options that DBMoto has to offer, including setting up multiple replications simultaneously, performing synchronization between two or more systems, customizing replications with selecting mapping and/or scripts and so on.

Here's how to get started.

1. Decide what type of data replication you want to perform.

[Refresh](#) Copy entire data set from source to target once only

[Continuous Refresh](#) Copy entire data set from source to target according to a schedule

[One-way mirroring \(with or without initial refresh\)](#) Initially copy entire data set from source to target, then copy only changes to the source database according to a schedule. This approach is also known as Change Data Capture.

[Synchronization \(with or without initial refresh\)](#) Initially copy entire data set from one system to the other, then keep changes to the two databases synchronized according to a schedule

Click on one of the links above for detailed steps on how to create a replication, or use the overview below to guide you.

2. Using the DBMoto Management Center, create a connection to the source database table(s).
3. Create a connection to the target database table(s).
4. Set replication properties using the Replication wizard or the Multiple Replications wizard.
5. Enable the replication(s).
6. Start the Data Replicator using the DBMoto Service Monitor in the Windows Notification Area.

## Steps for Replicating a Table Using Refresh Mode

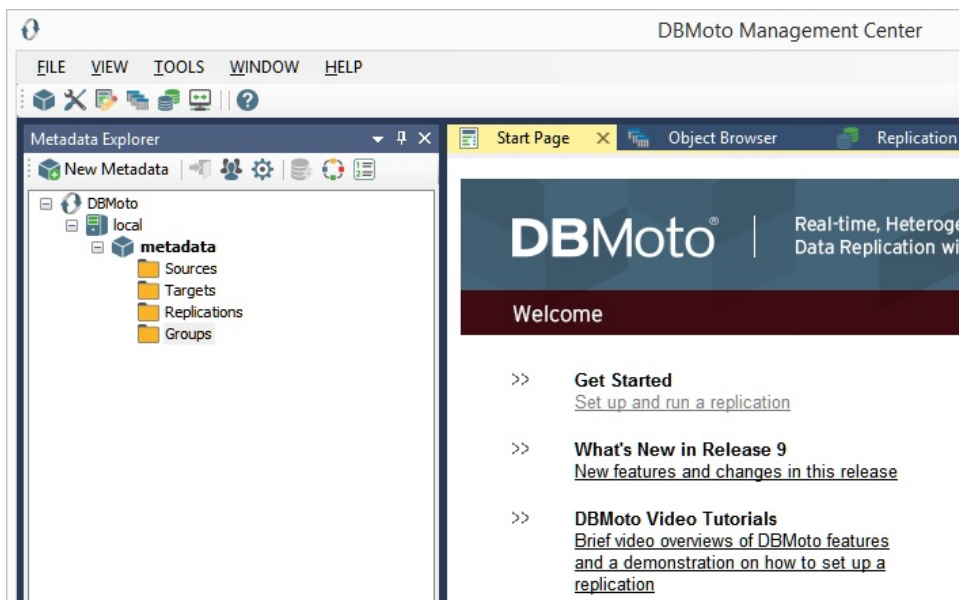
[Set up database connections](#)>[Create a target table](#)>[Define the Replication](#)>[Run the Replication](#)

A refresh is a one time complete replication from source to target table, according to replication settings and scripts. You can control the timing of the replication, identify the columns to be replicated and add scripts to transform data during replication. The source and target databases can be on the same or different database servers and platforms. For example, you can replicate an IBM Db2 for i (iSeries/AS400) table to an Oracle database or a SQL Server table to Db2 for i.

## 1. Set Up Database Connections

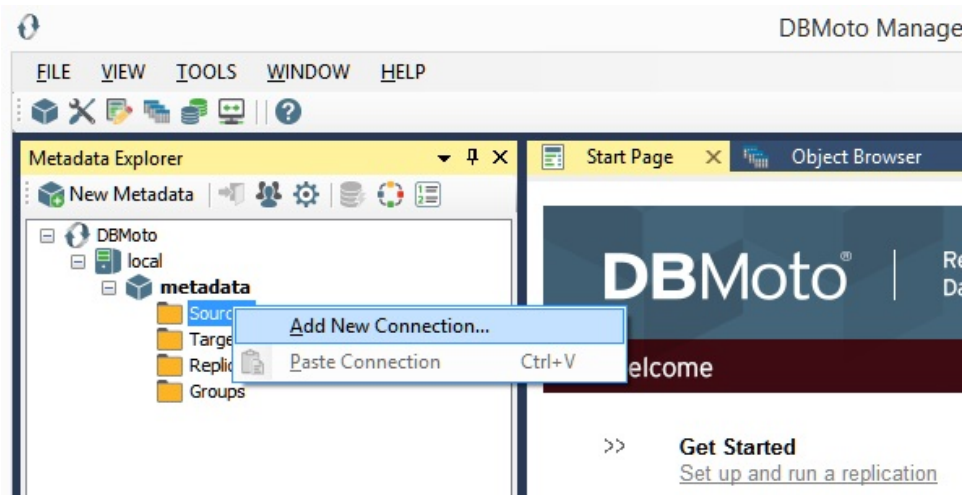
1. Make sure you have [database connections via a .NET data provider](#) to the supported databases for source database tables and target database tables.
2. Start DBMoto Management Center.

DBMoto provides a default database (Microsoft SQL Server CE) for your DBMoto metadata, all the information that DBMoto needs to store about your replication setup.

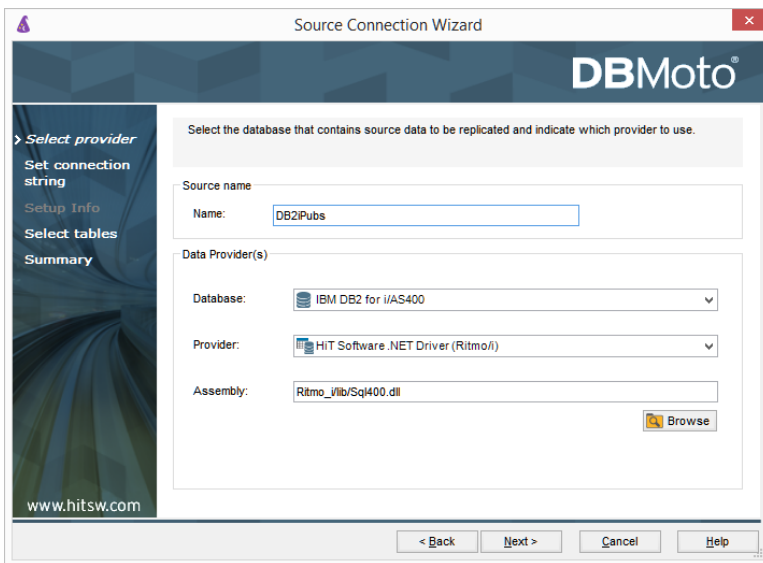


3. In the Metadata Explorer, expand the **metadata** node to view the **Sources** and **Targets** nodes.
4. Select the **Sources** node.
5. From the right mouse button menu, choose **Add New Connection**.

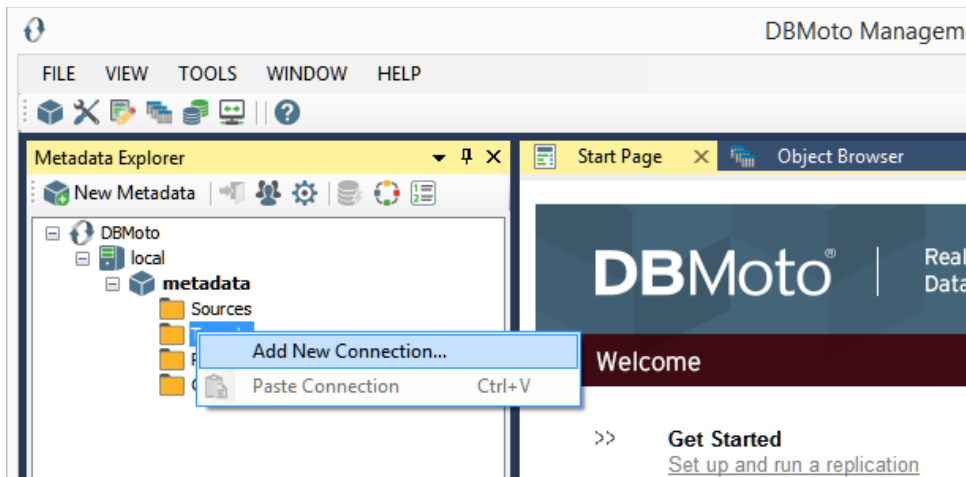




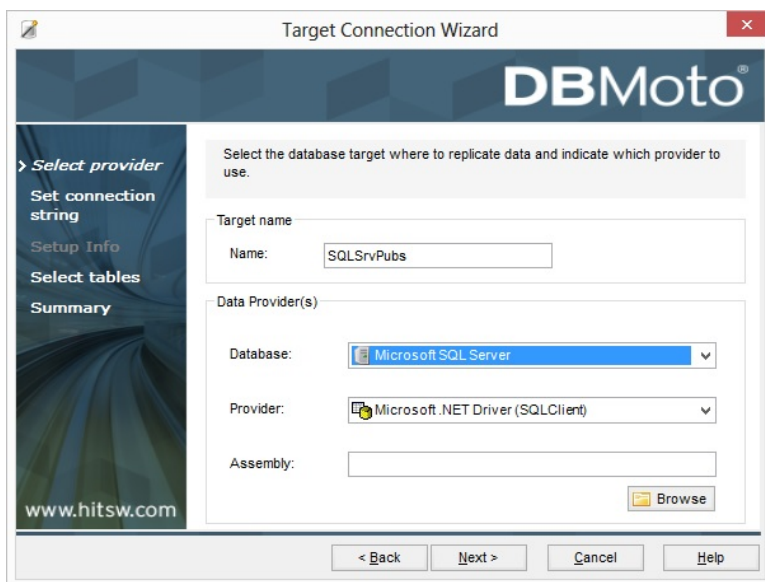
6. In the [Source Connection Wizard](#), follow steps to add a connection string and test the connection to the database. Check the [HiT Software knowledge base article on data providers](#) before entering a value in the **Assembly** field.



7. Choose the tables that you plan to replicate.
8. Complete the wizard.
9. Select the **Targets** node.
10. From the right mouse button menu, choose **Add New Connection**.



- In the [Target Connection Wizard](#), follow steps to add a connection string and test the connection to the database. Check the [HiT Software knowledge base article on data providers](#) before entering a value in the **Assembly** field.

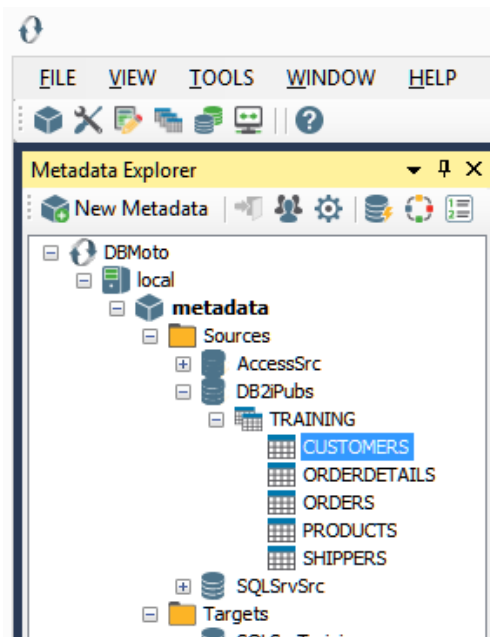


- Choose the tables to which you plan to replicate.  
If the table does not exist, continue to the next screen without selecting a table. [Create a target table](#) once the wizard is complete.
- Complete the wizard.

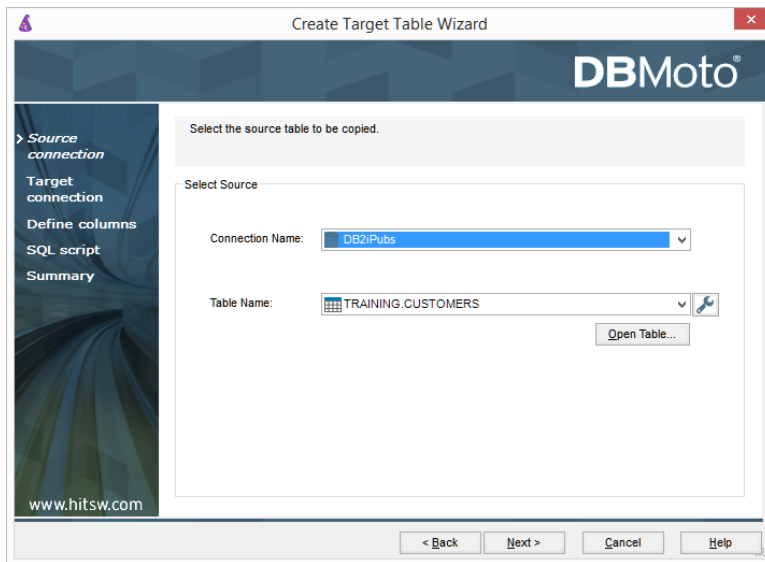
## 2. Create a Target Table

If you are replicating to a database where no target table exists, you need to create the target table before defining the replication. If the target table already exists in the database to which you are replicating, go to [Defining the Replication](#).

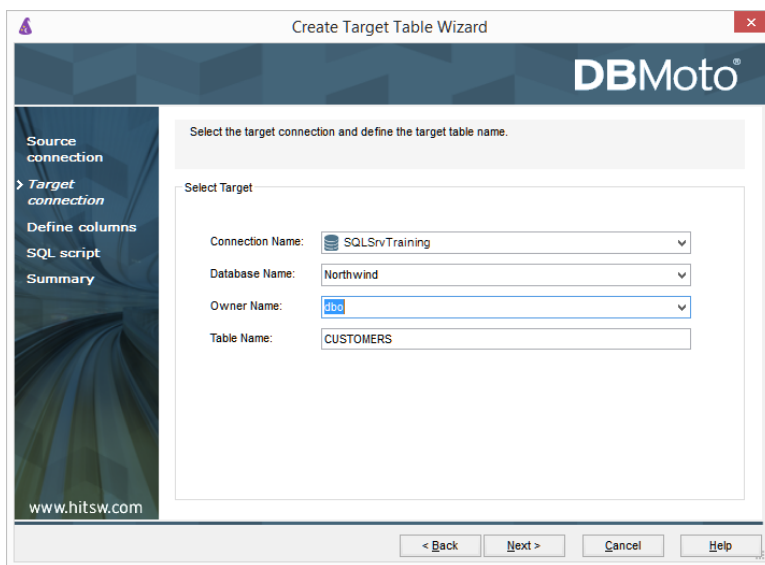
1. Expand the Metadata Explorer tree to display the table that contains the data you want to replicate.
2. Select the table and drag it to the target connection name to open the [Create Target Table wizard](#).



3. In the **Select Source Connection** screen, verify that the correct connection is displayed. You can modify the source connection and table names using the drop-down lists.

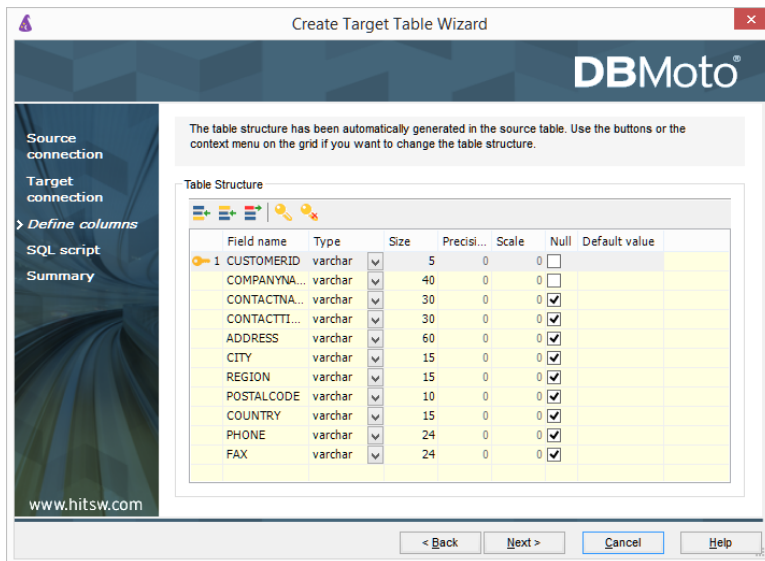


4. If you want more information about the table before proceeding, click **Open Table...**
5. Click **Next** to go to the **Target Connection** screen.



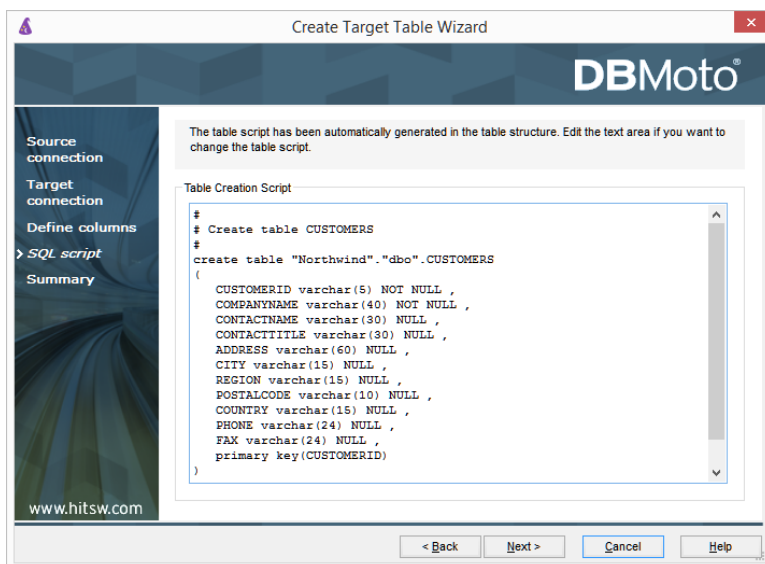
6. Choose a target connection name from the drop-down list that includes all the target connections you have created in DBMoto.
7. Select the appropriate qualifiers for your target database.
8. As needed, edit the name for the table you wish to create in the target database. This table will contain the replicated data.

9. Click **Next** to go to the **Define Columns** screen.



10. Review the columns that will be created in the target table. You can add or remove columns as well as designate one or more columns as a primary key.

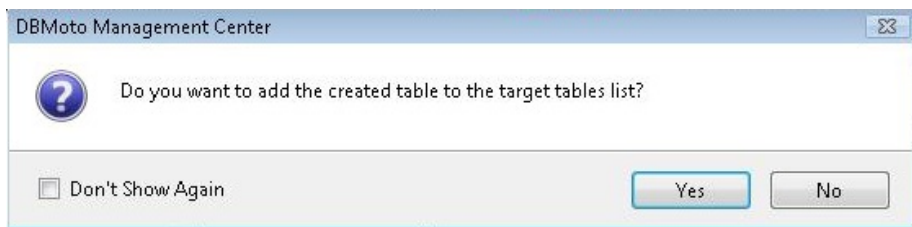
11. When you have finished editing the target table columns, click **Next** to go to the **SQL Script** screen.



12. If necessary, edit the SQL script that generates the table.

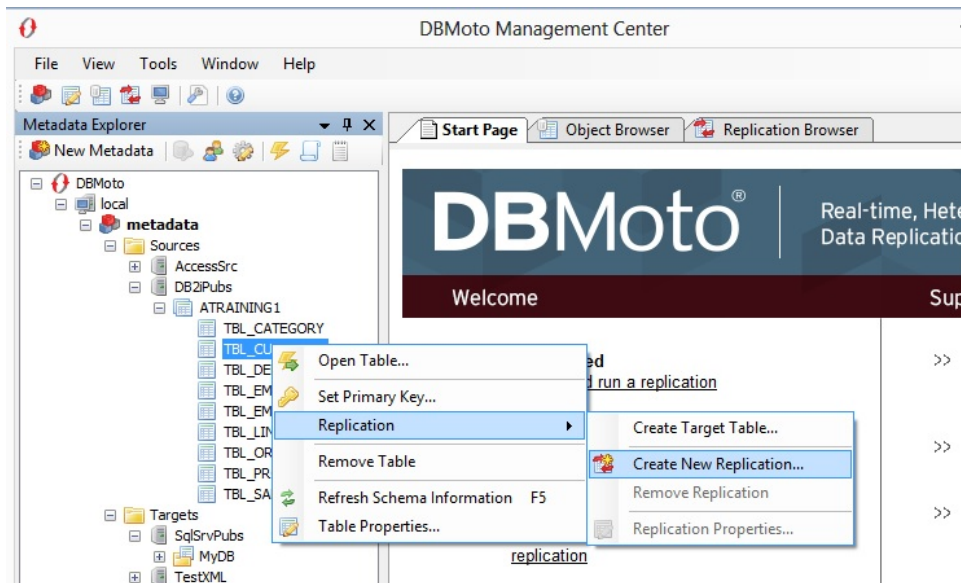
13. Click **Next** to review the wizard settings in the **Summary** screen.

14. To create additional tables, check the option **Create another table**.  
This opens another Create Target Table wizard when this wizard is complete.
15. To go directly to creating a replication once this wizard is complete, check the option **Proceed with the Definition of a Replication**.  
This opens the [Replication wizard](#) when the Create Target Table wizard is complete.
16. Click **Finish** to create the target table.
17. Note that the new table is not automatically displayed in the Metadata Explorer. Click **Yes** to add the newly created table to the list of target tables in the Metadata Explorer.



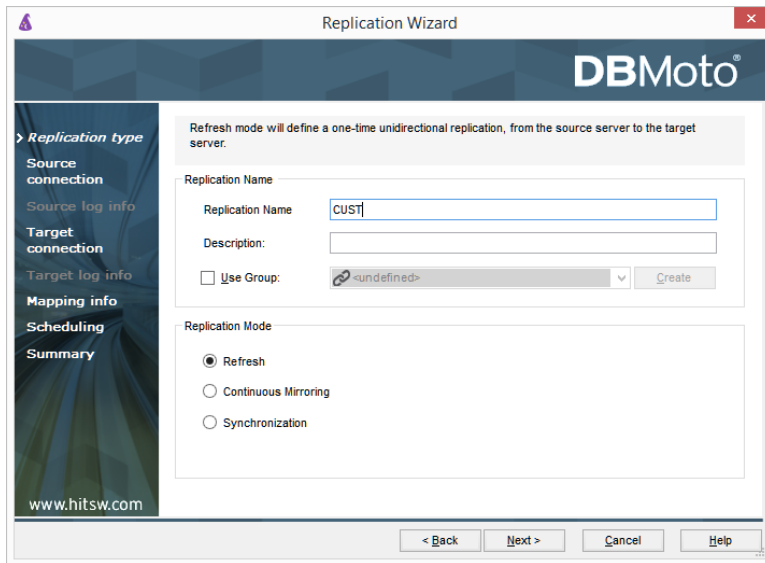
### 3. Define the Replication

1. Expand the Metadata Explorer to display the table that contains the data you want to replicate.
2. Select the table.
3. From the right mouse button menu, choose **Replication** then **Create New Replicaton....**

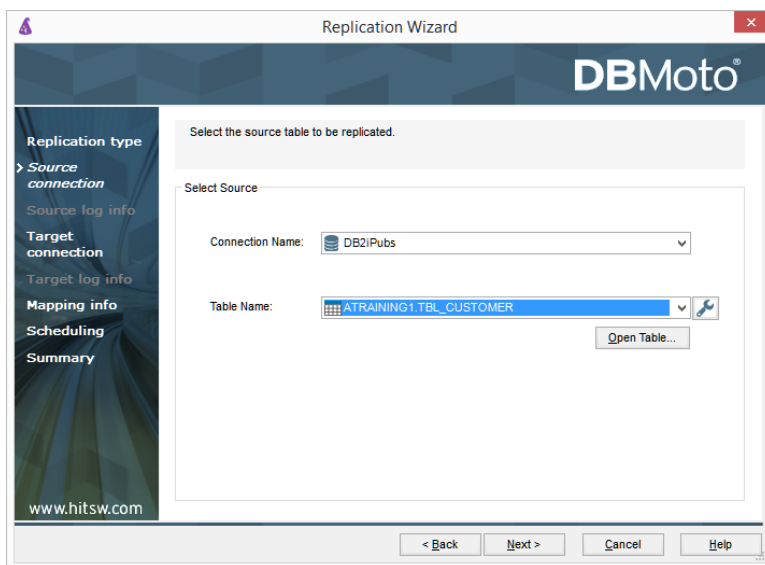


4. In the **Define Replication Type** screen, type a name to identify the replication.

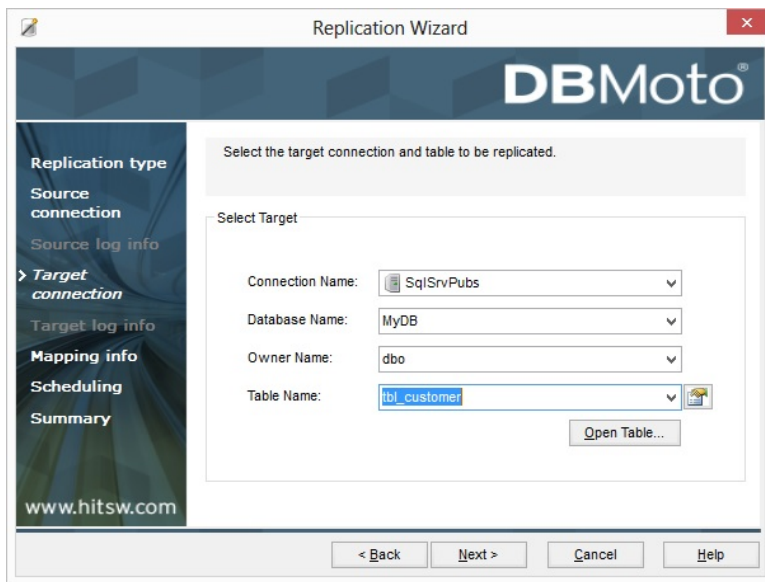
5. Optionally provide a description of the replication.
6. In the Replication Mode area, choose Refresh (a once-only, one-way replication).



7. Click **Next** to go to the Select Source Connection screen.
8. Choose the source name from the drop-down list that includes all the source connections you have created in DBMoto.

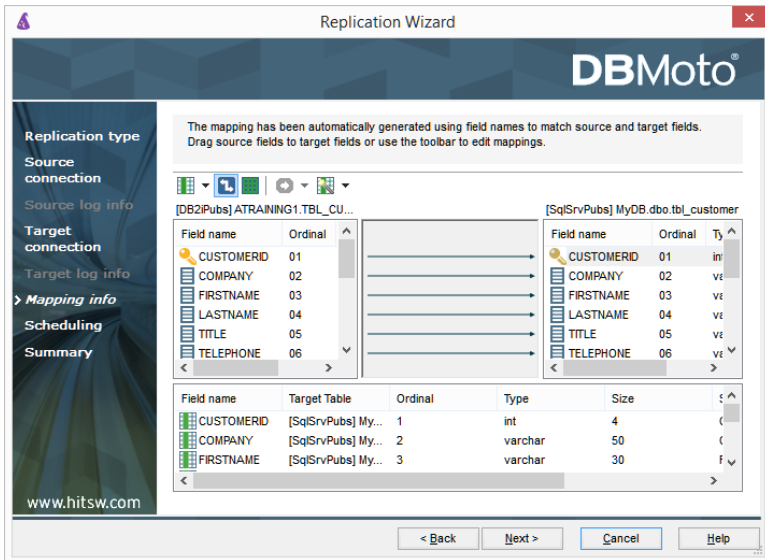


9. Choose the table that you want to replicate from the drop-down list.
10. If you want more information about the table before proceeding, click **Open Table...**
11. Click **Next** to go to the **Select Target Connection** screen.
12. Choose a target database name from the drop-down list that includes all the target connections you have created in DBMoto.



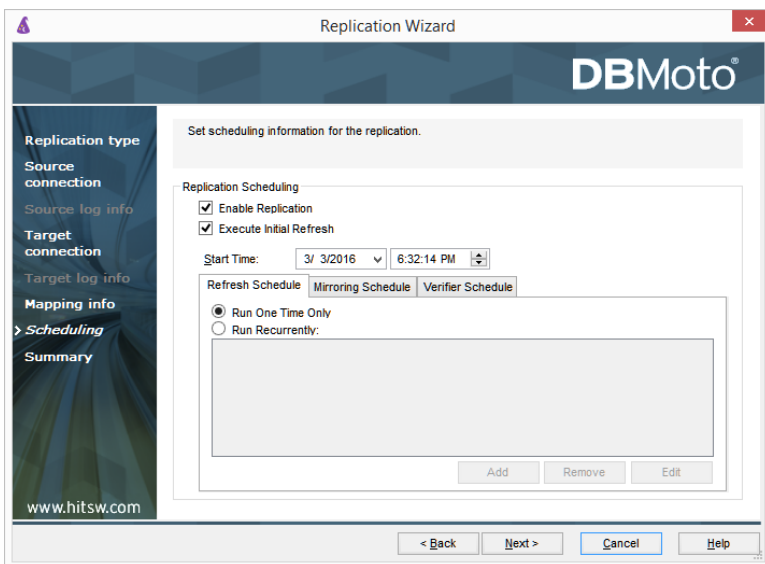
13. Choose the table to which you want to replicate from the drop-down list.  
If no tables are listed, you need to exit the wizard and add or create a target table.
14. If you want more information about the table before proceeding, click **Open Table...**
15. Click **Next** to go to the **Set Mapping Info** screen.





Source and target columns with the same name are automatically mapped. You can also map columns by dragging the target column and dropping it on the source column, or you can map a column to an expression. For more information about mapping, check the [Replication Wizard help topic](#). An alternative is to [write a script](#) to set mappings at runtime.


16. Click **Next** to go to the Scheduling screen.

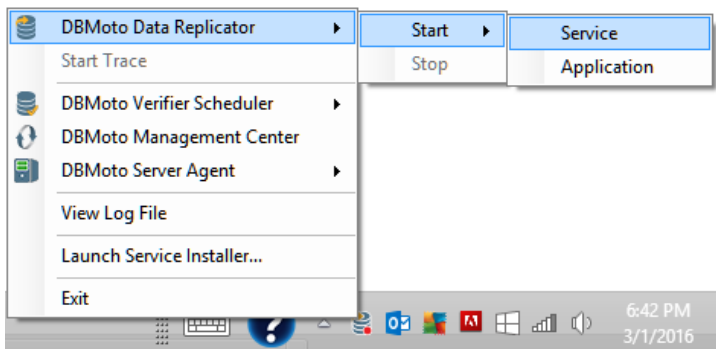


17. Make sure the **Enable Replication** option is checked. This is required for the replication to run.

18. Set a start time for the replication. The **Start Time** field indicates the time at which the Data Replicator will begin considering the replication for execution.
19. Select how you want to run the replication:
  - Once only: the replication will run once at the time specified in **Start Time**.
  - Recurrently: the replication will run according to the schedule you set by clicking the Schedule button to open the [Scheduler dialog](#).
20. Click **Next** to go to the **Summary** screen.
21. Click **Finish** to complete the wizard.

#### 4. Run the Replication

If you installed The DBMoto Data Replicator as a service during DBMoto setup, you just need to start the service using the DBMoto Service Monitor program  in the Windows Notification Area.




The replication that you have scheduled should start at the specified time. Use the [Replication Monitor](#) tab in the DBMoto Management Center to track the progress of the replication.

If you would like to install the DBMoto Replicator as a service:

- From the Windows Desktop **Start** menu, choose **Programs**, then **HiT Software DBMoto**, then **Service Installer**.
- Manage the service from DBMoto Service Monitor program (located in the DBMoto install folder or on the Windows **Start > Programs > Startup** menu).
- Use the [Replication Monitor](#) tab in the Management Center to track the progress of the replication.

To run the DBMoto Data Replicator interactively:

- In the Windows Notification Area, select  the DBMoto Service Monitor icon.
- From the right mouse button menu, choose **DBMoto Data Replicator**, then **Start** then **Application**. The replication that you have scheduled should start at the specified time.
- Use the [Replication Monitor](#) tab in the Management Center to track the progress of the replication.

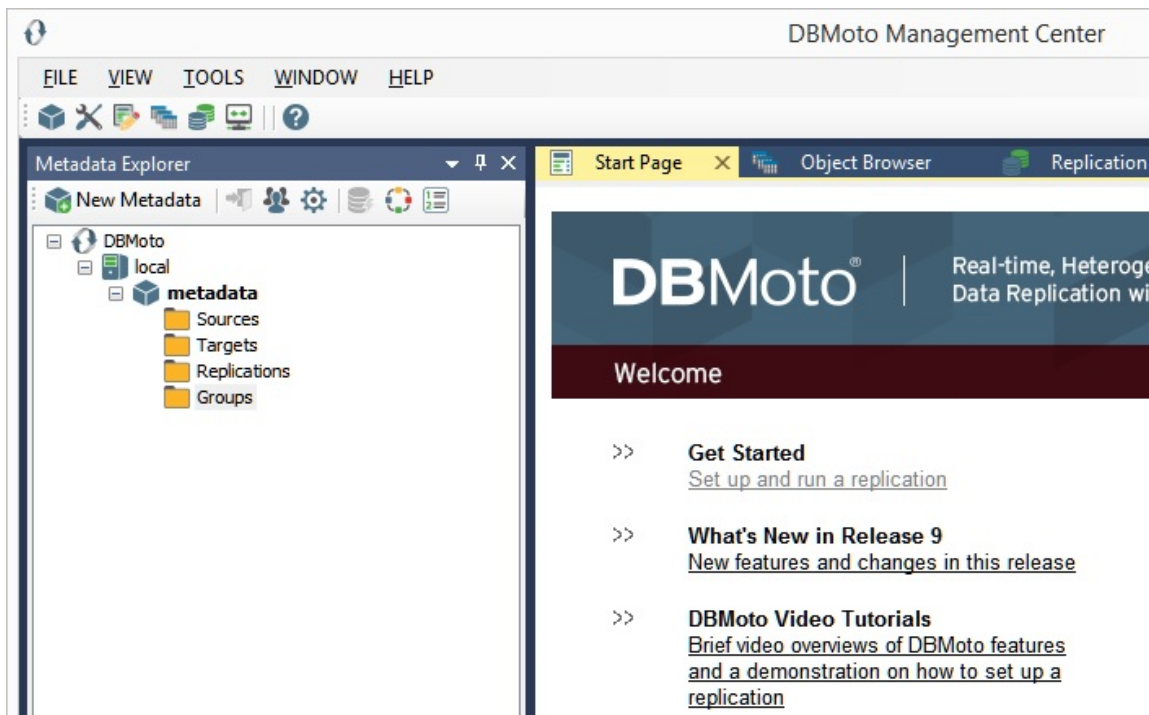
## Steps for Replicating with One-Way Mirroring

[Set up source database connection](#)>
[Enable transactional replication](#)>
[Set up target database connection](#)>
[Create a target table](#)>
[Define the Replication](#)>
[Run the Replication](#)

One-way mirroring provides a continuous update of a replicated table based on changes to the source database that have been recorded in the database server log. Typically, this involves an initial refresh operation, as described above, to set up the target table. Then you can define the replication settings to check the transaction log on the source database at regular intervals. Any changes found in the log are applied to the target database.

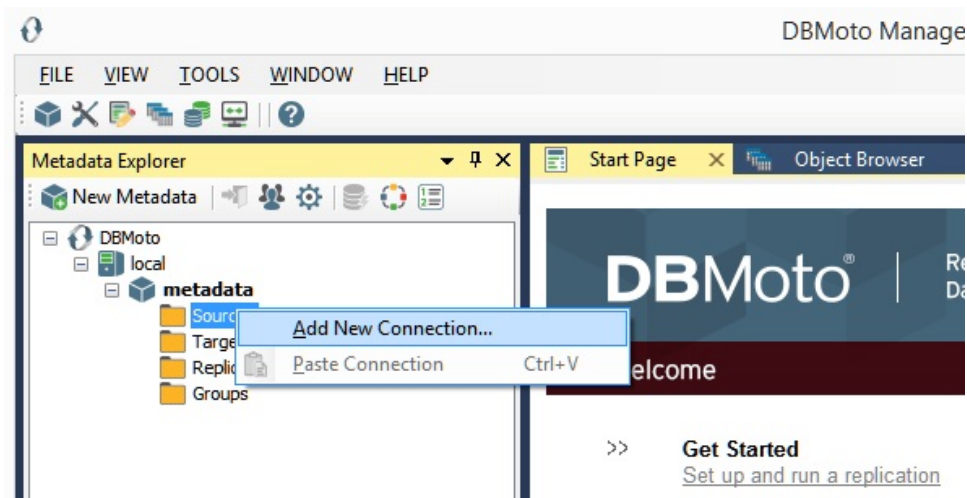
### 1. Set Up Source Database Connection

1. Make sure you have [database connections via a .NET data provider](#) to the supported databases for source database tables and target database tables.
2. Start DBMoto Management Center.  
 DBMoto provides a default database (Microsoft SQL Server CE) for your DBMoto metadata, all the information that DBMoto needs to store about your replication setup.

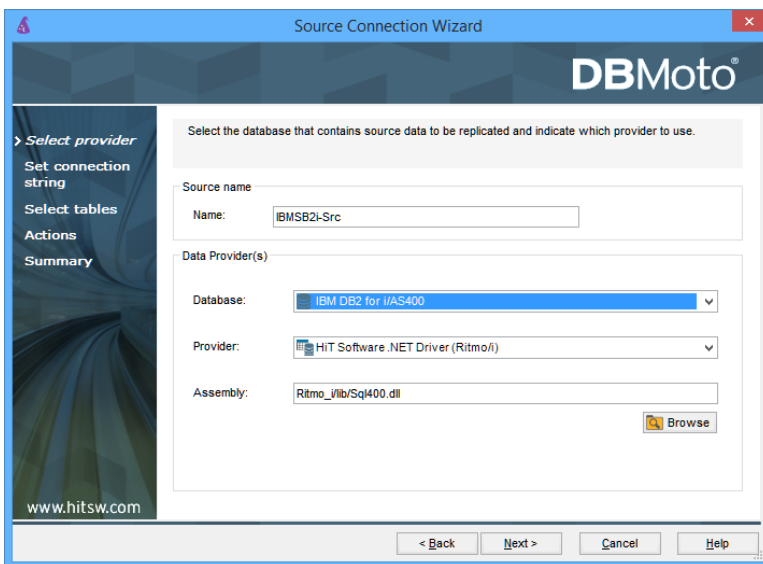


3. In the Metadata Explorer, expand the metadata node to view the **Sources** and **Targets** nodes.
4. Select the **Sources** node.

5. From the right mouse button menu, choose **Add New Connection**.

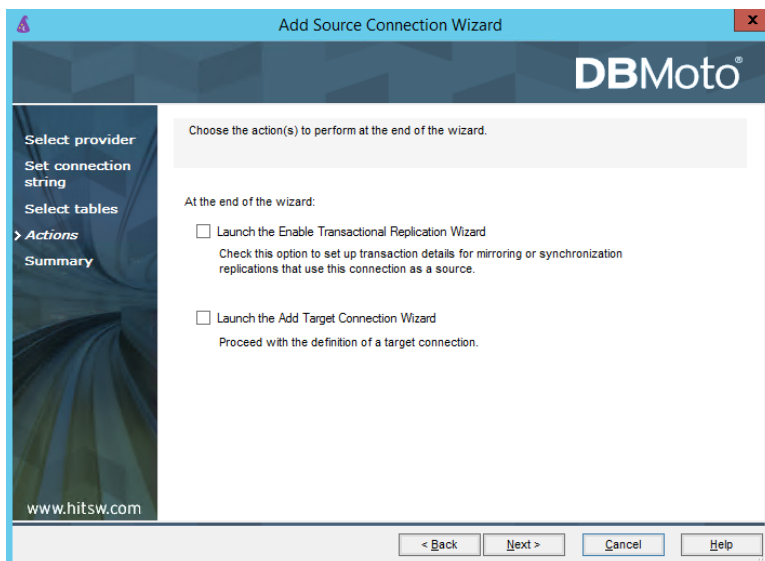


6. In the [Add Source Connection Wizard](#), follow steps to add a connection string and test the connection to the database. Check the Supported Provider List in the [Help Center](#) before entering a value in the **Assembly** field.



7. In the **Select Tables** screen, choose the tables that you plan to replicate.

8. In the **Actions** screen, check the option **Launch the Enable Transactional Setup Wizard**.



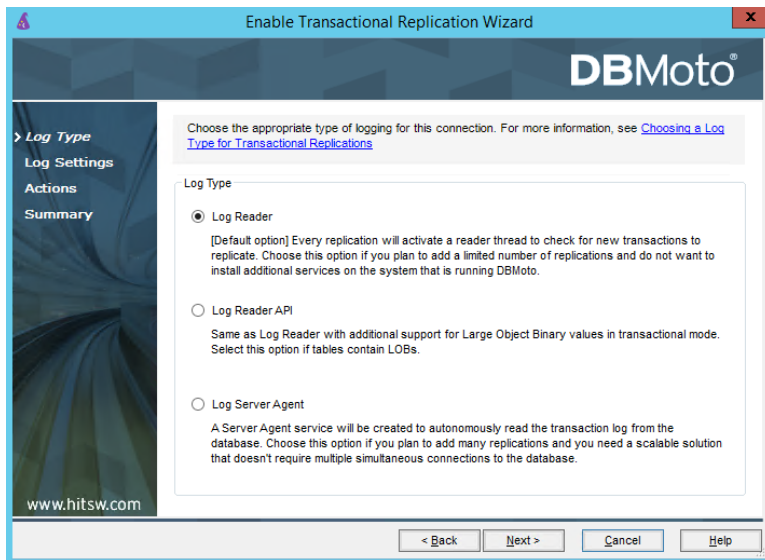
9. Complete the wizard.

## 2. Enable Transactional Replication

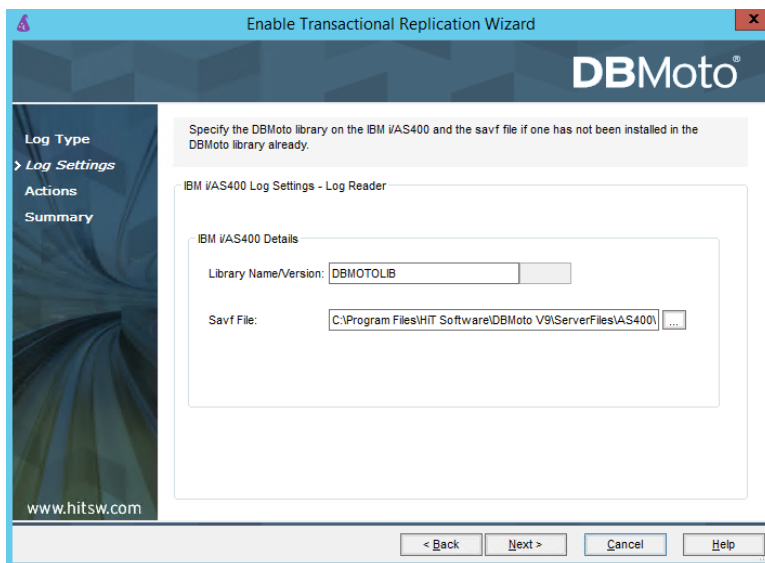
This section assumes you have checked the Source Connection wizard option to launch the Enable Transactional Replication wizard. To open the wizard from the Management Center, choose the connection in the Metadata Explorer, then right-click to choose Transactional Setup > Enable...

In the [Enable Transactional Replication wizard](#):

1. Select the type of transactional replication to use. The options depend on the source database and can include [log reader](#), [log server agent](#), [triggers](#), [plus log reader API](#) (for IBM Db2 for i only)



2. Click **Next** to enter the log settings. The fields and appropriate values depend on the [database and log type](#).

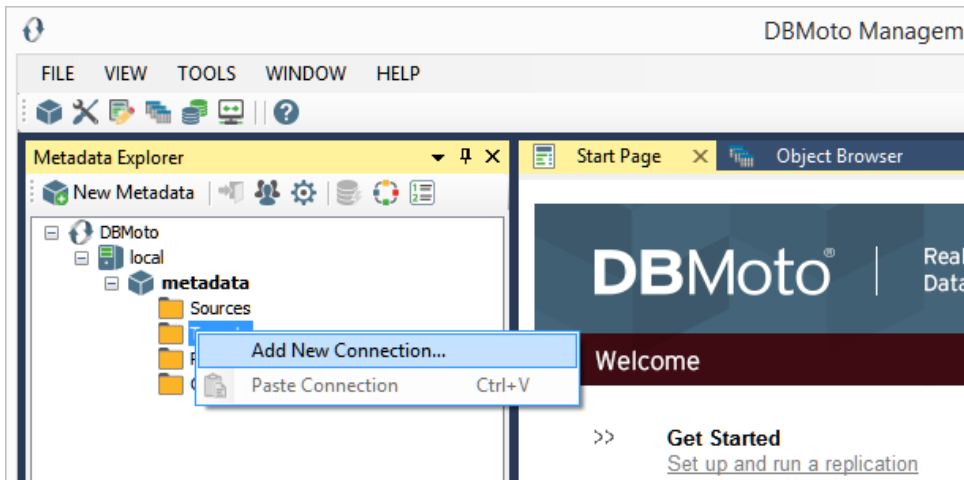


3. Click **Next** to verify your settings against the source connection to the database. If any information is missing, you will not be able to proceed.
4. In the Actions screen, check the option to launch the **Add Target Connection wizard**.
5. Click **Next** to review your changes.
6. Click **Finish** to complete the wizard.

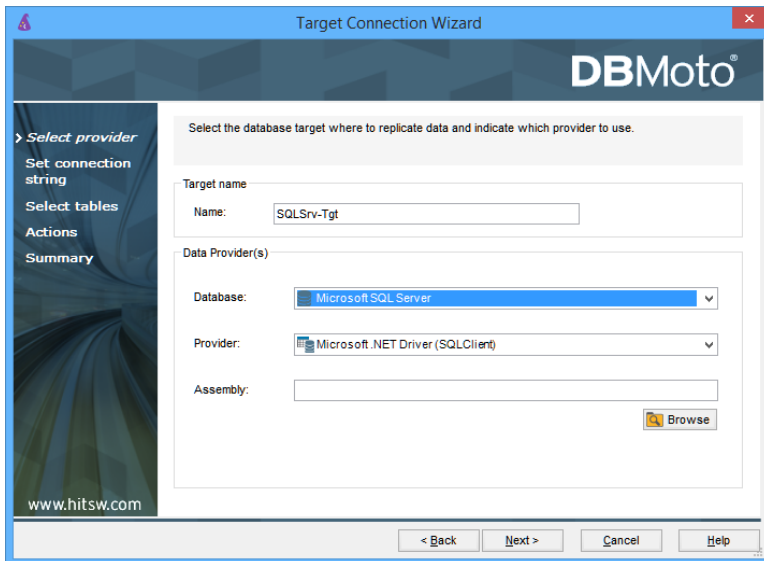
The source connection is now set up for transactional replications.

### 3. Set Up Target Database Connection

This section assumes you have checked the **Enable Transactional Replication wizard** option to launch the Add Target Connection wizard. If not, to open the wizard from the Management Center, choose **Targets** in the Metadata Explorer, then right-click to choose **Add New Connection...**



1. In the [Target Connection Wizard](#), follow steps to add a connection string and test the connection to the database. Check the Supported Provider List in the [Help Center](#) before entering a value in the **Assembly** field.



2. Choose the tables to which you plan to replicate.

If the table does not exist, continue to the next screen without selecting a table. [Create a target table](#) once the wizard is complete.

3. Complete the wizard.

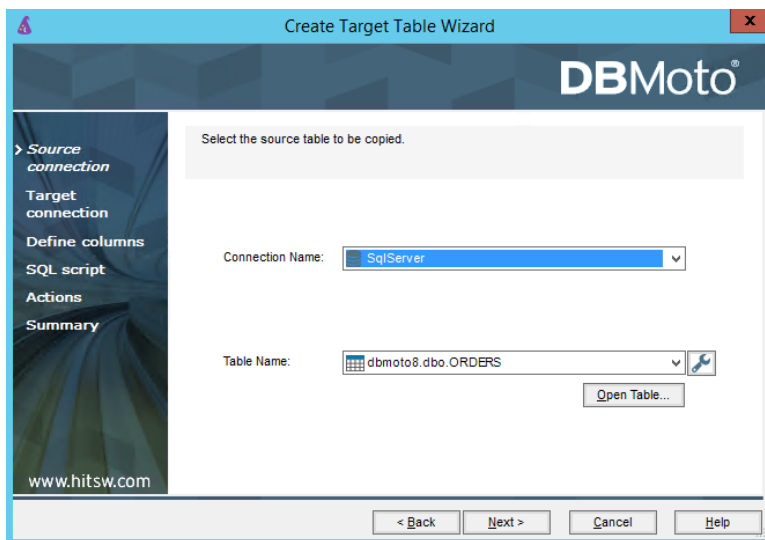
#### 4. Create a Target Table

If the target table does not exist, use the Create Target Table wizard to create the table.

1. In the Metadata Explorer, drag the source table to the target connection to open the Create Target Table wizard.

The **Select Source Connection** screen is already filled out with the table you selected.

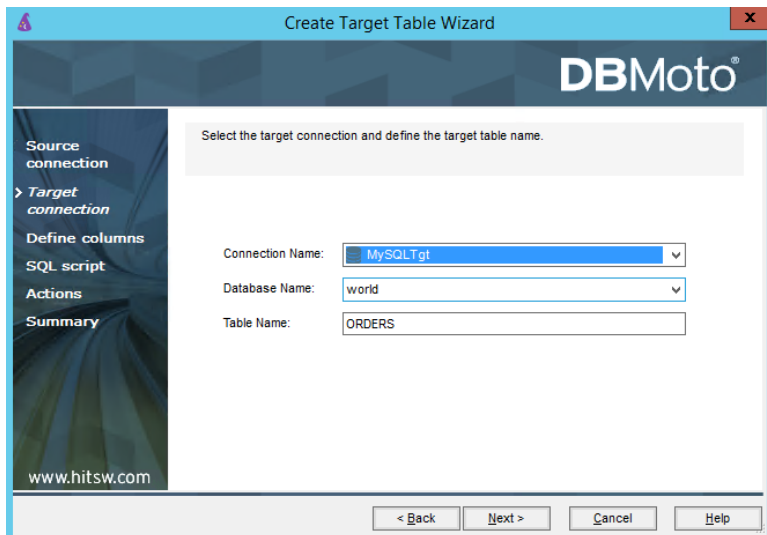
2. In the **Select Source Connection** screen, the source table information has already been provided.



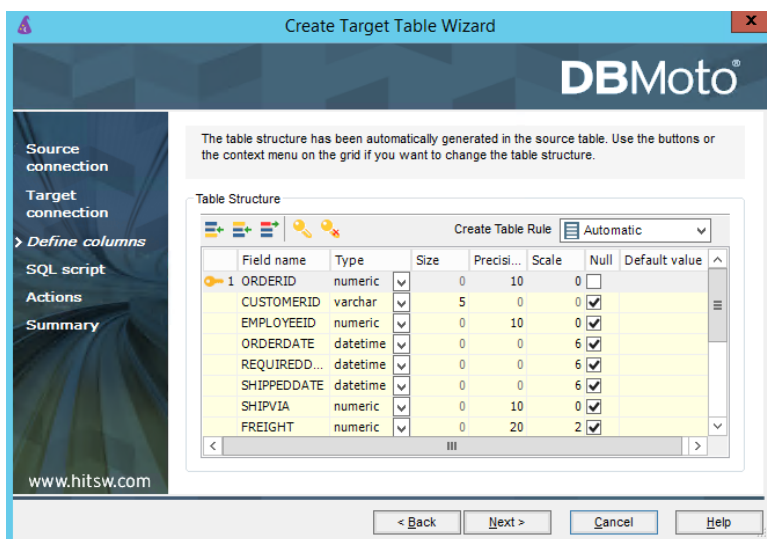
3. If you want more information about the table before proceeding, click **Open Table....**

4. Click **Next** to go to the **Select Target Connection** screen.

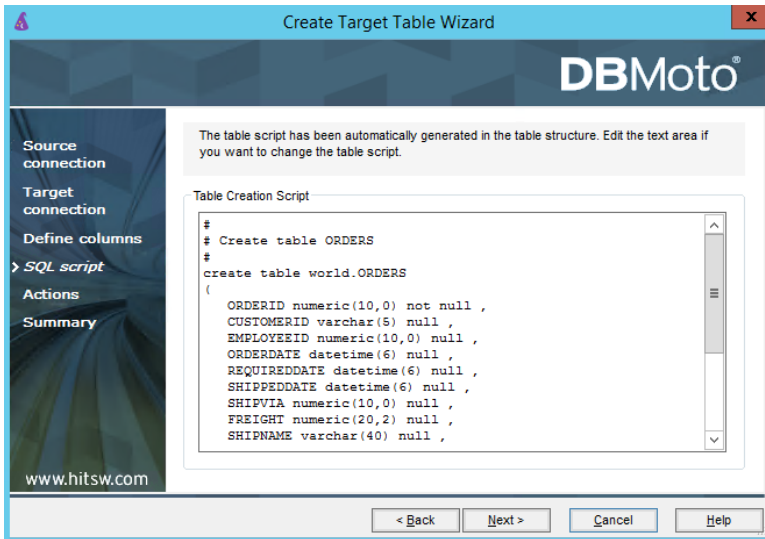




5. The target connection details have been provided. Change the table name if needed.
6. Click **Next** to go to the **Define Columns** screen.



7. Review the columns that will be created in the target table. You can add or remove columns as well as designate one or more columns as a primary key.
8. When you have finished editing the target table columns, click **Next** to go to the **SQL Script** screen.



9. If necessary, edit the SQL script that generates the table.
10. Click **Next** to display the Actions screen.
11. To create additional tables, check the **Launch Create Target Table wizard** option.  
This opens another Create Target Table wizard when this wizard is complete.
12. To go directly to creating a replication once this wizard is complete, check **Launch Create Replication wizard** option.  
This opens the [Replication wizard](#) when the Create Target Table wizard is complete.
13. Click **Next** to view a summary.
14. Click **Finish** to create the target table.
15. Note that the new table is not automatically displayed in the Metadata Explorer. Click **Yes** to add the newly created table to the list of target tables in the Metadata Explorer.

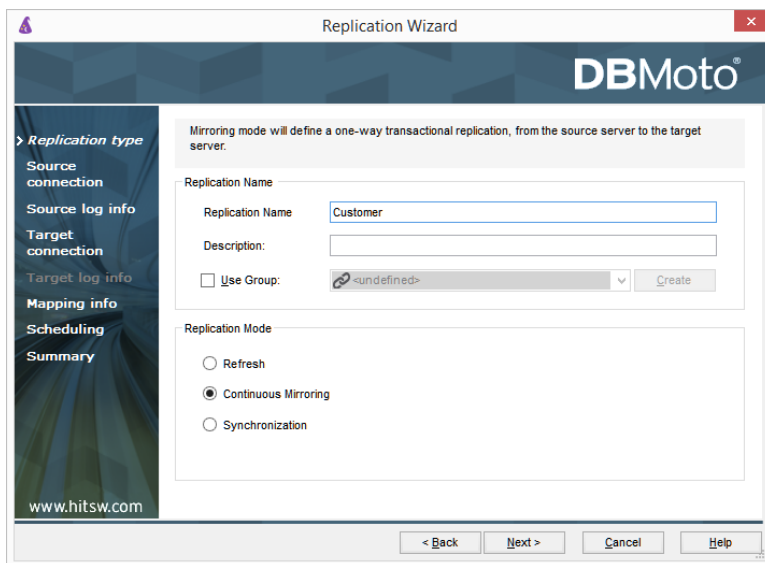


## 5. Define the Replication

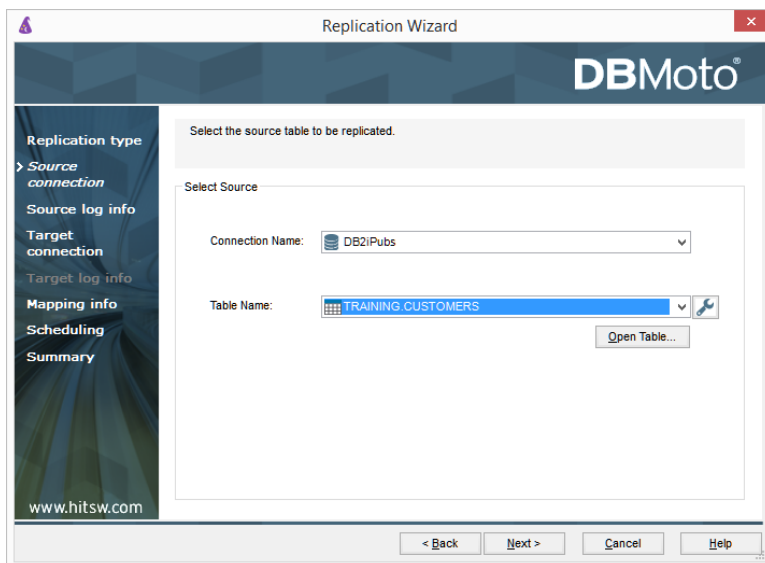
This section assumes you have checked the Create Replication wizard option to launch the Create Replication wizard. If not, to open the wizard from the Management Center, choose the table you want to replicate in the Metadata Explorer, then right-click to choose **Replication > Create New Replication...**

1. In the **Define Replication Type** screen, type a name to identify the replication.
2. Optionally provide a description of the replication.

3. In the Replication Mode area, choose **Continuous Mirroring**.



4. Click **Next** to go to the Select Source Connection screen.

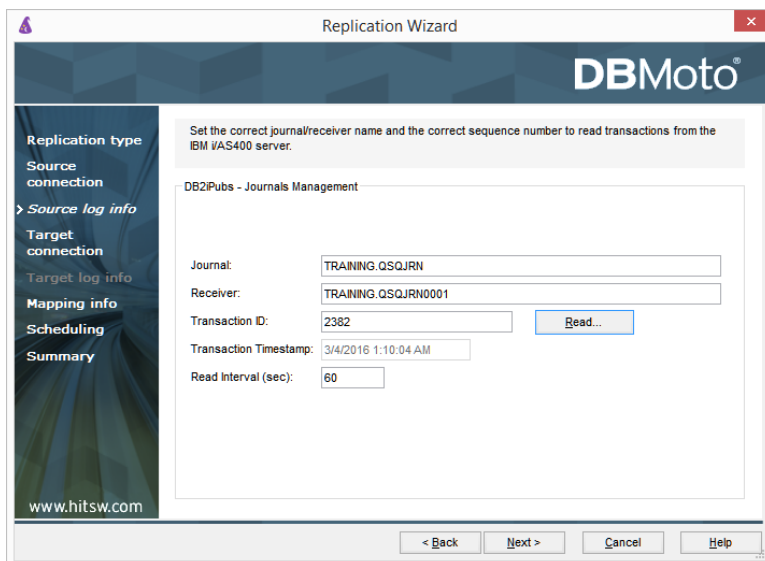


5. Choose the source name from the drop-down list that includes all the source connections you have created in DBMoto.

6. Choose the table that you want to replicate from the drop-down list.

7. If you want more information about the table before proceeding, click **Open Table....**

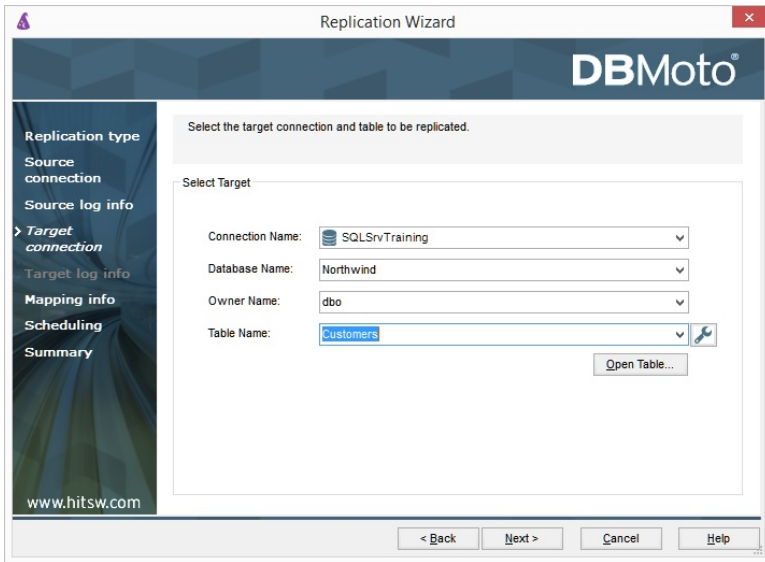
8. Click **Next** to go to the **Source Log Info** screen.



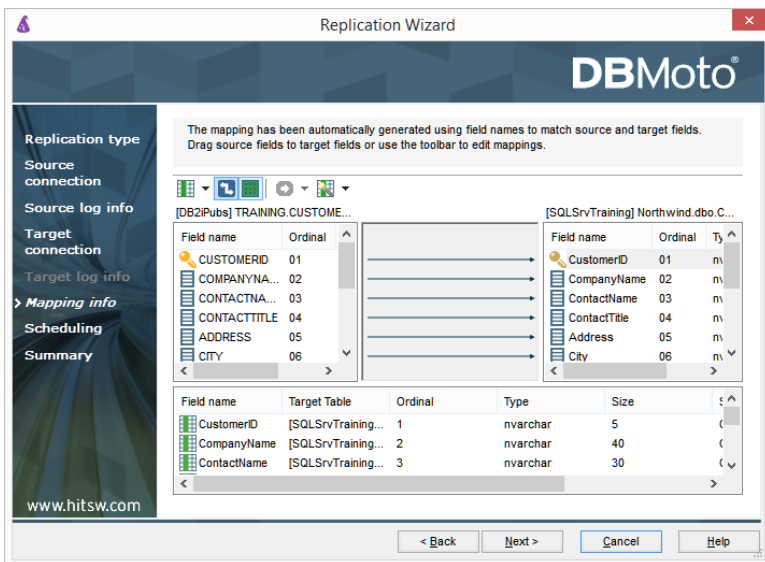
The first two fields on this screen depend on the source database you are using. In this explanation, we assume you are using IBM Db2 for i (iSeries or AS/400). Check the help for the [Replication Wizard](#) for details on the other databases.

The Journal field is automatically filled in by retrieving the information from Db2 for i. The Receiver field will be automatically filled in after setting the Transaction ID. You do not need to modify these fields. However, if the library that you have specified as a source is not journaled, you will need to ask your system administrator to journal the library.

9. In the Transaction ID field, click **Read** to open the Read Point dialog.
10. Choose either the current transaction or a transaction ID based on a time and date.
11. Click **OK** to add the value to the **Source Log Info** screen.
12. Set the value of the **Read Interval** field to the frequency with which you want DBMoto to check the transaction log for new events to mirror.
13. Click **Next** to go to the **Select Target Connection** screen.



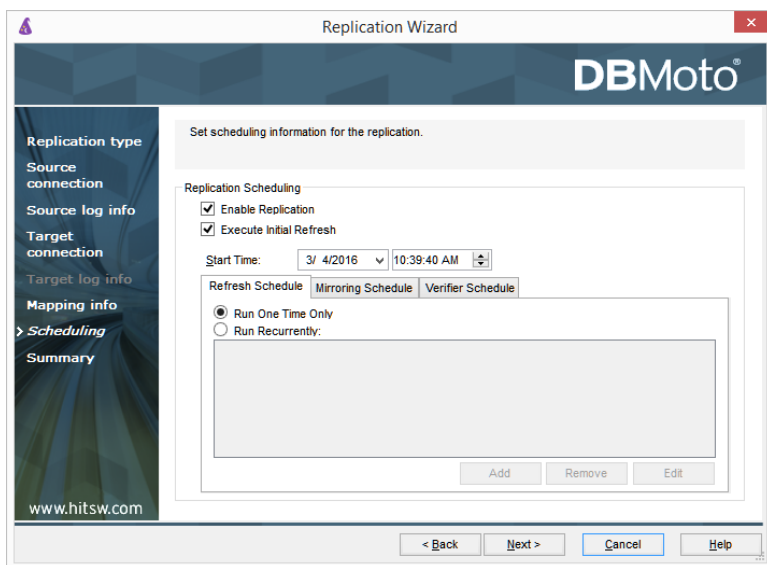
14. Choose a target source name from the drop-down list that includes all the target connections you have created in DBmoto.
15. Choose the table to which you want to replicate from the drop-down list.  
If no tables are listed, you need to exit the wizard and add or create a target table.
16. If you want more information about the table before proceeding, click **Open Table...**
17. Click **Next** to go to the **Set Mapping Info** screen.



Source and target columns with the same name are automatically mapped. You can also map columns by dragging

the target column and dropping it on the source column, or you can map a column to an expression. For more information about mapping, check the [Replication Wizard help topic](#). An alternative is to [write a script](#) to set mappings at runtime.

18. Click **Next** to go to the **Scheduling** screen.



19. Make sure the **Enable Replication** option is checked. This is required for the replication to run.

20. Set a start time for the replication. The **Start Time** field indicates the time at which the Data Replicator will begin considering the replication for execution.

21. Check the option to **Execute Initial Refresh** if needed.

If you check this option, a full replication will be performed from the source to the target table, prior to starting the mirroring process where only incremental changes will be replicated.

22. Go to the Mirroring Schedule tab.


23. Select how you want to run the replication:

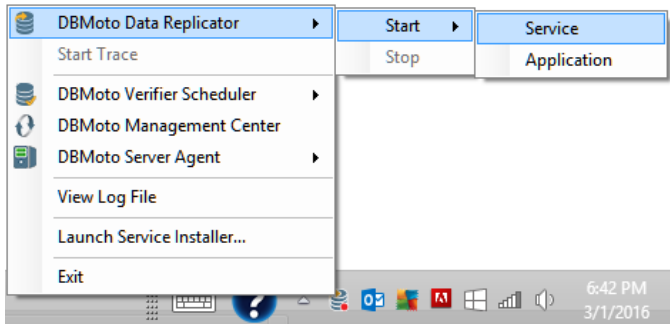
- **Run Continuously:** the transaction log will be checked for changes to the table using the frequency that you specified on the Set Log Info tab. Any changes will be replicated to the target table.
- **Schedule Interruptions:** the replication process will run as above, except for interruptions specified when you click the Schedule button in the [Scheduler dialog](#).

24. Click **Next** to go to the **Summary** screen.

25. Click **Finish** to complete the wizard.

## 4. Run the Replication

If you installed The DBMoto Data Replicator as a service during DBMoto setup, you just need to start the service using the DBMoto Service Monitor program  in the Windows Notification Area.




The replication that you have scheduled should start at the specified time. Use the [Replication Monitor](#) tab in the DBMoto Management Center to track the progress of the replication.

If you would like to install the DBMoto Replicator as a service:

- From the Windows Desktop **Start** menu, choose **Programs**, then **HiT Software DBMoto**, then **Service Installer**.
- Manage the service from DBMoto Service Monitor program (located in the DBMoto install folder or on the Windows **Start > Programs > Startup** menu).
- Use the [Replication Monitor](#) tab in the Management Center to track the progress of the replication.

To run the DBMoto Data Replicator interactively:

- In the Windows Notification Area, select  the DBMoto Service Monitor icon.
- From the right mouse button menu, choose **DBMoto Data Replicator**, then **Start** then **Application**. The replication that you have scheduled should start at the specified time.
- Use the [Replication Monitor](#) tab in the Management Center to track the progress of the replication.

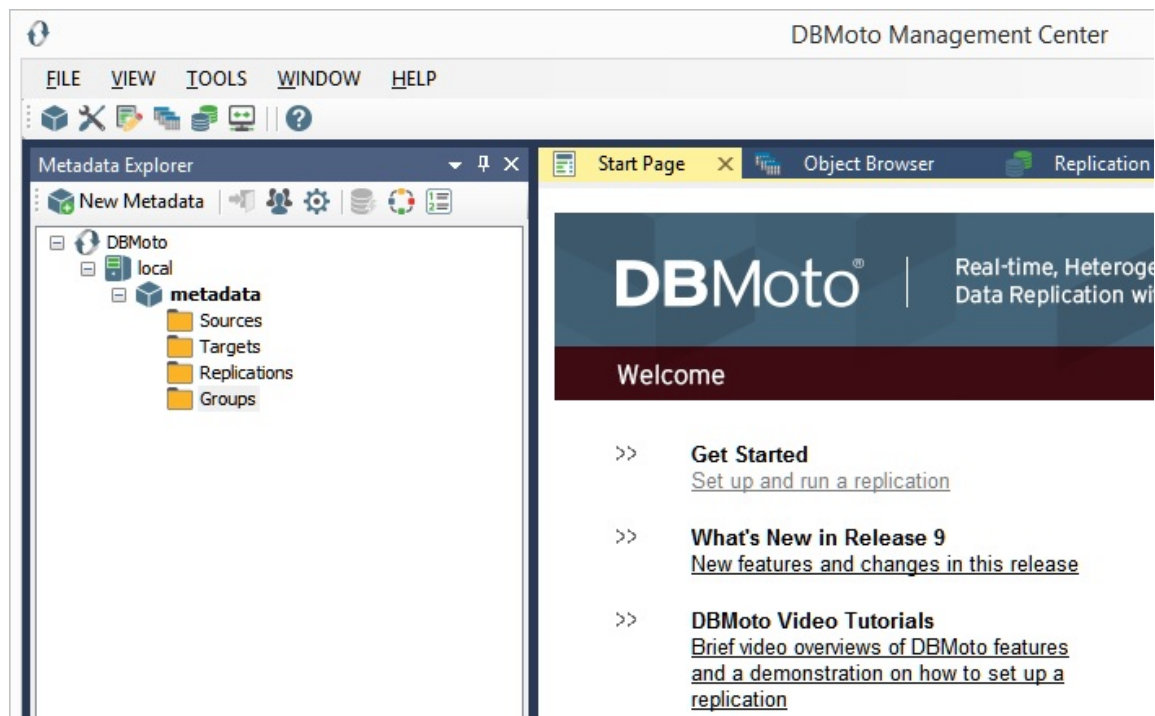
## Steps for Replicating with Synchronization

[Set up source database connection](#)>[Enable transactional replication](#)>[Set up target database connection](#)>[Create a target table](#)>[Enable target transactional replication](#)>[Define the Replication](#)>[Run the Replication](#)

### 1. Set Up Database Connections

1. Make sure you have [database connections via a .NET data provider](#) to the supported databases for source database tables and target database tables.
2. Start DBMoto Management Center.

DBMoto provides a default database (Microsoft SQL Server CE) for your DBMoto metadata, all the information that DBMoto needs to store about your replication setup.



3. In the Metadata Explorer, expand the metadata node to view the **Sources** and **Targets** nodes.

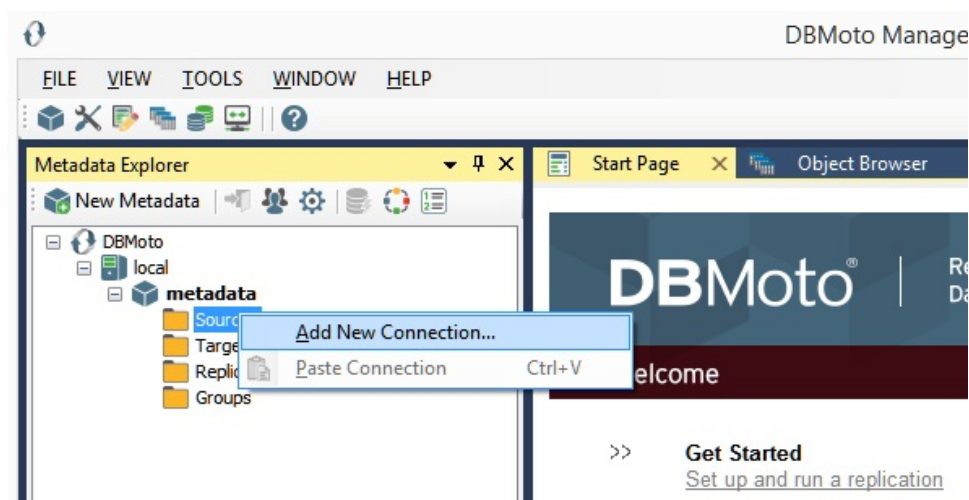
For tables involved in a synchronization, both tables serve as source and target tables.

Changes to Table A (source) can cause Table B to be updated (target), but changes to Table B



(source) can also cause Table A (target) to be updated. When defining a synchronization in DBMoto, you need to specify one connection as the source connection and one connection as the target connection although they will both be used for source and target operations. Note that if the replication involves tables that have not yet been created, you should set the connection for the new tables as the target connection so that you can use DBMoto to create the tables based on tables defined in the source connection.

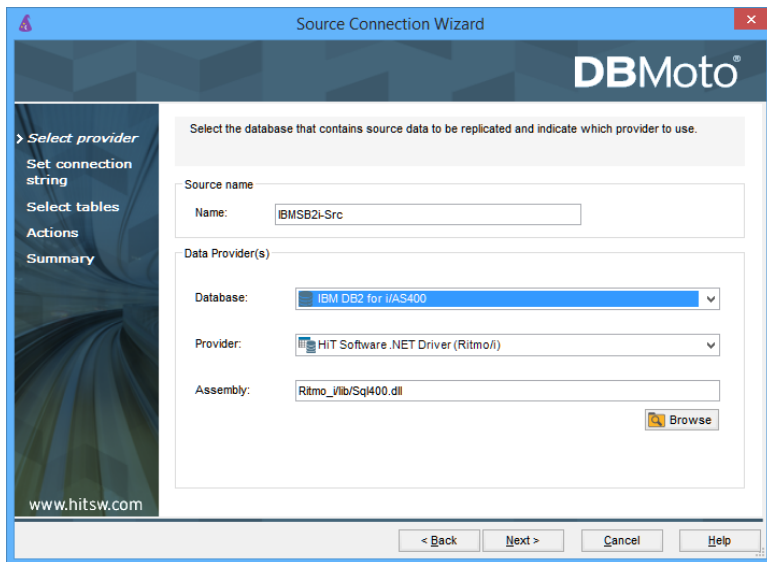
4. Select the **Sources** node.
5. From the right mouse button menu, choose **Add New Connection**.



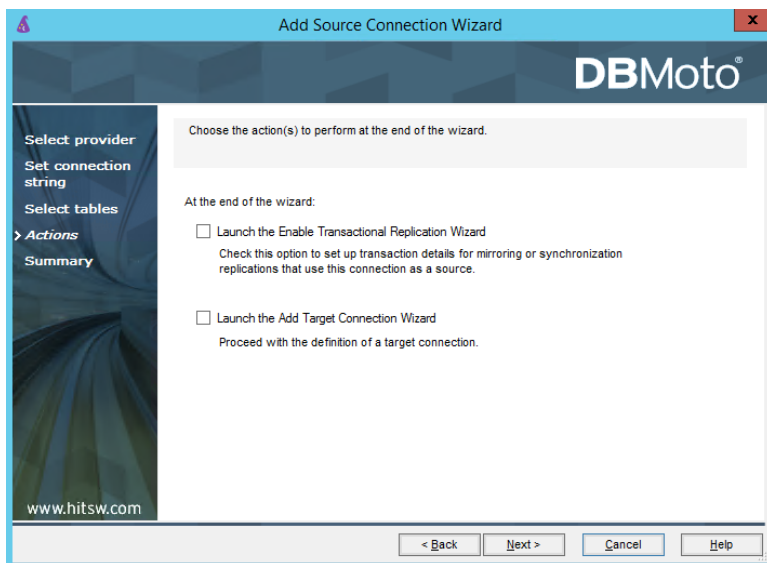
6. In the [Add Source Connection Wizard](#), follow steps to add a connection string and test the connection to the database.

Check the Supported Provider List in the [Help Center](#) before entering a value in the **Assembly** field.

Note that 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.



7. In the **Select Tables** screen, choose the tables that you plan to replicate.
8. In the **Actions** screen, check the option **Launch the Enable Transactional Setup Wizard**.



9. Complete the wizard.

## 2. Enable Transactional Replication

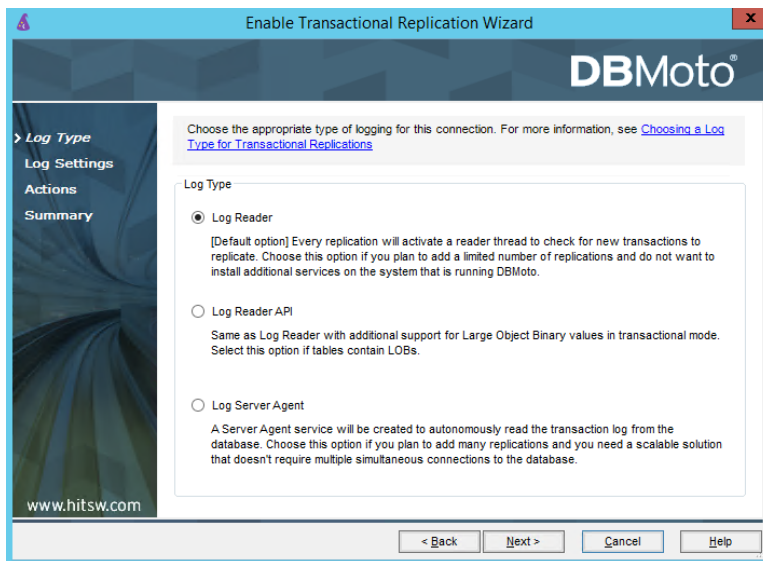
This section assumes you have checked the Source Connection wizard option to launch the Enable Transactional Replication wizard. If not, to open the wizard from the Management Center, choose the connection in the Metadata Explorer, then right-click to choose **Transactional**

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.

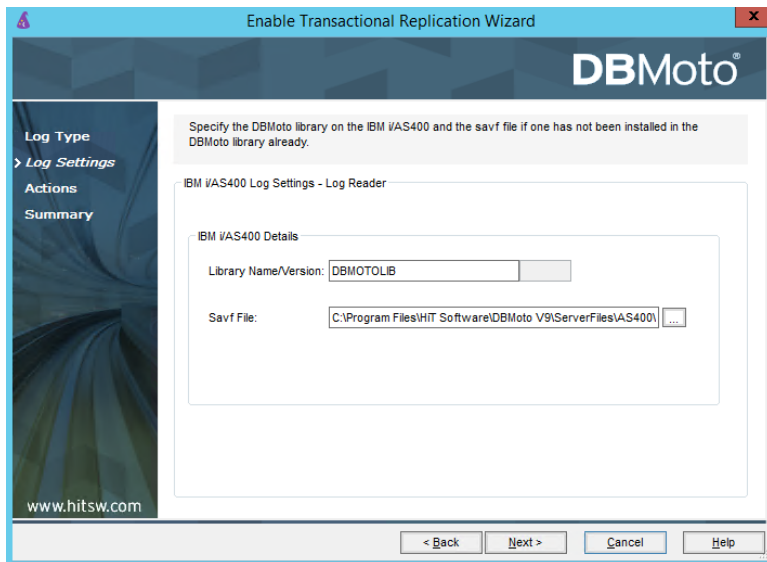
## Setup > Enable...

In the [Enable Transactional Replication wizard](#):

1. Select the type of transactional replication to use. The options depend on the source database and can include [log reader, log server agent, triggers, plus log reader API](#) (for IBM Db2 for i only)



2. Click **Next** to enter the log settings. The fields and appropriate values depend on the [database and log type](#).

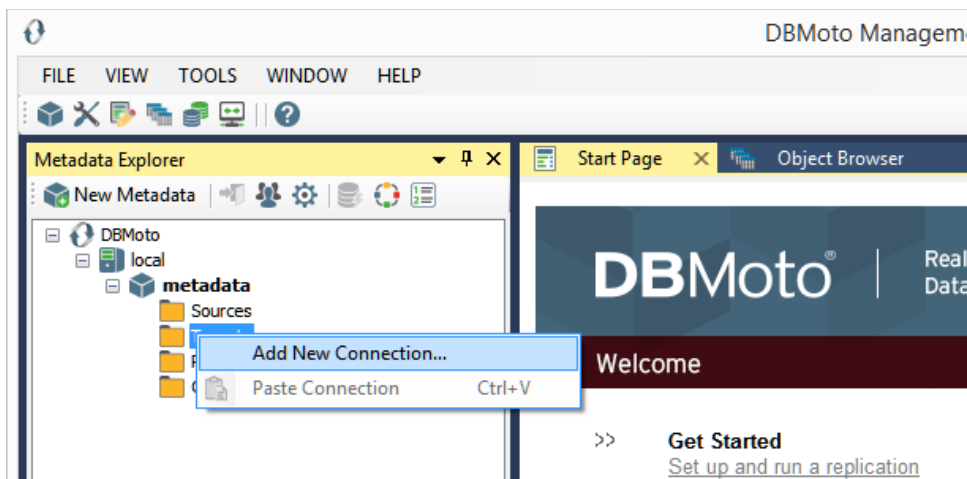


3. Click **Next** to verify your settings against the source connection to the database. If any information is missing, you will not be able to proceed.
4. In the Actions screen, check the option **Launch Add Target Connection wizard**.
5. Click **Next** to review your changes.
6. Click **Finish** to complete the wizard.

The source connection is now set up for transactional replications.

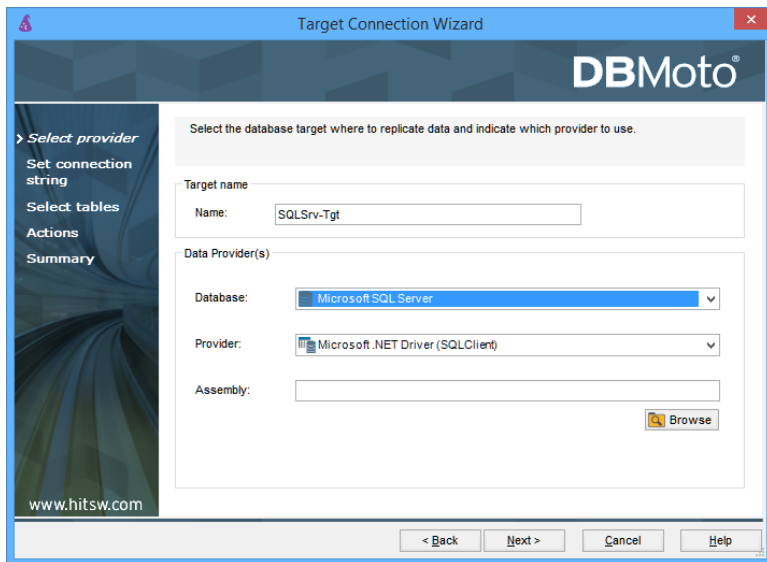
### 3. Set Up Target Database Connection

This section assumes you have checked the Add Target Connection wizard option to launch the Add Target Connection wizard. If not, to open the wizard from the Management Center, choose **Targets** in the Metadata Explorer, then right-click to choose **Add New Connection...**



7. In the [Add Target Connection Wizard](#), follow steps to add a connection string and test the connection to the database. Check the Supported Provider List in the [Help Center](#) before entering a value in the **Assembly** field.

Note that 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.

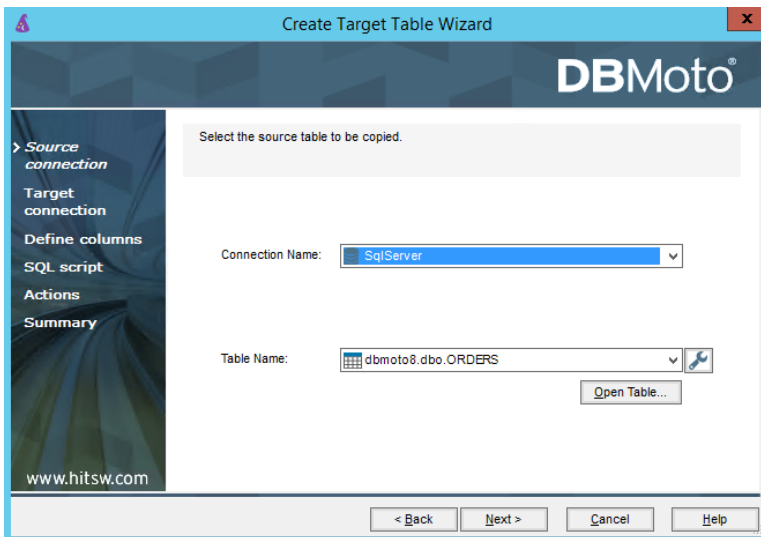


8. Choose the tables to which you plan to replicate.  
If the table does not exist, continue to the next screen without selecting a table.
9. Complete the wizard.

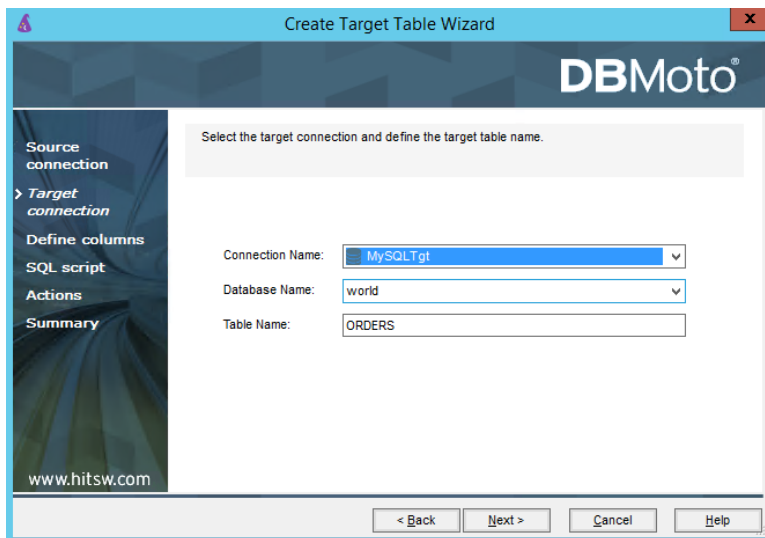
#### 4. Create a Target Table

If the target table does not exist, use the Create Target Table wizard to create the table.

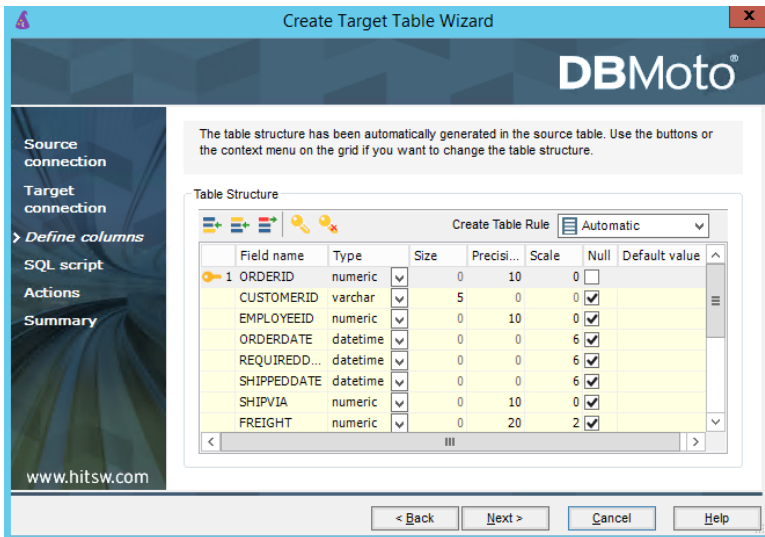
1. In the Metadata Explorer, drag the source table to the target connection to open the Create Target Table wizard.  
The **Select Source Connection** screen is already filled out with the table you selected.,



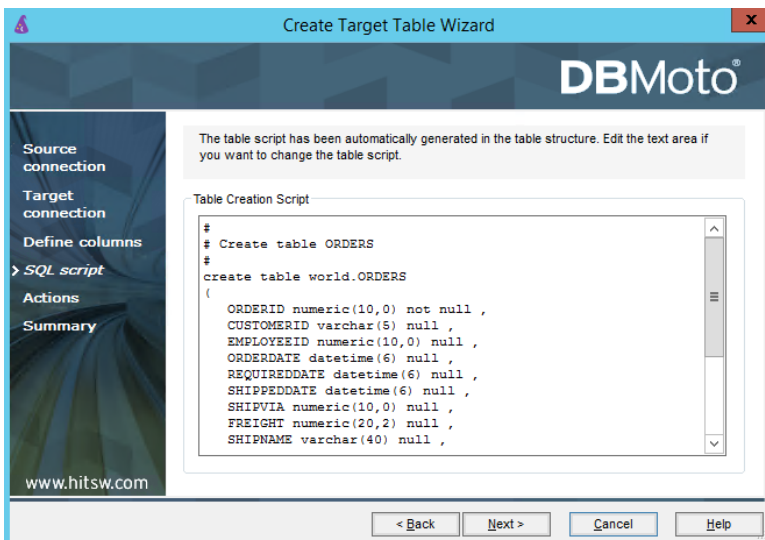
2. If you want more information about the table before proceeding, click [Open Table...](#)
3. Click **Next** to go to the **Select Target Connection** screen.  
The screen fields have already been completed with the target connection details.



4. Click **Next** to go to the **Define Columns** screen.



5. Review the columns that will be created in the target table. You can add or remove columns as well as designate one or more columns as a primary key.
6. When you have finished editing the target table columns, click **Next** to go to the **SQL Script** screen.



7. If necessary, edit the SQL script that generates the table.
8. Click **Next** to display the Actions screen.
9. To create additional tables, check the **Launch Create Target Table wizard** option.  
This opens another Create Target Table wizard when this wizard is complete.



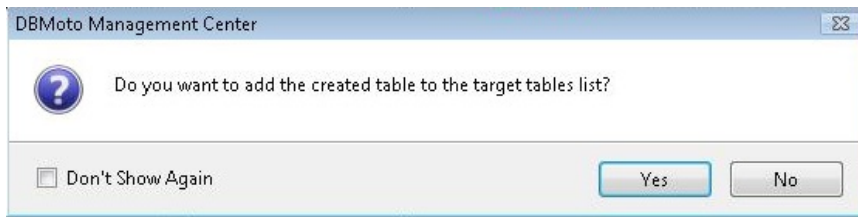
10. To go directly to creating a replication once this wizard is complete, check **Launch Create Replication wizard** option.

This opens the [Replication wizard](#) when the Create Target Table wizard is complete.

11. Click **Next** to view a summary.

12. Click **Finish** to create the target table.

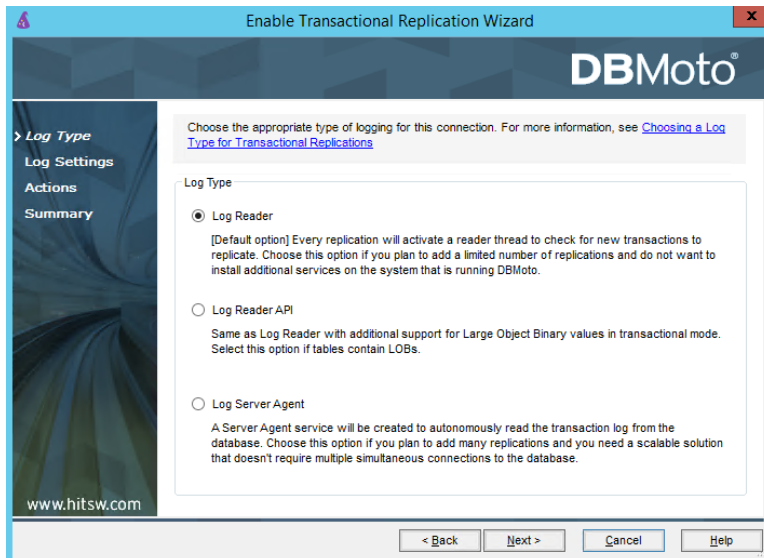
13. Note that the new table is not automatically displayed in the Metadata Explorer. Click **Yes** to add the newly created table to the list of target tables in the Metadata Explorer.



## 5. Enable Target Transactional Replication

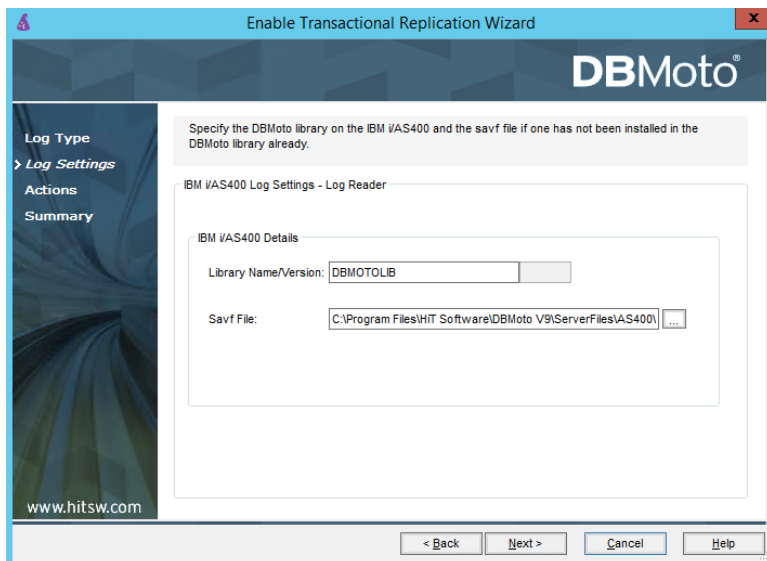
1. Choose the target connection in the Metadata Explorer, then right-click to choose **Transactional Setup > Enable...**

2. In the [Enable Transactional Replication wizard](#), select the type of transactional replication to use. The options depend on the source database and can include [log reader](#), [log server agent](#), [triggers, plus log reader API](#) (for IBM Db2 for i only)



3. Click **Next** to enter the log settings. The fields and appropriate values depend on the [database](#)

[and log type.](#)



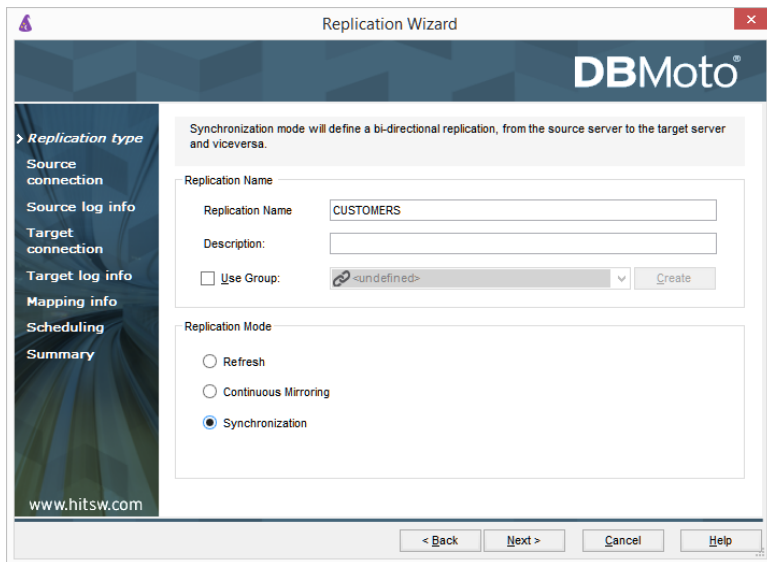
4. Click **Next** to verify your settings against the source connection to the database. If any information is missing, you will not be able to proceed.
5. In the Actions screen, check the option **Launch Create Replication wizard**.
6. Click **Next** to review your changes.
7. Click **Finish** to complete the wizard.

The target connection is now set up for synchronization with the source database.

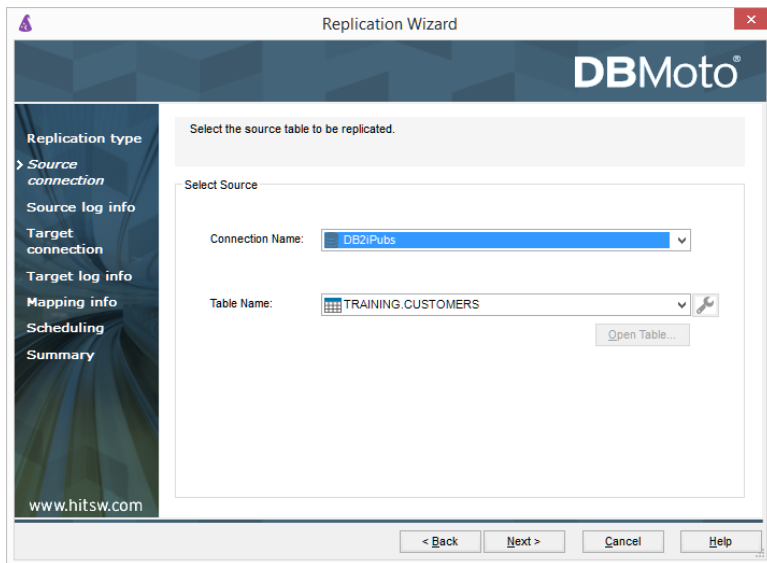
## 6. Define the Replication

This section assumes you have checked the Create Replication wizard option to launch the Create Replication wizard. If not, to open the wizard from the Management Center, choose the table you want to replicate in the Metadata Explorer, then right-click to choose **Replication > Create New Replication...**

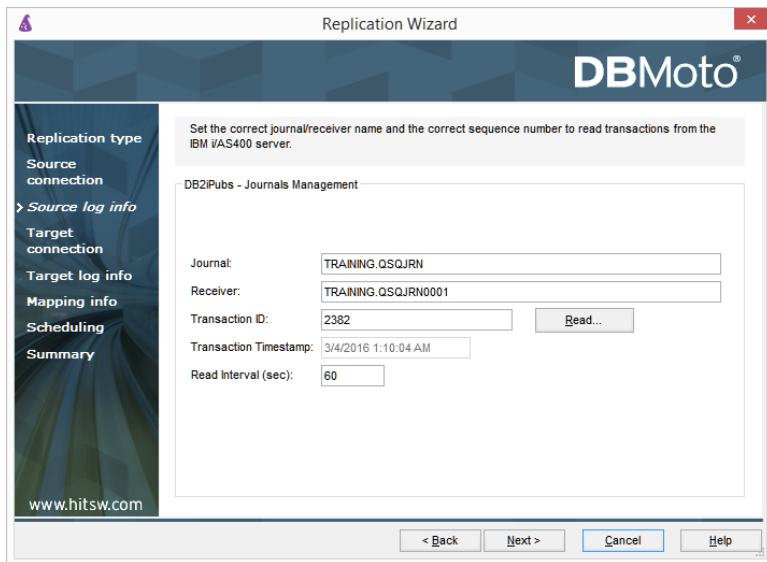
1. In the **Define Replication Type** screen, type a name to identify the replication.
2. Optionally provide a description of the replication.
3. In the Replication Mode area, choose **Synchronization**.



4. Click **Next** to go to the **Select Source Connection** screen.



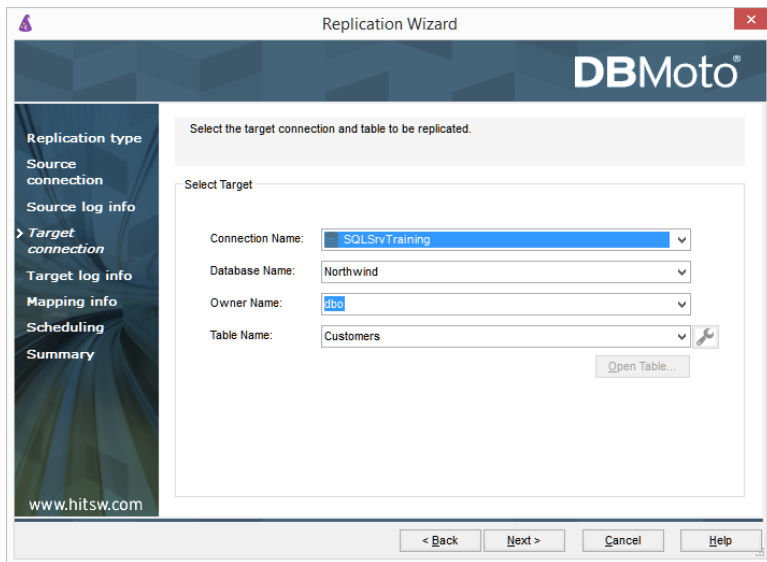
5. Choose the source name from the drop-down list that includes all the source connections you have created in DBMoto.
6. Choose the table that you want to replicate from the drop-down list.
7. If you want more information about the table before proceeding, click [Open Table...](#)
8. Click **Next** to go to the **Source Log Info** screen.



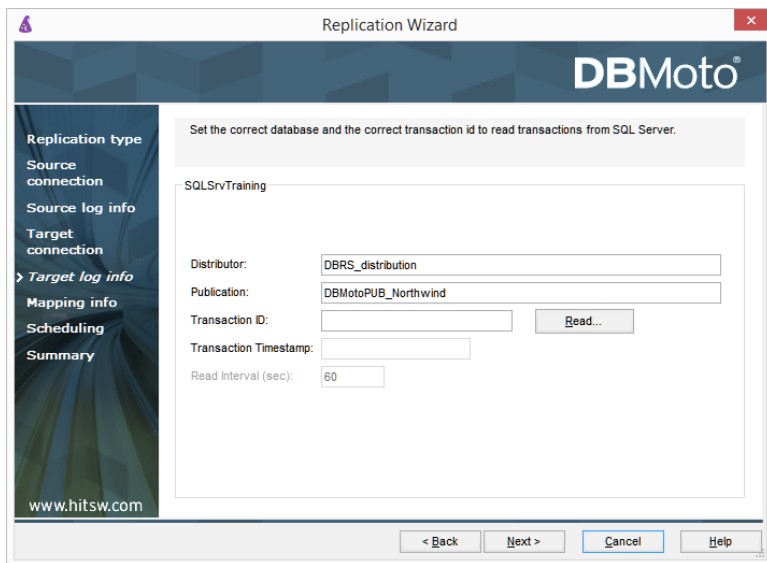
The first two fields on this screen depend on the source database you are using. In this explanation, we assume you are using IBM Db2 for i (iSeries or AS/400). Check the help for the [Replication Wizard](#) for details on the other databases.

The Journal field is automatically filled in by retrieving the information from Db2 for i. The Receiver field will be automatically filled in after setting the Transaction ID. You do not need to modify these fields. However, if the library that you have specified as a source is not journaled, you will need to ask your system administrator to journal the library.

9. In the **Transaction ID** field, click **Read** to open the Read Point dialog.
10. Choose either the current transaction or a transaction ID based on a time and date.
11. Click **OK** to add the value to the **Source Log Info** screen.
12. Set the value of the **Read Interval** field to the frequency with which you want DBMoto to check the transaction log for new events to mirror.
13. Click **Next** to go to the **Select Target Connection** screen.

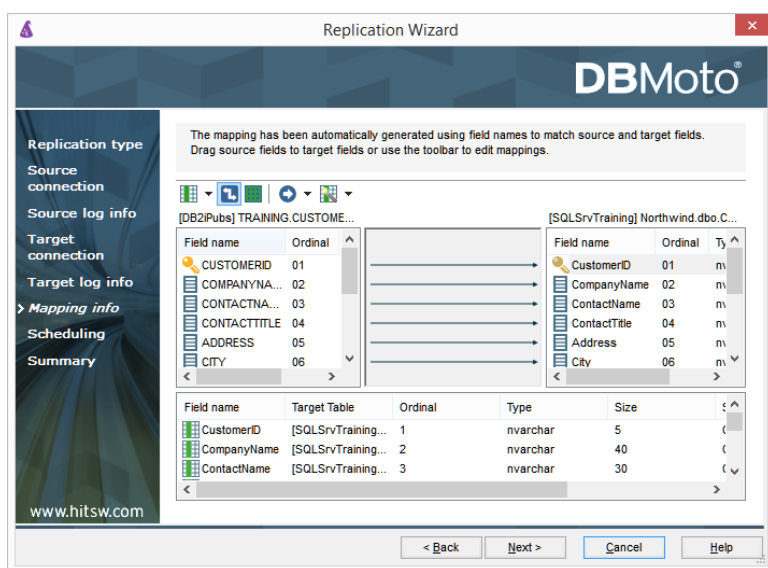


14. Choose a target source name from the drop-down list that includes all the target connections you have created in DBMoto.
15. Choose the table to which you want to replicate from the drop-down list.  
If no tables are listed, you need to exit the wizard and add or create a target table.
16. If you want more information about the table before proceeding, click **Open Table...**
17. Click **Next** to go to the **Target Log Info** screen.

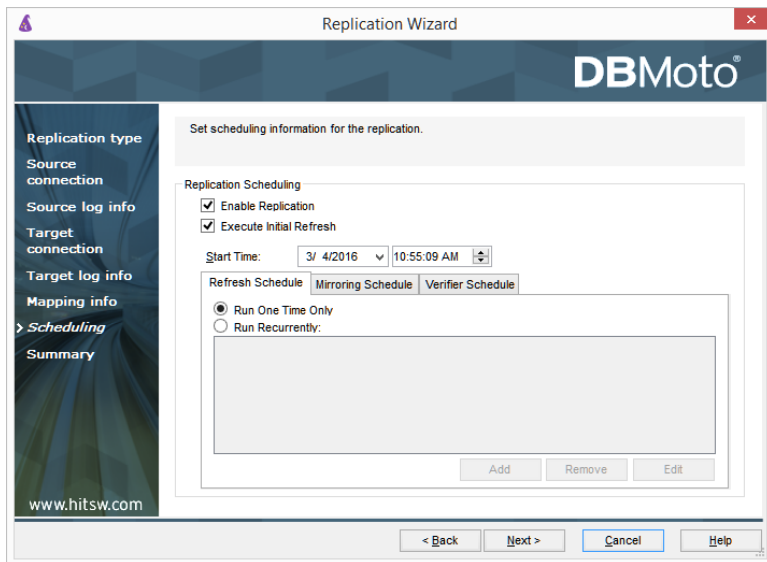


Note: If you encounter a connection error at this stage, check that you have used the

- appropriate IDs for your connections by going to the [Connection Properties dialog](#) for your source and target connections. For SQL Server, you need a non system administrator login for replication and a system administrator login for accessing the transaction log.
18. In the **Transaction ID** field, click **Read** to open the Read Point dialog.
  19. Choose either the current transaction or a transaction ID based on a time and date.
  20. Click **OK** to add the value to the **Target Log Info** screen.
  21. Set the value of the **Read Interval** field to the frequency with which you want DBMoto to check the transaction log for new events to mirror.
  22. Click **Next** to go to the **Set Mapping Info** screen.



- Source and target columns with the same name are automatically mapped. You can also map columns by dragging the target column and dropping it on the source column, or you can map a column to an expression. For more information about mapping, check the [Replication Wizard help topic](#). An alternative is to [write a script](#) to set mappings at runtime.
23. Click **Next** to go to the **Scheduling** screen.



24. Make sure the **Enable Replication** option is checked. This is required for the replication to run.

25. Set a start time for the replication. The **Start Time** field indicates the time at which the Data Replicator will begin considering the replication for execution.

26. Check the option to **Execute Initial Refresh** if needed.

If you check this option, a full replication will be performed from the source to the target table, prior to starting the synchronization process where only incremental changes will be replicated. The initial refresh is always performed from the source connection to the target connection. Note that any transaction submitted during the time that the refresh is running might not be replicated. It is strongly suggested that you avoid updating the designated source and target tables until the refresh is done.

27. Go to the Mirroring Schedule tab.


28. Select how you want to run the replication:

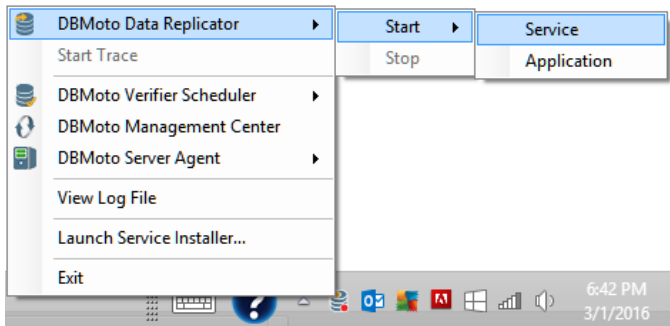
- **Run Continuously:** the transaction log will be checked for changes to the table using the frequency that you specified on the Set Log Info tab. Any changes will be replicated to the target table.
- **Schedule Interruptions:** the replication process will run as above, except for interruptions specified when you click the Schedule button in the [Scheduler dialog](#).

29. Click **Next** to go to the **Summary** screen.

30. Click **Finish** to complete the wizard.

## 4. Run the Replication

If you installed The DBMoto Data Replicator as a service during DBMoto setup, you just need to start the service using the DBMoto Service Monitor program  in the Windows Notification Area.




The replication that you have scheduled should start at the specified time. Use the [Replication Monitor](#) tab in the DBMoto Management Center to track the progress of the replication.

If you would like to install the DBMoto Replicator as a service:

- From the Windows Desktop **Start** menu, choose **Programs**, then **HiT Software DBMoto**, then **Service Installer**.
- Manage the service from DBMoto Service Monitor program (located in the DBMoto install folder or on the Windows **Start > Programs > Startup** menu).
- Use the [Replication Monitor](#) tab in the Management Center to track the progress of the replication.

To run the DBMoto Data Replicator interactively:

- In the Windows Notification Area, select  the DBMoto Service Monitor icon.
- From the right mouse button menu, choose **DBMoto Data Replicator**, then **Start** then **Application**.

The replication that you have scheduled should start at the specified time.

- Use the [Replication Monitor](#) tab in the Management Center to track the progress of the replication.





## Chapter 2: Setting Up Your Environment

### Requirements

The following are suggested minimum requirements for the environment where you will run the DBMoto Data Replicator. The actual requirements depend very much on your environment and your data replication needs. For more information, contact HiT Software Technical Support at [dbmoto-support@hitsw.com](mailto:dbmoto-support@hitsw.com).

- The system where DBReplicator is running should have a processor of at least 2 GH. In a production environment with fewer than 75 replications, dual-core processors are recommended, and for more than 75 replications, quad-core processors are recommended for efficient processing.
- For the system where the DBMoto Data Replicator is running, 4 GB memory is recommended, and at least 5GB hard disk space.
- .NET Framework 4.0 or 4.5 must be installed.
- You need a .NET connection to the database(s) that you want to use in replication (except in the case of Netezza where an ODBC driver is supported.) Check [HiT Software's web site for a list of supported drivers and providers](#).
- If using Oracle as a source, target or metadata database, the Oracle client must be installed on the same system as the one where DBMoto is running.
- If using MySQL as a source for mirroring via a transaction log, you need to use MySQL version 5.1.5 or above and you should contact HiT Software technical support staff at [dbmoto-support@hitsw.com](mailto:dbmoto-support@hitsw.com) to obtain an additional component, available on request only.
- If using IBM Db2 for i (iSeries/AS400), the operating system version must be V3R2 or later.
- If using IBM Db2 as a source database, the following versions are supported:
  - Db2 UDB v. 7.2 or higher
  - Db2 for OS390 v6 or higher

### Managing a DBMoto License

When you first download DBMoto from HiT Software's web site, you should receive your DBMoto evaluation license via email. The email consists of a text message with your company name and license key. A license file called `dbmotolicense_<company>.txt` is attached to the email.

During installation of DBMoto, you are required to supply the pathname to the license file.

If at any time you need to update your DBMoto license (typically when you purchase DBMoto and receive a permanent license), you can use the [License Information dialog](#) in the Management Center. Note that you must always have a single valid license file for DBMoto. New license files received from HiT Software should contain ALL your license details. Therefore, when you import a new license file, it will replace the previous license file.

1. In the DBMoto Management Center Metadata Explorer, select the server for which you want to modify the license.
2. From the right mouse button menu, choose **Manage Licenses**.
3. In the License Information dialog, click **Import License**.

4. In the new dialog, provide the pathname to the license file (dbmotolicense\_ *company*.txt) you received via email from HiT Software.
5. Click **OK** to close the dialog.
6. Verify your new license in the License Information dialog.
7. If the license key is not valid, double-check that you have entered all information correctly. Contact HiT Software technical support if you encounter further problems. (email: [dbmoto-support@hitsw.com](mailto:dbmoto-support@hitsw.com).)

To view existing license information at any time:

1. In the DBMoto Management Center Metadata Explorer, select the server for which you want to modify the license.
2. From the right mouse button menu, choose **Manage Licenses**.
3. Click **Export** to print all license information to a file.

## Installing DBMoto

DBMoto is distributed as a setup file and must be installed on a Microsoft Windows system that meets the [configuration requirements](#). DBMoto can be installed as a **standard installation** (entire DBMoto environment including DBMoto Management Center, DBMoto Replicator and Server Agent) or as a **client** (DBMoto Management Center only for managing replications remotely.) You can choose to install DBMoto on the system running the source database, or the system running the target database, or an independent system that has network access to the source and target database systems. Check [Running DBMoto on a Wide Area Network](#) for recommendations.

In addition to installing the DBMoto files, you need to:

- Install and test .NET Data Providers for your source and target databases. For a list of all supported databases, check [http://www.hitsw.com/support/kbase/DBMoto/Providers\\_DBMoto.htm](http://www.hitsw.com/support/kbase/DBMoto/Providers_DBMoto.htm).
- Verify that your databases are set up appropriately to use DBMoto using the [Source and Target Database Setup Guidelines](#)

Follow the steps below to install and set up DBMoto server.

1. Check that the system where you are installing DBMoto server meets the [DBMoto requirements](#).
2. Make sure that you have Microsoft .NET framework 4 or 4.5 installed on the system where you will be using DBMoto.
3. Make sure that you have a DBMoto 8.x license key file available.
4. Run the setup.
5. When you reach the **Setup Type** screen, click the **Standard Installation** option.
6. In the **Customer Information and Setup Options** screen, click **Import**, then supply the path name to the license file that you received via email.


7. If you want to install the DBReplicator as a Windows service, check the option, but note that the service does not start automatically by default.

The advantage of installing the data replicator as a service is that you can set it to run automatically when your server starts up again after failing. Without the DBReplicator service, you will need to run the DBReplicator manually each time your system restarts.

In addition to installation of DBMoto files, the setup program:

- Creates an entry under **HiT Software DBMoto V8** on the **Start** menu on the Windows desktop.
  - Starts the DBMoto Server Agent
  - Adds the local server (the system on which you installed DBMoto) to the DBMoto Management Center list of servers
  - Places the DBMServiceMonitor program in the Windows Notification area (System Tray.) You can use this program to [start the DBReplicator service](#) after creating metadata tables in the DBMoto Management Center.
8. On your Windows desktop, start the DBMoto Management Center in one of the following ways:
    - On your Windows desktop, double-click the DBMoto Management Center icon.



- In the Windows Notification area (System Tray), right click  **DBMoto** and choose **DBMoto Management Center** from the right mouse button menu.
  - Choose **Start**, then **All Programs**, then **HiT Software DBMoto V8**, then **Management Center** to start setting up replications.
9. In the DBMoto Management Center, you can:
    - [Add other DBMoto servers to monitor/manage](#)
    - [Establish accounts and security permissions for DBMoto users](#)
    - Create a replication. Use the following help topics to guide you:
      - [Steps for Replicating a Table Using Refresh Mode](#)
      - [Steps for Replicating with One-Way Mirroring](#)
      - [Steps for Replicating with Synchronization](#)

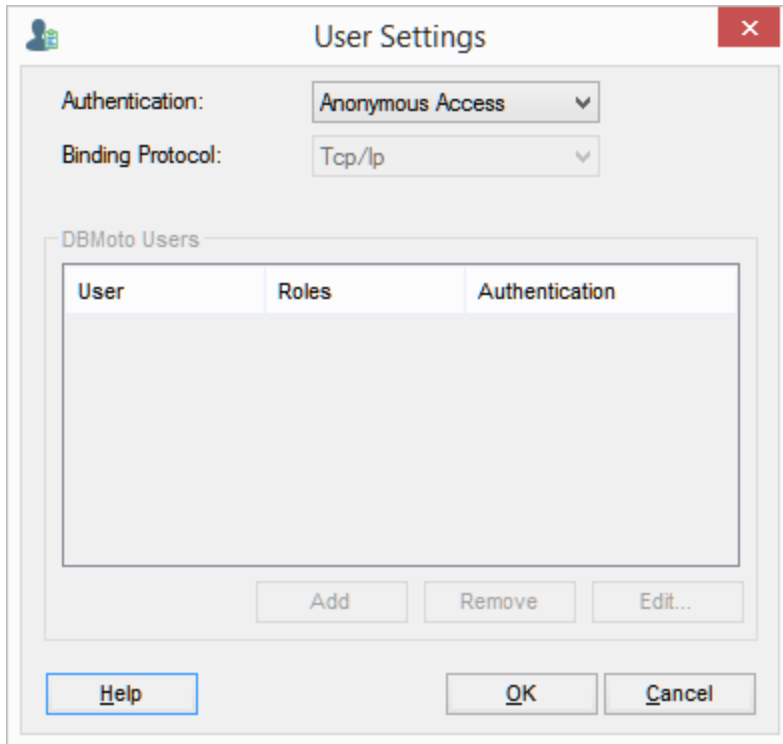
## Setting Up DBMoto Users

NOTE: To establish or modify user profiles, you must be running the DBMoto Management Center for a local installation of the server. It is not possible to modify user profiles for remote servers.

You can secure your DBMoto environment by creating user profiles with appropriate access permissions to create, manage and modify connections and replications. When you first install and run the DBMoto server's Management Center, a Windows service called DBMoto Server Agent is started automatically and a connection is made between the DBMoto Management Center and the local server (where both DBMoto components were installed.) The default name for this server appears as "local" in the Management Center. From the Management Center, you can set up restricted access to the server via user login IDs and user roles. Initially, DBMoto is set up with security disabled, thereby avoiding the security mechanism altogether. To enable security and set up user profiles for the local server:

1. In the DBMoto Management Center Metadata Explorer, select the **local** entry, and, from the right mouse button menu, choose **Manage Users....**
2. In the User Settings dialog, select the [type of security](#) you want to use.

Anonymous Access	Leave this option if you do not want to restrict access to DBMoto in any way.
DBMoto Authentication	Allows you to establish login IDs and passwords for users, but also requires the <a href="#">installation of an X.509 certificate</a> on the client system (where the DBMoto Management Center is running).
Windows Authentication	DBMoto uses your Microsoft Windows login and ID to control access to the DBMoto Management Center and server agent.
Certificate Authentication	Requires the <a href="#">installation of an X.509 certificate</a> but does not allow specific user login IDs.



For more information on security options in DBMoto, see [DBMoto Server Client Security Options](#).

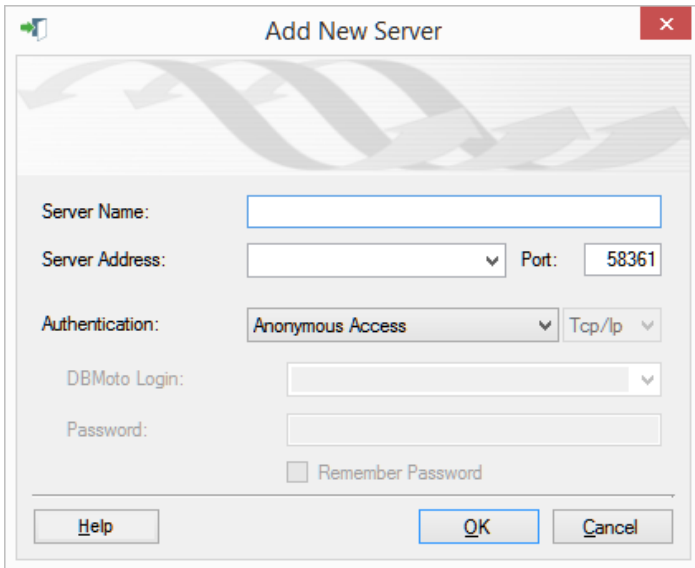
**NOTE:** If you set up a user ID with DBMoto Authentication, or Certificate Authentication, you will not be able to use DBMoto until the certificate is correctly installed. Generate and install the certificate before setting up authentication.

3. Select the type of Binding Protocol to use.. For more information on appropriate binding protocols for use with DBMoto, see [DBMoto Server Client Security Options](#).

Authentication	Binding Protocol Options
Anonymous Access	Inactive. Uses TCP/IP binding protocol
DBMoto Authentication	TCP/IP or HTTP binding protocol. TCP/IP is suitable for intranet access, HTTP is recommended for internet access.
Windows Authentication	Inactive. Uses TCP/IP binding protocol

Certificate Authentication	TCP/IP or HTTP binding protocol. TCP/IP is suitable for intranet access, HTTP is recommended for internet access.
----------------------------	---

- Click **Add** to open the Create DBMoto Login dialog.



- In the User Definition tab, check the Authentication type in the drop-down menu.

### Windows Authentication

This option sets authority for a domain user. The default value in the DBMoto Login field is the current Windows user login. This value can be edited. The password field is disabled in the dialog because the existing Windows password is expected.

### DBMoto Authentication

This option allows you to set a user name and password that will subsequently be managed by DBMoto.

- In the Permissions tab, select a User Role from the drop-down list.

The pre-defined roles set permissions for the types of operations that user would typically perform. You can customize the permissions for a user by selecting the value for the permission in the list, then choosing True or False

from the drop-down list of values. Note that the role automatically becomes Custom in this case.

<b>Administrator</b>	Permission for all operations.
<b>Auditor</b>	Permission to audit and monitor replications.
<b>Custom</b>	Permissions selected manually.
<b>Operator</b>	Permission to set up and change replication settings but cannot work on administrative tasks (such as change license, change user permissions)
<b>Public</b>	Permission to audit and monitor replications, back up metadatas and change metadata settings (such as starting/stopping traces or changing email parameters for email notification).
<b>SecurityAdmin</b>	Permission to manage security, Does not normally need to change replication settings or work on creating new metadata.

See [Create DBMoto Login Dialog](#) for more information about specific permissions and a comparison of the user roles.

7. Click **Create** to create the new user.

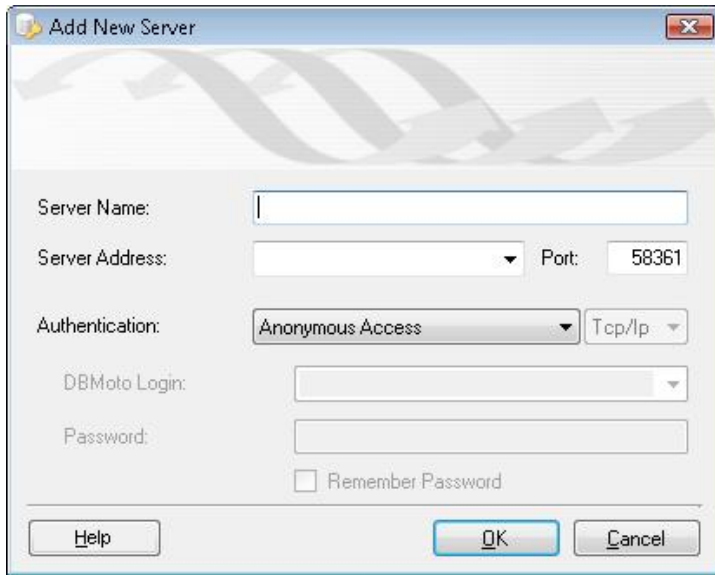
## Adding a DBMoto Server to the Management Center

When you first install DBMoto Server, the Management Center already has a connection to the local server, the system where you are running the Management Center and the DBMoto Replicator. However, you can view and manage multiple DBMoto installations from the Management Center, as long as you have a user profile which allows you to access remote DBMoto servers. The user profile must be established on the remote servers before you attempt to connect from the Management Center.

In the DBMoto Management Center Metadata Explorer:

1. Select the root entry DBMoto.
2. From the right mouse button menu, choose Add New Server...
3. In the Add New Server dialog, type the server name.





4. In the Server Address field, type the IP address for the server.
5. In the Port field, set the port number for DBMoto to use in contacting the server.
6. Choose the Authentication type from the drop-down list.

The authentication type, login and password must match a user profile already established on the DBMoto server to which you are trying to connect.

NOTE: If you set up a user ID with DBMoto Authentication, or Certificate Authentication, you will not be able to use DBMoto until the certificate is correctly installed. Generate and install the certificate before setting up authentication.

7. Choose the protocol.
 

Choose TCP/IP for intranet communication with a DBMoto client, or HTTP for Internet communication with a client. For more information, check DBMoto Server Client Security Options.
8. Enter login and password information if needed (active for DBMoto and Windows Authentication only).
9. Click OK to add the server to the Metadata Explorer.

## DBMoto Client and Server Security Options

DBMoto can be used in several different configurations, and each configuration has different data security needs. In the descriptions below, DBMoto client can be either an installation of the Management Center or an application using the DBMoto API to control the Data Replicator.

## DBMoto Client and Server Agent on a Single System

Security threats are typically not present, so Anonymous Authentication is appropriate, unless you want to define user IDs control access to DBMoto tools. In this case, Windows authentication may be appropriate because it does not require the installation of an X.509 certificate. DBMoto uses TCP/IP as a protocol for both Anonymous Authentication and Windows Authentication.

## DBMoto Client and Server Agent in an Intranet Trusted Environment

This type of environment would include the situation where an additional Management Center is installed on a second system within a secure intranet for monitoring or use by a second user. Security threats are typically not present, so Anonymous Authentication is appropriate, unless you want to define user IDs control access to DBMoto tools. In this case, Windows authentication may be appropriate because it does not require the installation of an X.509 certificate. DBMoto uses TCP/IP as a protocol for both Anonymous Authentication and Windows Authentication.

It is also possible to establish DBMoto Authentication, setting up user IDs and passwords specific to DBMoto. However, DBMoto authentication also requires the [installation of a X.509 security certificate](#) on the system where the Server Agent is installed. Choose TCP/IP as the protocol in this scenario.

## DBMoto Client and Server Agent in an Internet Environment

The Server Agent and the DBMoto client application, a custom application using the DBMoto API, for example, run on different servers over the internet. In this case, communication occurs over a firewall and messages may be redirected over channels which are not securable. The recommended authentication modes are DBMoto or Certificate Authentication, using the HTTP protocol to cross the firewall.

DBMoto authentication requires the [installation of a X.509 security certificate](#) in the server environment.

Certificate Authentication requires the [installation of a X.509 security certificate](#) in the client and server environments.

## Installing a DBMoto Certificate

X.509 certificates are required for certain authentication options when setting up user access to DBMoto via the [User Settings dialog](#). When choosing either the DBMoto Authentication or the Certificate Authentication options, you need to generate and install a security certificate. Typically, certificates are obtained from a trusted external commercial certificate authority. Identify an X.509 certificate provider through your company's IT department or using a web search.

[Installing a Certificate for DBMoto Authentication](#)

[Installing Certificates for Certificate Authentication](#)

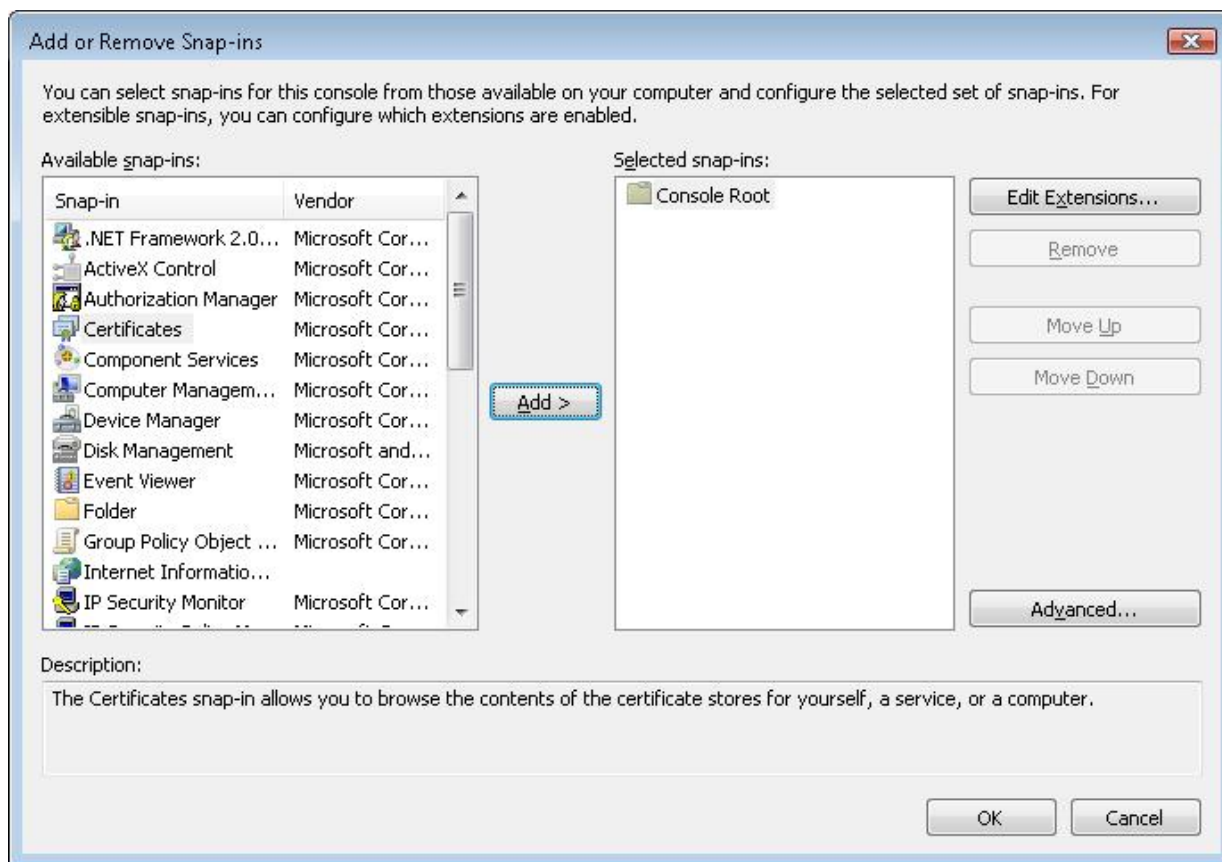
**NOTE:** If you set up a user ID with DBMoto Authentication, or Certificate Authentication, you will not be able to use DBMoto until the certificate is correctly installed. Generate and install the certificate before setting up authentication.

## Certificate Requirements

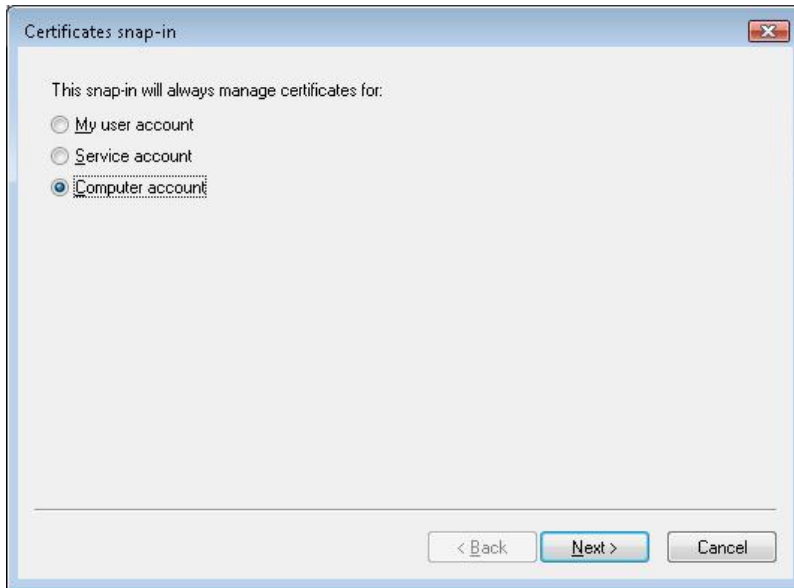
The certificate must come from a trusted certificate authority and be compatible with X.509 standards. The certificate name must be DBMoto.

## Installing a Certificate for DBMoto Authentication

1. On the system where the DBMoto Server Agent is running, open a Microsoft Management Console (MMC) from the Windows Command tool by typing `mmc.exe` at the command line.
2. In the MMC console, from the **File** menu, select **Add/Remove Snap In**.
3. In the **Add or Remove Snap-ins** dialog, select **Certificates** in the left column.

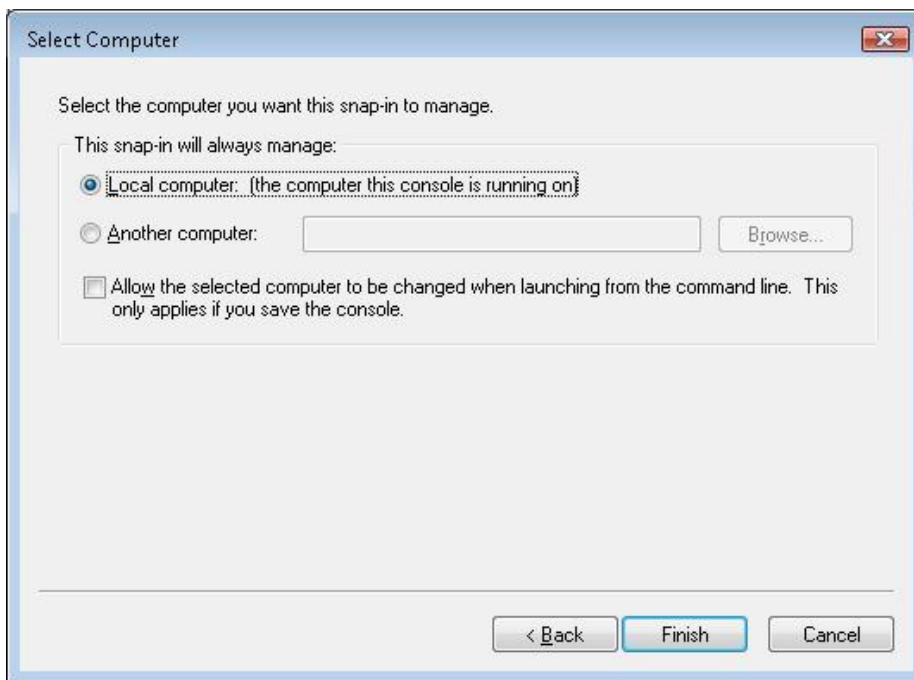


4. Click **Add**.
5. In the **Certificates Snap-in** step, select **Computer account**.



6. Click **Next**.

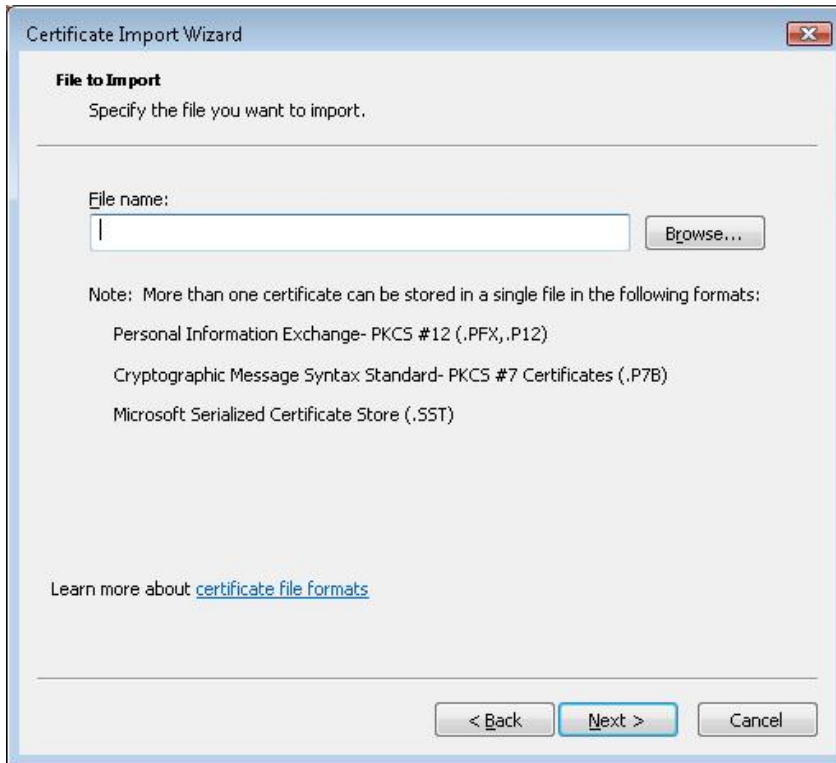
7. In the **Select Computer** step, select **Local Computer**.



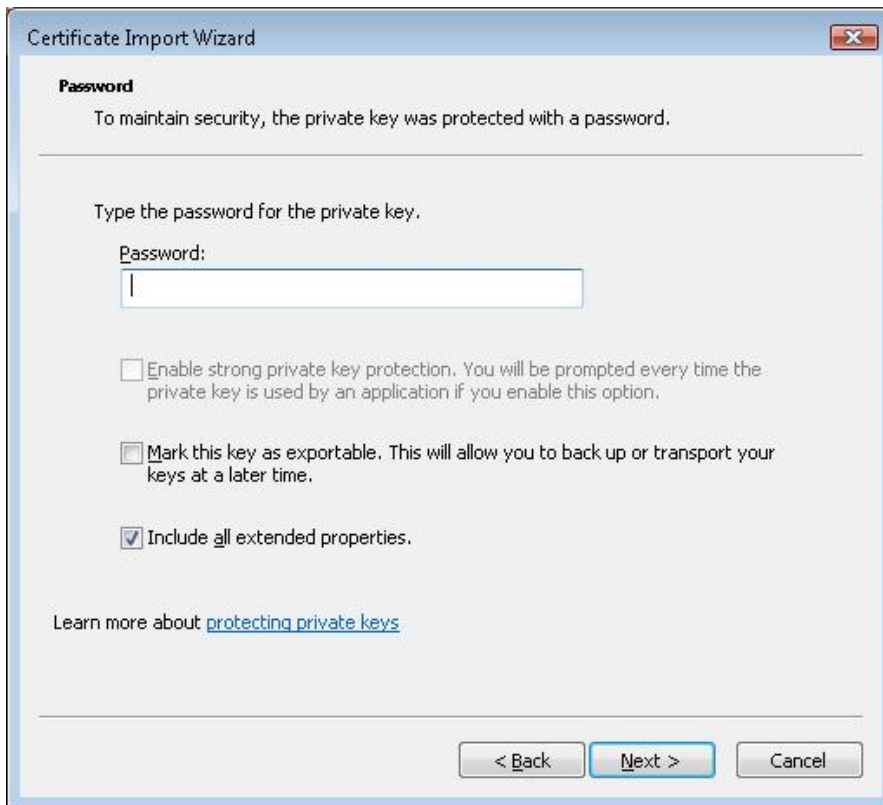
8. Click **Finish**.

9. Click **OK** in the **Add or Remove Snap-ins** dialog.

10. In the MMC Console, expand the **Certificates** node under **Console Root**.
11. Expand the **Trusted People** node.
12. Select **Certificates** and, from the right mouse button menu, choose **All Tasks** then **Import**.
13. In the **Certificate Import Wizard**, select the X.509 certificate file you generated for DBMoto.



14. Click **Next**.
15. Enter the password for the certificate.

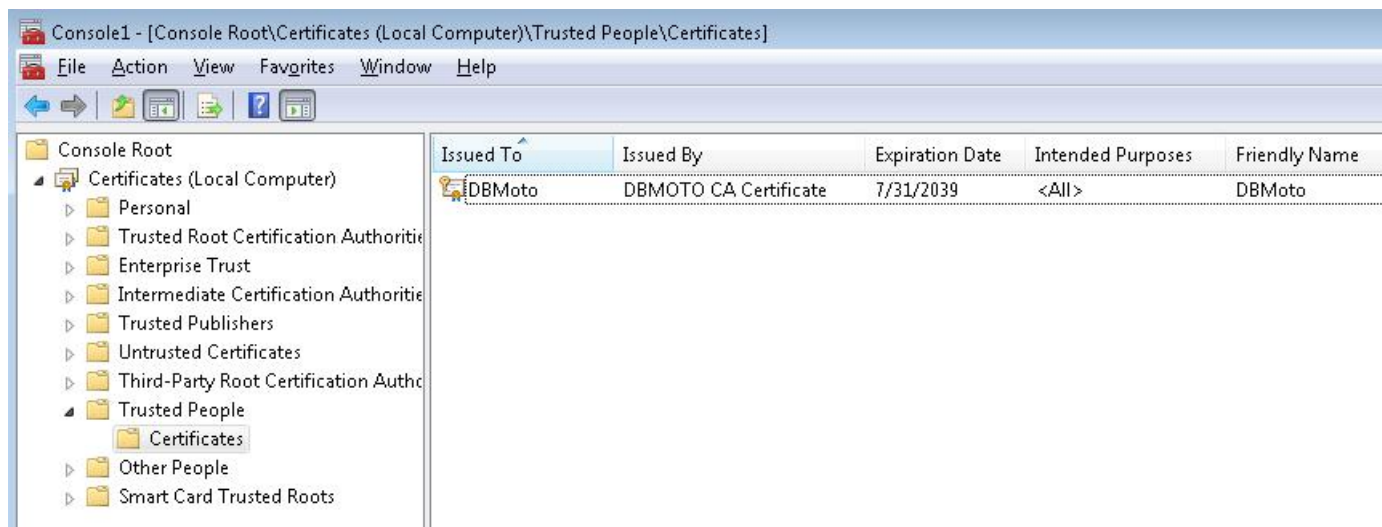


16. Click **Next**.

17. As needed, select the certificate store location.

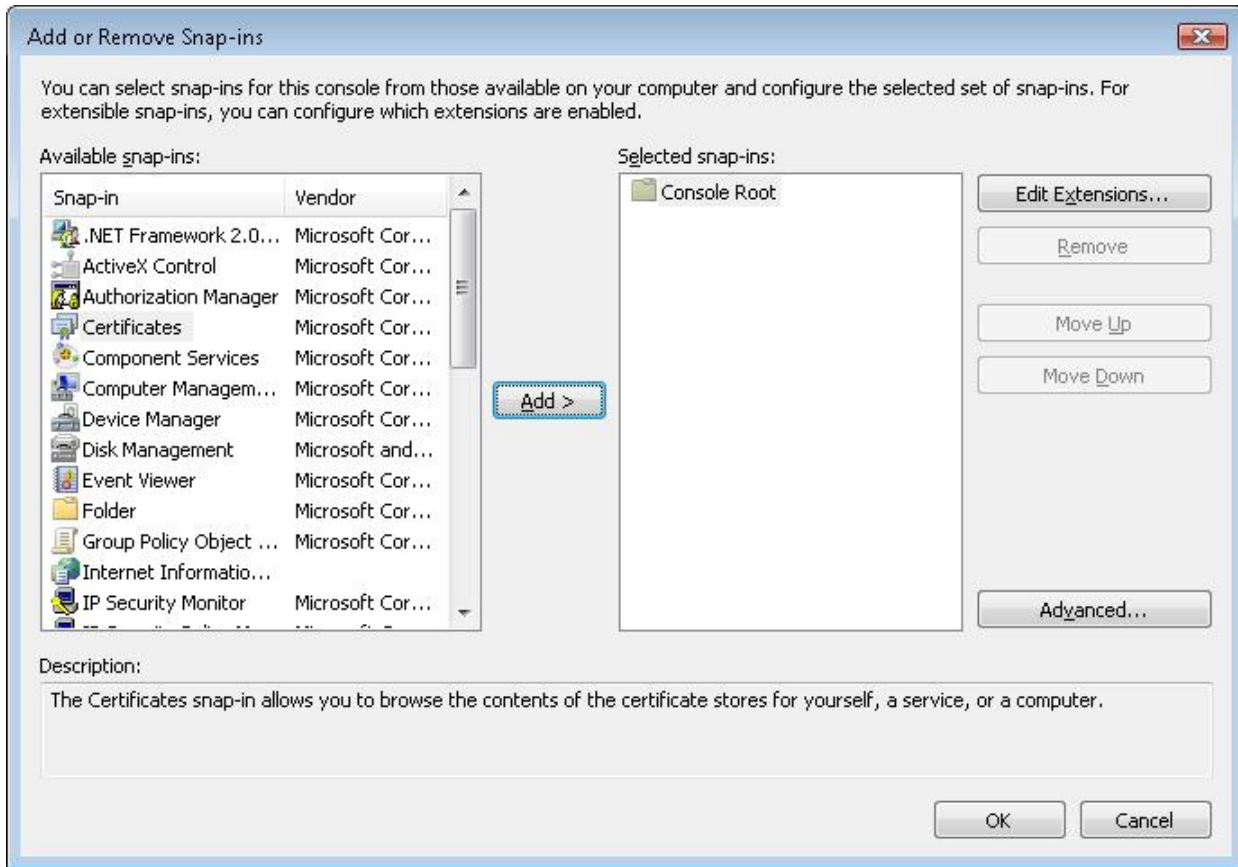
18. Click **Finish** to complete the wizard.

19. To view the certificates installed, expand the **Certificates** node in the MMC and expand the **Trusted People** node.



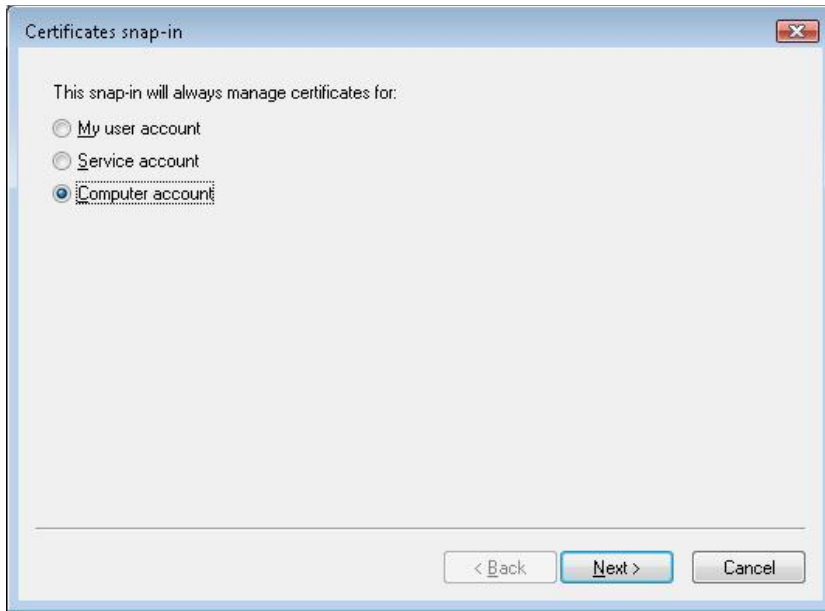
### Installing Certificates for Certificate Authentication

1. On the system where the DBMoto Server Agent is running, open a Microsoft Management Console (MMC) from the Windows Command tool by typing `mmc.exe` at the command line.
2. In the MMC console, from the **File** menu, select **Add/Remove Snap In**.
3. In the Add or Remove Snap-ins dialog, select **Certificates** in the left column.



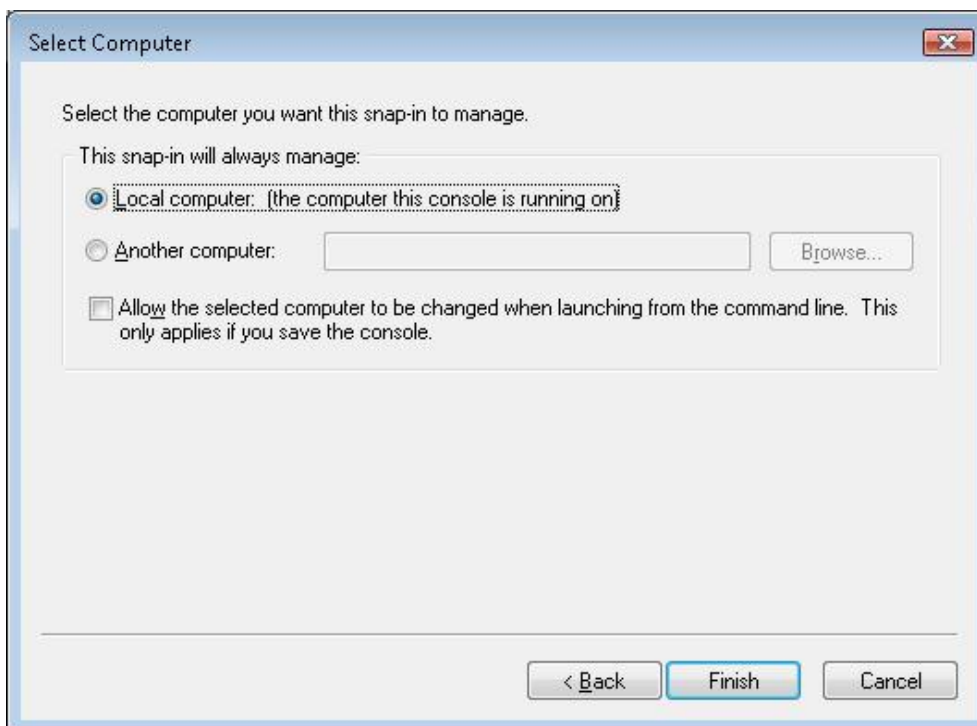
4. Click **Add**.
5. In the **Certificates Snap-in** step, select **Computer account**.



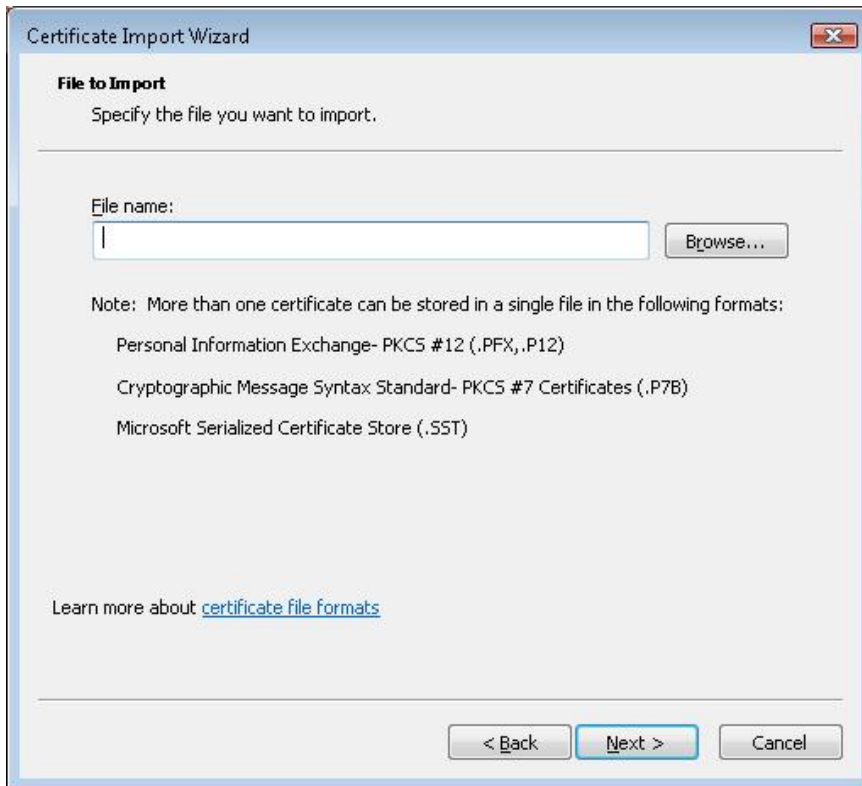


6. Click **Next**.

7. In the **Select Computer** step, select **Local Computer**.



8. Click **Finish**.
9. Click **OK** to close the Add or Remove Snap-ins dialog.
10. In the MMC Console, expand the **Certificates** node under **Console Root**.
11. Expand the **Trusted People** node.
12. Select **Certificates** and, from the right mouse button menu, choose **All Tasks** then **Import**.
13. In the Certificate Import Wizard, select the X.509 certificate file you generated for DBMoto.



14. Click **Next**.
15. Enter the password for the certificate.

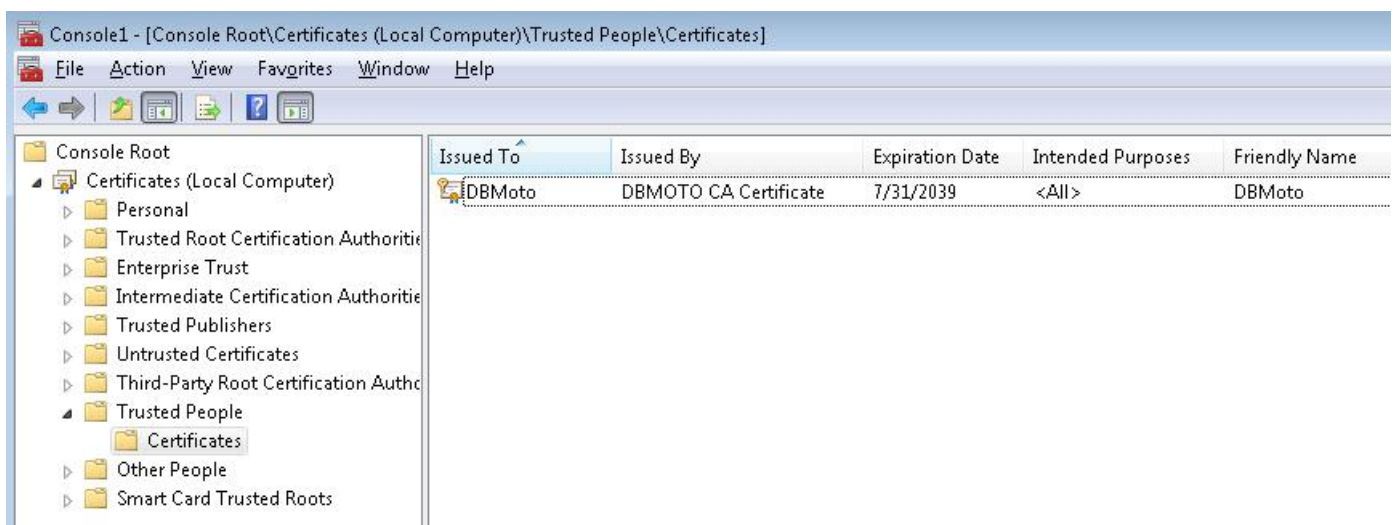


16. Click **Next**.

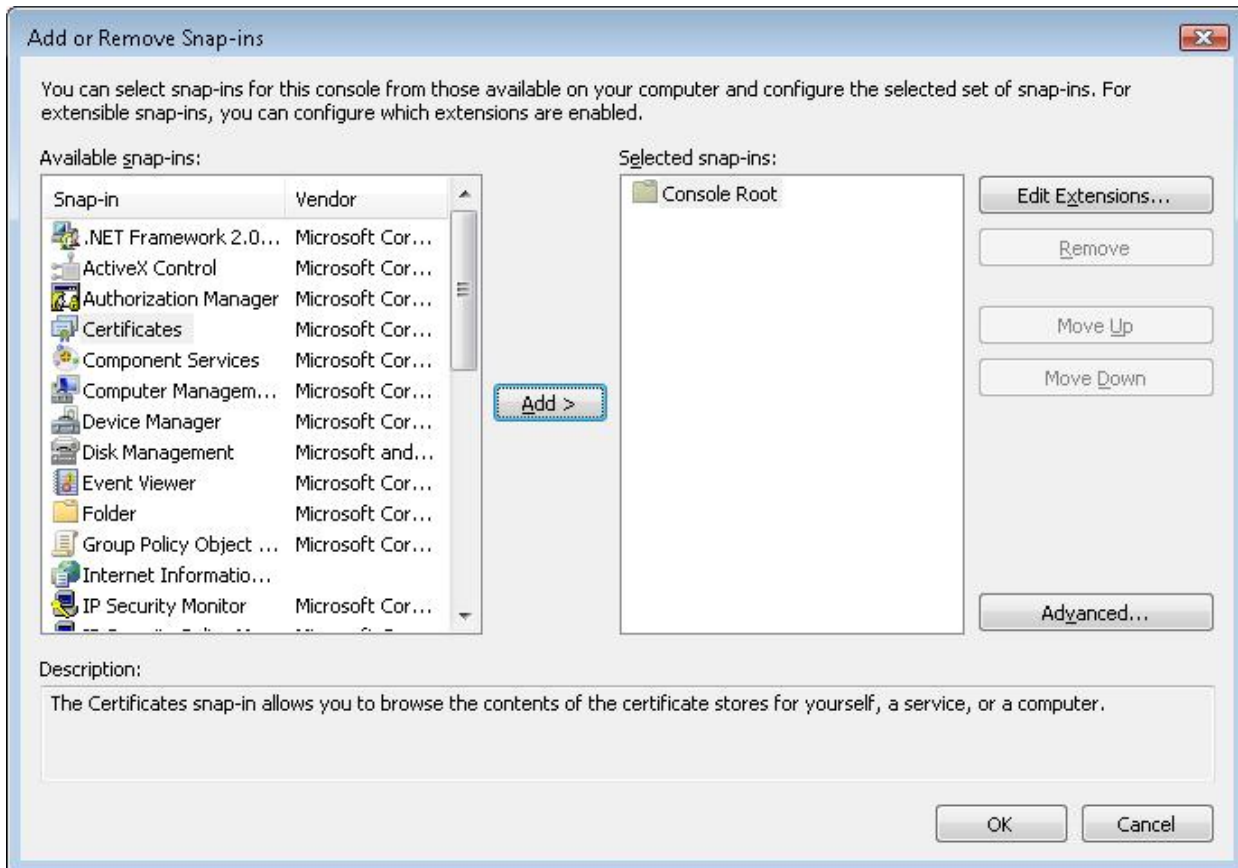
17. As needed, select the certificate store location.

18. Click **Finish** to complete the wizard.

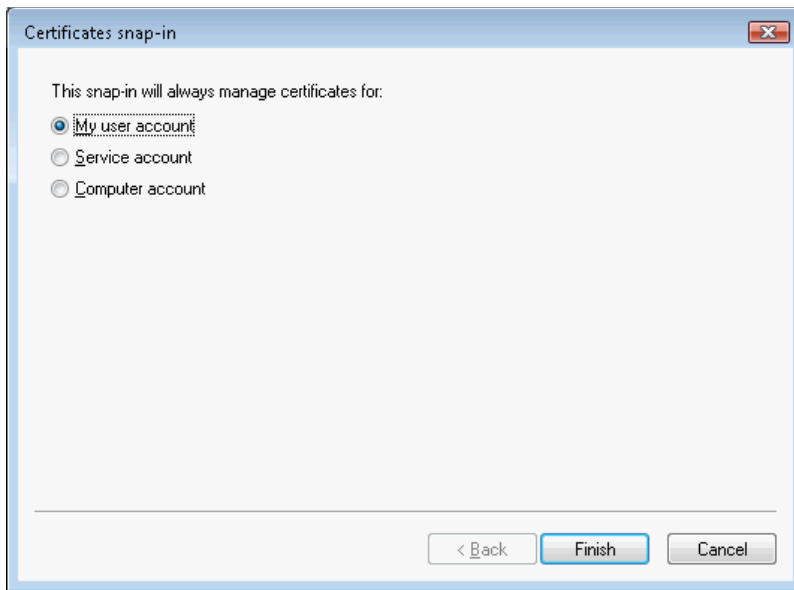
19. To view the installed certificate, expand the **Certificates** node in the MMC and expand the **Trusted People** node.



- 20. On the system where the DBMoto Client (Management Center or custom application), open a Microsoft Management Console (MMC) from the Windows Command tool by typing `mmc.exe` at the command line.
- 21. In the MMC console, from the **File** menu, select **Add/Remove Snap In**.
- 22. In the Add or Remove Snap-ins dialog, select **Certificates** in the left column.



- 23. Click **Add**.
- 24. In the Certificates Snap-in step, select **My user account**.



25. Click **Finish**.

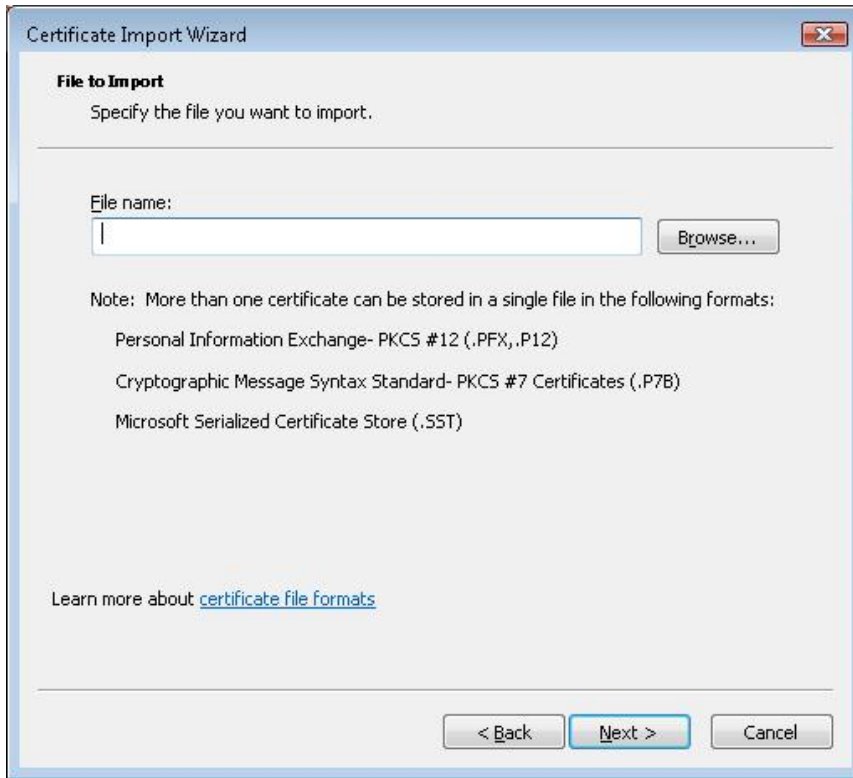
26. Click **OK** to close the Add or Remove Snap-ins dialog.

27. In the MMC Console, expand the **Certificates** node under **Console Root**.

28. Expand the **Trusted People** node.

29. Select **Certificates** and, from the right mouse button menu, choose **All Tasks** then **Import**.

30. In the Certificate Import Wizard, select the X.509 certificate file you generated for DBMoto.



31. Click **Next**.

32. Enter the password for the certificate.

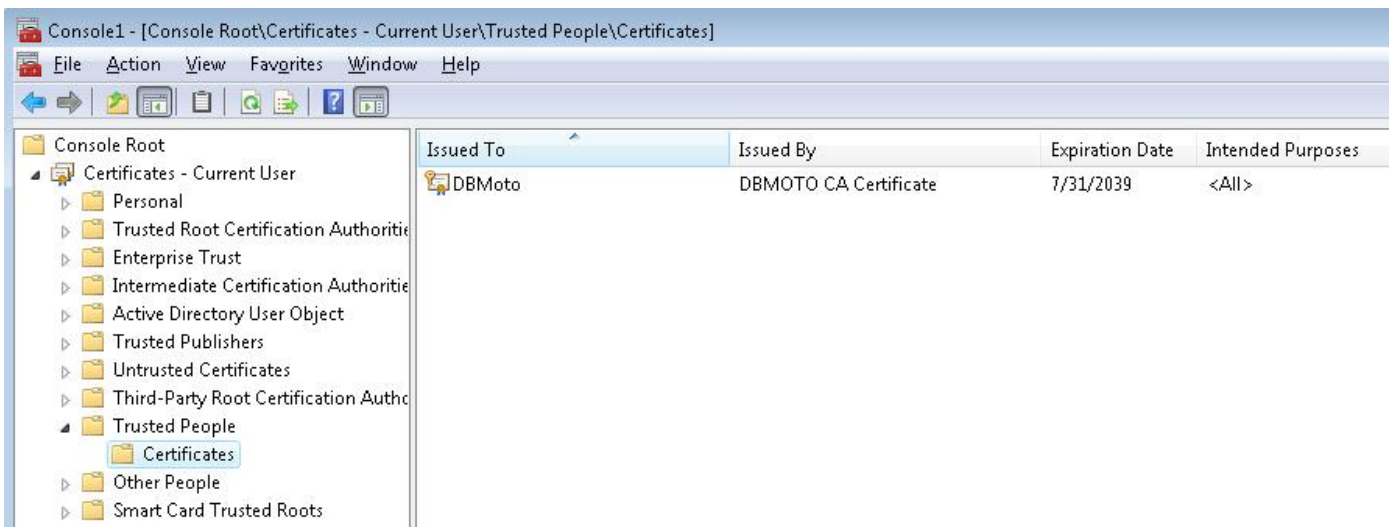


33. Click Next.

34. As needed, select the certificate store location.

35. Click Finish to complete the wizard.

36. To view the installed certificate, expand the Certificates node in the MMC and expand the Trusted People node.



## Source and Target Database Setup Guidelines

Before attempting to run a replication, check that your source and target databases are set up to work with DBMoto. In most cases, you will need to make no changes or very minor changes in your source database setup, such as introducing a login specific to DBMoto users or setting up a specific file on the database server. However, if you skip the recommended steps, your replication may not work as expected.

### Source Database Guidelines

NOTE: If you are setting up synchronization between two or more databases, all databases are effectively "source" databases. Look for instructions on all databases in this section. You do not need to go to the section on Target Database Guidelines.

RDBMS	RDBMS Setup Summary	Links to Details
Gupta SQLBase	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>Create or assign tables for Master Table and Log Table (DBMS tools)</li> <li>Assign a tablespace for the Master table and Log Tables (DBMS tools)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">Trigger-based Transactional Replication Settings</a></p> <p><a href="#">Trigger-based Transactional Replications</a></p>
IBM Db2 for i (iSeries/AS400)	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>Set user ID authorities appropriately (DBMS tools)</li> <li>Make sure tables are journaled and receivers are set up appropriately (DBMS tools)</li> <li>Create a library on Db2 (DBMoto Management Center)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">IBM Db2 for i Connection Recommendations</a></p> <p><a href="#">IBM Db2 for i Transactional Replication Settings</a></p> <p>Log Server Agent Settings</p>
IBM Db2 LUW (via transaction log)	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>DBADM or SYSADM Login ID.</li> <li>Db2 Log settings as follows: LOGRETAIN: Recovery LOGARCHMETH1:LOGRETAIN or</li> </ul>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">IBM Db2 Connection Requirements</a></p> <p><a href="#">IBM Db2 LUW Transactional Replication Settings</a></p>



	<p>DISK</p> <ul style="list-style-type: none"> <li>• Copy file from &lt;DBMotoInstallDir&gt;/ServerFiles/DB2U DB to &lt;Db2 Home Directory&gt;/function on Db2 server.</li> <li>• Rename copied file to db2udbreadlog02.dll (db2udbreadlog02 for LINUX/AIX)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	
<p>IBM Db2 LUW and z/OS (via triggers)</p>	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>• Use Db2 UDB v. 7.2 or higher, or Db2 for OS390 v6 or higher</li> <li>• Obtain Package creation privileges for the first connection (DBMS tools)</li> <li>• Set security mechanism in DBMoto (DBMoto Management Center)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">IBM Db2 Connection Requirements</a></p> <p><a href="#">IBM Db2 LUW Transactional Replication Settings</a></p> <p><a href="#">Trigger-based Transactional Replication Settings</a></p> <p><a href="#">Trigger-based Transactional Replications</a></p>
<p>IBM Informix (via transaction log)</p>	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>• Use Informix v 11.5 FC3 or above</li> <li>• Execute the script <b>syscdcv1.sql</b> on the server</li> <li>• Database should have logging enabled</li> <li>• DBMoto user should have CONNECT and DBA privileges to source database and syscdcv1 database</li> <li>• Install and test the latest Informix Client SDK (currently version 3.70) with the ESQL option and the .NET Data Provider</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">Informix Server and Client Setup</a></p> <p><a href="#">Informix Transactional Replication Settings</a></p>

<p>IBM Informix (via triggers)</p>	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>• Create or assign tables for Master Table and Log Table (DBMS tools)</li> <li>• Assign a tablespace for the Master table and Log Tables (DBMS tools)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">Informix Transactional Replication Settings</a></p> <p><a href="#">Trigger-based Transactional Replications</a></p> <p><a href="#">Trigger-based Transactional Replication Settings</a></p>
<p>IBM PureData (Netezza)</p>	<p><b>Mirroring Source:</b></p> <ul style="list-style-type: none"> <li>• Set user ID authorities appropriately (DBMS tools). CREATE TABLE authority needed for setup</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">IBM PureData Transactional Replication Settings</a></p>
<p>Microsoft SQL Server (via transaction log)</p>	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>• Define two connections to the database: a non system administrator connection and a system administrator connection. (DBMS tools)</li> <li>• Set up the distributor (DBMoto Management Center)</li> <li>• Establish primary keys on tables (or use trigger-based replication option.) (DBMoto Management Center or DBMS tools)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">SQL Server Transactional Replication Settings</a></p>
<p>Microsoft SQL Server (via Log Server Agent)</p>	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>• Define two connections to the database: a non system administrator connection and a system administrator connection. (DBMS tools)</li> <li>• Set up the distributor (DBMoto Management Center)</li> </ul>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">SQL Server Transactional Replication Settings</a></p> <p>Log Server Agent Settings</p>

	<ul style="list-style-type: none"> <li>Establish primary keys on tables (or use trigger-based replication option.) (DBMoto Management Center or DBMS tools)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	
Microsoft SQL Server (via triggers)	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>Create or assign tables for Master Table and Log Table (DBMS tools)</li> <li>Assign a tablespace for the Master table and Log Tables (DBMS tools)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">SQL Server Transactional Replication Settings</a></p> <p><a href="#">Trigger-based Transactional Replications</a></p> <p><a href="#">Trigger-based Transactional Replication Settings</a></p>
Microsoft Azure SQL Database (via triggers)	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>Create or assign tables for Master Table and Log Table (DBMS tools)</li> <li>Assign a tablespace for the Master table and Log Tables (DBMS tools)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">Trigger-based Transactional Replications</a></p> <p><a href="#">Trigger-based Transactional Replication Settings</a></p>
MySQL (via transaction log)	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>Modify the MySQL Server Instance Configuration file, my.ini (Desktop editor)</li> <li>Grant REPLICATION SLAVE privilege to the MySQL user ID that will be used by DBMoto (DBMS tools)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">MySQL Server Setup</a></p> <p><a href="#">MySQL Transactional Replication Settings</a></p>
MySQL (via Log Server Agent)	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>Configure DBMoto Log Server Agent (Windows Service) settings in the Connection wizard, providing file locations for log details.</li> </ul>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">MySQL Server Setup</a></p> <p><a href="#">MySQL Transactional Replication Settings</a></p>

	<p><b>Refresh:</b> No changes needed</p>	Log Server Agent Settings
MySQL (via triggers)	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>• Create or assign tables for Master Table and Log Table (DBMS tools)</li> <li>• Assign a tablespace for the Master table and Log Tables (DBMS tools)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">Trigger-based Transactional Replication Settings</a></p> <p><a href="#">Trigger-based Transactional Replications</a></p>
Oracle (via transaction log)	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>• If database is set to ARCHIVELOG mode, check Read Archived Logs options when configuring the source connection (DBMoto Management Center).</li> <li>• For Oracle versions 8.1.7 up to 9i, set dictionary file (DBMS tools then DBMoto Management Center)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">Setup for Different Oracle Versions</a></p> <p><a href="#">Oracle Transactional Replication Settings</a></p>
Oracle (via Log Server Agent)	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>• Configure DBMoto Log Server Agent (Windows Service) settings in the Connection wizard, providing file locations for log details.</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">Setup for Different Oracle Versions</a></p> <p><a href="#">Oracle Transactional Replication Settings</a></p>
PostgreSQL (via Log Server Agent)	<p><b>Mirroring Source</b></p> <ul style="list-style-type: none"> <li>• Edit the postgresql.conf file to set wal_level = logical max_replication_slots = 3 track_commit_timestamp = on</li> <li>• Configure DBMoto Log Server Agent (Windows Service) settings in the</li> </ul>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">PostgreSQL Transactional Replication Settings</a></p> <p>PostgreSQL Database Setup</p> <p>Log Server Agent Settings</p>

	<p>Connection wizard, providing file locations for log details.</p> <p><b>Refresh:</b> No changes needed</p>	
SAP HANA	<p><b>Mirroring Source</b></p> <ul style="list-style-type: none"> <li>• Create or assign tables for Master Table and Log Table (DBMS tools)</li> <li>• Assign a tablespace for the Master table and Log Tables (DBMS tools)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">Trigger-based Transactional Replication Settings</a></p> <p><a href="#">Trigger-based Transactional Replications</a></p>
SAP Sybase SQL Anywhere	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>• Create or assign tables for Master Table and Log Table (DBMS tools)</li> <li>• Assign a tablespace for the Master table and Log Tables (DBMS tools)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">Trigger-based Transactional Replication Settings</a></p> <p><a href="#">Trigger-based Transactional Replications</a></p>
SAP Sybase ASE	<p><b>Mirroring Source/Synchronization:</b></p> <ul style="list-style-type: none"> <li>• Create or assign tables for Master Table and Log Table (DBMS tools)</li> <li>• Assign a tablespace for the Master table and Log Tables (DBMS tools)</li> </ul> <p><b>Refresh:</b> No changes needed</p>	<p><a href="#">Help Center Database Access Provider List (.NET and ODBC)</a></p> <p><a href="#">Trigger-based Transactional Replication Settings</a></p> <p><a href="#">Trigger-based Transactional Replications</a></p>

## Target Database Guidelines

NOTE: If you are setting up synchronization between two databases, both databases are effectively "source" databases. Look for instructions on both databases in the Source Database Guidelines section.

For mirroring and refresh replications, no special setup steps are needed for target databases (including IBM PureData, Teradata, Vectorwise and HP Vertica), other than checking that you are using a [Supported Data Providers/Version for DBMoto](#) (.NET, OLE DB and ODBC.) For replication to Hadoop, [the DBMoto target is a file](#).

### SAP HANA and SAP Sybase IQ Targets

For mirroring and refresh replications to SAP HANA and SAP Sybase IQ targets:

- Set the **FTP Server, FTP Port, FTP User, FTP Password and Import Path** properties in the [Connection Properties dialog](#)
- For an SAP Sybase IQ target, enable the **allow\_read\_client\_file** and **allow\_write\_client\_file** options of the Sybase IQ 15.0 server. See [http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc00608.0480/html/etl\\_ug/BABEIJGI.htm](http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc00608.0480/html/etl_ug/BABEIJGI.htm)

## Choosing a Log Type for Transactional Replications

The [Enable Transactional Replication wizard](#) allows you to configure access to data changes on a source database (for mirroring and synchronization) or a target database (for synchronization). Depending on the source database, available options for identifying data changes are:

### Log Reader

Reads the native database transaction log using a DBMoto reader thread for each replication. This option is the default choice and works well if you have a limited number of replications and do not want to install additional services on the system that is running the DBMoto Server Agent component.

### Log Server Agent

Creates a Windows service on the system running the DBMoto Server Agent that makes only a single connection to the database to read the binary log for all replications in a source connection. The service autonomously reads the native database transaction log and writes all changes from the tables being replicated into intermediate binary log files (a DBMoto proprietary format) in a specified folder. The DBMoto Replicator reads the changes from the intermediate binary log files.

Choose this option if you plan to add many replications and you need a scalable solution that optimizes access to the native database transaction log.

Below is a summary of the advantages and disadvantages of replicating the Log Reader transactional replication mode versus using the Log Server Agent transactional replication mode:

	Log Reader	Log Server Agent
<b>Advantages</b>	<p><b>Simple to configure</b></p> <p><b>Standalone configuration</b></p> <p>Each replication connects directly to the source</p>	<p><b>Performance</b></p> <p>The Log Server only uses ONE instance of the native database log reader to query for data changes – regardless of the number of</p>

	<p>database and does not rely on other components – which at times may be stopped or in error or in any other state.</p>	<p>replications and groups. The Log Server queries the database for all changes in a data source at the same time and then stores them in intermediate log files. The replications and groups then read changes from the intermediate log files (very efficient.)</p> <p><b>Less physical log file retention required</b></p> <p>Because the Log Server runs continuously in the background, it can capture data changes more rapidly, and it stays running independently even when DBMoto goes down. It requires the database to retain physical log files for shorter periods.</p>
<p><b>Disadvantages</b></p>	<ol style="list-style-type: none"> <li>1. Query performance decreases as the number of replications or groups increases because each replication or group requires DBMoto to create a reader thread to query the database for changes. For example, if a data source has 2 single replications and 10 group replications, DBMOTO would need to create 12 threads to query the database for data changes.</li> <li>2. As performance decreases, the database has to retain physical log files longer until all changes are returned to DBMOTO.</li> <li>3. If DBMOTO goes off line, the database also has to retain physical log files until DBMOTO restarts and queries all the changes.</li> </ol>	<ol style="list-style-type: none"> <li>1. Additional file management: an extra application, the Log Server Folder, and Log Files Folder. However, the application management tasks are not complex.</li> </ol>

**Triggers**

Creates a trigger on the database for each replication to log data from committed transactions. This option is useful when neither the Log Reader or the Log Server Agent meet the needs of your environment. For example, use triggers when:

- There are no primary keys and the source is Microsoft SQL Server.
- All column data (before and after image) is needed. Some standard log readers only return record changed values

- Performance is better because only relevant information is logged and DBMoto does not read the entire database log
- The source database is MySQL versions prior to 5.1.5 or you are performing MySQL synchronization replications


### Log Reader API

This option is available only when replicating from IBM Db2 for i, and is specifically designed for replications that include Large Object Binaries.

## Creating Database Connections

With DBMoto, you can use a .NET data provider, an OLE DB Provider or an ODBC driver to connect to the databases used in your replication. Check [here](#) for a list of supported databases and drivers/providers.

You will need connections to the following databases:

- Metadata database (Default connection to Microsoft SQL Server CE provided)  
The database you use to store tables containing information and settings for the replication process. DBMoto provides a default database connection to SQL Server CE (included with DBMoto) so you do not need to create a connection for your metadata. However, you can choose to store metadata in a different database by creating a metadata connection to the database. To use a different database for your metadata, in the Metadata Explorer, choose  and complete the [Metadata Connection wizard](#).
- Source database  
The database that contains the tables you want to replicate.
- Target database  
The database that will contain the replicated tables.

Follow the steps below to set up DBMoto for replication.

1. For each database you are planning to use in your replication project:
  - Install and configure one of the supported data access products. See [http://www.hitsw.com/support/kbase/DBMoto/Providers\\_DBMoto.htm](http://www.hitsw.com/support/kbase/DBMoto/Providers_DBMoto.htm) for a current list of supported providers.
  - From the data access product, test the connection to the database.
  - Create a connection string for the data access product/database you are using. Check the documentation for the data access product for information on how to do this.
2. In the DBMoto Management Center Metadata Explorer, expand the node for your metadata to view the **Sources** and **Targets** nodes.
3. Review the Source and Target Database Setup Guidelines for your source database to learn all the steps involved in setting up the connection correctly.
4. Select the Sources node.
5. From the right mouse button menu, choose Add New Connection.
6. In the Source Connection Wizard, follow steps to add a connection string and test the connection to the database.




7. Choose the tables that you plan to replicate.
8. Complete the wizard.
9. Select the Targets node.
10. From the right mouse button menu, choose Add New Connection.
11. In the Target Connection Wizard, follow steps to add a connection string and test the connection to the database.
12. Choose the tables to which you plan to replicate.
13. Complete the wizard.
14. You are now ready to create target tables or create a replication.

## Copying Database Connections

Once you have created a source or target connection in the DBMoto Management Center, you can copy that connection, then edit its properties as needed. This option can save you time when you have multiple similar connections to create.

To copy a database connection:

1. Expand the Metadata Explorer to display source and/or target connections.
2. Select a connection in the list (connections are displayed with a  icon.)
3. From the right mouse button menu, choose Copy Connection.
4. Select the location where you want to paste the new connection (either Sources or Targets.)
5. From the right mouse button menu, choose Paste Connection.
6. Modify the connection name by choosing Rename Connection from the right mouse button menu.
7. Add tables to the connection by choosing Select Tables from the right mouse button menu.
8. Modify other properties of the connection by choosing Connection Properties from the right mouse button menu.

## Database-Specific Settings

### IBM Db2 for i (AS400/iSeries)

#### Using 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 Ritmo page on our 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.

See [IBM Db2 for i Connection Recommendations](#) for information on the type of permissions you need to set up a connection to IBM Db2 for i.

## 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 text editor.

## Creating 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 DBMoto 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 in the Management Center.

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. V5R4 and above. For use with Log Reader transaction mode. Includes stored procedure JRNSQMAPI to allow up to 300 replications from the same data source.	DBMLIB54.SAVF
V6R1 and above. For use with Log Reader API transaction mode. Includes stored procedure JRNSQMAPI to allow up to 300 replications from the same data source.	DBMLIBAPI61.SAVF

- Run the DOS command prompt and change the working directory to C:\DBMLib.

```
C:>cd C:\DBMLib
```

- Run an FTP session followed by the Db2 system IP address

```
C:\DBMLib> ftp 111.111.111.111
```

- Insert your username and password when prompted. Make sure that your user ID has write permissions and QSECOFR privileges.
- Make QGPL the current directory.

```
ftp> quote cwd QGPL
```

- Create an empty Savefile on the Db2 system.

```
ftp> quote rcmd CRTSAVF FILE(QGPL/DBMLIBSAVF) AUT(*ALL)
```

- 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 Version V4R2

#### 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 V5ROMO

#### 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**

Procedure JRNSQNMAPI required when planning to create more than 30 replications.

```

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 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

```

```

CREATE PROCEDURE DBMOTOLIB.JRNSQNMAPI

```

```

(IN KEYS VARCHAR(10000) FOR BIT DATA,
INOUT JOUR CHAR(20),
INOUT ENDSEQNUM CHAR(20),
INOUT TMSP CHAR(20),
INOUT SRN CHAR(10),
INOUT SRL CHAR(10),
INOUT ERRFLG CHAR(1),
INOUT MSG CHAR(500))
SPECIFIC DBMOTOLIB.JRNSQNMAPI
LANGUAGE C
EXTERNAL NAME 'DBMOTOLIB/JRNSQNMAPI'
PARAMETER STYLE GENERAL WITH NULLS

```

### Operating System Versions V6R1 and above

Required for transactional replication mode Log Reader API. Procedure JRNSQNMAPI required when planning to create more than 30 replications.

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

CREATE PROCEDURE DBMOTOLIB.JRNSQNMAPI

(IN KEYS VARCHAR(10000) FOR BIT DATA,
INOUT JOUR CHAR(20),
INOUT ENDSEQNUM CHAR(20),
INOUT TMSP CHAR(20),
INOUT SRN CHAR(10),
INOUT SRL CHAR(10),
INOUT ERRFLG CHAR(1),
INOUT MSG CHAR(500))
SPECIFIC DBMOTOLIB.JRNSQNMAPI
LANGUAGE C
EXTERNAL NAME 'DBMOTOLIB/JRNSQNMAPI'
PARAMETER STYLE GENERAL WITH NULLS

```

## IBM Db2 for i Connection Recommendations

When using DBMoto with IBM i/iSeries/AS400 systems, your user ID needs the authority to run the following commands:

- DSPOBJD on the schema used for replication and to access the following system tables:
  - QSYS2.SYSTABLES
  - QSYS2.SYSCOLUMNS
  - QSYS2.SYSCST
  - QSYS2.SYSKEYCST



Additionally, only if 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 mirroring and synchronization, the user ID should be used exclusively for DBMoto so that transactions executed by DBMoto can be easily identified in the journal.

## i/iSeries/AS400 System Journals and Receivers

If you are performing mirroring or synchronization with an 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.

**New for DBMoto 7:** It is now possible to create and use journals with minimized entries with DBMoto:

CRTJRN JRN(DBRSWRK/DBRSJ) JRNRCV(DBRSWRK/DBRSJ00001) AUT(\*ALL)MINENTDTA(\*FLDBDY)

Adding the MINENTDTA(\*FLDBDY) option to the CRTJRN command will decrease the size of journal entries as follows. The Minimized Entry Specific Data (MINENTDTA) parameter for an object type allows entries for that object type, in this case a database physical file, to be minimized. While \*FILE, \*DTAARA, and \*FLDBDY values are allowed the MINENTDTA parameter in the CRTJRN command, DBMoto supports only \*FLDBDY (field boundaries.) This means that the minimizing will occur on field boundaries. The entry specific data will be viewable and may be used for auditing purposes.

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

STRJRNP 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. Create 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 before, and therefore excludes a manual intervention: you are giving the system the appropriate instructions in order to manage receivers by itself.

In order to allow the receiver to be unbound but not deleted, you need to 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 (aka 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 continues 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.

## IBM Db2 UDB and z/OS

### Using the Ritmo/DB2 .NET Provider

The Ritmo/DB2 .NET provider is included with your DBMoto release and installed as part of the DBMoto setup. The version of Ritmo/DB2 that is provided with DBMoto does not include the Ritmo Toolbox. For more information about Ritmo features, check [the Ritmo page on our web site](#).

The Ritmo/DB2 files can be found in the DBMoto installation folder, in a separate folder called **Ritmo DB2**.

The following topic may be useful when using Ritmo/DB2 with DBMoto.

See [IBM Db2 Connection Requirements](#) for information on the type of permissions you need to set up a connection to IBM Db2.

### Enabling a trace

The file RitmoDb2.xml contains configuration settings for Ritmo and can be found in the folder where Ritmo was installed. You can set trace file names for high and/or low level traces and enable the traces from this file. Note that the traces 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\_DB2 folder.
2. Open the file RitmoDb2.xml in a text editor.
3. Modify the trace entries in bold below. For easy access to the tracefile, specify a pathname. The default path depends on your version of Windows. Note that the <traceflag> entry should be set to **true** to enable the trace.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<configSections>
<section name="trace" />
</configSections>
<trace>
<high>
  <tracefile>sqlldb2.trc</tracefile>
  <traceflag>true</traceflag>
</high>
</trace>
</configuration>
```

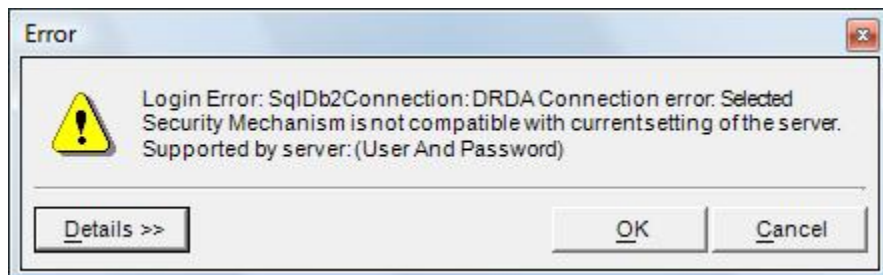
4. Save the file and exit the text editor.

### IBM Db2 Connection Requirements

When using DBMoto with IBM Db2, you need the following information to set up a connection.

## General Requirements

- User and password for the Db2 with appropriate privileges (see below)
- Security Mechanism used on the Db2 server. The value specified in DBMoto must match whatever setting is specified on the Db2 server. The options available are User Only, User and Password, Encrypted Password and Encrypted User and Password. In DBMoto version 6.5 and up, select the value from the drop-down list in the Security Mechanism field. If the value you choose does not match the value on the Db2 server, you will see the following error when attempting to connect.



**Note:** If you need to change this setting at any time, you may need to delete the connection and recreate it for the changed setting to take effect.

- IP address and port to reach Db2
- Name of Db2 catalog (Database Server Name)
- Package collection  
The account used to make a connection the first time requires permissions to create a package where some prepared SQL statements (DECLARE CURSOR) will be stored. The package can be created in a collection specified in the configuration parameter "r;Package Collection." The account must have CREATE, BIND and GRANT privileges for objects in the collection. These privileges are needed only for the first connection. Once the package has been created, for all users of that package the privileges are no longer required. (A package can later be dropped using the DROP PACKAGE SQL statement if needed.)
- Host Code Page (CCSID) for the Db2 catalog: the code of the language used in Db2 (ex. 280 Italian, 037 US English, 1252 ANSI etc.) In most cases, the appropriate choice is 1208 - Unicode UTF-8. However, you should check with your Db2 administrator.

## Requirements for Log-based Mirroring/Synchronization with Db2 LUW

- Either the DBADM or the SYSADM login ID.  
For DBMoto to access the Db2 log, the user ID specified in the DBMoto Connection dialog/wizard must have Database Administrator privileges.
- Db2 Log settings as follows. See [Setting Parameters for the Db2 Log](#) for more information.

LOGARCHMETH1 parameter should be set to LOGRETAIN or DISK

NOTE: Be sure to complete a full offline backup of the database after making any changes to the log settings.

- Copy file to Db2 Server.
  1. On your Windows desktop, open the DBMoto install folder, and go to the `ServerFiles/DB2UDB` folder.
  2. Locate the appropriate file for your operating system and Db2 version.

For example, the file `aix53_9010_db2udbreadlog03` is for the IBM AIX 5.3 operating system, Db2 version 9.1, whereas the file `win_10010_db2udbreadlog03.dll` is for Microsoft Windows, Db2 version 10.1.

NOTE: If you do not find a file for your operating system or Db2 version, [contact HiT Software Technical Support](#).

3. Copy the appropriate file from the `ServerFiles/DB2UDB` folder in your DBMoto install directory to:

`<DB2 Home Directory>/function`

on your Db2 server (the file contains stored procedures that DBMOTO calls to access the log).

4. Rename the copied file to `db2udbreadlog03.dll` (`db2udbreadlog03` for UNIX).

### Setting Parameters for the Db2 Log

By default, the IBM Db2 logging type is Circular, which means the log files are written in a circular fashion. To make the log files accessible for reading, Db2 requires the logging type to be Archive.

If the database is not already in Archive mode, you need to:

1. Change the logging type. To change the database logging type to Archive, change the parameter LOGARCHMETH1 value to either LOGRETAIN or DISK.
2. Perform a full offline backup (required by Db2 right after the logging type is changed.)

HiT Software highly recommends that the above actions are carried out by the Db2 Database Administrator.

### Setting Parameter LOGARCHMETH1 examples

Below are examples showing how to change the LOGARCHMETH1 parameter and to perform a backup. Again, it is highly recommend that the operations are carried out by the DBA. Refer to your database documentation for more information on the steps below.



- Set the value of LOGARCHMETH1 to LOGRETAIN.

```
CONNECT TO SAMPLE;  
UPDATE DATABASE CONFIGURATION USING LOGARCHMETH1 LOGRETAIN IMMEDIATE;  
CONNECT RESET;
```

- Set the value of LOGARCHMETH1 to DISK

```
CONNECT TO SAMPLE;  
UPDATE DATABASE CONFIGURATION USING LOGARCHMETH1 DISK:f:/logs/ IMMEDIATE;  
CONNECT RESET;
```

- Back up the database

Note that, depending on the size of your database, this step can take several hours.

```
BACKUP DATABASE SAMPLE TO "/tmp" WITH 2 BUFFERS BUFFER 1034 PARALLELISM 1  
COMPRESS WITHOUT PROMPTING;
```

For more information, see IBM Db2 Version 9.7's 'Configuration parameters summary' on the IBM website:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.admin.config.doc/doc/r0005181.html>.

## MySQL Server Setup

If you are planning to use MySQL version 5.1.5 or above as your source database for [mirroring](#) using a transaction log, you need to perform an additional installation step on the MySQL server: configure MySQL so that DBMOTO can read the database's binary log.

Before DBMOTO can read the binary log, row-based logging must be enabled for the database. To enable row-based logging, properties must be set in the MySQL Server Instance Configuration file.

1. Find the file "my.ini" which can be found in the directory where MySQL is installed.  
This file is the "MySQL Server Instance Configuration File".
2. Open the my.ini.
3. Scroll to the end of the file and add the following properties:

```
#For DBMoto replication  
log-bin=binlog  
binlog-format=ROW  
server-id=111
```



- `log-bin=binlog`  
In case users want to locate them, the binlog files are found in directory specified by "datadir" in "my.ini"
- `binlog-format=ROW`  
You can force the replication format by starting the MySQL server with `--binlog-format=type`. When set, all replication slaves connecting to the server will read the events according to this setting. ROW causes replication to be row-based.
- `server-id=<unique_value>`  
For each server participating in replication, you should pick a positive integer in the range from 1 to 232 – 1, and each ID must be different from every other ID in use by any other replication master or slave. For example, you could set the value as 123:  
`server-id=123`

4. Save the my.ini file.

5. Grant the following privileges to the MySQL user ID that will be used by DBMoto to connect to the database. For example, if the user ID that is used to replicate data is dbmoto:

```
GRANT SELECT, PROCESS, FILE, SUPER, REPLICATION CLIENT, REPLICATION SLAVE,  
RELOAD ON *.* TO dbmoto@'%';  
Flush Privileges;
```

6. Restart the server.

## PostgreSQL .NET Provider Setup for BulkInsert

If replicating with PostgreSQL, use the .NET provider [recommended by PostgreSQL](#). To use the bulkinsert feature for increased performance:

1. The driver DLL has to be installed using the [Microsoft Global Assembly Cache Tool](#) (GAC). If the provider is not registered, you will receive the following error when attempting to use the provider:

```
Error creating the command for writing to the target (Replication: 'EMP1' -  
Target table: 'public.test1')  
  
System.IO.FileNotFoundException: Could not load file or assembly 'Npgsql,  
Version=2.2.3.0, Culture=neutral, PublicKeyToken=5d8b90d52f46fda7' or one of  
its dependencies. The system cannot find the file specified.  
  
File name: 'Npgsql, Version=2.2.3.0, Culture=neutral,  
PublicKeyToken=5d8b90d52f46fda7'  
  
at ....
```

The commands for installing the PostgreSQL DLLs are as follows when running from the location where the GAC tool is installed:

```
gacutil.exe -i C:\Npgsql-2.2.3-net40\Npgsql.dll
```

```
gacutil.exe -i C:\Npgsql-2.2.3-net40\Mono.Security.dll
```

2. The provider version number stored in the DBReplicator.exe.config file (in the DBMoto install folder) should match the provider version that you are using. To change the version of the provider, specify the "newVersion" in the DBReplicator.exe.config file as follows:

```
<dependentAssembly>
  <assemblyIdentity name="Npgsql" publicKeyToken="5d8b90d52f46fda7"
culture="neutral" ></assemblyIdentity>
  <bindingRedirect oldVersion="2.2.3.0"
newVersion="2.2.3.0"></bindingRedirect>
</dependentAssembly>
```

## Setup for Different Oracle Versions

The Oracle client must be installed on the same system as DBMoto.

Oracle v 8.1.7 and higher can be used as a source for replication in refresh and mirroring modes, and a target for replication in synchronization. Replications are based on Oracle redo log files which require use of a dictionary file. The dictionary file, starting from Oracle 9.0, has become "online" and it is automatically created and managed by the DBMS, but for previous versions (8.1.7 up to 9.0), it must be created manually. Once created, you need to inform DBMoto about its name and location to allow the mirroring to take place.

Your DBMoto user account should be granted permissions to read the Oracle LogMiner (if using Oracle as a source), and write to the target tables (if using Oracle is a target.)

### Oracle 9.0 or Higher

When creating a [source connection](#) for Oracle (or a source or [target connection](#) if you plan to do synchronization), check that the **Use Online Dictionary** option is selected in the [Setup Info screen](#) of the Source Connection wizard.

### Oracle 8.1.7 to 9.0

1. Create the dictionary in the database. The user ID should have Database Administrator privileges.

Using PL/SQL, run the following command:

```
EXECUTE DBMS_LOGMNR_D.BUILD('<dictionary name>', '<dictionary path>')
```

2. When creating a [source connection](#) for Oracle (or a source or [target connection](#) if you plan to do synchronization), check that the **Use Flat File Dictionary** option is selected in the [Setup Info screen](#) of the Source Connection wizard.
3. In the **Dictionary File** field, type the name of the dictionary file that you created above.

## Informix Server and Client Setup

If you are planning to use Informix version 11.5 FC3 or above as your source database for [mirroring](#) or [synchronization](#) using a transaction log, you need to perform additional installation steps on the Informix server.

1. Check that you are using Informix IDS database version 11.5 FC3 or above
2. Ask a Database Administrator (DBA) to execute the script syscdcv1.sql that comes with the database to create the database syscdcv. The tables are required by DBMoto.
3. Make sure that The database containing the source data to be replicated was created with logging enabled. In the unlikely event that logging is not enabled, a DBA can use the ondblog utility to change the logging mode (together with the onbar utility to backup the database after the logging mode is changed.)
4. For the user ID that will be used to connect the database from DBMoto, grant CONNECT and DBA privileges both to the source database and to the syscdcv1 database. For example:

```
grant connect to "dbmouser1"  
grant dba to "dbmouser1"
```

5. Make sure that the user ID is in the informix-admin group.

The following steps should be performed on the client machine where DBMoto is installed and run.

1. Install the latest Informix Client SDK (currently version 3.70) with the ESQL option and the .NET Data Provider.
2. Use the SetNet32 utility to define the server.
3. Verify the connection to the database using the ILogin.exe demo.
4. Set up the permissions for the Windows user that runs DBMoto to create and delete Windows services.

## SAP Sybase ASE Setup for BulkInsert

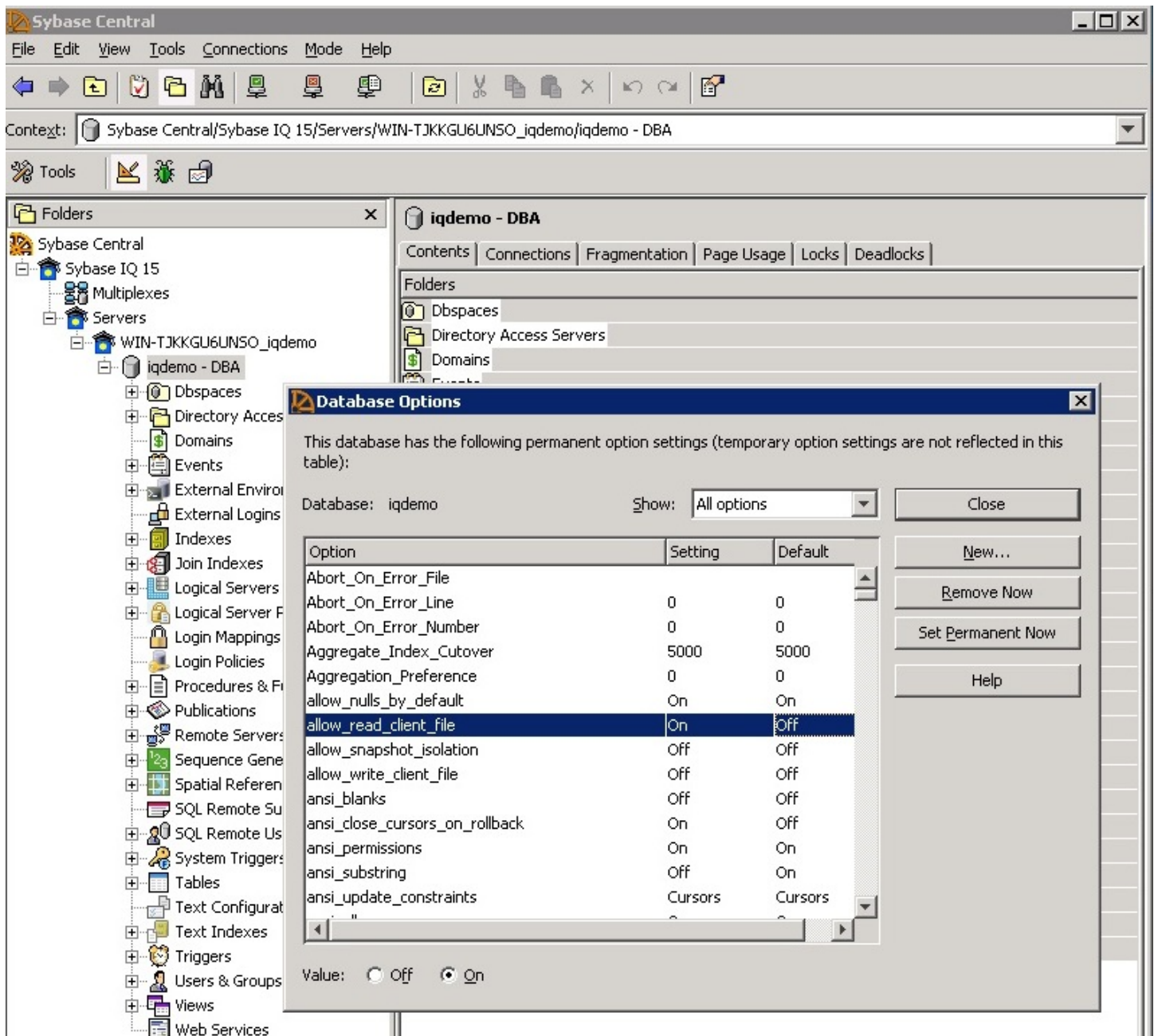
If replicating with SAP Sybase ASE using bulk insert, use the .NET provider Sybase.AdoNet4.AseClient.dll with version 16.0 and above of Sybase ASE. To use the bulkinsert feature for increased performance:

1. Bulk insert should be enabled on the Sybase ASE database. This is often not the default.
2. The connection string should contain an extra token: "EnableBulkLoad=2;"
3. The charset should be set as follows: "Charset=iso\_1;"

## SAP Sybase IQ Setup

For an SAP Sybase IQ target, enable the **allow\_read\_client\_file** and **allow\_write\_client\_file** options on the Sybase IQ server. Refer to the Sybase documentation for details on setting the options:

<http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc00608.0480/html/etl ug/BABEJIGI.htm>



Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.

## Replicating To and From a Flat File

When replicating from a relational database to a flat file, DBMoto allows you to replicate data in two different ways, or modes:

- **Refresh**

A one-time complete replication from source table to flat file, according to replication settings and scripts. You can control the timing of the replication, identify the columns to be replicated and add scripts to transform data during replication.

- **One-way mirroring**

A continuous update of a replicated table to a flat file based on changes to the source database that have been recorded in the database server log. You can define the replication settings to check the transaction log on the source database at regular intervals. Any changes found in the log would be applied to the target file.

When replicating from a flat file to a relational database, DBMoto allows you to replicate data in a single way, or mode:

- **One-way mirroring**

A continuous update of a database table based on data retrieved from a file.

Replicating to and from flat files is useful under certain circumstances in DBMoto, such as:

- Replicating relational data to a JSON format file.
- Replicating relational data to Hadoop.

## Chapter 3: Designing Replications

### Replication Types

DBMoto allows you to replicate data between database tables in the following ways:

- **Refresh**

A one time complete replication from source to target table, according to replication settings and scripts. You can control the timing of the replication, identify the columns to be replicated and add scripts to transform data during replication. The source and target databases can be on the same or different database servers and platforms. For example, you can replicate a Db2 table to an Oracle database or a SQL Server table to IBM Db2 for i (iSeries/AS400).

Note that when performing a refresh replication, DBMoto deletes all target records before inserting records from the source table. To avoid truncation of all records from the target table, you can [write a script](#) for the [Refresh\\_onBeforeTruncate](#) event.

[Steps for Replicating a Table Using Refresh Mode](#)

- **Continuous refresh**

A regularly scheduled refresh replication as described above. The schedule is defined in the replication settings.

[Steps for Replicating a Table Using Refresh Mode](#)

- **One-way mirroring**

A continuous update of a replicated table based on changes to the source database that have been recorded in the database server log. Typically, this involves an initial refresh operation, as described above, to set up the target table. Then you can define the replication settings to check the transaction log on the source database at regular intervals. Any changes found in the log would be applied to the target database.

[Steps for Replicating with One-Way Mirroring](#)

- **Synchronization or two-way mirroring**

Synchronization is appropriate when changes occur in both tables involved in the replication. Your replication settings ensure that both tables maintain the same state by checking the logs on each table and performing updates on each table as needed.

[Steps for Replicating with Synchronization](#)

### Planning the Replication Process

During the planning phase of a replication project, you should gather all the business requirements that DBMoto will need to fulfill, and precisely determine how the target tables will be derived from the source tables.

The DBMoto Administrator and the Database Administrator (whether it is one person who does both or two different persons) will need to ask the following questions before developing a plan for the replication process:

#### Business Requirements Questions

- Is DBMoto being used for system backup? Data sharing? Data propagation?
- Will the users need current data, or will they be able to work with data that is not current up to the second?
- What is an acceptable level of latency for the replicated data? (1 hour, 1 day, 1 week?)
- How often do replications need to be scheduled for?

- Which data should be blocked (from users, groups of users)?
- Will the users need to be able to update the replicated data, or will they only need to read the data?
- Will all the users and user groups need the same data, or will they need different subsets of the data?
- How will the users access the data? (OS, Hardware, Network)
- Are there periods when insert/delete/update activity on source tables is more frequent?

## DBMoto Configuration Questions

- How many tables will need to be replicated? How many databases? Use the [Database Replication Scenarios](#) and [Table Replication Scenarios](#) to help you determine your needs.
- What types of replication are needed? ([Refresh](#), [Mirroring](#), [Synchronization](#))
- If the users must be able to update the replicated data, is it a problem if someone else updates another copy of the data (same row), elsewhere, at the same time? If it is a problem, how can you prevent such a conflict from occurring, or deal with update conflicts if you cannot prevent them?
- If users need different subsets of rows, do you have a subsetting criteria present in all the tables? Is this criteria contained in a column that can be updated in place by an existing application? Will some users need different subsets of columns?
- How much data has to be replicated? What is the level of volatility (number of updates/inserts/deletes per hour or per day)?
- Are the existing tables normalized, and do you always follow the relational model recommendations?
- Will the headquarters of your organization need consolidated data from geographically dispersed data?
- Are there special filtering needs?

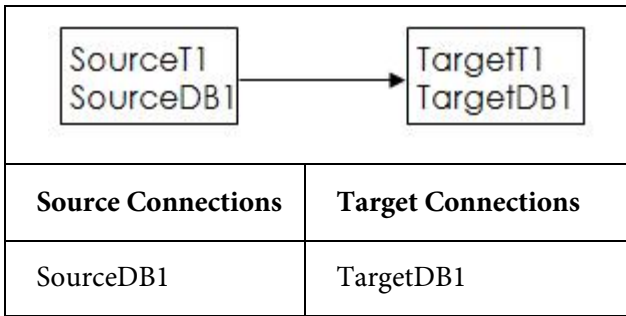
## Table Replication Scenarios

Replication in DBMoto is defined at the table level, so there is a replication definition for each source-target table pair. To replicate an entire database, see [Database Replication Scenarios](#).

### One to One Table Replication

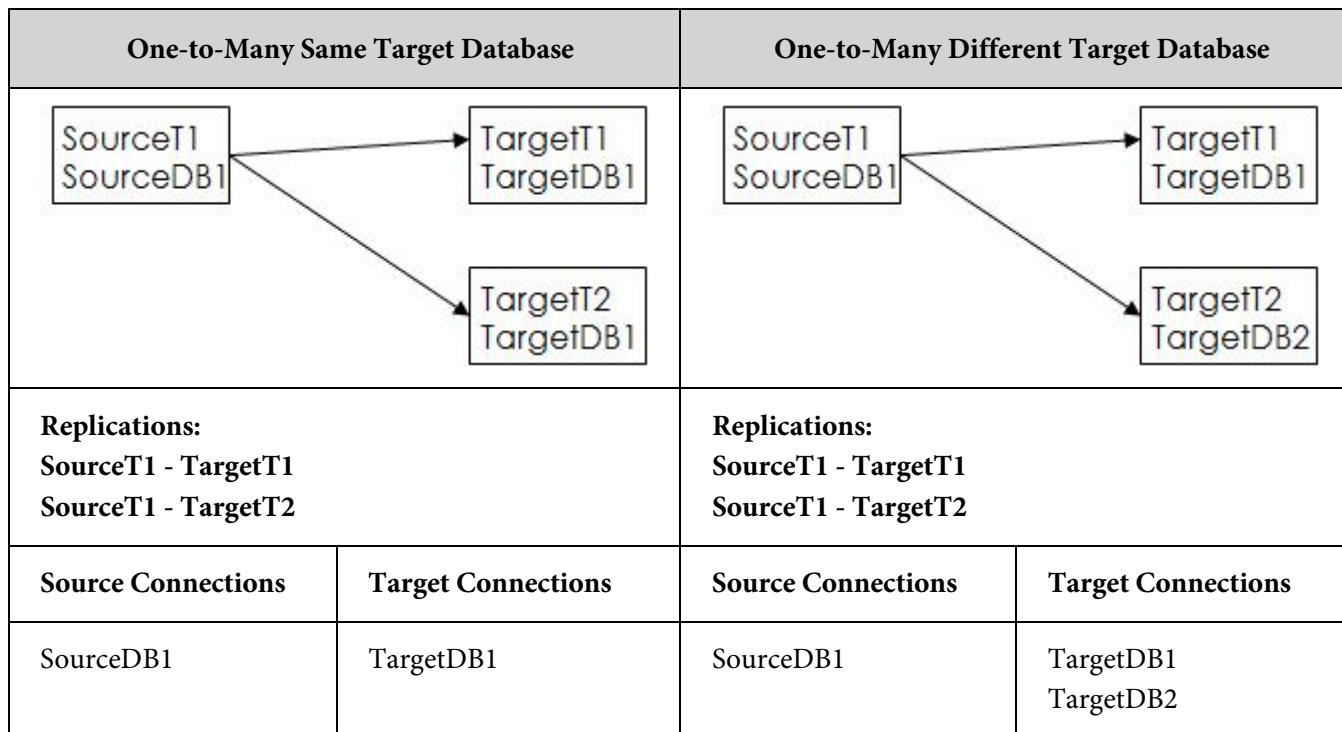
This is the simplest case where a source table is replicated to a target table that may be on the same or a different database server. For this type of replication, you need to define a [source database connection](#) and a [target database connection](#). Then you need to specify replication details that include the type of replication, the scheduling and the mapping between source and target fields in the table. You can also write a script to transform values in the source table for the target table.





### One to Many Table Replication

In this scenario, a source table is replicated to more than one target table where the target tables may be on the same or a different database server. For this type of replication, you need to define a [source database connection](#) and one or more [target database connections](#). You will need a connection for each database server. Then you need to specify replication details for each source-target pair that include the type of replication, the scheduling and the mapping between source and target fields in the table. You can also write a script to transform values in the source table for the target table. For example, in the Customers source table, you might replicate all the customer data to target table A, but just the customer ID and the fields containing financial information to target B. In this example, you would need the following source and target connections:

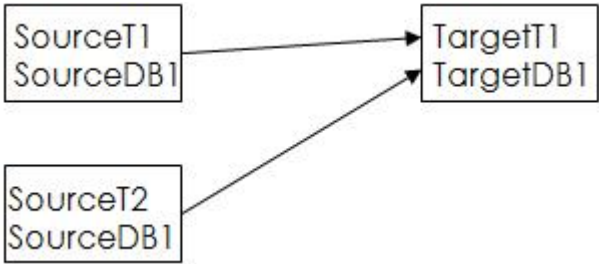
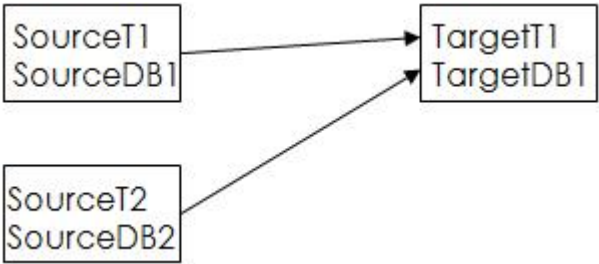




## Many to One Table Replication

In this scenario, multiple source tables are replicated to a single target table. For this type of replication, you need to define [source database connections](#) and a [target database connection](#). You need to specify replication details for each source-target pair that include the type of replication, the scheduling and the mapping between source and target fields in the table. You can also write a script to transform values in the source table for the target table. DBMoto does not provide tools to manage possible conflicts when replicating to a table from different sources. When planning the replications, you need to avoid possible conflicts. For example, if performing a Refresh, you may want to skip the truncation step by [writing a script](#) for the [Refresh\\_onBeforeTruncate event](#).

In replicating two different source tables to a single target table, you would need the following source and target connections:

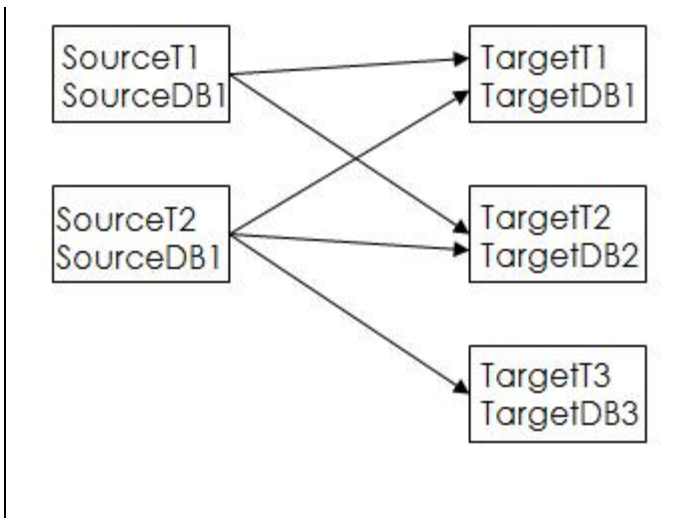
Many-to-One Same Source Database		Many-to-One Different Source Database	
			
<b>Replications:</b> SourceT1 - TargetT1 SourceT2 - TargetT1		<b>Replications:</b> SourceT1 - TargetT1 SourceT2 - TargetT1	
Source Connections	Target Connections	Source Connections	Target Connections
SourceDB1	TargetDB1	SourceDB1 SourceDB2	TargetDB1

## Many to Many Table Replications

In this scenario, multiple source tables are each replicated to one or more target tables. Carefully analyze the relationships for each source-target pair and create source and target connections based on the descriptions above in [Many to One](#)

[Table Replication](#) and [One to Many Table Replication](#). DBMoto does not provide any specific features for handling this type of scenario. In this example, you would need the following source and target connections:

Many-to-Many Same Source and Same Target Databases		Many-to-Many Different Source and Same Target Database	
<p><b>Replications:</b>            SourceT1 - TargetT1            SourceT1 - TargetT2            SourceT2 - TargetT1            SourceT2 - TargetT2            SourceT2 - TargetT3</p>		<p><b>Replications:</b>            SourceT1 - TargetT1            SourceT1 - TargetT2            SourceT2 - TargetT1            SourceT2 - TargetT2            SourceT2 - TargetT3</p>	
Source Connections	Target Connections	Source Connections	Target Connections
SourceDB1	TargetDB1 TargetDB1 TargetDB2 TargetDB2 TargetDB3	SourceDB1 SourceDB2	TargetDB1 TargetDB1 TargetDB2 TargetDB2 TargetDB3
Many-to-Many Same Source and Different Target Databases			



**Replications:**

- SourceT1 - TargetT1**
- SourceT1 - TargetT2**
- SourceT2 - TargetT1**
- SourceT2 - TargetT2**
- SourceT2 - TargetT3**

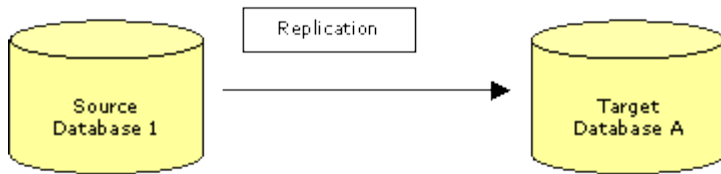
Source Connections	Target Connections
SourceDB1	TargetDB1 TargetDB1 TargetDB2 TargetDB2 TargetDB3

## Database Replication Scenarios

Database replications typically fall into one of the following categories.

### One to One Database Replication

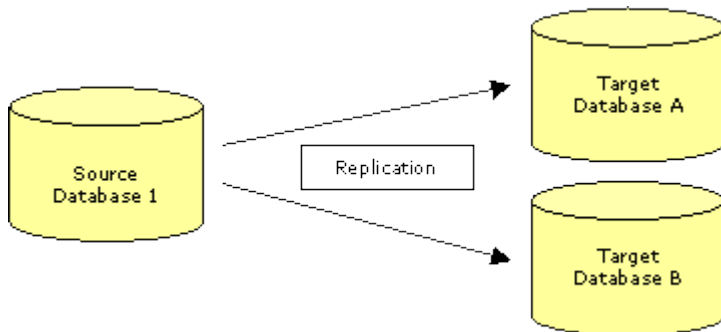
The easiest and most common scenario involves one source database and one target database, where DBMoto reads data from the source database or log/journal, and writes to the target database. The replication may involve one table or several tables. If the replication involves more than one table, you can set up replications for each table quickly and efficiently using the [Multiple Replications wizard](#). See also [Table Replication Scenarios](#) for additional scenarios.



For this type of replication, you will need to define one [source connection](#) and one [target connection](#) before defining replication details for each table involved in the replication. Replication details depend on whether you intend to perform a [refresh replication](#), perform [synchronization](#), or set up [mirroring](#) from the source to the target database.

## One to Many Database Replication

In this scenario, data from a single source database must be replicated to multiple target databases. The replication may involve one table or several tables on the source and target sides. See [Table Replication Scenarios](#) for more information.

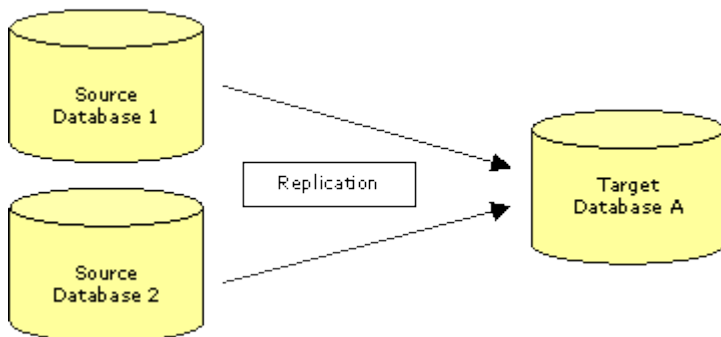


For this type of replication, you will need to define one [source connection](#) and one [target connection](#) for each database to which you are replicating data. You will also need to define replication details for each table involved in the replication. For example, if you have two tables on the source database, one of which needs to be replicated to a SQL Server target database, and the other to an Oracle target database, you will need to set up two replications. Replication details depend on whether you intend to perform a [refresh replication](#), perform [synchronization](#), or set up [mirroring](#) from the source to the target database.

This approach could be useful in the following conditions:

- Department databases are present
- Security policies require several target databases
- You need to reduce network traffic by replicating data only where necessary

## Many to One Database Replication



This scenario often occurs when consolidating enterprise data. You will need to define one [source connection](#) for each database from which you are replicating. If you are replicating to different tables in the target database, you can set up a single [target database connection](#) and define the tables to replicate in the [Replication wizard](#). If you are replicating from tables in multiple source databases to a single table in the target database, a restriction applies. Even though you are targeting a single database, connections to the same table cannot be shared so you will need to set up a target connection that matches each source connection using the same table. The replication requires a unique target connection. You will also need to define replication details for each table involved in the replication. For example, if you have two tables on two source databases that both need to be replicated to the target database, you will need to set up two replications. Replication details depend on whether you intend to perform a [refresh replication](#), perform [synchronization](#), or set up [mirroring](#) from the source to the target database.

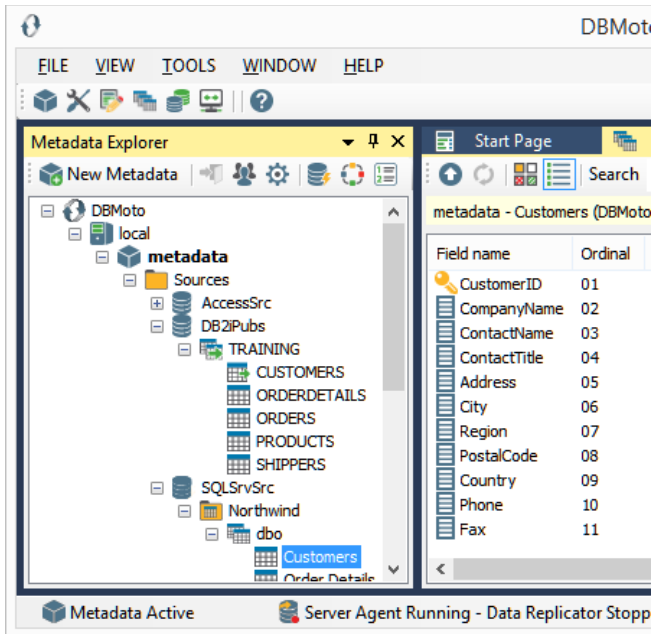
## Primary Key Settings for Mirroring and Synchronization

For transaction-based replications (mirroring and synchronization), it is critical to be able to uniquely identify records in the source and target databases, before performing UPDATE and DELETE operations during replication. Database applications typically make use of primary keys to uniquely identify records.

DBMoto can use primary keys that have been previously set in the source and target databases to identify records during replication. It is also possible to create primary keys within the DBMoto environment for use only during replications with DBMoto. Therefore, if you have no primary key set in the database, you can create one within DBMoto, or, if you have set a primary key in the database, you can override the primary key setting by creating a key within DBMoto.

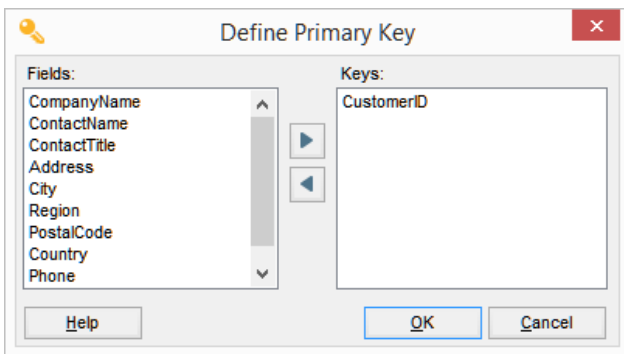
**NOTE:** If you choose to set one or more table columns as primary key in DBMoto, it is critical that each record contains a **unique value**. If the values are not unique for each record, the results of any replication using the table are unpredictable. The primary key columns defined in DBMoto are used to match source and target records based on their values. DBMoto does not enforce primary key constraints internally, and is not able to determine whether a transaction will be applied to multiple records or whether it will generate duplicate values.

When you create a source or target connection and display a table in the DBMoto Management Center, any primary key that is defined in the database is displayed with a key icon to the left of the row name. In the example below, the **authors** table has a primary key set on the **au\_id** column.



To create a primary key in DBMoto:

1. In the Metadata Explorer, select the source or target table for which you want to create a key.
2. From the right mouse button menu, choose **Set Primary Key** to open the [Define Primary Key dialog](#).



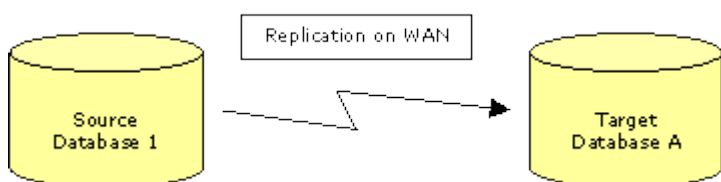
In the **Fields** column of the Define Primary Key dialog, there is a list of fields eligible to be primary keys. It does not include any nullable fields. The **Keys** column of the dialog shows any primary keys that have already been set. If a primary key is set in the database, it is displayed in the **Keys** column of the dialog.

3. To add a new primary key, select a field name in the **Fields** column, then click the right-facing arrow to move the field into the **Keys** column of the dialog. You can use any number of fields (to the limit set by the database you are using) to create a primary key, as long as they can uniquely identify any record.

4. To remove an existing primary key, select the field name in the **Keys** column, then click the left-facing arrow to move the field into the **Fields** column of the dialog.
5. When you have set the primary key to your liking, click **OK** in the Define Primary Key dialog.

## Running DBMoto on a Wide Area Network

DBMoto has been designed to run on wide-area networks (WANs) using client/server technology. However, if source and target databases are connect to DBMoto via a low-speed network, the DBMoto installation should be carefully planned. You can minimize the effects of a low-speed network by following the guidelines below.



### Install DBMoto on a source database LAN when:

- Data to be replicated represents a small portion of the total source data. If you are using filters and performing mapping to only a small number of fields, executing filters and mapping locally (on the same system, or within the same local network) will add minimal load to the WAN.

### Install DBMoto on a target database LAN when:

- The replication process uses several lookup functions on the target database (e.g. when running scripts.) It is more efficient to access target data locally (on the same system, or within the same local network.)

## Trigger-based Transactional Replications

A database trigger is code that is automatically executed in response to certain events on a database table. DBMoto supports trigger-based transactional replications (mirroring or synchronization) with the following databases as a source:

IBM Db2 LUW  
 IBM Db2 z/OS  
 Microsoft SQL Server  
 MySQL  
 Gupta SQL Base  
 Informix  
 Sybase SQL Anywhere

Note that transaction log-based replication is usually the preferred option for [IBM Db2 LUW](#), [Microsoft SQL Server](#) and [MySQL](#)

If you plan to define a trigger-based replication (mirroring or synchronization) with any of the above databases, you need to provide information in the [Source](#) and/or [Target](#) Connection wizards so that triggers can be created to log table changes for replication.

For each table involved in the replication, DBMoto creates 3 triggers in the source table that fire when a specific event occurs on a record:

- INSERT trigger which fires when a new record is being inserted in the table
- UPDATE trigger which fires when a record is modified
- DELETE trigger which fires when a record is deleted

If the replication is later deleted, the triggers are removed by DBMoto. However, note that if you change a replication from mirroring to refresh, the triggers on the source table are not deleted. All transactions will continue to be recorded in the log tables. If you are not planning to reset the replication to mirroring, it is better to delete the replication, so that the triggers are removed, and create a new refresh replication.

Data retrieved using the triggers is stored in log tables that are specified in your Source/Target connection. The master log table can be an existing table or one created specifically to hold DBMoto information. It contains general information about the transactions, like user name, timestamp, table name. A log table (`_DBM__LOG_x`) is also created for each source table in the replication, and contains the data changes identified by the triggers, as well as trigger objects `_DBM__TRG_OBJS`.

Note that DBMoto does not create a tablespace. If you want to have a table space named DBMOTO, you must create it beforehand using a SQL tool such as the [Execute SQL Query dialog](#) in the Management Center. Run a statement like

```
CREATE TABLESPACE DBMOTO
```

When creating a connection, it is important to set the retention time to keep the log table size under control. The higher the value, the more data is kept in the log tables. Try to estimate the number of transactions occurring in all the source tables during a retention period and be sure that the database and table space have enough storage capacity for all those transactions. The DBReplicator (engine) cleans up the log tables periodically, based on the retention setting in the connection dialog. If the engine is not running, the log tables are not cleaned up. This might create space problems in the database as the logs grow in size. If you stop the engine and you are not planning to run it again, be sure to remove all the mirroring synchronization replications.

In addition, if you have many table replications in a single group, using a single connection, all the replications share a master log table. Access to the log table for each source table can become a bottleneck if there are many transactions using the same master log and log tables. DBMoto may report errors about locked tables during replication. Although DBMoto is able to recover from these errors and continue replicating, a better approach is to prevent the errors by splitting the replications into multiple groups with multiple connections and multiple master log tables. First, create



multiple source connections to the database. Use the Transaction Log Type field in the Connection Properties of each connection to open the [Setup Info dialog](#) and create a new master log table for each connection.

During replication:

- When a record is inserted in the source table, the INSERT trigger fires and inserts one record in the master table and one record in the log table associated with the source table. The record inserted in the log table contains all the original values of the INSERT statement.
- When a record is deleted from a table, the DELETE trigger fires and inserts one record in the master table and one record in the log table associated with the source table. The record inserted in the log table contains the key values of the deleted record.
- When a record is updated, the UPDATE trigger fires and inserts one record in the master table and two records in the log table associated with the source table. The two records inserted in the log table contain all the record values before and all the records after the update.

Your system administrator needs to create and define appropriate table spaces and databases to hold the log tables. They should be large enough to handle the expected amount of replication data.

## Resolving Conflicts During Synchronization

In synchronization, or two-way mirroring mode, the assumption is that changes will be made to both tables involved in the replication, and the changes may be different in each table. The goal of synchronization, then, is to make sure that both tables are kept up to date with one another by applying changes from table A to table B and changes to table B to table A.

If you choose synchronization, as your replication type, you should be prepared to handle the situation where changes have occurred at the same time on the same record in both tables involved in the replication. For example, if the respective transaction logs show, at the same mirroring interval, that the Price field in a Product record is updated on SQL Server and also updated on Db2, you need to decide which update is applied to both tables. By default, the changes made to the table in the database defined as the source connection are propagated to the table in the target database.

After defining your replication using the [Replication wizard](#), or [Multiple Replications wizard](#), go to the Preferences tab in the [Replication Properties dialog](#) to specify how conflicts should be handled:

- **SourceServerWins:** This is the default value. Changes applied to the table defined in the source connection are also applied to the table defined in the target connection(s), overriding any changes that have occurred in the target connection table.
- **TargetServerWins:** Changes applied to the tables defined in the target connections are also applied to the table defined in the source connection, overriding any changes that have occurred in the source connection table. This option also requires you to set a priority for the target tables involved so that, if changes occur in more than one target table, DBMoto can determine which changed value to use. Set the priority for target connections as follows.
  1. In the Metadata Explorer, select the group.
  2. From the right mouse button menu, choose **Group Properties**.

3. In the Group Properties dialog, go to the **Preferences** tab.
  4. Click in the **Conflict Resolver Priority** field to view the target connections.
  5. Order the target connections by selecting the connection, then using the up and down arrows to determine its priority.
- **FirstComeWins:** The timestamps of the changes in the tables designated as source and target are compared and the change that applied earliest is applied to all tables.
  - **LastComeWins:** The timestamps of the changes in the tables designated as source and target are compared and the change that applied last is applied to all tables.
  - **Use Script:** This option generates an event that can be handled by writing a function [Replication\\_onConflict](#). The script editor can be accessed from the General Tab Use Script option. You can [write a function](#) that handles the values from each table in any way you wish using VB .NET.



# Chapter 4: Managing Replications in the Management Center

## Modifying/Adding Replication Settings

After creating a replication using the Replication wizard or Multiple Replications wizard, you can modify some replication properties as follows. Note that if you have created a replication as part of a group, you cannot modify the replication properties without first removing the replication from the group.

1. [Disable the replication.](#)
2. [Open the Replication Properties dialog.](#)
3. [Modify any of the editable replication properties.](#)
4. [Enable the replication.](#)

### Disable the Replication

In the DBMoto Management Center:

1. Display the Replication Browser tab.
2. Select the replication you want to modify.
3. On the right mouse button menu, uncheck the **Enable Replication** option.

### Open the Replication Properties Dialog

In the DBMoto Management Center Metadata Explorer:

1. Expand the Replications folder.
2. Select the replication you want to modify.
3. From the right mouse button menu, choose [Replication Properties](#).

### Modify/Add Property Values

Edit properties as needed:

- **General tab/Use script:** Add [scripts](#)
- **General tab/Mapping:** Change mappings
- **General tab/Replication Mode:** Change the replication mode. Note that if you change the replication mode from a transactional replication to a refresh replication involving SQL Server, Db2 or SQLBase, you will be asked if you want to remove objects/components that were created on the database server to support your transactional replication. For example, if using SQL Server, a distributor is created for you.
- **Scheduler tab:** Change the replication schedule
- **Preferences tab:** Modify [thread settings](#) and command/record pool sizes for performance
- **Preferences tab:** For synchronization replications, set a [conflict resolution approach](#)
- **Preferences tab:** Change the isolation level for transactional replications
- **Preferences tab:** Set a SQL WHERE condition for refresh replications to filter the records replicated.

### Enable the Replication

In the DBMoto Management Center:

1. Display the Replication Browser tab.

2. Select the replication you have modified.
3. On the right mouse button menu, check the **Enable Replication** option.

## Copying Replications

If you need to copy a replication (or replications) you have created, you can use one of the following options:

- [Copy a replication in the same metadata, but using different connections.](#)
- [Copy one or more replications to a different metadata](#), for example when moving from a test to a production environment

### Copying a Replication with New Connections in the Current Metadata

Use the [Export Replications wizard](#) to create a new replication based on settings in an existing replication. For example, if you have created a replication that mirrors data from Oracle to SQL Server, and you wish to mirror the same data to a MySQL database, use the Export Replications wizard to define the new target connection and copy the remaining replication information to a new replication.

1. Create the database connections that will be used for the copied replication.
2. Select the replication you want to copy.  
You can also copy a group of replications if they share the same source and the same target connections.
3. From the right mouse button menu, choose Export Replications.
4. Follow steps in the [Export Replications wizard](#) to set new source and/or target connections for the replication.

### Copying Replications to a Different Metadata

With a backup of the metadata containing the replications you want to copy, you can restore the selected replications in the desired metadata using the Custom Restore option and the [Restore Metadata wizard](#).

1. Back up the metadata containing the replications you want to copy: select the metadata in the Metadata Explorer, then, from the right mouse button menu, choose **Backup Metadata**.
2. Select the metadata into which you want to copy the replications.


If the metadata does not yet exist, click  in the Metadata Explorer to create a new metadata set.

3. From the right mouse button menu of the selected metadata, choose **Restore Metadata**.
4. In the Restore Metadata Options dialog, choose the Custom Restore option.
5. Follow steps in the [Restore Metadata wizard](#) to select the replications to copy.

## Copying Database Connections

Once you have created a source or target connection in the DBMoto Management Center, you can copy that connection, then edit its properties as needed. This option can save you time when you have multiple similar connections to create.

To copy a database connection:

1. Expand the Metadata Explorer to display source and/or target connections.
2. Select a connection in the list (connections are displayed with a  icon.)
3. From the right mouse button menu, choose **Copy Connection**.
4. Select the location where you want to paste the new connection (either **Sources** or **Targets**.)
5. From the right mouse button menu, choose **Paste Connection**.
6. Modify the connection name by choosing **Rename Connection** from the right mouse button menu.
7. Add tables to the connection by choosing **Select Tables** from the right mouse button menu.
8. Modify other properties of the connection by choosing **Connection Properties** from the right mouse button menu.

## Using the Management Center while Replications are Running

When the Data Replicator is running, possible actions in the Management Center are as follows:

- You can **create and enable new replications** (both transactional and refresh.) The Data Replicator is able to add replications to its tasks without first stopping then restarting the replicator.
- You can **add new source and target connections**
- You can **disable/enable any replication**
- You can **update any disabled replication**
- You **cannot update an enabled replication**. To modify an enabled replication, first disable it, then make changes, then re-enable the replication.

## Managing and Monitoring Replications on Remote Servers

If you need to monitor or manage replications remotely (for example, if you have DBMoto installed on several systems and want to manage the installations from a single location), it is possible to:

- [Add remote DBMoto servers](#) to any standard installation of DBMoto.
- Install the DBMoto Management Center only, then [add remote DBMoto servers](#) to the Management Center.

If you choose to manage replications remotely, there are a few issues to consider:

- When you install DBMoto to be run from a remote Management Center, the metadata connection has to be accessible from the remote system and point to a unique server name identifiable in the local network. Local-only connection strings such as those that might be used with Microsoft SQL Server CE or Microsoft Access do not work unless remote access is specifically established (for example, if pointing to a local database file, make sure that the file is shared and accessible from the remote machine).
- Each Management Center component installed remotely connects to one (or more) server agent(s) to retrieve authorization/security information and metadata information for the server. The Management Center application will be able to monitor and partially modify remote DBMoto configurations. When the Management Center is used for the local system, additional permissions are available, such as the ability to modify server security and server configuration settings, set the default metadata, add or remove metadata, change license, and so on.

- User control over local and remote installations of the DBMoto server depends on the established user ID and permissions. In a default DBMoto installation, there are no user restrictions. [DBMoto users](#) can be defined in the [User Settings dialog](#), available in the Metadata Explorer.

## Chapter 5: Handling Multiple Replications

### Defining Multiple Table Replications

If you need to replicate all or many of the tables in one database to another database, you can define the replications using the [Multiple Replications wizard](#). This allows you to set the source database, target database and scheduling details just once for all replications involved. The wizard also creates tables in the target database if necessary. While the replications are scheduled to run together, you can optimize performance by first [creating a group](#) for the replications.

Follow the steps below to create a group then create multiple replications.


1. In the DBMoto Management Center Metadata Explorer, select a metadata.
2. For this metadata, create a source connection and a target connection for the databases you are using in the replication.
3. Select the **Groups** item.
4. From the right mouse button menu, choose **Add New Group**.
5. In the [Replication Group Wizard](#), enter a name and, optionally, a description for the group.
6. Click **Next** to display the **Select Replications** screen.
7. Click **Next** to go to the Summary screen right away.
8. Click **Finish** to create the group.
9. In the Metadata Explorer, select the source connection to the database that you want to replicate.
10. From the right mouse button menu, choose **Create Multiple Replications**.
11. In the Define replication type screen, enter a base replication name to identify the table replications that will be created at the end of the wizard. Each table replication will have the base replication name followed by a sequential number to identify it.
12. Check the **Use Group** option.
13. From the drop-down list, select the group you created above.
14. Choose the [type of replication](#) you want to perform.
15. Click **Next** to go to the **Select source connection** screen.
16. Make sure that the correct source connection is selected.
17. Click **Next** to go to the **Select target connection** screen.
18. Select a target connection.
19. Click **Next** to go to the **Set replications** screen.

All tables in the database are automatically selected for replication.



20. If you do not want to replicate all tables, double-click in the check box next to each table that you do not wish to replicate. This will remove the check mark.

By default, source table fields are replicated to fields of the same name in target tables.

21. If you wish to change the source table field - target table field mapping, click  in the right column of the table containing the field.
22. In the [Fields Mapping dialog](#), set the mapping to a field, an expression or no corresponding target.
23. Click **OK** in the Fields Mapping Dialog..
24. Click **Next** to go to the **Scheduling** screen.

Note that if **Enable Replications** is checked and you are running DBReplicator as a service, the replications will start automatically after they have been created at the time you have specified. If you want to hold off on running the replications, uncheck the **Enable Replications** checkbox.

25. [Set the schedule](#) for all replications that have been defined.
26. Click **Next** to go to the **Summary** screen.
27. Review your settings, then click **Start** to create the replications.
28. Click **Finish** to close the wizard.
29. Check that all replications have been created in the Metadata Explorer by expanding the Replications node.

When you have completed the wizard, you will see that a replication has been created for each table that you selected in the wizard. The replication properties are set so that the replications will occur at the same time and within a single group.

## Managing Multiple Concurrent Replications

If you want to run multiple replications concurrently, you have two options:

- [Create a replication group](#)  
The advantage of this option is that you optimize database connections and, if you are mirroring data, optimize access to the transaction log. The transaction log is read only once per group, instead of once per replication where distinct reading threads would be generated that may also be scheduled and executed at different times. All replications in the group must have the same source and target connections. All replications are assigned the same scheduling properties, as well as mirroring/refresh properties, so you do lose some flexibility by grouping replications.
- Set individual replication properties so that the replications are scheduled together  
For each replication, you need to set or edit the scheduling properties so that the replications are scheduled at the same time. This allows you to set separate mirroring/refresh properties for each replication and [set a priority for each](#)

[replication](#). With this option, you gain in flexibility but lose in performance when replications have a common source and target database connection.

## Setting Priorities for Concurrent Replications

You can schedule multiple replications to run at the same time because each replication runs as a separate thread. You can also assign a priority to each replication from the [Replication Properties dialog](#).

1. In the DBMoto Management Center Metadata Explorer, select the replication.
2. From the right mouse button menu, choose **Replication Properties**.
3. In the Replication Properties dialog, go to the [Preferences tab](#).
4. In the **Thread Priority** field, click the down arrow to display a list of priorities.
5. Select the appropriate priority for the replication.
6. Click **OK** to close the Replication Properties dialog.

## Creating Replications with Different Connection Settings

In some situations, you may need to create multiple replications using the same replication settings, but changing either the source or target connection. For example, in replicating tables from IBM Db2 for iSeries/AS400 to SQL Server, there may be several SQL Server databases that need to be updated with the same information. You can create one replication, then use the [Export Replications wizard](#), to create a copy of the replication settings, changing only the target database information.

1. Create source and target connections for all databases involved in replications.
2. Create a replication: [Refresh Mirroring Synchronization](#)
3. From the **Source** menu, choose **Export Replications**.
4. Follow instructions in the wizard to supply the new source and/or target connection settings.

## Synchronizing Data Among Multiple Tables

In the simplest cases, replication is between two tables, a source table and a target table. The replication can be achieved via a full snapshot from the source table to the target table, or via change data capture where only changed data is replicated from the source to the target. Replication can also be two-way between source and target, where changes are expected on both systems, and both systems need to stay synchronized.

In more complex cases, data in tables on several systems needs to stay synchronized. Follow the steps below to synchronize data among more than two tables.

1. [Set up a synchronization replication](#) between the first two tables.
2. In the Replication wizard, be sure to add the replication to a group, creating a new group as needed.

3. Use the Replication wizard to set up an additional replication in the same group, between the source table you specified above and an additional "target" table.
4. Continue to set up replications as in step 3 for all the remaining tables to be synchronized.
5. Make sure that all replications involved in the multi-table synchronization are in the same group.

## Editing Replication Properties

Note that if you edit replication properties for any replication in the group, it is best to do it from the group menu.

1. In the Metadata Explorer, select the group.
2. From the right mouse button menu, choose **Edit Replications**.
3. In the Replication Properties dialog, use the **Replication** drop-down list to select the replication to edit, or choose **All Replications** to make changes that affect all replications in the group.

## Resolving Conflicts During Replication

DBMoto provides a default conflict resolution strategy for situations where multiple changes to the same field are detected: SourceServerWins. There are four additional options that can be set in the Replication Properties dialog:

- **TargetServerWins:** Changes applied to the tables defined in the target connections are also applied to the table defined in the source connection, overriding any changes that have occurred in the source connection table. This option also requires you to set a priority for the target tables involved so that, if changes occur in more than one target table, DBMoto can determine which changed value to use. Set the priority for target connections as follows.
  1. In the Metadata Explorer, select the group.
  2. From the right mouse button menu, choose Group Properties.
  3. In the [Group Properties dialog](#), go to the Preferences tab.
  4. Click in the Conflict Resolver Priority field to view the target connections.
  5. Order the target connections by selecting the connection, then using the up and down arrows to determine its priority.
- **FirstComeWins:** The timestamps of the changes in the tables designated as source and target are compared and the change that applied earliest is applied to all tables.
- **LastComeWins:** The timestamps of the changes in the tables designated as source and target are compared and the change that applied last is applied to all tables.
- **Use Script:** This option generates an event that can be handled by writing a function [Replication\\_onConflict](#). The script editor can be accessed from the General Tab Use Script option. You can [write a function](#) that handles the values from each table in any way you wish using VB .NET.

## Creating a Replication Group

A replication group allows you to set replication properties (scheduling and preferences) for all replications in the group at the same time. All replications in the group must have the same source and target connections. This is useful because it optimizes database connections and transaction log access (if mirroring.)

When a replication is put into a group, it should be considered as part of a whole. If you believe that a replication is likely to need to be taken out of the group several times to do a full refresh (or for whatever reason) it would probably be better to keep it out of the group, or put it into another group to which it logically belongs. Also, if considering the addition of a huge table to a group, it is probably better to keep it as a single replication to avoid unnecessary replication of the table if a problem should occur during replication of the group.

When constraints exist on tables, all the tables with constraints should be placed in a group to ensure that transactions on those tables occur in the same order they were performed on the system being replicated. If you are setting up a group of refresh replications where tables contain foreign key relations, you need to set the [order of replications](#) to maintain integrity across tables.

The recommended approach for using groups is to first create a group, then, as you create replications, create them in the group. This saves time because you only need to set replication properties once for the group. You can also create replications first, then add replications to the group. However, any settings for the replication you are adding will be overridden by the group replication settings.

To create a replication group:

1. In the Metadata Explorer, select the **Groups** item.
2. From the right mouse button menu, choose **Add New Group**.
3. In the [Replication Group Wizard](#), enter a name and, optionally, a description for the group.
4. Click **Next** to display the **Select Replications** screen.
5. Check each replication that you want to include in the group.  
If you have not yet created a replication, click **Next** to go to the Summary screen right away.
6. Click **Next** to go to the Summary screen.
7. Click **Finish** to create the group.

## Adding a Replication to a Group

The recommended approach for using groups is to first create a group, then, as you create replications, create them in the group. This saves time because you only need to set replication properties once for the group. You can also create replications first, then add replications to the group. However, any settings for the replication you are adding will be overridden by the group replication settings.

Certain conditions apply when adding a replication to a group:

- All replications in the group must use the same source connection
- All replications in the group must use the same target connection
- All replications in the group must be the same type (i.e. refresh, mirroring, mirroring with initial refresh, synchronization)
- If replicating from SQL server using mirroring, all replications in the group must use the same source database.

- If replicating from IBM Db2 for i/iSeries/AS400 using mirroring, all replications in the group must use the same journal for source tables.

To add a replication to a group:

1. In the Metadata Explorer, select the group to which you want to add a replication.
2. From the right mouse button menu, choose **Group Properties**.
3. In the Group Properties dialog, check the replication that you want to include in the group.
4. Click **OK**.

If the group already contains one or more replications, the properties of the replication you are adding change to match the properties of other replications in the group.

## Setting Properties for a Group

Groups have two sets of properties:

- **Group Properties**  
Apply to the group as a whole
- **Replication Properties**  
Apply to each replication in the group

You can modify both the group properties and the group replication properties.

## Setting or Changing Group Properties

Group properties apply to the set of replications within the group. You can edit these properties via the [Group Properties dialog](#).

1. Select the group in the Metadata Explorer.
2. From the right mouse button menu, choose **Group Properties...**
3. In the [Group Properties dialog](#), modify the properties as needed.
4. Click **OK** to close the dialog.

## Setting or Changing Replication Properties for a Group

Grouping replications allows you to set replication properties (scheduling and preferences) for all replications in the group at the same time. If you first create a group then create each replication in the group, you only need to set replication properties once for the group. You can also create replications first, then add replications to the group. However, any settings for the replication you are adding will be overridden by the group replication settings.

To change replication settings for a group:

1. In the Metadata Explorer, select the group.
2. From the right mouse button menu, choose **Edit Replications**.

3. In the [Replication Properties dialog](#), modify the properties as needed.
4. Click **OK**.

## Ordering Refresh Replications with Foreign Key Relations

When setting up a replication group to refresh tables with foreign key relations, you need to order replications based on how foreign keys are created because a foreign key record cannot be inserted in the foreign key table before the corresponding key record is inserted in the key table. For example, with Employee and Department tables where a foreign key is defined for an Employee to a Department, you cannot insert an employee record if the employee's department does not exist. The referential integrity of the database does not allow that. During the refresh operation, DBMoto should insert the departments first and then all the employees. However, when tables are deleted (during the truncation phase, prior to the insertion of records), the rule is reversed: a department record cannot be deleted if there is an employee record that has a reference to the department.


Using the example above, you would need to set up the refresh operation for the Department and Employee tables so that all the tables are first truncated in the reverse order to which they are refreshed:

1. Truncate Employee
2. Truncate Department
3. Insert records in Department
4. Insert records in Employee

However, the default order for truncation and subsequent insertion is as follows:

1. Truncate Table1
2. Insert records in Table1
3. Truncate Table2
4. Insert records in Table2

The [Group Properties dialog](#) in the DBMoto Management Center provides a way for you to set up the correct truncation and insertion order:

1. [Create a replication group](#) for all the refresh replications on tables with foreign key relations.
2. Create each replication and [add](#) it to the replication group.
3. For each replication with one or more foreign keys, verify that the foreign key relation is visible in the Management Center Object Browser by selecting the table with the foreign key in the Metadata Explorer. Foreign key relations are displayed with an icon showing a grey key .

Note that if there is any error retrieving a foreign key, DBMoto simply does not show the column as a foreign key.

4. Verify that the foreign key relationship has been defined correctly in the target table, because that is where records need to be deleted in the correct order.
5. When you have created all replications in the group, in the Metadata Explorer, select the group.
6. From the right mouse button menu, choose Group Properties to open the [Group Properties dialog](#).
7. On the Preferences tab, set the Truncate Inverse field to True.  
This causes related tables to be truncated in the correct order to maintain data integrity.
8. On the General tab, click Reorder Replications.  
DBMoto checks the foreign key relations and reorders the replications so that they will execute in an order that maintains data integrity at all times.

## Chapter 6: Database-Specific Replication Issues

### Replicating from Multi-Member Tables

If you plan to replicate data from a Db2 multi-member table running on i (iSeries/AS400), you have the following options available:

- Replicate the entire table (all members) by setting up a replication to the main table. Follow the [Steps for Replicating a Table Using Refresh Mode](#), or the [Steps for One-Way Mirroring Between Tables](#).
- Replicate one or more member tables by creating aliases for each of the member tables you plan to use, then setting up replications using the aliases. The steps below provide more information.

### Replicating Member Tables

1. On your System i, create a table alias for each member table from which you want to replicate. The SQL syntax for creating a table alias is as follows:

```
CREATE ALIAS EMPLOYEE_EUROPE FOR EMPLOYEE(EMPEUROPE)
```

where EMPLOYEE\_EUROPE is the alias created for member EMPEUROPE of multi-member table EMPLOYEE.

2. Follow the [Steps for Replicating a Table Using Refresh Mode](#), or the [Steps for One-Way Mirroring Between Tables](#) to set up a replication using the alias you have created on the Db2 system and the target table.

If you are replicating from several member tables, you can [set up multiple replications](#) using the [Multiple Replications wizard](#).

### Using a Row ID to Identify a Record (Oracle)

For transaction-based replications (mirroring and synchronization), it is critical to be able to uniquely identify records in the source and target databases, before performing UPDATE and DELETE operations during replication. Database applications typically make use of primary keys to uniquely identify records.

DBMoto can use primary keys that have been previously set in the source and target databases to identify records during replication. It is also possible to [create primary keys within the DBMoto environment](#) for use only during replications with DBMoto. If, however, you are replicating from Oracle and you do not have a primary key set on the source table, you can use the Oracle Row ID to identify records during replication.

To use the Row ID to identify each record:

1. In the target table, create an additional column of type String.
2. If you have not yet created the replication, when you get to the Set Mapping Info screen of the [Replication wizard](#) or [Multiple Replications wizard](#), choose to map the newly created column to an expression.

If you have created the replication, in the [Replication Properties dialog](#), open the [Fields Mapping dialog](#) and choose to map the newly created column to an expression.

3. In the [Expression Generator](#), expand the tree to display the Values node.



4. Expand the Values node to display Log Field values.
5. Select the !RecordID value.  
The !RecordID value represents the Oracle Row ID.
6. Click **OK** in the Expression Generator.

You have now set the value of the column that you created above to the Oracle Row ID.

Any time you reorganize a source table (to compress deleted rows, etc.), DBMoto needs to perform a full refresh of all the target tables mapped using the Row ID.

## Using a Relative Record Number to Identify a Record (IBM Db2 for i)

For transaction-based replications (mirroring and synchronization), it is critical to be able to uniquely identify records in the source and target databases, before performing UPDATE and DELETE operations during replication. Database applications typically make use of primary keys to uniquely identify records.

DBMoto can use primary keys that have been previously set in the source and target databases to identify records during replication. It is also possible to [create primary keys within the DBMoto environment](#) for use only during replications with DBMoto. If, however, you are replicating from IBM Db2 for i (iSeries/AS400) and you do not have a primary key set on the source table, you can use the RRN (relative record number--a hidden column that uniquely identifies the record) to identify records during replication.

To use the RRN to identify each record:

1. In the target table, create an additional column of type LONG.
2. If you have not yet created the replication, when you get to the Set Mapping Info screen of the [Replication wizard](#) or [Multiple Replications wizard](#), choose to map the newly created column to an expression.

If you have created the replication, in the [Replication Properties dialog](#), open the [Fields Mapping dialog](#) and choose to map the newly created column to an expression.

3. In the [Expression Generator](#), expand the tree to display the Values node.
4. Expand the Values node to display Log Field values.
5. Select the RecordID value.  
The RecordID value represents the Db2 for i (iSeries/AS400) Relative Record Number.
6. Click **OK** in the Expression Generator.

You have now set the value of the column that you created above to the RRN.

Any time you reorganize a source table (to compress deleted rows, etc.), DBMoto needs to perform a full refresh of all the target tables mapped using the RRN.


## Setting the CCSID for Tables and Columns

DBMoto allows you to set the CCSID (language) for tables and/or columns using the [Table Properties dialog](#). This is useful if you are working with a table that contains columns with characters in different languages or if you need to change the language setting for the entire table. For example, if a column in a n IBM Db2 for i (iSeries/AS400) table is defined as CCSID 37, but in reality it contains data in Arabic, you can set the CCSID for that column to Arabic.

To set the CCSID for an entire table:

1. Select the source table in the DBMoto Management Center.
2. Select **Table Properties** from the right mouse button menu.
3. Edit the value of the **Character CCSID** property. You can obtain the CCSID values from your Db2 documentation.
4. Click **OK** to close the dialog.


To set the CCSID for a specific column:

1. Follow the steps above to display the Table Properties dialog.
2. In the Character CCSID field, click  to open the Field CCSID Settings dialog.
3. Set the CCSID value for columns as needed.
4. Click **OK** to close the dialog.
5. Click **OK** to close the Table Properties dialog.

## Customizing Character Data by Overriding the CCSID Setting

In certain conditions, you might want to customize the current codepage (CCSID) default values during read operations with user-defined character (UDC) mapping or override an existing character with a different one. For example, you could have a Db2 table where there are null string values and you want them show up as blank characters in your application.

To customize the current code page, follow the steps in the example below. The example assumes that you want to override character conversion in CCSID 280 (Italian) so that null values are converted to blank characters. The EBCDIC value in the Db2 table for null is 00, and the decimal value for blank characters in the Microsoft Windows environment is 32.

1. In the DBMoto Management Center Metadata Explorer, select the IBM Db2 for i/iSeries/AS400 connection.
2. From the right mouse button menu, choose **Connection Properties...**
3. In the [Source Connection Properties dialog](#), select the **Connection** field, then click  the Ellipsis button to view the connection properties for the Ritmo/i provider.
4. Scroll to the **Override CCSID** field.

5. Type the value for the CCSID you wish to override. In this example, the value would be 280.
6. Click **OK** to close the Connection Properties dialog for Ritmo/i properties.
7. Click **OK** to close the Source Connection Properties dialog.
8. Create an empty text file named <ccsid number>.txt. In this example, the file name would be 280.txt.
9. Open the file in a text editor.
10. Add the characters 00,32 derived from the links below.
11. To get the right decimal values for the mapping, view the IBM table at: <http://www-03.ibm.com/systems/i/software/globalization/codepages.html>
12. Open the Italian codepage (Italy - CECP (00280)): <http://www-03.ibm.com/systems/i/software/globalization/pdf/cp00280z.pdf>.
13. Then open the Windows, Latin 1 codepage (01252): <http://www-03.ibm.com/systems/i/software/globalization/pdf/cp01252z.pdf>. The blank character is 20hex= 32 decimal.
14. Save the file 280.txt.
15. Put the file in the same folder as the Ritmo/i Sql400.dll, usually C:\Program Files\HiT Software\DBMoto V6\Ritmo\_i\lib.
16. If you need to override more than one code page just write additional CCSID numbers separated by commas.

To override more than one character in a file, type additional character pairs on separate lines. For example, assume the file 280.txt already contains the pair 00,32. To add a character conversion where occurrences of "a" would be converted to "@":

Table 280: "a" (81 hex = 129 decimal)

Table 1252: "@" (40 hex = 64 decimal)

File 280.txt would contain the following:

00,32

129,64

## Synchronization Limitation with IBM Db2 for i

If you are planning a synchronization replication with IBM Db2 for i (iSeries/AS400) as a source or target database, note that any CLRPFM commands which DBMoto encounters in the journal will not be propagated to the destination table. DBMoto synchronization does not support the use of clear on physical members, although clear commands are supported for mirroring replications.

To work around this restriction, avoid calling the CLRPFM command, replacing it with DELETE FROM TABLE without adding a WHERE condition.

## Writing a Script to Determine Receiver Use

If you are managing replications with IBM Db2 for i (AS400/iSeries) as a source database, you may want to identify which receivers are in use at any time so that unused receivers can be deleted. DBMoto provides a global script function called [GetReceiversInUse](#) so that you can identify all the receivers in use for replications using a specific connection to Db2. There are two ways you might use this function.

### 1. Receiver Names are Auto-Generated with Sequential Names

The first (and simplest) approach can be applied when receiver names are sequentially autogenerated in the format <rec\_prefix>nnn, where nnn is a sequential number. This occurs when the JRNRCV(\*GEN) parameter is used in the change journal. If you run the function below, you obtain a list of receivers in use by DBMoto under that connection.

```
GetReceiversInUse("r;MyISeriesConnection, True, True)
```

The list of receivers returned is sorted by (in order): Journal Library, Journal Name, Receiver Library and Receiver Name. For instance, you could obtain these receivers as output:

```
JRNLIB1.JOURNAL1  RECLIB1.RECV0003
JRNLIB1.JOURNAL1  RECLIB1.RECV0004
JRNLIB1.JOURNAL1  RECLIB1.RECV0005
JRNLIB1.JOURNAL1  RECLIB1.RECV0006
JRNLIB1.JOURNAL2  RECLIB2.RECV0015
JRNLIB1.JOURNAL2  RECLIB2.RECV0016
```

**Warning:** The results from this function show only receivers in use by DBMoto. Other receivers may be used by other applications. Do not remove receivers unless you have verified that they are not needed.

This result shows that, under JOURNAL1, you can remove RECLIB1.RECV001 and RECLIB1.RECV002 because they are not in use by DBMoto; under JOURNAL2, you can remove RECLIB2.RECV0001 through RECLIB2.RECV0014 because they are not in use by DBMoto.

With this information, you could write a script function using the event LogReader\_onReceiverChanged to detect the current list of receivers not in use and then run the command to remove them physically from the i(iSeries/AS400) system.

### 2. Receiver Names are NOT Sequentially Named

If sequential naming is not guaranteed, there is no way to detect if a receiver, based on its name, is in the receiver chain of one specific receiver. In this case, a script would need to store the receiver chain for each journal in a shared list in memory, but preferably in a file. For instance, the LogReader\_onReceiverChanged event might return the previous and the new receiver:

```
JRNLIB1.JOURNAL1  RECLIB1.RECV_A    old receiver
JRNLIB1.JOURNAL1  RECLIB1.RECV_B    new receiver
```

Every time the receiver changes, its name needs to be stored in a list. At some point the list might look like this:

```
JRNLIB1.JOURNAL1 RECLIB1.RECV_A  
JRNLIB1.JOURNAL1 RECLIB1.RECV_B  
JRNLIB1.JOURNAL1 RECLIB1.RECV_C  
JRNLIB1.JOURNAL1 RECLIB1.RECV_D  
JRNLIB1.JOURNAL1 RECLIB1.RECV_E  
JRNLIB1.JOURNAL1 RECLIB1.RECV_F
```

Notice that letters A through F have been used to simplify the sample, but the receivers could have any names. If at this point you run [GetReceiversInUse](#) and you find out that the first receiver in use is RECLIB1.RECV\_C, you know that receivers RECLIB1.RECV\_A and RECLIB1.RECV\_B can be removed because they appear earlier in the list.

## Chapter 7: Writing Scripts

### Writing Scripts in DBMoto

DBMoto supports three types of scripts:

- [Global scripts](#)  
A global script is a shared module that defines a global class of functions. It can be used to define general user functions that need to be called from inside a replication script or an expression script.
- [Replication scripts](#)  
A script that is defined to run when a specific event occurs during replication.
- [Expression scripts](#)  
A script used in the Mapping Dialog when a column value needs to be mapped not simply to a source column but to a value retrieved after calculation.

DBMoto scripts can be written using C# or Visual Basic .NET. C# is the default scripting language for all script types. However, it is possible to change the scripting language for a specific metadata in the Global Script Editor.

### Changing the Script Language

To change the script language for all scripts associated with a metadata:

1. In the Metadata Explorer, select the metadata.
2. From the right mouse button menu, choose **Global Script**.
3. In the [Global Script Editor](#), change the language in the Script Language drop down list.

Note that any scripts already saved in a DBMoto Script Editor will become commented code when the script language is changed.

### Visual Basic .NET Tips for Developers

Note that, while there are many similarities between Visual Basic and Visual Basic .NET, there are also some subtle differences. For information about the Visual Basic .NET language, see <http://msdn.microsoft.com/vbasic/using/>

- Use `IsNothing` in place of Visual Basic's `IsNull`
- Always use parentheses for the arguments of a function, even if there are no arguments
- Always use typed parameters. Define the return type of your functions.
- In a global script, define all custom procedures (Subs and Functions) using the keyword 'Shared', in order to define them as static functions.
- Always compile a script before executing it. The VB.NET compiler is stronger than the old VBA compiler and it is able to find errors at an earlier stage.

### Writing a Custom Create Table Rule

When setting up a replication, it is often the case that the replication source table does not exist on the target table. DBMoto can be used to create the target table via the [Create Target Table wizard](#) which generates the SQL code for the target database environment. While it is possible to modify the way that the table is created in the target database by editing the SQL code in the wizard, this approach is not very convenient in situations where many target tables will be

created with the same type of modification (adding a column, for example.) DBMoto also provides a metadata-level feature to customize the way that target tables are created. Here is a brief overview of the steps involved.

1. Create a global script with a definition for a CreateTableRule class.

The global script template contains a new class definition:

```
public class CreateTableRule : ICreateTableRule
```

2. Compile and save the script.
3. Edit the Target Connection Property CREATE TABLE Custom Rule to specify use of the new class in creating target tables.

The default value for this property is 'Automatic.' After a custom rule has been defined in the global script, it can be selected as a value in this field, and used by default for all tables created for the connection.

A second property, Default Mapping Rule, can be set if you have also created a mapping rule in the global script (see example below.)

4. In the [Create Target Table wizard](#), or the [Multiple Replications wizard](#) select or deselect the new rule as needed for each table.

## Step-by-Step Example

Below is a complete example that shows the steps to implement an automatic rule for target table creation together with a field mapping rule to map source to target fields after table creation.

1. In the DBMoto Management Center Metadata Explorer, select the metadata for which you want to create a global script.
2. From the right mouse button menu, select **Global Script**.
3. In the [Global Script Editor](#), edit the template for CreateTableRule.
4. Write a CreateTableAttribute definition to determine the function name that appears in the Target Connection Properties dialog, the Create Table wizard and the Multiple Replications wizard. The first string contains the text that will appear in drop-down menus. The second string contains text that will serve as a tooltip description of the function.

```
public class CreateTableRule : ICreateTableRule
{
    [CreateTableRuleAttribute("Audit Table New", "Create columns for an audit replication")]
    public bool MyCustomAuditTable (List<ColumnClass> aTargetFields)
    {
        ColumnClass colClass;
        colClass = new ColumnClass();
        colClass.Name = "REC_ID";
    }
}
```

```

colClass.AllowNull = false;
colClass.TypeName = "integer";
colClass.PrimaryKeyPos = 1;
aTargetFields.Insert(0, colClass);

colClass = new ColumnClass();
colClass.Name = "TID";
colClass.AllowNull = true;
colClass.TypeName = "varchar";
colClass.Size = 50;
aTargetFields.Add(colClass);

colClass = new ColumnClass();
colClass.Name = "TTS";
colClass.AllowNull = true;
colClass.TypeName = "bigint";
aTargetFields.Add(colClass);

colClass = new ColumnClass();
colClass.Name = "UserID";
colClass.AllowNull = true;
colClass.TypeName = "varchar";
colClass.Size = 20;
aTargetFields.Add(colClass);

return true;
}
}

```

This function takes a list of ColumnClass objects as input. This is the ColumnClass definition:

```

public class ColumnClass
{
    public string Name;
    public int CCSID;
    public string TypeName;
    public int Size;
    public int Precision;
    public int Scale;
    public int PrimaryKeyPos;
    public bool AllowNull;
    public string Default;
}

```

Whenever a target table is to be created, the function will pass columns in the format of a list of ColumnClass objects. Within the function, new columns can be appended at specified positions in relation to existing columns, or columns can be removed or column definitions changed.



In this example, a REC\_ID integer column is added at position 0 (notice the Insert at index 0), specifying the PrimaryKeyPos = 1. PrimaryKeyPos is a value which, if other than 0, indicates that the field will be set as part of the primary key definition, at the specified position if not taken by other existing fields, or at the first available position.

Three more columns are added at the end of the columns list, TID, TTD and User\_ID. For each new column, the datatype, size, precision and scale as requested by the database are specified. This means that every time a specific Create Table Rule is written, it is critical to consider the target database to which the rule is applied.

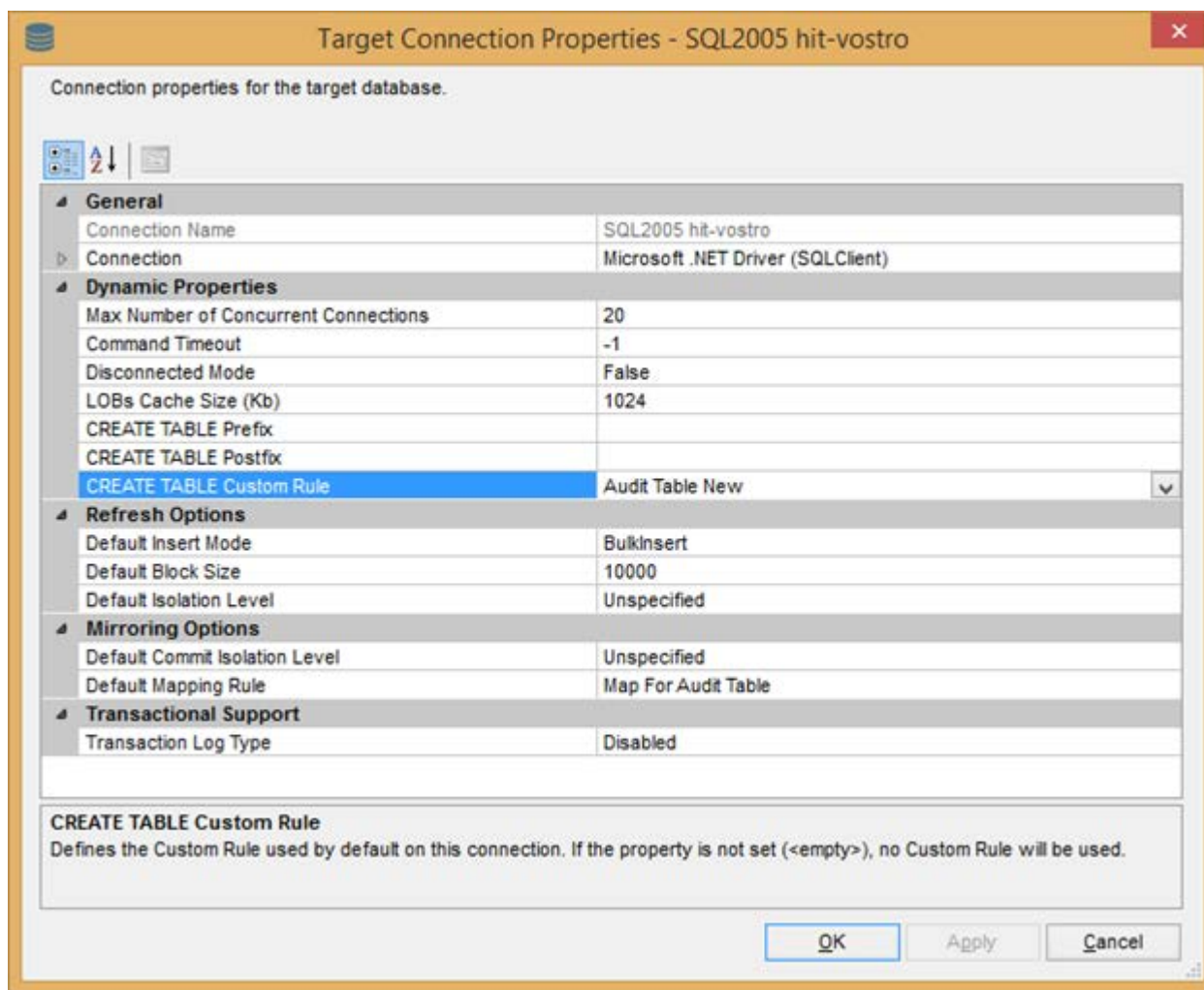
5. In the Global Script Editor, edit the MappingRule template to add a custom mapping rule that will be used to map columns of the above 'audit' tables:

```
public class MappingRule : IMappingRule
{
    [MappingRuleAttribute("Map For Audit Table", "Match names considering the
    additional fields on audit tables")]

    public bool AuditTableMapping (bool bIsForth, string sSourceName, int
    iSourceOrdinal, string sSourceType, bool bIsSourcePrimaryKey, bool
    bIsSourceNullable, int iSourceSize, short sSourcePrecision, short sSourceScale,
    string sTargetName, int iTargetOrdinal, string sTargetType, bool
    bIsTargetPrimaryKey, bool bIsTargetNullable, int iTargetSize, short
    sTargetPrecision, short sTargetScale, ref System.Text.StringBuilder sExpression)
    {
        if (String.Compare(sTargetName, "TTS", true) == 0)
        {
            sExpression.Append("[TransactionTS]");
            return true;
        }
        else if (String.Compare(sTargetName, "TID", true) == 0)
        {
            sExpression.Append("[TransactionID]");
            return true;
        }
        else if (String.Compare(sTargetName, "UserID", true) == 0)
        {
            sExpression.Append("[UserID]");
            return true;
        }
        else
            return String.Compare(sSourceName, sTargetName, true) == 0;
    }
}
```

6. Save and compile the script.

7. In the Metadata Explorer, select the target connection for which you have defined the rules.
8. From the right mouse button menu, choose Connection Properties.
9. In the Target Connection Properties dialog, scroll to the CREATE TABLE Custom Rule property.

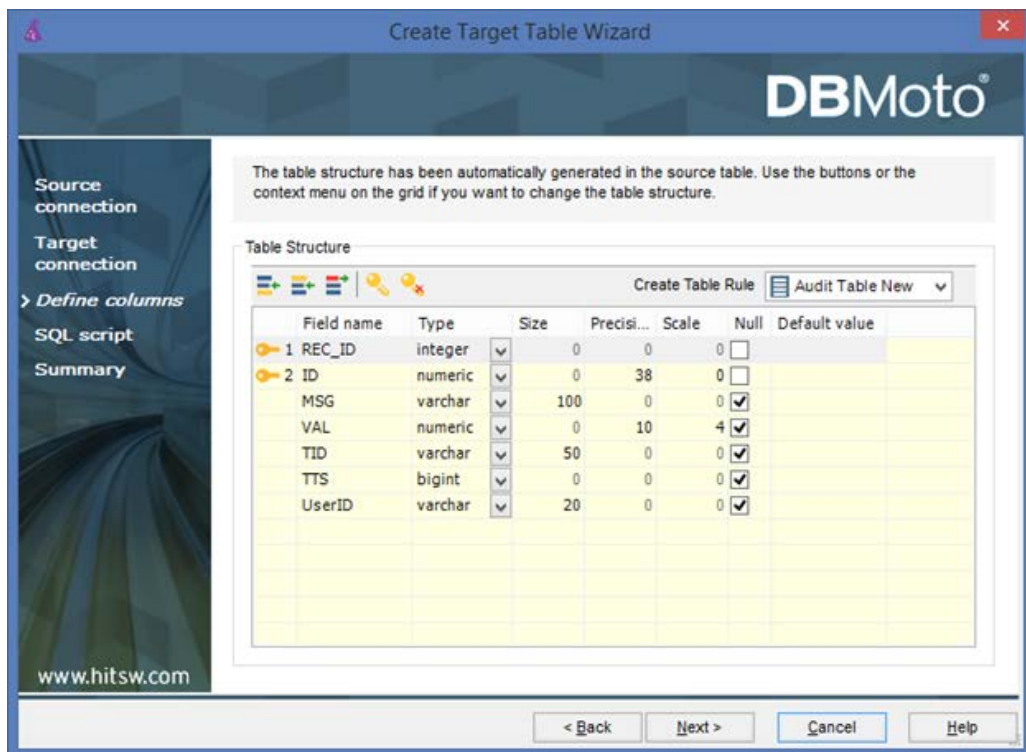


10. The default value of for the property is 'Automatic' which means create tables as they are on the source.
11. From the drop-down menu, select the 'Audit Table New' rule you created above.
12. In the Default Mapping Rule field, select 'Map for Audit Table' created above.
13. Click OK to apply the changes.

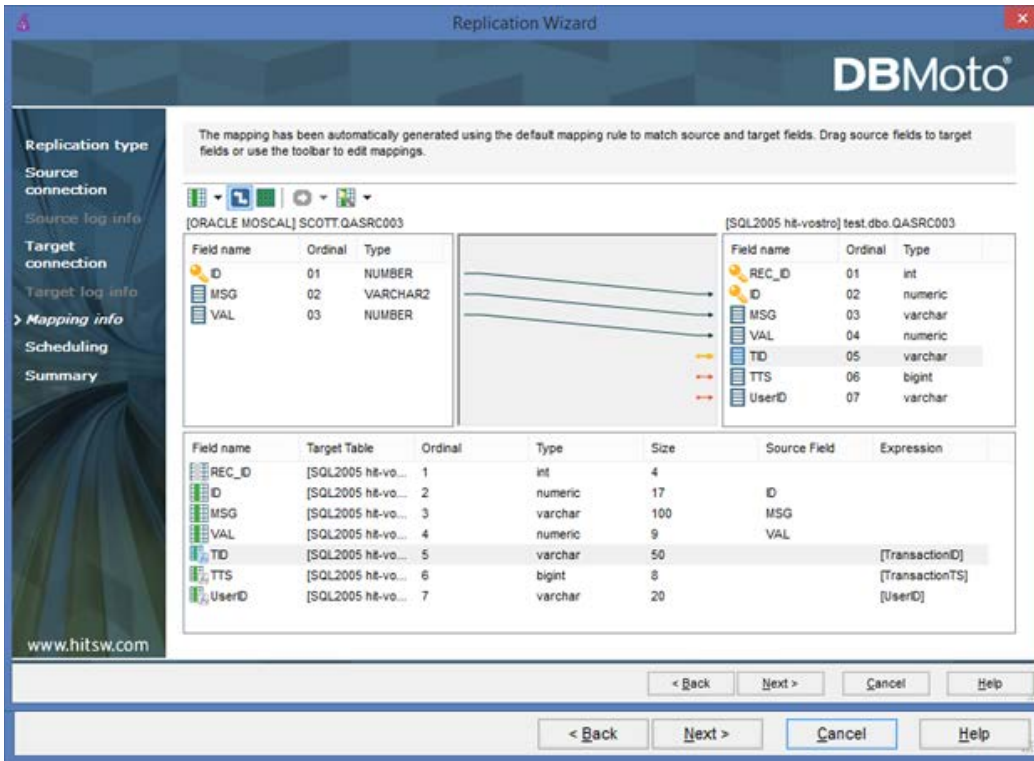
The custom functions are applied either when using the Create Target Table Wizard or when creating multiple replications in the Multiple Replications wizard.

## Create Target Table Wizard

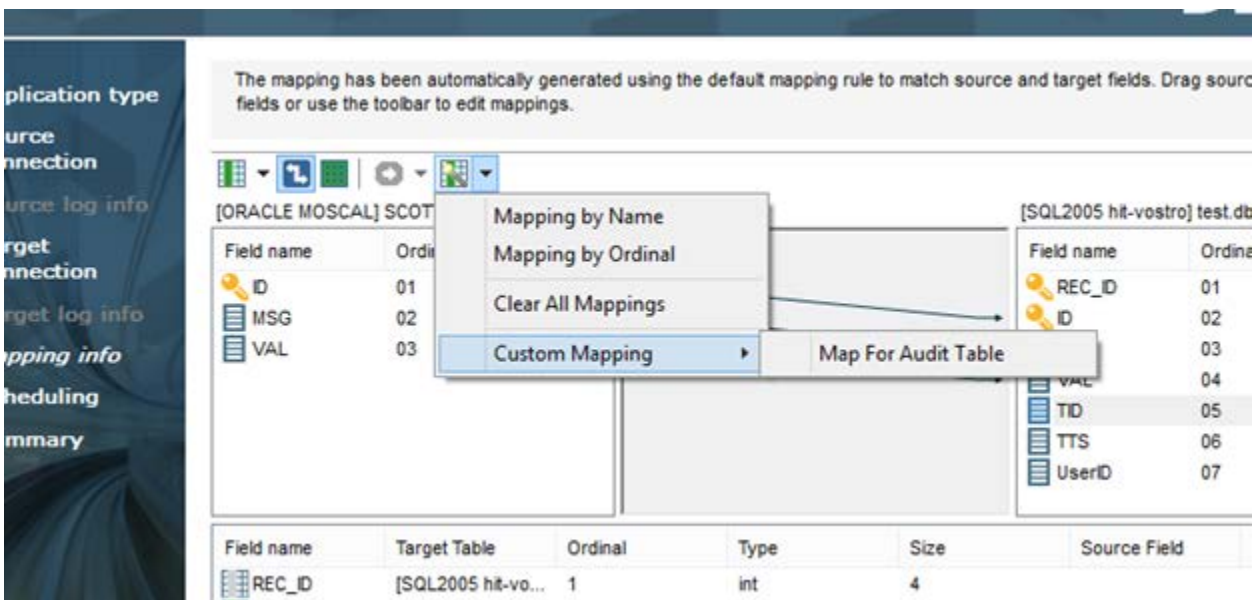
In the Create Target Table wizard, the Define columns screen displays a Create Table Rule drop-down menu, with the Audit Table New rule. The list of columns is modified as specified in the rule. You can change the value of Create Table Rule by selecting a different item from the menu.



In the Create New Replication wizard, the Mapping Info step initializes the mappings by adding the custom mapping rule defined in the script function.

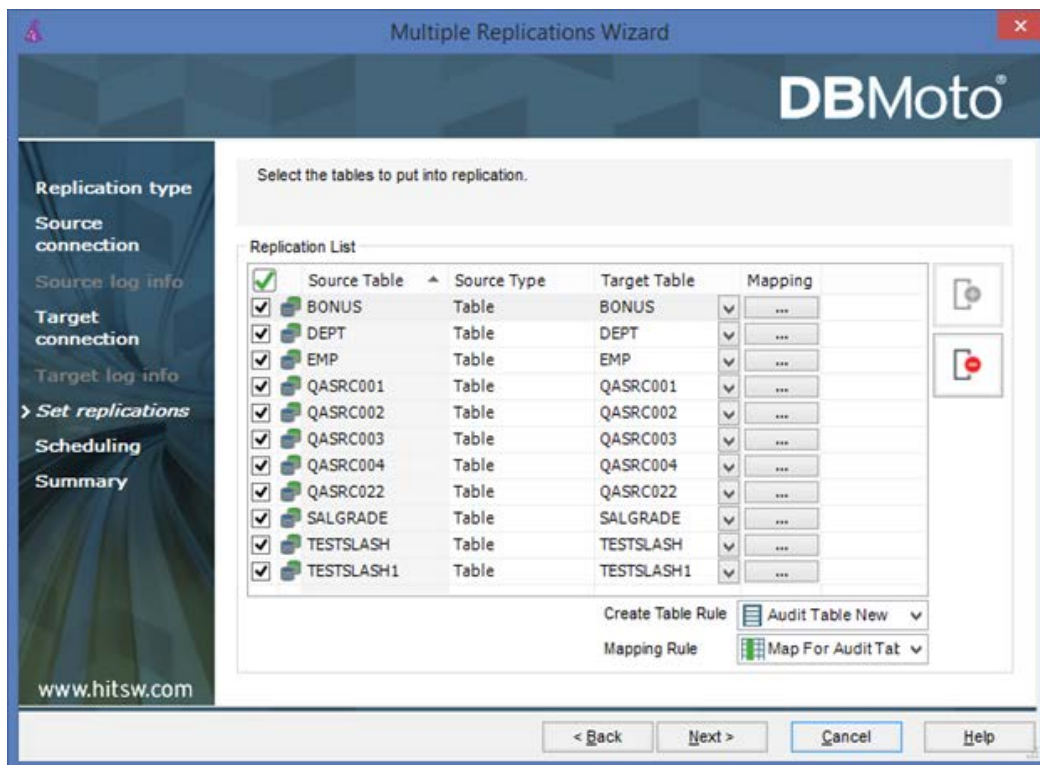


You can modify the mapping either manually or selecting a different mapping function from the right most button in the toolbar.



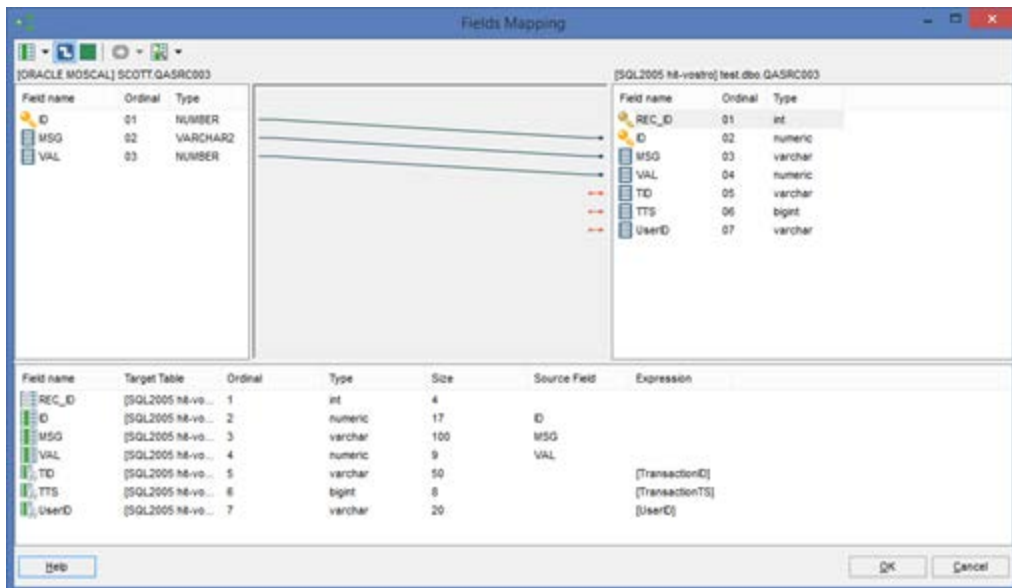
## Multiple Replications Wizard

Both the Create Table Rule and the Mapping Rule appear in the Set Replications screen, with the default values for the connection automatically selected.



The created tables and replications have the additional fields and custom mappings:





## Writing a Field Mapping Function

This type of function, a mapping rule, can be used to identify which source and target columns to map when [defining a replication](#). The default approach is to map source table columns to target table columns with matching names. If, for example, source table column names and target table column names use a different prefix, you could write a function to compare names after skipping the prefix as in the VB.NET and C# example below.

### VB.NET Example

```
Public Class MappingRule : Inherits IMappingRule
```

```
<MappingRuleAttribute("Match By Prefix", "Match names considering the prefix  
'fld_')>
```

```
Public Function CustomMapping_ByNamePrefixed (
    ByVal sSourceName As String,
    ByVal iSourceOrdinal As Integer,
    ByVal sTargetName As String,
    ByVal iTargetOrdinal As Integer) As Boolean

    If sTargetName.Length >= 4 Then
        Return (String.Compare(sSourceName, sTargetName.SubString(4), True) = 0)
    Else
        Return False
    End If
End Function
```

```
<MappingRuleAttribute("Match By Suffix", "Match names considering the suffix  
'_fld')>
```

```
Public Function CustomMapping_ByNameSuffixed (  
    ByVal sSourceName As String,  
    ByVal iSourceOrdinal As Integer,  
    ByVal sTargetName As String,  
    ByVal iTargetOrdinal As Integer) As Boolean  
  
    If sSourceName.Length >= 4  
        Return (String.Compare(sSourceName.SubString(4), sTargetName, True) = 0)  
    Else  
        Return False  
    End If  
End Function
```

```
<MappingRuleAttribute("Match Custom", "Match names considering a specific custom  
rule")>
```

```
Public Function MyCustomMapping (  
    ByVal bIsForth As Boolean,  
    ByVal sSourceName As String,  
    ByVal iSourceOrdinal As Integer,  
    ByVal sSourceType As String,  
    ByVal sTargetName As String,  
    ByVal iTargetOrdinal As Integer,  
    ByVal sTargetType As String,  
    ByRef sExpression As System.Text.StringBuilder) As Boolean  
    /// ...  
End Function
```

```
<MappingRuleAttribute("Match Custom Extended Params", "Match names considering  
a specific custom rule")>
```

```
Public Function MyCustomMapping (  
    ByVal bIsForth As Boolean,  
    ByVal sSourceName As String,  
    ByVal iSourceOrdinal As Integer,
```

```

        ByVal sSourceType As String,
        ByVal bIsSourcePrimaryKey As Boolean,
        ByVal bIsSourceNullable As Boolean,
        ByVal iSourceSize As Integer,
        ByVal sSourcePrecision As Short,
        ByVal sSourceScale As Short,
        ByVal sTargetName As String,
        ByVal iTargetOrdinal As Integer,
        ByVal sTargetType As String,
        ByVal bIsTargetPrimaryKey As Boolean,
        ByVal bIsTargetNullable As Boolean,
        ByVal iTargetSize As Integer,
        ByVal sTargetPrecision As Short,
        ByVal sTargetScale As Short,
        ByVal sExpression As System.Text.StringBuilder) As Boolean

    /// ...
End Function

```

End Class

## C# Example

```

using System;
using System.Data;
using DBMotoPublic;
using DBMotoScript;
namespace DBRS
{
    public class GlobalScript : IGlobalScript
    {
    }
    public class MappingRule : IMappingRule
    {
        [MappingRuleAttribute("Match By Prefix", "Match names considering the
prefix 'fld_')]
        public bool CustomMapping_ByNamePrefixed (String sSourceName, int
iSourceOrdinal, String sTargetName, int iTargetOrdinal)
        {

```



```

        if (sTargetName.Length >= 4)
        {
            return (String.Compare(sSourceName, sTargetName.Substring(4), true)
== 0);
        }
        else
        {
            return false;
        }
    }

    [MappingRuleAttribute("Match By Prefix", "Match names considering the prefix
'fld_')]
    public bool CustomMapping_ByNameSuffixed (String sSourceName, int
iSourceOrdinal, String sTargetName, int iTargetOrdinal)
    {
        if (sSourceName.Length >= 4)
        {
            return (String.Compare(sSourceName.Substring(4), sTargetName, true) ==
0);
        }
        else
        {
            return false;
        }
    }

    [MappingRuleAttribute("Match Custom", "Match names considering a specific
custom rule")]
    public bool MyCustomMapping(bool bIsForth, String sSourceName, int
iSourceOrdinal, String sSourceType, String sTargetName, int iTargetOrdinal,
String sTargetType, ref System.Text.StringBuilder sExpression)
    {
        // ...
        return true;
    }

```

```
}  
public class GlobalEvents : IGlobalEvents  
{  
}  
}
```

## Parameters

The possible parameters for the functions are:

**bIsForth** - TRUE or FALSE depending on whether mapping is forward or backward direction (for synchronization). This allows you to write a script that handles both directions of mapping. For instance, if **bIsForth** is True, then apply mapping by name; if **bIsForth** is False then apply a different rule.

**sSourceName** - The source  
**iSourceOrdinal** - The source column ordinal position (expressed as an integer)

**sSourceType** - The source column type (original database type). This can be used to apply custom mapping rules that work on specific datatypes. For instance, you could write a rule that applies a Trim function to the mapping for all VARCHAR columns.

**bIsSourcePrimaryKey** - Specifies whether the source field is a primary key

**bIsSourceNullable** - Specifies whether the source field is nullable.

**iSourceSize** - Source field size expressed as an integer

**sSourcePrecision** - Source field precision expressed as a short

**sSourceScale** - Source field scale expressed as a short

**sTargetName** - The target column name

**iTargetOrdinal** - The target column ordinal position (expressed as an integer)

**sTargetType** - The target column type (original database type).

**bIsTargetPrimaryKey** - Specifies whether the target field is a primary key

**bIsTargetNullable** - Specifies whether the target field is nullable. Allows you to check if the target field is not nullable, and thereby avoid cases where NULL values are passed to the target.

**iTargetSize** - Target field size expressed as an integer

**sTargetPrecision** - Target field precision expressed as a short

**sTargetScale** - Target field scale expressed as a short

sExpression - A reference parameter that can be returned from the function back to DBMoto. This allows you to apply specific expressions to the mapping. For example:

```
If sTargetType = "VARCHAR" Then
    sExpression.Append("MyGlobalFunction([" + sSourceName + "])")
End If
```

The example above defines an expression that will call the global function MyGlobalFunction for all target columns of type VARCHAR.

## Implementing the Mapping Rule

1. Select the metadata for which you want to create a global script.
2. From the right mouse button menu, select **Global Script**.
3. In the [Global Script Editor](#), insert your mapping rule or rules below the line:

```
VB.NET: Public Class MappingRule : Inherits IMappingRule
```

```
C#: public class MappingRule : IMappingRule
```

4. Write a MappingRuleAttribute definition to determine what appears on the Automatic Mapping menu in the [Set Mapping Info screen of the Replication Wizard](#) or the [Fields Mapping dialog](#). The first string contains the text that will appear as a submenu of the Custom Mapping menu item. The second string contains text that will serve as a tooltip description of the function.
5. Write your function using the examples above as a guideline. The parameters of your function must match those in the examples above.

These parameters can then be used in the function.

6. Compile the global script to see that the function compiles correctly. It will be available in the Automatic Mapping menu only if it compiles correctly.

## Writing a Global Script

A global script is a shared module that defines a global class of functions or defines event handlers for global events. It can be used to define general user functions that need to be called from inside a replication script or an expression script.

Global scripts can use or define the following sets of functions:

- Microsoft.VisualBasic Namespace Functions  
A .NET Framework Class Library that makes available Visual Basic 6 functions such as [Strings] Left, Mid, Right, Instr; [Math] Cos, Arc, Power and so on.

- [DBMoto Global Functions](#)  
Functions implemented as part of DBMoto.
- [DBMoto Global Script Events](#)  
Events that apply to all replications.
- [User Functions](#)  
Functions that users define in the Global Script dialog typically for use from within Replication scripts.
- [Field Mapping Rules or Functions](#)  
Functions used to determine how to map source and target columns when defining a replication.

Use the Global Script Editor to write functions which are then typically used in [replication scripts](#) and [expressions](#). For example, you may have a conversion function that you use in several different replication scripts. Instead of repeating the conversion function for each replication script, it can be defined in a global script and then called from each replication script.

1. In the DBMoto Management Center Metadata Explorer, select the metadata for which you want to write a global script.
2. From the right mouse button menu, choose **Global Script**.

The Global Script Editor opens to display a stub for the script:

```
Imports Microsoft.VisualBasic
Imports DBMotoPublic
Imports DBMotoScript
Imports System
Namespace DBRS
    Public Class GlobalScript : Inherits IGlobalScript
    End Class
    Public Class MappingRule : Inherits IMappingRule
    End Class
    Public Class GlobalEvents : Inherits IGlobalEvents
    End Class
End Namespace
```

3. In the GlobalScript class, declare functions as follows:

```
Public Shared Function ConvertToDate(Param1 as Integer) as Date
'function body
End Function
```



4. Define your functions.

Declare event handlers as in the following example for the Record\_OnExecuteError event. Remember to specify the attribute GlobalEventAttribute.

```
<GlobalEventsAttribute("Record_OnExecuteError", "Define a general event for the event
OnExecuteError")> _
```

```
Public Shared Sub MyErrorHandler (ByVal sReplOrGroupName As String,
                                ByVal recTarget As DBMotoPublic.IRecord,
                                ByVal e As Exception,
                                ByRef bRetryExecute As Boolean,
```

```
ByRef iSleep As Integer,
ByVal iIteration As Integer)
```

5. If you have added any libraries to the list of imports in the script, click  to open the References dialog and add the path to the library.
6. Use the Compile button  to check your script syntax.  
Any script problems are displayed in a separate dialog.
7. Correct any syntax errors before attempting to run a replication.

[Define mapping rules or functions](#) as needed to be used in mapping source and target fields.

## Writing a Replication Script

A script that is defined to run when a specific event occurs during replication. Replication scripts can use:

- Standard Visual Basic .NET functions
- Microsoft.VisualBasic Namespace Functions

A .NET Framework Class Library that makes available Visual Basic 6 functions such as [Strings] Left, Mid, Right, Instr; [Math] Cos, Arc, Power and so on.

- User functions defined in a global script
- Specific replication script [events](#), [methods](#) and [properties](#).

Use the Replication Script Editor to write a script that is run when a specific event occurs during replication. The script should consist of one or more functions selected from the drop-down lists at the top of the Editor.

1. In the DBMoto Management Center Metadata Explorer, select the replication for which you want to write a script.
2. From the right mouse button menu, choose **Replication Properties**.
3. On the General tab, check the **Use Script** option.
4. Click the **Script** button.

The Replication Script Editor opens to display a stub for the script:

```
Imports Microsoft.VisualBasic
Imports DBMotoPublic
Imports DBMotoScript
Imports DBRS.GlobalScript

Namespace DBRS
    Public Class ReplicationScript : Inherits IReplicationScript
    End Class
End Namespace
```

From the left drop-down list, select a category to specify when the script should be run:



Category	Events	Event Type	Explanation
Refresh	<a href="#">onBeforeTruncate</a> <a href="#">onAfterTruncate</a> <a href="#">onBeforeRefresh</a> <a href="#">onAfterRefresh</a> <a href="#">onPrepareRefresh</a>	<a href="#">Writer</a> <a href="#">Writer</a> <a href="#">Writer</a> <a href="#">Writer</a> <a href="#">Reader</a>	Available events apply to refresh replications only.
LogReader	<a href="#">onPrepareMirroring</a> <a href="#">onBeforeMirroring</a> <a href="#">onAfterMirroring</a> <a href="#">onReceiverChanged</a>	<a href="#">Reader</a> <a href="#">Reader</a> <a href="#">Reader</a> <a href="#">Reader</a>	Available events apply only to mirroring and synchronization replications. onReceiverChanged applies only to replications involving System i/iSeries/AS400.
Record	<a href="#">onBeforeMapping</a> <a href="#">onAfterMapping</a> <a href="#">onBeforeExecute</a> <a href="#">onAfterExecute</a>	<a href="#">Reader</a> <a href="#">Reader</a> <a href="#">Writer</a> <a href="#">Writer</a>	Available events apply to each record that is considered for replication. In the case of refresh replications, the events would apply to every record in the source/target table. In the case of mirroring and synchronization replications, the events would apply to records found in the transaction log.
Replication	<a href="#">onCriticalError</a> <a href="#">onConflict</a> <a href="#">onLateConflict</a>	<a href="#">Writer</a> <a href="#">Reader</a> <a href="#">Reader</a>	The onCriticalError event applies to all replication types. It has been replaced by global events <a href="#">Record onExecuteError</a> and <a href="#">Replication onError</a> . onConflict and onLateConflict should be used to <a href="#">resolve conflicts</a> only for synchronization replications.

- From the right drop-down list, select [an event](#) for which you want to write a script.  
This drop-down list displays events appropriate for the function you selected above. If no function was selected, the list remains empty.  
Events are always added directly below the ReplicationScript class as in the example below.

```
Imports Microsoft.VisualBasic
Imports DBMotoPublic
Imports DBMotoScript
Imports DBRS.GlobalScript
```

```
Namespace DBRS
    Public Class ReplicationScript : Inherits IReplicationScript

        Public Overrides Sub Record_onBeforeMapping(recSource As IRecord, ByRef
AbortRecord As Boolean, ByRef DisableReplication As Boolean)


            End Sub
        End Class
    End Namespace
```

6. Type the script in the editor pane.
7. If you want the behavior of your script to depend on the type of SQL operation that is being performed during replication (INSERT, UPDATE or DELETE), be sure to specify the operation type as in the example below.



```
Public Overrides Sub Record_onAfterMapping(recSource As IRecord, recTarget As
IRecord, ByRef AbortRecord As Boolean)
    Select Case recTarget.OperationType
        Case enmOperationType.Insert
            If recTarget.GetValueAfter("SID") = Nothing Then
                recTarget.SetValueAfter("NAME", Nothing)
            Else
                UpdateSTUDENTFields (recTarget)
            End If
        Case enmOperationType.Update
            If recTarget.GetValueAfter("SID") = Nothing Then
                recTarget.SetValueAfter("NAME", Nothing)
            ElseIf recTarget.GetValueBefore("SID") = Nothing Then
                'Field SID was null and now it has a value
                UpdateSTUDENTFields (recTarget)
            ElseIf recTarget.GetValueBefore("SID")
                recTarget.GetValueAfter("SID") Then
                'Field SID has changed
                UpdateSTUDENTFields (recTarget)
            End If
        End Select
    End Sub
```

8. Use [IRecord methods](#) to access values in the source and target records. Be sure to use only those methods which make sense for the operation type (INSERT, UPDATE or DELETE).



9. Typically, only those columns that have been mapped for replication (during replication setup) are available for use within scripts. However, if you need to access a column value that is not mapped, you can set that column as "Use Unmapped" in the [Replication Properties dialog](#).
10. Be aware of script objects that need to be accessed by more than one event and be prepared to lock an object that is accessed by both [Reader script events](#) and [Writer script events](#). [Example](#).
11. Use the [Expression Generator](#)  as needed to identify functions and syntax that you can use in the script.

If you have defined functions in the [Global Script Editor](#), the function names and prototypes will be listed under User Functions in the Expression Generator.

12. If you have added any libraries to the list of imports in the script, click  to open the References dialog and add the path to the library.
13. Use the Compile button  to check your script syntax.  
Any script problems are displayed in a separate dialog.
14. Correct any syntax errors before attempting to run a replication.

## Writing an Expression Script

An expression script is used in the [Fields Mapping Dialog](#) when a column value needs to be mapped not simply to a source column but to a value retrieved after calculation. Alternatively, [Mapping functions, or rules](#), can be used to define a rule which applies to all source-target pairs which meet the conditions specified in the rule.

Expression scripts can use:

- User functions defined in a global script
- All Microsoft.VisualBasic (.NET Framework Class Library) functions, as listed in the Expression Editor dialog.

To set up a mapping to an expression:

1. In the [Replication Properties dialog](#), or in the Replication wizard, open the [Fields Mapping dialog](#).
2. Select a target field in the list and choose the **Map to Expression** option from the right mouse button menu.
3. In the [Expression Generator](#), type your expression or build it by expanding the tree to select items.  
Functions defined in a global script are available from the User Functions node in the tree.
4. To refer to source fields in an expression script, indicate the field between square brackets, with no quotes. For example, to refer to the field "EmployeeID" in the source table, use the syntax [EmployeeID].
5. Click **OK** to close the Expression Generator and complete your expression.

## Writing a Function to use in Scripts and Expressions

Even though you can use the full range of C# or VB .NET syntax when writing a DBMoto script, there are times when you need to write a custom function to use in scripts or expressions. You can use the [Global Script Editor](#) to write and compile the function, then call it from a replication script or expression. Here is an example in VB.NET of how you might write and use a function that trims values and accepts null values:

```
Public Shared Function SafeTrim(objString as Object) as Object

    'If the parameter is null return a null value
    If objString Is Nothing Then
        Return System.DBNull.Value
    Return Trim(objString.ToString())
End Function
```

1. In the DBMoto Management Center Metadata Explorer, select the metadata where you want to use the function.
2. From the right mouse button menu, choose **Global Script**.

The Global Script Editor opens to display a stub:

```
Imports Microsoft.VisualBasic
Imports DBMotoPublic
Imports DBMotoScript
Namespace DBRS
    Public Class GlobalScript : Inherits IGlobalScript
    End Class
    Public Class MappingRule : Inherits IMappingRule
    End Class
    Public Class GlobalEvents : Inherits IGlobalEvents

    End Class
End Namespace
```

3. Type your function between the "Public Class GlobalScript : Inherits IGlobalScript" line and the corresponding "End Class" line so that it looks as follows:



```
Imports Microsoft.VisualBasic
Imports DBMotoPublic
Imports DBMotoScript
Namespace DBRS
    Public Class GlobalScript : Inherits IGlobalScript

        Public Shared Function SafeTrim(objString as Object) as Object
            'If the parameter is null return a null value
            If objString Is Nothing Then Return System.DBNull.Value
            Return Trim(objString.ToString())
        End Function
    End Class
    Public Class MappingRule : Inherits IMappingRule
    End Class
```

```
Public Class GlobalEvents : Inherits IGlobalEvents
```

```
End Class
```

```
End Namespace
```

4. If you have added any libraries to the list of imports in the script, click  to open the References dialog and add the path to the library.
5. Click **Compile**  to check your script syntax.  
Any script problems are displayed in a separate dialog.
6. Correct any syntax errors.

Now you can use the function from a replication script or expression. The steps below show you how to use a function in an expression.

1. If you are using the function in an existing replication, either stop the Data Replicator or disable the replication.
2. For an existing replication, right-click on the replication and choose **Replication Properties**, then click **Mapping** to display the Mapping Editor. For new replications, perform the following steps when you get to the mapping portion of replication creation.
3. In the left hand pane of the Mapping Editor, right-click on each field that will be used as a parameter for the function and make sure **Use Unmapped** is checked. If it is not checked, select the menu item to check it.
4. Right-click on the target field that will be using the generated value and choose **Map to Expression**.
5. In the Expression Generator window, expand the User Functions folder and click on (all) to see if your function is listed.

If the function is not displayed, check to make sure that you have used the proper Public Shared qualifier for the function. See step 3 above for more information.

6. Double-click on the function name to use it in the Expression Editor.
7. Either type in the names of the fields you want to use in the function or expand Values, then Fields and double click on each field name to add it to the Expression Editor. Note that field names should be enclosed in square brackets [] as in the example below. In this case, the value of the field MyText is being passed to the function SafeTrim (defined in step 3 above).

```
SafeTrim([MyText])
```

8. Click **OK** to close the Expression Generator window.
9. Click **OK** to close the Mapping window.
10. Click **OK** to close the Replication Properties window.
11. Either start the Data Replicator or enable the replication to test that your function is working.

## Handling Events for INSERT, UPDATE and DELETE Operations

When writing an event-handling function for a replication, you often need to consider which type of operation (INSERT, UPDATE, DELETE) is being performed during the replication. In your [replication script](#), use the record's [OperationType](#) property to identify the type of operation being performed, as in the following VB.NET example.

```
Public Overrides Sub Record_onAfterMapping(recSource As IRecord, recTarget As IRecord, By
Ref AbortRecord As Boolean)
    if (recSource.OperationType = enmOperationType.Update)
        AddLog(recSource.GetValueBefore(0), 0)
    End if
End Sub
```

Depending on the operation type, different functions for accessing source and target values are available.

### INSERT Operations

**Replication Types:** Refresh (all records) and mirroring/synchronization (insert operations obtained from the transaction log)

**Record Functions Available:**

Source Record	Target Record
<a href="#">GetValueAfter(index)</a>	<a href="#">GetValueAfter(index)</a>
<a href="#">GetValueAfter(col)</a>	<a href="#">GetValueAfter(col)</a>
<a href="#">SetValueAfter(index, value)</a>	<a href="#">SetValueAfter(index, value)</a>
<a href="#">SetValueAfter(col, value)</a>	<a href="#">SetValueAfter(col, value)</a>

**Note:** Depending on the event function you are using, source or target record objects may not be available. Check the parameters for the event to see which objects you can use.

### UPDATE Operations

**Replication Types:** Mirroring/synchronization (update operations obtained from the transaction log)

**Record Functions Available:**

Source Record	Target Record
<a href="#">GetValueBefore(index)*</a>	<a href="#">GetValueBefore(index)*</a>
<a href="#">GetValueBefore(col)*</a>	<a href="#">GetValueBefore(col)*</a>

<a href="#">SetValueBefore(col, value)**</a>	<a href="#">SetValueBefore(col, value)**</a>
<a href="#">SetValueBefore(index, value)**</a>	<a href="#">SetValueBefore(index, value)**</a>
<a href="#">GetValueAfter(index)</a>	<a href="#">GetValueAfter(index)</a>
<a href="#">GetValueAfter(col)</a>	<a href="#">GetValueAfter(col)</a>
<a href="#">SetValueAfter(index, value)</a>	<a href="#">SetValueAfter(index, value)</a>
<a href="#">SetValueAfter(col, value)</a>	<a href="#">SetValueAfter(col, value)</a>

\* Values before an operation are not always available in the log and therefore these functions may return Null.

\*\* These functions are rarely used.

**Note:** Depending on the event function you are using, source or target record objects may not be available. Check the parameters for the event to see which objects you can use.

## DELETE Operations

**Replication Types:** Mirroring/synchronization (delete operations obtained from the transaction log)

**Record Functions Available:**

Source Record	Target Record
<a href="#">GetValueBefore(index)*</a>	<a href="#">GetValueBefore(index)*</a>
<a href="#">GetValueBefore(col)*</a>	<a href="#">GetValueBefore(col)*</a>
<a href="#">SetValueBefore(col, value)**</a>	<a href="#">SetValueBefore(col, value)**</a>
<a href="#">SetValueBefore(index, value)**</a>	<a href="#">SetValueBefore(index, value)**</a>

\* Values before an operation are not always available in the log and therefore these functions may return Null.

\*\* These functions are rarely used.

**Note:** Depending on the event function you are using, source or target record objects may not be available. Check the parameters for the event to see which objects you can use.

## Global Script Functions (IGlobalScript Class)

The DBMoto global functions for use in [Global scripts](#) are described below. If using these functions from a global script event, call them using the IGlobalScript class (e.g., `IGlobalScript.AddLog`).

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.

[AddLog](#)

[SendMail](#)

[GetRecordInfo](#)

[GetJSONRecordInfo](#)

[GetReceiversInUse](#)

## AddLog (String, enmLogMessageType)

Note: this function supersedes AddLog(str As String, eType As Integer)

Writes a log message in the DBMoto.log file.

### VB.NET syntax

```
Public Shared Sub AddLog (str As String, eType As enmLogMessageType)
```

### C# syntax

```
public static void AddLog(string str, enmLogMessageType eType)
```

### Parameters

- str = message to log
- eType = enumerator enmLogMessageType with the possible values: Information = 0, Warning = 1, Error = 2

### Example [VB.NET syntax]

```
AddLog ("The current record has been inserted", 0)
```

### Example [C# syntax]

```
AddLog("The current record has been inserted", 0)
```

## AddLog (String, enmLogMessageType, Record, enmRecordImage)

Note: this function supersedes AddLog(String, Int, Record, enmRecordImage)

Writes a log message that includes the specified record information in the DBMoto.log file.

### VB.NET Syntax

```
Public Shared Sub AddLog (str As String, eType As enmLogMessageType, record As  
DBMotoPublic.IRecord, eRecordImage As enmRecordImage)
```

### C# Syntax

```
public static void AddLog(string str, enmLogMessageType eType, DBMotoPublic.IRecord  
record, enmRecordImage eRecordImage)
```

### Parameters

- str = message to log
- eType = enumerator enmLogMessageType with the possible values: Information = 0, Warning = 1, Error = 2

- record = IRecord object to be logged along with the message
- eRecordImage = enumerator enmRecordImage for type of record print required. The record image uses the Values After of the record if it comes from an INSERT operation, otherwise the Values Before. The parameter can have the value 1, 2 or 4:
  - 1 = KeyValues - Print only the primary key columns values
  - 2 = Image - Print the whole image of the record
  - 4 = LogInfo - Print the log information (Transaction ID, Transaction Timestamp)

#### Example [VB.NET syntax]

```
Public Overrides Sub Record_onAfterMapping(recSource As IRecord, recTarget As IRecord, ByRef AbortRecord As Boolean, ByRef DisableReplication As Boolean)
    Dim S as String
    AddLog("The current record has been inserted", 0, recSource, 4)
End Sub
```

#### SendMail (subject, body)

Sends an email using the SMTP settings defined in the [Mail tab](#) of the [Data Replicator Options dialog](#).

#### VB.NET syntax

```
Public Shared Sub SendMail (sSubject As String, sMessageBody As String)
```

#### C# syntax

```
public static void SendMail(string sSubject, string sMessageBody)
```

#### Parameters

- sSubject = subject of the email
- sMessageBody = body of the email

#### Example [VB.NET syntax]

```
SendMail ("Message from DBMoto", "An error occurred that requires intervention by the system administrator.")
```

#### Example [C# syntax]

```
SendMail("Message from DBMoto", "An error occurred that requires intervention by the system administrator.")
```

#### SendMail (subject, body, to)

Sends an email using the SMTP settings defined in the [Mail tab](#) of the [Data Replicator Options dialog](#).

#### VB.NET syntax

```
Public Shared Sub SendMail (sSubject As String, sMessageBody As String, sRcptTo As String)
```

## C# syntax

```
public static void SendMail(string sSubject, string sMessageBody, string sRcptTo)
```

## Parameters

- sSubject = subject of the email
- sMessageBody = body of the email
- sRcptTo = the email address of the intended recipient

## Example [VB.NET syntax]

```
SendMail ("Message from DBMoto", "An error occurred that requires intervention by the  
system administrator.", "somebody@company.com")
```

## Example [C# syntax]

```
SendMail("Message from DBMoto", "An error occurred that requires intervention by the  
system administrator.", "somebody@company.com")
```

## SendMail (sSubject, sMessageBody, sSMTPServer, sSMTPPort, sRcptFrom, sRcptTo, bAuthentication, sUser, sPassword, bUseSSL)

Sends an email using the parameters defined when the function is called. When parameters are not passed, the Mail settings in the [Data Replicator Options dialog](#) will be used instead.

## VB.NET syntax

```
Public Shared Sub SendMail (sSubject As String, sMessageBody As String, sSMTPServer As  
String, sRcptFrom As String, sRcptTo As String)
```

## C# syntax

```
public static void SendMail(string sSubject, string sMessageBody, string sSMTPServer,  
string sRcptFrom, string sRcptTo)
```

## Parameters

- sSubject = subject of the email
- sMessageBody = body of the email
- sSMTPServer = the SMTP server for outgoing mail
- sSMTPPort = the SMTP server port for outgoing mail
- sRcptFrom = the email address from which you want to send the message
- sRcptTo = the email address of the intended recipient.
- bAuthentication = True if authentication should be used, otherwise False
- sUser = If bAuthentication is True, provide a user ID.
- sPassword = If bAuthentication is True, provide a password



- bUseSSL = True if SSL (Secure Sockets Layer) for email encryption should be used, otherwise False.

#### Example [VB.NET syntax]

```
SendMail ("Message from DBMoto", "An error occurred that requires intervention by the
system administrator.", "www.smtp.com", "DBMoto Notification Agent",
"somebody@company.com")
```

#### Example [C# syntax]

```
SendMail("Message from DBMoto", "An error occurred that requires intervention by the
system administrator.", "www.smtp.com", "DBMoto Notification Agent",
"somebody@company.com")
```

### GetRecordInfo(Record, enmRecordImage)

Returns a string containing formatted information about the record.

#### VB.NET syntax

```
Public Shared Function GetRecordInfo (record As DBMotoPublic.IRecord, eRecordImage As
enmRecordImage) As String
```

#### C# syntax

```
public static string GetRecordInfo(DBMotoPublic.IRecord record, enmRecordImage
eRecordImage)
```

#### Parameters

- record = IRecord object to be logged along with the message
- eRecordImage = enumerator enmRecordImage for type of record print required. The record image uses the Values After of the record if it comes from an INSERT operation, otherwise the Values Before. The parameter can have the value 1, 2 or 4:

1 = KeyValues - Print only the primary key columns values

2 = Image - Print the whole image of the record

4 = LogInfo - Print the log information (Transaction ID, Transaction Timestamp)

#### Example [VB.NET syntax]

```
Imports Microsoft.VisualBasic
Imports DBMotoPublic
Imports DBMotoScript
Imports DBRS.GlobalScript
Namespace DBRS
    Public Class ReplicationScript : Inherits IReplicationScript
        Public Overrides Sub Record_onAfterMapping(recSource As IRecord, recTarget As
IRecord,
                                                    ByRef AbortRecord As Boolean,
                                                    ByRef DisableReplication As Boolean)
            Dim S as String = GetRecordInfo(recSource, enmRecordImage.KeyValues)
            AddLog("The current record has been inserted: " + S, 0)
        End Sub
    End Class
End Namespace
```

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.

```
End Class
End Namespace
```

### Example [C# syntax]

```
using System;
using System.Data;
using DBMotoPublic;
using DBMotoScript;
namespace DBRS
{
    public class ReplicationScript : IReplicationScript
    {
        public override void Record_onAfterMapping(DBMotoPublic.IRecord recSource,
DBMotoPublic.IRecord recTarget, ref bool AbortRecord, ref bool DisableReplication)
        {
            string s = GlobalScript.GetRecordInfo(recSource, enmRecordImage.KeyValues);
            GlobalScript.AddLog("The current record has been inserted: " + s,
0);
        }
    }
}
```

### GetJSONRecordInfo

All four variants of this function below are helper functions that can be useful when mapping a generic structured or unstructured data into a JSON column type.

#### GetJSONRecordInfo(record,eRecordImage)

Returns a string in JSON format. For example:

```
{
  "FIELD1" = "Value1",
  "FIELD2" = "Value2",
  "FIELD3" = "Value3"
}
```

Object names and values are picked up from the IRecord object (and use the fields based on the enmRecordImage parameter).

#### VB.NET syntax

```
Public Shared Function GetJSONRecordInfo(ByVal record As DBMotoPublic.IRecord,
ByVal eRecordImage As enmRecordImage) As String
```

#### C# syntax

```
public static string GetJSONRecordInfo(DBMotoPublic.IRecord record,
enmRecordImage eRecordImage)
```

## GetJSONRecordInfo(args)

Returns a string in JSON format. For example:

```
{  
  "FIELD1" = "Value1",  
  "FIELD2" = "Value2",  
  "FIELD3" = "Value3"  
}
```

Object names and values are picked from a list of generic <propname1, propvalue1, propname2, propvalue2, ...>.

### VB.NET syntax

```
Public Shared Function GetJSONRecordInfo(ParamArray ByVal args() As Object) As  
String
```

### C# syntax

```
public static string GetJSONRecordInfo(params object[] args)
```

## GetJSONRecordInfo(T value)

Returns the JSON format for a specific object of type T. This function can be used when the value passed is not a string value but something else (such as a DateTime, a GUID, etc.)

### VB.NET syntax

```
Public Shared Function GetJSONRecordInfo(Of T)(ByVal value As T) As String
```

### C# syntax

```
public static string GetJSONRecordInfo<T>(T value)
```

## GetJSONRecordInfo T (jsonString)

The inverse of the function above. Given a certain value in JSON string format, it parses back the string and returns the object (of type T) from which it was generated.

### VB.NET syntax

```
Public Shared Function GetJSONRecordInfo(Of T)(ByVal jsonString As String) As T
```

### C# syntax

```
public static T GetJSONRecordInfo<T>(string jsonString)
```

## GetReceiversInUse

Returns a list of receivers in use by the replications in a DBMoto connection as a ReceiversInfo object. The definition of the ReceiverInfo class is given below. For more information on using this function, see [Writing a Script to Determine Receiver Use](#).

## VB.NET syntax

```
Public Shared Function GetReceiversInUse (ConnectionName As String, IsSource As Boolean,  
OnlyActiveReplications As Boolean) As ReceiverInfo()
```

## C# syntax

```
public static ReceiverInfo[] GetReceiversInUse(string ConnectionName, bool IsSource, bool  
OnlyActiveReplications)
```

## Parameters

- ConnectionName = Name of the IBM Db2 for i connection for which to retrieve the receivers in use
- IsSource = True if ConnectionName refers to a source connection, false otherwise
- OnlyActiveReplications = If true, retrieve only receivers used by active (Enabled) replications, and skip the disabled ones

## Example [VB.NET syntax]

```
Imports Microsoft.VisualBasic  
Imports DBMotoPublic  
Imports DBMotoScript  
Imports DBRS.GlobalScript  
Imports System.Data  
Namespace DBRS  
    Public Class ReplicationScript : Inherits IReplicationScript  
        Public Overrides Sub LogReader_onBeforeMirroring(bSource as Boolean)  
            Dim arrReceivers As ReceiverInfo() = GetReceiversInUse ("AS400", True, True)  
            Dim recInfo As ReceiverInfo  
            For Each recInfo in arrReceivers  
                AddLog("Receiver in use: " + recInfo.JournalLibrary + "." +  
recInfo.JournalName + "/"  
                + recInfo.ReceiverLibrary + "." + recInfo.ReceiverName, 0)  
            Next  
        End Sub  
    End Class  
End Namespace
```

### Example [C# syntax]

```
using System;
using System.Data;
using DBMotoPublic;
using DBMotoScript;
namespace DBRS
{
    public class ReplicationScript : IReplicationScript
    {
        public override void LogReader_onBeforeMirroring(bool bSource)
        {
            ReceiverInfo[] arrReceivers = GlobalScript.GetReceiversInUse("AS400", true,
true);
            foreach (ReceiverInfo recInfo in arrReceivers)
            {
                GlobalScript.AddLog("Receiver in use: " + recInfo.JournalLibrary + "." +
recInfo.JournalName + "/" + recInfo.ReceiverLibrary + "." + recInfo.ReceiverName, 0);
            }
        }
    }
}
```

### Other

This function returns a value of type ReceiverInfo. The definition for the class is below.

```
Public Class ReceiverInfo
    Public JournalLibrary As String
    Public JournalName As String
    Public ReceiverLibrary As String
    Public ReceiverName As String
End Class
```

## Global Script Events

Define global event handlers in the GlobalEvents class in the Global Script. Each global event must have an attribute section called 'GlobalEventsAttribute' specifying the internal event that the function is handling.

### GlobalEventsAttribute

An attribute with two parameters, the event name and a help string. This attribute must always be declared when defining an error handler for the [Record\\_OnExecuteError](#) and [Replication\\_OnError](#) events.

Parameter 1: Required. A string containing the event name. The value must match the name of the event being handled.

Parameter 2: Required. A string containing a description or help message for the event handler. This string is not currently used by DBMoto, but because the parameter is required, you must at least provide an empty string "".

### Record\_OnExecuteError

This event occurs whenever an error on execution of a single record occurs. As a [writer event](#), it occurs on target operations, such as INSERT, DELETE, UPDATE. Use the bRetryExecute option to force an attempt to rerun an execute

operation on a record when the operation initially fails. For instance, if a single record insert fails with a timeout error, setting `bRetryExecute` to true and setting values for `iSleep` and `iIteration` parameters, might allow the operation to succeed a few milliseconds later. Note that the replication event [Record\\_OnBeforeExecute](#) is generated only once, no matter how many times the record operation is retried using this parameter. Also the replication event [Record\\_OnAfterExecute](#) is generated once only if the record operation succeeds and not at all if the record operation fails.

The [GlobalEventsAttribute](#) first parameter must indicate the event name "Record\_OnExecuteError".

### VB.NET Syntax

```
<GlobalEventsAttribute("Record_OnExecuteError", "Define a general event for the event
OnExecuteError")>
Public Shared Sub MyErrorHandler (ByVal sReplOrGroupName As String,
                                ByVal recTarget As DBMotoPublic.IRecord,
                                ByVal e As Exception,
                                ByRef bRecoverReplication As Boolean,
                                ByRef bAbortRecord As Boolean,
                                ByRef bDisableReplication As Boolean,
                                ByRef bRetryExecute As Boolean,
                                ByRef iSleep As Integer,
                                ByVal iIteration As Integer)
```

**Note:** Function name 'MyErrorHandler' can be modified.

### C# Syntax

```
[GlobalEventsAttribute("Record_OnExecuteError", "Define a general event for the event
OnExecuteError")]

public static void MyErrorHandler(string sReplOrGroupName, DBMotoPublic.IRecord
recTarget, Exception e, ref bool bRecoverReplication, ref bool bAbortRecord, ref bool
bDisableReplication, ref bool bRetryExecute, ref int iSleep, int iIteration)
```

### Parameters

Name	Type	Description
sReplOrGroupName	String	Read only. The name of the replication or group that generated the event.
recTarget	<a href="#">IRecord</a>	Read only. The object that represents the target record for which the error was generated. You can access specific information in the record using the methods available in the <a href="#">IRecord interface</a> .
e	Exception	Read only. The VB .NET Exception object for this error.
bRecoverReplication	Boolean	The default is False. If set to True, the replication is set to recovery mode

bAbortRecord	Boolean	The default is false. If set to True, the record being processed is not replicated
bDisableReplication	Boolean	The default is False. If set to True, regardless of the RetryExecute value, the replication exits signaling error message in the log and disabling the replication.
bRetryExecute	Boolean	The default value is False. Set this parameter to True if you want to try the target record operation again before recording the error in the DBMoto log. For instance, if a single record insert fails with a timeout error, setting bRetryExecute to True and setting a value for the iSleep parameter might allow the operation to succeed a few milliseconds later. The example below demonstrates how to use this parameter. Note that the replication event <a href="#">Record_OnBeforeExecute</a> is generated only once, no matter how many times the record operation is retried using this parameter. Also the replication event <a href="#">Record_OnAfterExecute</a> is generated once only if the record operation succeeds and not at all if the record operation fails.
iSleep	Integer	When bRetryExecute is set to True, use this parameter to indicate a delay value before retrying the operation. The value should be in milliseconds. The default value is 0. The example below demonstrates how to use this parameter.
iIteration	Integer	Read only. For use in conjunction with the bRetryExecute parameter. iIteration keeps track of the number of times the target record operation is attempted. The start value is 0. When the target record operation is attempted the first time, this value is set to 1. The event handler will continue to retry the operation until bRetryExecute is set to false. The example below demonstrates how you might avoid an infinite loop on an error.

### VB.NET Example

```
<GlobalEventsAttribute("Record_OnExecuteError", "Define a general event for the
event EventName")> _
Public Shared Sub MyErrorHandler (ByVal sReplOrGroupName As String,
    ByVal recTarget As DBMotoPublic.IRecord,
    ByVal e As Exception,
    ByRef bRecoverReplication As Boolean,
    ByRef bAbortRecord As Boolean,
    ByRef bDisableReplication As Boolean,
```

```

        ByRef bRetryExecute As Boolean,
        ByRef iSleep As Integer,
        ByVal iIteration As Integer)
Dim s As String

    s = s + "-- Called MyErrorHandler to catch the error in replication or group
'" + sReplOrGroupName + "'" + Environment.NewLine
    s = s + "-- Exception: " + e.ToString() + Environment.NewLine
    s = s + "-- Iteration: " + iIteration.ToString()
    bRetryExecute = False

    ' Disable replication if the error is a SQL Server timeout error
    If e.Message.IndexOf("System.Data.SqlClient.SqlException: Timeout expired.")
>= 0 Then
        bAbortRecord = True
        bDisableReplication = True
        s = s + " - Replication disabled"
    Else ' retry
        If iIteration >= 3 Then
            bRetryExecute = False
            s = s + " - No Retry"
        Else
            bRetryExecute = True
            iSleep = 3000
            s = s + " - Try Again after " + (iSleep/1000).ToString() + " seconds."
        End If
    End If

    IGlobalScript.AddLog (s, 1)
End Sub

```

### C# Example

```

public class GlobalEvents : IGlobalEvents
{
    /// <param name="sReplOrGroupName"></param>
    /// <param name="recTarget"></param>
    /// <param name="e"></param>
    /// <param name="bRecoverReplication"></param>
    /// <param name="bAbortRecord"></param>
    /// <param name="bDisableReplication"></param>
    /// <param name="bRetryExecute"></param>
    /// <param name="iSleep"></param>
    /// <param name="iIteration"></param>

    [GlobalEventsAttribute("Record_OnExecuteError", "Define a general event for
the event EventName")]

    public static void MyErrorHandler(String sReplOrGroupName,
DBMotoPublic.IRecord recTarget, Exception e, ref bool bRecoverReplication, ref
bool bAbortRecord, ref bool bDisableReplication, sref Boolean bRetryExecute, ref

```



```

int iSleep, int iIteration)
{
    String s = null;
    s = s + "-- Called MyErrorHandler to catch the error in replication or
group '" + sReplOrGroupName + "'" + Environment.NewLine;
    s = s + "-- Exception: " + e.ToString() + Environment.NewLine;
    s = s + "-- Iteration: " + iIteration.ToString();
    bRetryExecute = false;
    // Disable replication if the error is a SQL Server timeout error
    if (e.Message.IndexOf("System.Data.SqlClient.SqlException: Timeout expired.") >=
0) {
        bAbortRecord = true;
        bDisableReplication = true;
        s = s + " - Replication disabled";
    } else { // retry
        if (iIteration >= 3) {
            bRetryExecute = false;
            s = s + " - No Retry";
        } else {
            bRetryExecute = true;
            iSleep = 3000;
            s = s + " - Try Again after " + (iSleep / 1000).ToString() + "
seconds.";
        }
    }
    IGlobalScript.AddLog(s, 1);
}

```

## Replication\_OnError

This event occurs whenever an error on a replication occurs. Any type of error that is shown in the DBMoto Replication Monitor as an error or warning icon at the end of the replication operation generates this event. This event is invoked at the end of a refresh or at the end of the mirroring interval, and not at the moment the error happens.

The [GlobalEventsAttribute](#) first parameter must indicate the event name "Replication\_OnError".

### VB.NET Syntax

```
<GlobalEventsAttribute("Replication_OnError", "Define a general event for the event
Replication_OnError")>
```

```
Public Shared Sub MyErrorHandler (ByVal sReplOrGroupName As String, ByVal sMessage As
String,ByVal bWillDisable As Boolean, ByVal eReplStatus As enmReplStatus)
```

**Note:** Function name 'MyErrorHandler' can be modified.

### C# Syntax

```
[GlobalEventsAttribute("Record_OnExecuteError", "Define a general event for the event Replication_OnError")]
```

```
public static void MyErrorHandler(string sReplOrGroupName, string sMessage, bool bWillDisable, enmReplStatus eReplStatus)
```

### Parameters

Name	Type	Description
sReplOrGroupName	String	Read only. The name of the replication or group that generated the event.
sMessage	String	Read only. The error message/exception stack.
bWillDisable	Boolean	Read only. True if the error will disable the replication.
eReplStatus	enmReplStatus	<p>Read only. Specifies the replication status when the event is called using the enmReplStatus enumerator. Possible values for the enumerator are:</p> <ul style="list-style-type: none"> <li> <b>ErrorAndContinue</b>            An error was generated during the replication (refresh or mirroring interval) but the replication was able to continue. An example is when a transaction fails to delete a target record because the record was not found in the target table. In this case, DBMoto reports the error in the log and continues with the next record.         </li> <li> <b>Error</b>            An error was generated and the replication stopped executing. The replication will start again at the next interval. An example of this type of error is when a connection to the source or target database is broken. In such cases, DBMoto cannot continue with the replication and goes into recovery mode.         </li> <li> <b>Success</b> </li> </ul>

		<p>Not used in this event. When Replication_OnError is called, the replication is always in an error state</p>
--	--	--

### VB.NET Example

```
<GlobalEventsAttribute("Replication_OnError", "General event for the event OnError")> _
Public Shared Sub MyGlobalEvent (ByVal sReplicationName As String, ByVal sMessage As
String,ByVal bWillBeDisabled As Boolean, ByVal eReplStatus As enmReplStatus)
    Dim s As String
    s = DateTime.Now.ToString() + Environment.NewLine + Environment.NewLine
    s = s + "Error in replication '" + sReplicationName + "'" + Environment.NewLine +
Environment.NewLine
    s = s + "Replication Status: " + eReplStatus.ToString() + Environment.NewLine +
Environment.NewLine
    s = s + sMessage + Environment.NewLine + Environment.NewLine
    If bWillBeDisabled Then
        s = s + "The replication will be disabled." + Environment.NewLine +
Environment.NewLine
    End If
    s = s + "-----" + Environment.NewLine
    s = s + "This is an automatic message generated by the DBMoto script" +
Environment.NewLine
    s = s + "-----" + Environment.NewLine
    IGlobalScript.SendMail ("DBMoto Error", s)
End Sub
```

### C# Example

```
/// <param name="sReplicationName"></param>
/// <param name="sMessage"></param>
/// <param name="bWillBeDisabled"></param>
/// <param name="eReplStatus"></param>
[GlobalEventsAttribute("Replication_OnError", "General event for the event OnError")]
public void MyGlobalEvent(String sReplicationName, String sMessage, Boolean
bWillBeDisabled, enmReplStatus eReplStatus)
{
    String s = null;
    s = DateTime.Now.ToString() + Environment.NewLine + Environment.NewLine;
    s = s + "Error in replication '" + sReplicationName + "'" + Environment.NewLine +
Environment.NewLine;
    s = s + "Replication Status: " + eReplStatus.ToString() + Environment.NewLine +
Environment.NewLine;
    s = s + sMessage + Environment.NewLine + Environment.NewLine;
    if (bWillBeDisabled)
    {
        s = s + "The replication will be disabled." + Environment.NewLine +
Environment.NewLine;
    }
    s = s + "-----" + Environment.NewLine;
```

```

s = s + " This automatic message is generated by the DBMoto script" +
Environment.NewLine;
s = s + "-----" + Environment.NewLine;
IGlobalScript.SendMail("DBMoto Error", s);
}

```

## ReplicationManager\_onStart

This event occurs when the Data Replicator is started. It can be used to add a notification message to the DBMoto log, or to send email to an administrator. To write a handler for the event, position the cursor inside the class header "Public Class GlobalEvents : Inherits IGlobalEvents", then define the handler using the example below as guideline.

The [GlobalEventsAttribute](#) first parameter must indicate the event name "ReplicationManager\_OnStart".

### VB.NET Syntax

```

<GlobalEventsAttribute("ReplicationManager_OnStart", "Standard event for the event
OnStart")> _
Public Shared Sub OnStart ()

```

### C# Syntax

```

[GlobalEventsAttribute("ReplicationManager_OnStart", "Standard event for the event
OnStart")]
public static void OnStart()

```

### Parameters

None.

### VB.NET Example

```

<GlobalEventsAttribute("ReplicationManager_OnStart", "Standard event for the event
OnStart")> _
Public Shared Sub OnStart ()
    Dim s As String
    s = s + "-- Message from DBMoto - START"
    s = s + "-----" + Environment.NewLine
    s = s + "- This is an automatic message generated by the DBMoto script -" +
Environment.NewLine
    s = s + "-----" + Environment.NewLine
    IGlobalScript.SendMail ("DBMoto Replication Manager Message", s)
End Sub

```

### C# Example

```

[GlobalEventsAttribute("ReplicationManager_OnStart", "Standard event for the event
OnStart")]
public void OnStart()
{
    String s = null;
    s = s + "-- Message from DBMoto - START";
    s = s + "-----" + Environment.NewLine;
    s = s + "- This is an automatic message generated by the DBMoto script -" +
Environment.NewLine;
}

```

```
s = s + "-----" + Environment.NewLine;
IGlobalScript.SendMail("DBMoto Replication Manager Message", s);
}
```

## ReplicationManager\_onStop

This event occurs when the Data Replicator is started. It can be used to add a notification message to the DBMoto log, or to send email to an administrator. To write a handler for the event, position the cursor inside the class header "Public Class GlobalEvents : Inherits IGlobalEvents", then define the handler using the example below as a guideline.

The [GlobalEventsAttribute](#) first parameter must indicate the event name "ReplicationManager\_OnStop".

### VB.NET Syntax

```
<GlobalEventsAttribute("ReplicationManager_OnStop", "Standard event for the event
OnStop")> _
Public Shared Sub OnStop ()
```

### C# Syntax

```
[GlobalEventsAttribute("ReplicationManager_OnStop", "Standard event for the event
OnStop")]
public static void OnStop()
```

### Parameters

None.

### VB.NET Example

```
<GlobalEventsAttribute("ReplicationManager_OnStop", "Standard event for the event
OnStop")> _
Public Shared Sub OnStop ()
    Dim s As String
    s = s + "-- Message from DBMoto - STOP"
    s = s + "-----" +
Environment.NewLine
    s = s + "- This is an automatic message generated by the DBMoto script -" +
Environment.NewLine
    s = s + "-----" +
Environment.NewLine
    IGlobalScript.SendMail ("DBMoto Replication Manager Message", s)
End Sub
```

### C# Example

```
[GlobalEventsAttribute("ReplicationManager_OnStop", "Standard event for the event
OnStop")]
public void OnStop()
{
    String s = null;
    s = s + "-- Message from DBMoto - STOP";
    s = s + "-----" +
Environment.NewLine;
    s = s + "- This is an automatic message generated by the DBMoto script -" +
Environment.NewLine;
```

```

    s = s + "-----" +
Environment.NewLine;
    IGlobalScript.SendMail("DBMoto Replication Manager Message", s);
}

```

## VB.NET Global Script Function Examples

The following VB.NET example functions can be defined in a global script, then used in replication scripts and expressions to modify target field values during replication. For detailed steps on how to define a Global Script function, check [Writing a Function to Use in Scripts and Expressions](#).

### Accessing Values Using a Different Database Connection

If your replication needs record values which are accessible only by opening a separate database connection and running a query there, you can write the following script in the Global Script Editor, then call the script as indicated below.

The GetQueryValue function retrieves the first returned value from the passed in connection and query. If the parameters are illegal, the query returns an error or the query does not return a value, the function returns System.DBNull.Value.

```

Public Shared Function GetQueryValue(objConnection as Object, strQuery as String) as Object
    'If a parameter is null, return a null value
    If objConnection Is Nothing OR strQuery="" Then Return System.DBNull.Value
    'Declare variables that might be used
    Dim cmd As IDbCommand = Nothing
    Dim reader As IDataReader = Nothing
    Dim objValue as Object = System.DBNull.Value
    'Use the Try block to make sure we capture any exceptions that might be generated
    Try
        'Use a SyncLock to make sure this function is the only user of the connection since
        'the source or target connection might be in use.
        SyncLock(objConnection)
            'Create and execute the select to get the specified value
            cmd = objConnection.CreateCommand
            cmd.CommandText = strQuery
            reader = cmd.ExecuteReader
            'If we can read from the reader, the record is already there. Get
            'the information
            If reader.Read Then objValue = reader.GetValue(0)
        End SyncLock
        'If there are no exceptions so far, return the value
        reader.Close
        reader = Nothing
        cmd.Dispose
        Return objValue
    'Output exceptions to the DBMoto log
    Catch ex As Exception
        AddLog("Exception in GetQueryValue: " & ex.ToString,0)
    Finally
        if (not reader is Nothing) Then
            reader.Close
        end if
    End Try
End Function

```

```

    End If
    if (not cmd is Nothing) Then
        cmd.Dispose
    End If
End Try
'Return System.DBNull.Value because an exception has been generated if this point
'is reached.
Return System.DBNull.Value
End Function 'GetQueryValue

```

### Calling the Script from a Mapping Expression

1. If you are using the function in an existing replication, either stop the Data Replicator or disable the replication.
2. For an existing replication, right-click on the replication and choose **Replication Properties**, then click **Mapping** to display the Mapping Editor. For new replications, perform the following steps when you get to the mapping portion of replication creation.
3. Right-click on the target field that will be using the generated value and choose **Map to Expression**.
4. In the Expression Generator window, expand the User Functions folder and click on (all) to see if your function is listed.

If the function is not displayed, check to make sure that you have used the proper Public Shared qualifier for the function. See [Writing a Function to Use in Scripts and Expressions](#) for more information.

5. Type an expression similar to the following in the Expression Editor.

```
GetQueryValue(DBRS.SourceConnection, "SELECT Capital FROM Capitals WHERE abbrv='" &
[ABBRV] & "'")
```

modifying the SELECT statement to retrieve the information you need.

6. Click **OK** to close the Expression Generator window.
7. Click **OK** to close the Mapping window.
8. Click **OK** to close the Replication Properties window.
9. Either start the Data Replicator or enable the replication to test that your function is working.

### Date Conversion

Define this function in the Global Script Editor, then call it by mapping the source field to an expression on the target field. The expression would be DateConvert([DT]), where DT is the name of the source field. The DT field should have the **Use Unmapped** option checked.

```

Public Shared Function DateConvert(intDate)
    Dim cyy
    Dim yyyy
    Dim mm
    Dim dd
    cyy = CInt(intDate/10000)
    mm = CInt((intDate - cyy * 10000) / 100)
    dd = intDate Mod 100
    if (cyy < 100)

```

```

        yyyy = 1900 + cyy
    else
        yyyy = 2000 + (cyy Mod 100)
    End if
    DateConvert = DateSerial/yyyy, mm, dd)
End Function

```

## Julian Date Conversion

Define this function in the Global Script Editor, then call it by mapping the source field to an expression on the target field. The expression would be `JulianToDate(obj)`, where `obj` is the name of the source field. The source field should have the **Use Unmapped** option checked. This function takes a numeric Julian date in the format `YYYDDD` where `YYY` is the number of years since 1900 and `DDD` is the day of the year and returns a VB.Net date value or Null if the original value was null.

```

Public Shared Function JulianToDate(obj as Object) as Object
    If obj Is Nothing Then Return Nothing
    If Not IsNumeric(obj) Then Return Nothing
    Dim intYears as Integer
    intYears = Int(obj/1000)
    Dim intDays as Integer
    intDays = (obj - intYears * 1000) - 1
    Dim datDate as Date
    datDate = CDate("1/1/" & Str(1900 + intYears)).AddDays(intDays)
    Return datDate
End Function

```

## Combine Date and Time Fields

Define this function in the Global Script Editor, then call it by mapping the source field to an expression on the target field. The expression would be `CombineDateTime(objDate, objTime)`, where `objDate` is the name of the date source field and `objTime` is the name of the time source field. The source fields should have the **Use Unmapped** option checked.

```

Public Shared Function CombineDateTime(objDate as Object, objTime as Object) as Object
    'If the date object is nothing return a null value
    If objDate is Nothing Then Return System.DBNull.Value
    'If you want to return a time of midnight for null times, remove this line
    If objTime is Nothing Then Return System.DBNull.Value
    'If we don't have a legal date return a null value
    If Not IsDate(objDate) Then Return System.DBNull.Value
    Dim intHours as Integer
    intHours = Hour(objTime)
    Dim intMinutes as Integer
    intMinutes = Minute(objTime)
    Dim intSeconds as Integer
    intSeconds = Second(objTime)
    Dim datOutput as Date
    datOutput = DateAdd("h", intHours, objDate)
    datOutput = DateAdd("n", intMinutes, datOutput)
    datOutput = DateAdd("s", intSeconds, datOutput)
    Return datOutput
End Function

```



## Time Function that Handles Null Values

Define this function in the Global Script Editor, then call it by mapping the source field to an expression on the target field. The expression would be `SafeTimeValue(objTime)`, where `objTime` is the name of the time source field. The source field should have the **Use Unmapped** option checked. This function can be used to replace the built in `TimeValue` function because it checks for null values. When a record is deleted on SQL Server, only the primary key is replicated so when an attempt is made to modify the time value, the `TimeValue` routine fails because it has been passed a null value.

```
Public Shared Function SafeTimeValue(objTime as Object) as Object
    Dim intHours as Integer
    Dim intMinutes as Integer
    Dim intSeconds as Integer
    'Return a time of midnight for null or illegal times
    intHours = 0
    intMinutes = 0
    intSeconds = 0
    If Not objTime is Nothing Then
        If IsDate(objTime) Then
            intHours = Hour(objTime)
            intMinutes = Minute(objTime)
            intSeconds = Second(objTime)
        End If
    End If
    Dim datOutput as Date
    datOutput = DateAdd("h", intHours, datOutput)
    datOutput = DateAdd("n", intMinutes, datOutput)
    datOutput = DateAdd("s", intSeconds, datOutput)
    Return datOutput
End Function
```

## C# Global Script Function Examples

The following C# example functions can be defined in a global script, then used in replication scripts and expressions to modify target field values during replication. For detailed steps on how to define a Global Script function, check [Writing a Function to Use in Scripts and Expressions](#).

### Accessing Values Using a Different Database Connection

If your replication needs record values which are accessible only by opening a separate database connection and running a query there, you can write the following script in the Global Script Editor, then call the script as indicated below.

The `GetQueryValue` function retrieves the first returned value from the passed in connection and query. If the parameters are illegal, the query returns an error or the query does not return a value, the function returns `System.DBNull.Value`.

```
public object GetQueryValue(IDbConnection objConnection, String strQuery)
{
    // If a parameter is null, return a null value
    if (objConnection == null || strQuery == "")
        return System.DBNull.Value;
    // Declare variables that might be used
```

```

IDbCommand cmd = null;
IDataReader reader = null;
object objValue = System.DBNull.Value;
// Use the Try block to capture any exceptions that might be generated
try
{
    // Use a SyncLock to make sure this function is the only user of the
    // connection since the
    // source or target connection might be in use.
    lock (objConnection)
    {
        // Create and execute the select to get the specified value
        cmd = objConnection.CreateCommand();
        cmd.CommandText = strQuery;
        reader = cmd.ExecuteReader();
        // If we can read from the reader, the record is already there.
        // Get the information
        if (reader.Read())
            objValue = reader.GetValue(0);
    }
    // If there are no exceptions so far, return the value
    reader.Close();
    reader = null;
    cmd.Dispose();
    return objValue;
}
catch (Exception ex)
{
    // Output exceptions to the DBMoto log
    IGlobalScript.AddLog("Exception in GetQueryValue: " + ex.ToString(), 0);
}
finally
{
    if (!(reader == null))
        reader.Close();
    if (!(cmd == null))
        cmd.Dispose();
    //Return System.DBNull.Value because an exception has been generated
    //if this point is reached.
}
return System.DBNull.Value;
}

```

### Calling the Script from a Mapping Expression

1. If you are using the function in an existing replication, either stop the Data Replicator or disable the replication.
2. For an existing replication, right-click on the replication and choose **Replication Properties**, then click **Mapping** to display the Mapping Editor. For new replications, perform the following steps when you get to the mapping portion of replication creation.
3. Right-click on the target field that will be using the generated value and choose **Map to Expression**.

4. In the Expression Generator window, expand the User Functions folder and click on (all) to see if your function is listed.

If the function is not displayed, check to make sure that you have used the proper Public Shared qualifier for the function. See [Writing a Function to Use in Scripts and Expressions](#) for more information.

5. Type an expression similar to the following in the Expression Editor. Modify the SELECT statement to retrieve the information you need.

```
GlobalScript.GetQueryValue(DBRS.SourceConnection, "SELECT Capital FROM Capitals
WHERE abbrev='" + [ABBREV] + "'")
```

6. Click **OK** to close the Expression Generator window.
7. Click **OK** to close the Mapping window.
8. Click **OK** to close the Replication Properties window.
9. Either start the Data Replicator or enable the replication to test that your function is working.

## Date Conversion

Define this function in the Global Script Editor, then call it by mapping the source field to an expression on the target field. The expression would be DateConvert([DT]), where DT is the name of the source field. The DT field should have the **Use Unmapped** option checked.

```
public DateTime DateConvert(int intDate)
{
    int cyy;
    int yyyy;
    int mm;
    int dd;
    cyy = (int)(intDate / 10000);
    mm = (int)((intDate - cyy * 10000) / 100);
    dd = intDate % 100;
    if (cyy < 100)
        yyyy = 1900 + cyy;
    else
        yyyy = 2000 + (cyy % 100);
    DateTime dc = new DateTime(yyyy, mm, dd);
    return dc;
}
```

## Julian Date Conversion

Define this function in the Global Script Editor, then call it by mapping the source field to an expression on the target field. The expression would be JulianToDate(obj), where obj is the name of the source field. The source field should have the **Use Unmapped** option checked. This function takes a numeric Julian date in the format YYYYDDD where YYY is the number of years since 1900 and DDD is the day of the year and returns a VB.Net date value or Null if the original value was null.

```
public object JulianToDate(object obj)
{
    if (obj == null)
        return null;
    int intObj = Convert.ToInt32(obj);
    int intYears;
    intYears = intObj / 1000;
    int intDays;
    intDays = (intObj - intYears * 1000) - 1;
    DateTime datDate = new DateTime();
    datDate = Convert.ToDateTime("1/1/" + (1900 + intYears)).AddDays(intDays);
    return datDate;
}
```

## Combine Date and Time Fields

Define this function in the Global Script Editor, then call it by mapping the source field to an expression on the target field. The expression would be `CombineDateTime(objDate, objTime)`, where `objDate` is the name of the date source field and `objTime` is the name of the time source field. The source fields should have the **Use Unmapped** option checked.

```
public object CombineDateTime(String objDate, System.DateTime objTime)
{
    // If the date object is nothing return a null value
    if (objDate == null)
        return System.DBNull.Value;
    // To return a time of midnight for null times, remove this line
    if (objTime == null)
        return System.DBNull.Value;
    // If not a legal date, return a null value
    if (!IsDate(objDate))
        return System.DBNull.Value;
    int intHours;
    intHours = objTime.Hour;
    int intMinutes;
    intMinutes = objTime.Minute;
    int intSeconds;
    intSeconds = objTime.Second;
    DateTime datOutput;
    datOutput = DateTime.Now.AddDays(intHours);
    datOutput = DateTime.Now.AddDays(intMinutes);
    datOutput = DateTime.Now.AddDays(intSeconds);
    return datOutput;
}
```

## Time Function that Handles Null Values

Define this function in the Global Script Editor, then call it by mapping the source field to an expression on the target field. The expression would be `SafeTimeValue(objTime)`, where `objTime` is the name of the time source field. The source field should have the **Use Unmapped** option checked. This function can be used to replace the built in `TimeValue` function because it checks for null values. When a record is deleted on SQL Server, only the primary key is replicated so when an attempt is made to modify the time value, the `TimeValue` routine fails because it has been passed a null value.

```
public object SafeTimeValue(System.DateTime objTime)
{
    int intHours = 0;
    int intMinutes = 0;
    int intSeconds = 0;
    // Return a time of midnight for null or illegal times
    if (objTime != null)
    {
        if (IsDate(objTime.ToString()))
        {
            intHours = objTime.Hour;
            intMinutes = objTime.Minute;
            intSeconds = objTime.Second;
        }
    }
    System.DateTime datOutput = default(System.DateTime);
    datOutput = DateTime.Now.AddDays(intHours);
    datOutput = DateTime.Now.AddDays(intMinutes);
    datOutput = DateTime.Now.AddDays(intSeconds);
    return datOutput;
}
```

## Replication Script Events

The following events can be handled as part of a replication script.

<a href="#">Reader Events</a>	<a href="#">Writer Events</a>
<p>In the order they are called:</p> <ol style="list-style-type: none"> <li>1. <a href="#">Refresh_onPrepareRefresh</a> (Refresh only) or <a href="#">LogReader_onPrepareMirroring</a> (Mirroring only)</li> <li>2. <a href="#">Record_onBeforeMapping</a></li> <li>3. <a href="#">Record_onAfterMapping</a></li> <li>4. <a href="#">Replication_onConflict</a> (Synchronization only)</li> <li>5. <a href="#">Replication_onLateConflict</a> (Synchronization only)</li> </ol>	<p>In the order they are called:</p> <ol style="list-style-type: none"> <li>1. <a href="#">Refresh_onBeforeRefresh</a> (Refresh only)</li> <li>2. <a href="#">Refresh_onBeforeTruncate</a> (Refresh only)</li> <li>3. <a href="#">Refresh_onAfterTruncate</a> (Refresh only)</li> <li>4. <a href="#">LogReader_onBeforeMirroring</a> (Mirroring only)</li> <li>5. <a href="#">Record_onBeforeExecute</a></li> <li>6. <a href="#">Record_onExecuteError</a> (Defined in the <a href="#">global script</a>)</li> <li>7. <a href="#">Record_onAfterExecute</a> (Not called if record execution failed)</li> <li>8. <a href="#">LogReader_onReceiverChanged</a> (Mirroring only)</li> <li>9. <a href="#">Refresh_onAfterRefresh</a> (Refresh only)</li> <li>10. <a href="#">LogReader_onAfterMirroring</a> (Mirroring only)</li> <li>11. <a href="#">Replication_onCriticalError</a></li> <li>12. <a href="#">Replication_onError</a> (Defined in the <a href="#">global script</a>)</li> </ol>

For more about reader and writer events, check [Managing Reader and Writer Events in a Script](#).

## LogReader\_onPrepareMirroring

This event is fired before a mirroring operation is built and allows you to perform such operations as cancel the mirroring, record the mirroring start time and so on.

### VB.NET syntax

```
Public Overrides Sub LogReader_onPrepareMirroring(ByRef CancelMirroring As Boolean)
```

### C# syntax

```
public override void LogReader_onPrepareMirroring(ref bool CancelMirroring)
```

### Parameters

- CancelMirroring: A flag that allows you to cancel the mirroring operation for which the event is fired. The default value is False. Set the flag to True to cancel mirroring.

### Example [VB.NET syntax]

```
Public Overrides Sub LogReader_onPrepareMirroring(ByRef CancelMirroring As Boolean)
    Dim bNeedCancel As Boolean
    ' TODO:
    ' check some condition by which the mirroring would need to be canceled
    ' and applying the result to the bNeedCancel variable
    If bNeedCancel Then
        CancelMirroring = True
    End If
End Sub
```

### Example [C# syntax]

```
public override void LogReader_onPrepareMirroring(ref bool CancelMirroring)
{
    bool bNeedCancel = false;
    // TODO:
    // check some condition by which the mirroring would need to be canceled
    // and applying the result to the bNeedCancel variable
    if (bNeedCancel) {
        CancelMirroring = true;
    }
}
```

## LogReader\_onBeforeTransactionsRead

Now [LogReader\\_onBeforeMirroring](#). On installation of v 6.1, all scripts are updated to use the new event name [LogReader\\_onBeforeMirroring](#)

## LogReader\_onBeforeMirroring

This event will be activated only if you have defined a mirroring or synchronization replication which accesses the database transaction log. The event is fired immediately before attempting to read the transaction log.

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.

## VB.NET syntax

```
Public Overrides Sub LogReader_onBeforeMirroring (ByVal bSource As Boolean)
```

## C# syntax

```
public override void LogReader_onBeforeMirroring(bool bSource)
```

## Parameters

- bSource : When set to True, the event is related to the source connection. When set to False the event is related to the target

## Example [VB.NET syntax]

```
Public Overrides Sub LogReader_onBeforeMirroring(bSource As Boolean)
    AddLog("BeforeMirroring", 0)
End Sub
```

## Example [C# syntax]

```
public override void LogReader_onBeforeMirroring(bool bSource)
{
    GlobalScript.AddLog("BeforeMirroring", 0);
}
```

## LogReader\_onAfterTransactionsRead

Now [LogReader\\_onAfterMirroring](#). On installation of v 6.1, all scripts are updated to use the new event name [LogReader\\_onAfterMirroring](#)

## LogReader\_onAfterMirroring

This event will be activated only if you have defined a mirroring or synchronization replication which accesses the database transaction log. The event is fired immediately after reading the transaction log.

## VB.NET syntax

```
Public Overrides Sub LogReader_onAfterMirroring (ByVal bSource As Boolean)
```

## C# syntax

```
public override void LogReader_onAfterMirroring(bool bSource)
```

## Parameters

- bSource : When set to True, the event is related to the source connection. When set to False the event is related to the target

## Example [VB.NET syntax]

```
Public Overrides Sub LogReader_onAfterMirroring(bSource As Boolean)
    AddLog("AfterMirroring", 0)
End Sub
```

## Example [C# syntax]

```
public override void LogReader_onAfterMirroring(bool bSource)
```

```
{  
    GlobalScript.AddLog("AfterMirroring", 0);  
}
```

## LogReader\_onReceiverChanged

This event will be activated only if you have defined a mirroring or synchronization replication from Db2 which accesses the Db2 journal. Event fired when the iSeries receiver changes. This event could be used to delete an old receiver. However it should be used carefully especially when you have more than one replication sharing the same journal: you don't want to delete a receiver when one replication has changed it if there are other replications still using it. The best approach is to configure the iSeries to delete old receivers, for example, configuring it to keep only the last 2 receivers. Check the [HiT Software knowledge base](#) and the [Db2 Journal Information](#) topic for more about receivers.

### VB.NET syntax

```
Public Overrides Sub LogReader_onReceiverChanged (sOldRecvLib As String, sOldRecvName As String, sNewRecvLib As String, sNewRecvName As String)
```

### C# syntax

```
public override void LogReader_onReceiverChanged(string sOldRecvLib, string sOldRecvName, string sNewRecvLib, string sNewRecvName)
```

### Parameters

- sOldRecvLib: The old receiver library name
- sOldRecvName: The old receiver name
- sNewRecvLib: The new receiver library name
- sNewRecvName: The new receiver name

### Example [VB.NET syntax]

```
Public Overrides Sub LogReader_onReceiverChanged(sOldRecvLib As String, sOldRecvName As String, sNewRecvLib As String, sNewRecvName As String)  
    AddLog("Receiver Changed: OLD = " + sOldRecvLib + "/" + sOldRecvName + " NEW =  
    "+sNewRecvLib + "/" + sNewRecvName , 0)  
End Sub
```

### Example [C# syntax]

```
public override void LogReader_onReceiverChanged(string sOldRecvLib, string sOldRecvName, string sNewRecvLib, string sNewRecvName)  
{  
    GlobalScript.AddLog("Receiver Changed: OLD = " + sOldRecvLib + "/" + sOldRecvName +  
    " NEW = " + sNewRecvLib + "/" + sNewRecvName, 0);  
}
```

## Record\_onBeforeMapping

Event fired right before applying the mapping on the source table columns.



## VB.NET syntax

```
Public Overrides Sub Record_onBeforeMapping (recSource As IRecord, ByRef AbortRecord As Boolean, ByRef DisableReplication As Boolean)
```

## C# syntax

```
public override void Record_onBeforeMapping(IRecord recSource, ref bool AbortRecord, ref bool DisableReplication)
```

## Parameters

- recSource: An [IRecord](#) object that provides access to source record data, including the intended operation type (UPDATE, INSERT or DELETE)
- AbortRecord: By default this is set to False. If you set it to True, the record will be skipped during replication.
- DisableReplication: By default this is set to False. If you set it to True, the replication will be disabled.

## Example [VB.NET syntax]

```
' For each new inserted record, skip the replication if COMPANYNAME = 'TEST'

Public Overrides Sub Record_onBeforeMapping(recSource As IRecord, ByRef AbortRecord As Boolean, ByRef DisableReplication As Boolean)
    Dim obj As Object
    If recSource.OperationType = enmOperationType.Insert Then
        obj = recSource.GetValueAfter("COMPANYNAME")
        If obj Is Nothing OR IsDBNull(obj) Then
            Return
        End If
        If obj.ToString() = "TEST" Then
            AbortRecord = True
        End If
    End If
End Sub
```

## Example [C# syntax]

```
// For each new inserted record, skip the replication if COMPANYNAME = 'TEST'

public override void Record_onBeforeMapping(IRecord recSource, ref bool AbortRecord, ref bool DisableReplication)
{
    object obj = null;
    if (recSource.OperationType == enmOperationType.Insert) {
        obj = recSource.GetValueAfter("COMPANYNAME");
        if (obj == null | Information.IsDBNull(obj)) {
            return;
        }
        if (obj.ToString() == "TEST") {
            AbortRecord = true;
        }
    }
}
```

## Record\_onAfterMapping

Event fired right after applying the mapping on the source table columns.

### VB.NET syntax

```
Public Overrides Sub Record_onAfterMapping (recSource As IRecord, recTarget As IRecord,  
ByRef AbortRecord As Boolean, ByRef DisableReplication As Boolean)
```

### C# syntax

```
public override void Record_onAfterMapping(IRecord recSource, IRecord recTarget, ref bool  
AbortRecord, ref bool DisableReplication)
```

### Parameters

- recSource: An [IRecord](#) object that provides access to source record data, including the intended operation type (UPDATE, INSERT or DELETE)
- recTarget: An [IRecord](#) object that provides access to target record data, including the intended operation type (UPDATE, INSERT or DELETE)
- AbortRecord: By default this is set to False. If you set it to True, the record will be skipped during replication.
- DisableReplication: By default this is set to False. If you set it to True, the replication will be disabled.

### Example [VB.NET syntax]

```
' For each new inserted record, replicate the field in column 0 uppercase  
  
Public Overrides Sub Record_onAfterMapping(recSource As IRecord, recTarget As  
IRecord, ByRef AbortRecord As Boolean, ByRef DisableReplication As Boolean)  
    If recSource.OperationType = enmOperationType.Insert Then  
        if Not recTarget.GetValueAfter(0) Is Nothing Then  
            recTarget.SetValueAfter(0, recTarget.GetValueAfter(0).ToUpper)  
        End if  
    End If  
End Sub
```

### Example [C# syntax]

```
// For each new inserted record, replicate the field in column 0 uppercase  
  
public override void Record_onAfterMapping(IRecord recSource, IRecord recTarget, ref bool  
AbortRecord, ref bool DisableReplication)  
{  
    if (recSource.OperationType == enmOperationType.Insert) {  
        if ((recTarget.GetValueAfter(0) != null)) {  
            recTarget.SetValueAfter(0, recTarget.GetValueAfter(0).ToUpper);  
        }  
    }  
}
```

## Record\_onBeforeExecute

Event triggered right before changes to the record are executed at the target site. Note that this event and [Record\\_onAfterExecute](#) no longer have access to the source record values. Therefore, if you need to perform operations using the source value, use the [Record\\_OnAfterMapping](#) event instead.

Events associated with the execution phase of replication are run using a different thread from events associated with the mapping phase of replication. If you want the event to be triggered at the closest possible time to the actual replication of the record, you should use `Record_onBeforeExecute` or [Record\\_onAfterExecute](#).

### VB.NET syntax

```
Public Overrides Sub Record_onBeforeExecute (recTarget As IRecord, AbortRecord As Boolean, ByRef DisableReplication As Boolean)
```

### C# syntax

```
public override void Record_onBeforeExecute(IRecord recTarget, bool AbortRecord, ref bool DisableReplication)
```

### Parameters

- `recTarget`: An [IRecord](#) object that provides access to target record data, including the intended operation type (UPDATE, INSERT or DELETE)
- `AbortRecord`: By default this is set to False. If you set it to True, the record will be skipped during replication.
- `DisableReplication`: By default this is set to False. If you set it to True, the replication will be disabled.

### Example [VB.NET syntax]

```
' For each inserted and updated record, skip the replication if the new Value assigned to column 0 is 'TEST'
```

```
Public Overrides Sub Record_onBeforeExecute(recTarget As IRecord, ByRef AbortRecord As Boolean, ByRef DisableReplication As Boolean)
    Dim obj As Object
    If recSource.OperationType = enmOperationType.Insert Or recSource.OperationType = enmOperationType.Update Then
        obj = recTarget.GetValueAfter(0)
        if obj Is Nothing OR IsDBNull(obj) Then
            return
        End if
        if obj.ToString() = "TEST" Then
            AbortRecord = True
        End if
    End If
End Sub
```

### Example [C# syntax]

```
// For each inserted and updated record, skip the replication if the new Value assigned to column 0 is 'TEST'
```

```
public override void Record_onBeforeExecute(IRecord recTarget, ref bool AbortRecord, ref
bool DisableReplication)
{
    object obj = null;
    if (recSource.OperationType == enmOperationType.Insert | recSource.OperationType ==
enmOperationType.Update) {
        obj = recTarget.GetValueAfter(0);
        if (obj == null | Information.IsDBNull(obj)) {
            return;
        }
        if (obj.ToString() == "TEST") {
            AbortRecord = true;
        }
    }
}
```

## Record\_onAfterExecute

Event triggered right after changes to the record are executed at the target site. Note that this event and [Record\\_onBeforeExecute](#) no longer have access to the source record values. Therefore, if you need to perform operations using the source value, use the [Record\\_OnAfterMapping](#) event instead.

Events associated with the execution phase of replication are run using a different thread from events associated with the mapping phase of replication. If you want the event to be triggered at the closest possible time to the actual replication of the record, you should use [Record\\_onBeforeExecute](#) or [Record\\_onAfterExecute](#). Note that this event is generated only if the record operation succeeds and not at all if the record operation fails.

### VB.NET syntax

```
Public Overrides Sub Record_onAfterExecute (recTarget As IRecord, Failed As Boolean,
ByRef DisableReplication As Boolean)
```

### C# syntax

```
public override void Record_onAfterExecute(IRecord recTarget, bool Failed, ref bool
DisableReplication)
```

### Parameters

- recTarget: An [IRecord](#) object that provides access to target record data, including the intended operation type (UPDATE, INSERT or DELETE)
- Failed: A read-only input parameter. When True, the record operation failed.
- DisableReplication: By default this is set to False. If you set it to True, the replication will be disabled.

### Example [VB.NET syntax]

```
Public Overrides Sub Record_onAfterExecute(recTarget As IRecord, Failed As Boolean, ByRef
DisableReplication As Boolean)
    If Not Failed Then
        AddLog("Replication of single record succeeded", 0)    Else
        AddLog("Replication of single record failed", 2)
```

```
End If
End Sub
```

### Example [C# syntax]

```
public override void Record_onAfterExecute(DBMotoPublic.IRecord recTarget, bool Failed,
ref bool DisableReplication)
{
    if (!(Failed))
    {
        GlobalScript.AddLog("Replication of single record succeeded", 0);
    }
    else
    {
        GlobalScript.AddLog("Replication of single record failed", 2);
    }
}
```

## Refresh\_onBeforeTruncate

Event fired right before the truncation of the target table. If the value of the variable CancelTruncate is set to True, no truncation will occur.

### VB.NET syntax

```
Public Overrides Sub Refresh_onBeforeTruncate (ByRef CancelTruncate As Boolean, ByRef
Filter As System.String)
```

### C# syntax

```
public override void Refresh_onBeforeTruncate(ref bool CancelTruncate, ref System.String
Filter)
```

### Parameters

- CancelTruncate: indicates if the truncation has to be canceled.
- Filter: a string containing a SQL WHERE condition to be used as a filter to determine which records to truncate.

### Example [VB.NET syntax]

```
Public Overrides Sub Refresh_onBeforeTruncate(ByRef CancelTruncate As Boolean, ByRef
Filter As System.String)
    AddLog("Refresh_onBefore Started", 0)
    Filter = " ID < 10 "
    AddLog("Refresh_onBeforeTruncate Finished", 0)
End Sub
```

### Example [C# syntax]

```
public override void Refresh_onBeforeTruncate(ref bool CancelTruncate, ref System.String
Filter)
{
    AddLog("Refresh_onBefore Started", 0);
```

```
Filter = " ID < 10 ";  
AddLog("Refresh_onBeforeTruncate Finished", 0);  
  
}
```

## Refresh\_onAfterTruncate

Event fired right after the truncation of the target table.

### VB.NET syntax

```
Public Overrides Sub Refresh_onAfterTruncate ()
```

### C# syntax

```
public override void Refresh_onAfterTruncate()
```

### Example [VB.NET syntax]

```
Public Overrides Sub Refresh_onAfterTruncate()  
    AddLog("Truncate Finished", 0)  
End Sub
```

### Example [C# syntax]

```
public override void Refresh_onAfterTruncate()  
{  
    GlobalScript.AddLog("Truncate Finished", 0);  
}
```

## Refresh\_onPrepareRefresh

Event fired before refresh operation is built to allow the inclusion of a filter (WHERE condition). This is an alternative to providing a filter in the replication properties and allows you to change the filter each time a refresh is performed.

### VB.NET syntax

```
Public Overrides Sub Refresh_onPrepareRefresh(ByRef Filter As String, ByRef CancelRefresh  
As Boolean)
```

### C# syntax

```
public override void Refresh_onPrepareRefresh(ref string Filter, ref bool CancelRefresh)
```

### Parameters

- Filter: a string containing a SQL WHERE condition to be used as a filter when performing a refresh.
- CancelRefresh: A flag that allows you to cancel the refresh operation for which the event is fired. The default value is False. Set the flag to True to cancel a refresh.

### Example [VB.NET syntax]

This example shows how to define an incremental refresh by using the Refresh\_onPrepareRefresh event.

NOTE: this sample works only with inserted data. Updated and deleted records are not correctly replicated using the incremental refresh.

1. Define a source table in this way:

ID integer (incremental)

Name varchar(20)

2. Define a replication in refresh mode, scheduled to run recurrently.

3. Define the following replication script:

```

Namespace DBRS
    Public Class ReplicationScript : Inherits IReplicationScript
        ' create a variable m_LastID to store the last inserted id
        Public m_LastID As Long
        Public Overrides Sub Refresh_onBeforeTruncate(ByRef CancelTruncate As Boolean)
            ' truncate the target table only the first time
            If m_LastID > 0 Then
                CancelTruncate = True
            End If
        End Sub
        Public Overrides Sub Record_onAfterExecute(recTarget As IRecord)
            ' save the last inserted id
            If recTarget.OperationType = enmOperationType.Insert Then
                m_LastID = recTarget.GetValueAfter("ID")
            End If
        End Sub

        Public Overrides Sub Refresh_onPrepareRefresh(ByRef Filter As String)
            ' prepare the filter for the next refresh
            If m_LastID > 0 Then
                Filter = "ID > " + m_LastID.ToString()
            End If
        End Sub
    End Class
End Namespace

```

#### Example [C# syntax]

```

using System;
using System.Data;
using DBMotoPublic;
using DBMotoScript;
namespace DBRS
{
    public class ReplicationScript : IReplicationScript
    {
        long m_LastID = 0; //create a variable m_LastID to store the last inserted id

        public override void Refresh_onBeforeTruncate(ref bool CancelTruncate)
        {
            // truncate the target table only the first time
            if (m_LastID > 0)
            {

```

```

        CancelTruncate = true;
    }
}

public override void Record_onAfterExecute(DBMotoPublic.IRecord recTarget, bool
Failed)
{
    // save the last inserted id
    if (recTarget.OperationType == enmOperationType.Insert)
    {
        m_LastID = (long) recTarget.GetValueAfter("ID");
    }
}

public override void Refresh_onPrepareRefresh(ref bool CancelRefresh, ref string
Filter)
{
    // prepare the filter for the next refresh
    if (m_LastID > 0)
    {
        Filter = "ID > " + m_LastID.ToString();
    }
}
}
}
}

```

## Refresh\_onBeforeRefresh

Event fired right before starting the refresh.

### VB.NET syntax

```
Public Overrides Sub Refresh_onBeforeRefresh ()
```

### C# syntax

```
public override void Refresh_onBeforeRefresh()
```

### Example [VB.NET syntax]

```
Public Overrides Sub Refresh_onBeforeRefresh()
    AddLog("Refresh Started", 0)
End Sub
```

### Example [C# syntax]

```
public override void Refresh_onBeforeRefresh()
{
    GlobalScript.AddLog("Refresh Started", 0);
}
```

## Refresh\_onAfterRefresh

Event fired right after the execution of the refresh.



## VB.NET syntax

```
Public Overrides Sub Refresh_onAfterRefresh ()
```

## C# syntax

```
public override void Refresh_onAfterRefresh()
```

### Example [VB.NET syntax]

```
Public Overrides Sub Refresh_onAfterRefresh()  
    AddLog("Refresh Finished", 0)  
End Sub
```

### Example [C# syntax]

```
public override void Refresh_onAfterRefresh()  
{  
    GlobalScript.AddLog("Refresh Finished", 0);  
}
```

## Replication\_onCriticalError

This event, although still available, has been replaced by the global events [Replication\\_onError](#) and [Record\\_OnExecuteError](#).

This event is triggered when an error occurs during data replication (refresh, mirroring or synchronization) that prevents the replication operation from continuing. The error may either disable the replication, or block the replication from executing the current cycle. For example, a connection error on the source table will temporarily put a mirroring replication in a recovery state but will not disable it. An error during truncation before refresh will generate a critical error because it stops the refresh from executing. For some of these errors, DBMoto can recover and try the operation (mirroring or refresh) again at the next scheduled interval. For other errors, DBMoto has to stop the replication (for example if the source or target table is deleted). The message passed as a parameter to the event contains information regarding the status of the replication, describing if it is stopped or in recovery mode. This event is useful when a system administrator wants to be notified immediately (for example by email) when a critical error occurs.

Any other error (like a single record insert error) will generate a [global Replication\\_onError event](#) at the end of the current refresh operation or mirroring interval, but not a Replication\_onCriticalError event. Notice also that the Replication\_onError event is always generated even in presence of a critical error, so if you define a global script to handle the Replication\_onError event, you are sure it will be run in presence of any critical error generated. Check also the global [Record\\_OnExecuteError](#) event where an event is generated when an error occurs during a single record execute operation (INSERT, UPDATE, DELETE.)

## VB.NET syntax

```
Public Overrides Sub Replication_onCriticalError (sReplicationName As String, sMessage As String)
```

## C# syntax

```
public override void Replication_onCriticalError(string sReplicationName, string sMessage)
```

## Parameters

- sReplicationName: The name of the replication where the error has occurred.
- sMessage: The error message string.

### Example [VB.NET syntax]

```
Public Overrides Sub Replication_onCriticalError(sReplicationName As String, sMessage As String)
    AddLog("Critical Error in replication " + sReplicationName + " - Message: " + sMessage.Substring(0, sMessage.Length - 2), 2)
End Sub
```

### Example [C# syntax]

```
public override void Replication_onCriticalError(string sReplicationName, string sMessage)
{
    GlobalScript.AddLog("Critical Error in replication " + sReplicationName + " - Message: " + sMessage.Substring(0, sMessage.Length - 2), 2);
}
```

## Replication\_onConflict

This event is generated when a collision happens on a record during synchronization. A collision occurs when both the source and target database update/delete/insert the same record at the same time. The two records that generate the collision are passed as parameters to the event. The event function must resolve the collision by returning a new record that will be inserted. The record returned could be the source record, in this case the source connection wins the conflict, the target record (target wins), or a new record obtained merging the values of the two records.

### VB.NET syntax

```
Public Overrides Function Replication_onConflict (ByRef recSource As IRecord, ByRef recTarget As IRecord) As IRecord
```

### C# syntax

```
public override IRecord Replication_onConflict(IRecord recSource, IRecord recTarget)
```

## Parameters

- recSource: A record in the table defined by the source connection
- recTarget: A record in the table defined by the target connection

Returns: The record that will be inserted after executing the event function.

### Example [VB.NET syntax]

```
Public Overrides Function Replication_onConflict(recSource As IRecord, recTarget As IRecord) As IRecord
    AddLog("Conflict", 1)
End Function
```

### Example [C# syntax]

```
public override IRecord Replication_onConflict(IRecord recSource, IRecord recTarget)
{
    GlobalScript.AddLog("Conflict", 1);
}
```

### Replication\_onLateConflict

This event is generated when a very specific type of collision occurs on a record during synchronization. It may occasionally happen that between the time the log/journal is read in preparation for replication and the changes are recorded in the tables involved, a transaction occurs in one of the tables. If the transaction conflicts with the newly entered changes (as a result of synchronization), this event is triggered so that you can determine how you want to handle the conflict. The event is triggered during the synchronization iteration following the one where the conflict actually occurred.

### VB.NET syntax

```
Public Overrides Function Replication_onLateConflict (ByRef rec1 As IRecord, ByRef rec2
As IRecord, ByVal bSource As Boolean) As IRecord
```

### C# syntax

```
public override IRecord Replication_onLateConflict(ref IRecord rec1, ref IRecord rec2,
bool bSource)
```

### Parameters

- rec1: The transaction that was missed because it occurred at a critical time during synchronization
- rec2: The DBMoto transaction that occurred as a result of the synchronization replication

Note that rec1 and rec2 signify a conflict on the same table rather than a conflict between the tables involved in the synchronization.

- bSource: Boolean that when set to True means rec1 and rec2 were found to be in conflict in the table designated as a source table for the replication. When set to False, rec1 and rec2 occurred on the table designated as the target table for the replication.

Returns: The record that will be inserted after executing the event function.

## Replication Script Properties

The following properties are for use in any function in a [replication script](#).

[SourceConnection](#)

[InternalSourceConnection](#)

[SourceConnectionName](#)

[SourceTableName](#)

[TargetConnection](#)

[InternalTargetConnection](#)

[TargetConnectionName](#)

[TargetTableName](#)

[GroupName](#)

[ReplicationName](#)

### SourceConnection

Returns the connection for the source table. A reference to assembly 'System.Data.dll' is required in the Reference dialog.

DBMoto handles both the opening and the closing of the connection, but there are certain conditions (due to the use of threads in DBMoto) under which you need to lock the connection. If you are using more than one [reader event](#) or more than one [writer event](#) in the same replication, and those events use a connection object, that connection object needs to be synchronized to avoid the use of the same object inside different threads. To synchronize objects you can use the command SyncLock in VB.NET or lock in C# as in the following examples:

#### VB.NET SyncLock Example

```
Dim conn as IDbConnection = Nothing
Dim cmd as IDbCommand = Nothing
Dim reader as IDataReader = Nothing
Try
    conn = SourceConnection
    SyncLock(conn)
        cmd = conn.CreateCommand
        cmd.CommandText = "select count(*) from MARCO.dbo.EMP50"
        reader = cmd.ExecuteReader
        Dim iNumRecords = -1
        If reader.Read Then
            iNumRecords = reader.GetInt32(0)
        End If
        AddLog("Reader - Record in EMPNO: " & iNumRecords.ToString, 0)
    End SyncLock
Catch ex As Exception
    AddLog("Exception in Record_onAfterMapping: " & ex.ToString,0)
Finally
```

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.

```

if (not reader is Nothing) Then
    reader.Close
End If
if (not cmd is Nothing) Then
    cmd.Dispose
End If
End Try

```

### C# lock Example

```

public override void Record_onBeforeMapping(DBMotoPublic.IRecord recSource, ref bool
AbortRecord)
{
    IDbConnection conn = null;
    IDbCommand cmd = null;
    IDataReader reader = null;
    try
    {
        conn = SourceConnection;
        lock ((conn))
        {
            cmd = conn.CreateCommand();
            cmd.CommandText = "select count(*) from HITDB.dbo.EMP50";
            reader = cmd.ExecuteReader();
            int iNumRecords = -1;
            if (reader.Read())
            {
                iNumRecords = reader.GetInt32(0);
            }
            GlobalScript.AddLog("Reader - Record in EMPNO: " + iNumRecords.ToString(), 0);
        }
    }
    catch (Exception ex)
    {
        GlobalScript.AddLog("Exception in Record_onAfterMapping: " + ex.ToString(), 0);
    }
    finally
    {
        if ((reader != null))
        {
            reader.Close();
        }
        if ((cmd != null))
        {
            cmd.Dispose();
        }
    }
}

```

### VB.NET syntax

```
Public SourceConnection As System.Data.IDbConnection
```

## C# syntax

```
public System.Data.IDbConnection SourceConnection;
```

### Example [VB.NET syntax]

```
Public Overrides Sub Record_onBeforeMapping(recSource As IRecord, ByRef AbortRecord As Boolean)
    AddLog("SC connstring = " + SourceConnection.ConnectionString ,0)
    AddLog("SC timeout = " + SourceConnection.ConnectionTimeout.ToString() ,0)
    AddLog("SC db = " + SourceConnection.Database ,0)
    AddLog("SC state = " + SourceConnection.State.ToString() ,0)
End Sub
```

### Example [C# syntax]

```
public override void Record_onBeforeMapping(IRecord recSource, ref bool AbortRecord)
{
    GlobalScript.AddLog("SC connstring = " + SourceConnection.ConnectionString, 0);
    GlobalScript.AddLog("SC timeout = " + SourceConnection.ConnectionTimeout.ToString(), 0);
    GlobalScript.AddLog("SC db = " + SourceConnection.Database, 0);
    GlobalScript.AddLog("SC state = " + SourceConnection.State.ToString(), 0);
}
```

## InternalSourceConnection

Returns the .NET connection used internally by DBMoto for accessing to the source and to the target database, unlike `SourceConnection` which returns as clone of the connection object used by the Data Replicator engine. This property is useful in cases where a statement must be executed using the same connection as the one used by the engine. **WARNING:** Use this property with great care because improper use may compromise the success of the replication. Any error generated by running a script using this connection will affect the Data Replicator and the replication process. For example if the connection is closed by the script, the engine will fail. Typically it is preferable to use the [SourceConnection](#) property.

### VB.Net syntax

```
Public InternalSourceConnection As System.Data.IDbConnection
```

### C# syntax

```
public System.Data.IDbConnection InternalSourceConnection;
```

### Example [VB.NET syntax]

For the replication used in this example, the source database is IBM System i, and the target database is Microsoft SQL Server 2005. The source table contains an integer field which is mapped to a field with datatype `Int Identity` in SQL Server 2005. The `Int Identity` field is an auto number field so, without using a script such as the one below, the field in the source table which maps to the `Int Identity` field in the target will not be replicated. Using the following script, the data in the source field will be replicated to the target field correctly.

```
Public Overrides Sub Refresh_onBeforeRefresh()
```

```

        Dim conn as IDbConnection = Nothing
        Dim cmd as IDbCommand = Nothing
        Dim reader as IDataReader = Nothing
    Try
        conn = InternalTargetConnection
        cmd = conn.CreateCommand
        cmd.CommandText = "SET IDENTITY_INSERT SAMPLE.dbo.TarEmp ON"
        reader = cmd.ExecuteReader
    Catch ex As Exception
        AddLog("Exception in Internal Target Connection: " & ex.ToString,0)
    Finally
        if (not reader is Nothing) Then
            reader.Close
        End If
        if (not cmd is Nothing) Then
            cmd.Dispose
        End If
    End Try
End Sub

```

### Example [C# syntax]

For the replication used in this example, the source database is IBM System i, and the target database is Microsoft SQL Server 2005. The source table contains an integer field which is mapped to a field with datatype Int Identity in SQL Server 2005. The Int Identity field is an auto number field so, without using a script such as the one below, the field in the source table which maps to the Int Identity field in the target will not replicated. Using the following script, the data in the source field will be replicated to the target field correctly.

```

public override void Refresh_onBeforeRefresh()
{
    IDbConnection conn = null;
    IDbCommand cmd = null;
    IDataReader reader = null;
    try {
        conn = InternalTargetConnection;
        cmd = conn.CreateCommand;
        cmd.CommandText = "SET IDENTITY_INSERT SAMPLE.dbo.TarEmp ON";
        reader = cmd.ExecuteReader;
    }
    catch (Exception ex) {
        GlobalScript.AddLog("Exception in Internal Target Connection: " + ex.ToString, 0);
    }
    finally {
        if (((reader != null))) {
            reader.Close();
        }
        if (((cmd != null))) {
            cmd.Dispose();
        }
    }
}

```





## SourceConnectionName

Returns the name of the source connection as specified in DBMoto.

### VB.NET syntax

```
Public SourceConnectionName As String
```

### C# syntax

```
public string SourceConnectionName;
```

## SourceTableName

Returns the name of the source table used in the replication.

### VB.NET syntax

```
Public SourceTableName As String
```

### C# syntax

```
public string SourceConnectionName;
```

## TargetConnection

Returns the connection for the Target table. A reference to assembly 'System.Data.dll' is required in the Reference dialog.

DBMoto handles both the opening and the closing of the connection, but there are certain conditions (due to the use of threads in DBMoto) under which you need to lock the connection. See the [SourceConnection](#) property for more information.

### VB.NET syntax

```
Public TargetConnection As System.Data.IDbConnection
```

### C# syntax

```
public System.Data.IDbConnection TargetConnection;
```

### Example [VB.NET syntax]

```
Public Overrides Sub Record_onBeforeMapping(recSource As IRecord, ByRef AbortRecord As Boolean)
    AddLog("TC connstring = " + TargetConnection.ConnectionString ,0)
    AddLog("TC timeout = " + TargetConnection.ConnectionTimeout.ToString() ,0)
    AddLog("TC db = " + TargetConnection.Database ,0)
    AddLog("TC state = " + TargetConnection.State.ToString() ,0)
End Sub
```

### Example [C# syntax]

```
public override void Record_onBeforeMapping(IRecord recSource, ref bool AbortRecord)
{
    GlobalScript.AddLog("TC connstring = " + TargetConnection.ConnectionString, 0);
    GlobalScript.AddLog("TC timeout = " + TargetConnection.ConnectionTimeout.ToString(), 0);
    GlobalScript.AddLog("TC db = " + TargetConnection.Database, 0);
    GlobalScript.AddLog("TC state = " + TargetConnection.State.ToString(), 0);
}
```

```
}
```

## InternalTargetConnection

Returns the .NET connection used internally by DBMoto for accessing to the source and to the target database, unlike TargetConnection which returns as clone of the connection object used by the Data Replicator engine. This property is useful in cases where a statement must be executed using the same connection as the one used by the engine. **WARNING:** Use this property with great care because improper use may compromise the success of the replication. Any error generated by running a script using this connection will affect the Data Replicator and the replication process. For example if the connection is closed by the script, the engine will fail. Typically it is preferable to use the [TargetConnection](#) property.

### VB.Net syntax

```
Public InternalTargetConnection As System.Data.IDbConnection
```

### C# syntax

```
public System.Data.IDbConnection InternalTargetConnection;
```

### Example

See the [TargetConnection](#) property.

## TargetConnectionName

Returns the name of the target connection as specified in DBMoto.

### VB.NET syntax

```
Public TargetConnectionName As String
```

### C# syntax

```
public string TargetConnectionName;
```

## TargetTableName

Returns the name of the target table used in the replication.

### VB.NET syntax

```
Public TargetTableName As String
```

### C# syntax

```
public string TargetTableName;
```

## GroupName

Returns the name, if any, of the group to which the replication belongs.

### VB.NET syntax

```
Public GroupName As String
```

### C# syntax

```
public string GroupName;
```

## ReplicationName

Returns the name of the replication.

### VB.NET syntax

```
Public ReplicationName As String
```

### C# syntax

```
public string ReplicationName;
```

## IRecord Interface

Used as a parameter when writing scripts to handle certain [replication events](#) or the [Record\\_OnExecuteError global event](#), this interface provides access to records identified for replication either via transaction logs/journals during mirroring and synchronization replications, or via mapped source table records during refresh replications. Using the Get and Set methods, you can access and/or modify column values.

[ColumnCount \(Property\)](#)

[OperationType \(Property\)](#)

[IsSource \(Property\)](#)

[IsPrimaryKey](#)

[enmLogFields\(Enumerator\)](#)

[enmOperationType \(Enumerator\)](#)

[GetValueBefore](#)

[GetValueAfter](#)

[SetValueBefore](#)

[SetValueAfter](#)

[GetLogValue](#)

[SetLogValue](#)

[GetType](#)

[GetDBType](#)

[GetPrecision](#)

[GetScale](#)

[GetLength](#)

[GetName](#)

## ColumnCount (Property)

The number of columns in the IRecord object, that is, it returns the number of mapped columns (within the current replication) in the table to which the IRecord is related.

### VB.NET syntax

```
Public ReadOnly Property ColumnCount As Integer
```

### C# syntax

```
public int ColumnCount { get; }
```

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.

### Example [VB.NET syntax]

This example prints a message in the log to show how many mapped columns have been found before applying the mapping.

```
Public Overrides Sub Record_onBeforeMapping(recSource As IRecord, ByRef AbortRecord As Boolean)
    AddLog("Column Count = " & recSource.ColumnCount, 0)
End Sub
```

### Example [C# syntax]

This example prints a message in the log to show how many mapped columns have been found before applying the mapping.

```
public override void Record_onBeforeMapping(IRecord recSource, ref bool AbortRecord)
{
    GlobalScript.AddLog("Column Count = " + recSource.ColumnCount, 0);
}
```

## OperationType (Property)

A list of constants that define the type of SQL operation associated with the transaction represented in the record. This property is typically used when designing mirroring or synchronization replications which use a transaction log. It provides access to the transaction's SQL operation: INSERT, UPDATE, DELETE, CLEAR or Unknown. If setting up a refresh replication, the only operation type is INSERT. The datatype for this property is the enumerator [enmOperationType](#).

### VB.NET syntax

```
Public Property OperationType As enmOperationType
```

### C# syntax

```
public enmOperationType OperationType { get; set; }
```

### Example [VB.NET syntax]

The example below checks to see if the operation being performed is an update before proceeding to add the value before changes apply to the DBMoto log.

```
Public Overrides Sub Record_onAfterMapping(recSource As IRecord, recTarget As IRecord, ByRef AbortRecord As Boolean)
    if (recSource.OperationType = enmOperationType.Update)
        AddLog(recSource.GetValueBefore(0), 0)
    End if
End Sub
```

### Example [C# syntax]

The example below checks to see if the operation being performed is an update before proceeding to add the value before changes apply to the DBMoto log.

```
public override void Record_onAfterMapping(DBMotoPublic.IRecord recSource,
DBMotoPublic.IRecord recTarget, ref bool AbortRecord)
{
    if (recSource.OperationType == enmOperationType.Update)
    {
        GlobalScript.AddLog(" " + recSource.GetValueBefore(0), 0);
    }
}
```

### enmOperationType (Enumerator)

The datatype for the [OperationType](#) property.

Value	Name	Description
0	Unknown	Not typically used
1	Insert	A SQL INSERT operation
2	Update	A SQL UPDATE operation
3	Delete	A SQL DELETE operation
4	Clear	For Db2 for i. All values will be cleared from the record. This operation type is supported for mirroring replications only.
5	Internal	For internal use only.

### IsSource (Property)

This property returns True when the record object has been generated while replicating a transaction **from** a connection defined under the Source node in the DBMoto Management Center (often referred to as a source connection) **to** a connection defined under the Target node in the Management Center (target connection). In other words, all records replicated in a refresh or mirroring replication have this property set to True. In synchronization, it is set to True if the direction of the replication is source to target; False otherwise.

### VB.NET syntax

```
Public Property IsSource as Boolean
```

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.

## C# syntax

```
public bool IsSource { get; set; }
```

## Example [VB.NET syntax]

This sample function is defined in a synchronization replication and skips (i.e., doesn't propagate to the target) any Delete operations performed on the source table.

```
Public Overrides Sub Record_onBeforeExecute(recTarget As IRecord, ByRef AbortRecord As Boolean)
    If (recTarget.IsSource And recTarget.OperationType = enmOperationType.Delete) Then
        AbortRecord = True
    End If
End Sub
```

## Example [C# syntax]

This sample function is defined in a synchronization replication and skips (i.e., doesn't propagate to the target) any Delete operations performed on the source table.

```
public override void Record_onBeforeExecute(IRecord recTarget, ref bool AbortRecord)
{
    if ((recTarget.IsSource & recTarget.OperationType == enmOperationType.Delete)) {
        AbortRecord = true;
    }
}
```

## IsPrimaryKey

Provides a way to determine if a field is a primary key or part of a primary key.

### VB.NET syntax

```
Public Function IsPrimarykey (Index As Integer) As Boolean
Public Function IsPrimarykey (ColumnName As String) As Boolean
```

### C# syntax

```
public bool IsPrimarykey(int Index)
public bool IsPrimarykey(string ColumnName)
```

### Parameters

- Index: an integer that identifies the column for which you want to retrieve the primary key settings
- ColumnName: a string representing the column name for which you want to retrieve the primary key settings

### Return value

Returns true if the field is part of the primary key.

### Example [VB.NET syntax]

```
Public Overrides Sub Record_onAfterMapping(recSource As IRecord, recTarget As IRecord,
ByRef AbortRecord As Boolean)
    Dim T As Boolean
    T=recSource.IsPrimaryKey(1)
    AddLog("Is the second column a primary key? True or False: " + T.ToString, 0)
End Sub
```

### Example [C# syntax]

```
public override void Record_onAfterMapping(IRecord recSource, IRecord recTarget, ref bool
AbortRecord)
{
    bool T = false;
    T = recSource.IsPrimaryKey(1);
    GlobalScript.AddLog("Is the second column a primary key? True or False: " +
T.ToString, 0);
}
```

## GetValueBefore

Returns the value of the column indicated in the record before modification. If the value is not available in the transaction log, this function returns Null. Use this function in mirroring/synchronization replications if you want to retrieve a prior source or target value when the transaction log reports an UPDATE or DELETE operation on a record.

### VB.NET syntax

```
Public Function GetValueBefore (sColName As String) As Object
Public Function GetValueBefore (iIndex As Integer) As Object
```

### C# syntax

```
public object GetValueBefore(string sColName)
public object GetValueBefore(int iIndex)
```

### Parameters

- sColName: name of the column to be referred to.
- iIndex: index of the column to be referred to.

## Chapter 8: Exploring the Graphical User Interface

### Shortcuts to Creating a Replication

The DBMoto Management Center window usually offers several ways to accomplish the same goal:

- Via the buttons displayed in the toolbars
- Via the menus displayed when clicking the right mouse button over an item in the Management Center window.
- Via the main toolbar
- Via dragging and dropping items in the Metadata Explorer.

Here are some ways to speed up the process of creating a replication.

### Starting Wizards Automatically

The first step in setting up a replication in the Management Center is to create connections to your source and target databases.

Each wizard involved in the creation of a replication provides an option to automatically open the next wizard in the process.

In the Summary screen of the [Source Connection wizard](#), check the option to proceed with the definition of a target connection. Then, in the Summary screen of the Target Connection wizard check the option to open the [Replication Wizard](#).

### Using Drag and Drop to Create Tables and Replications

Once you have defined source and target connections, you can use the Metadata Explorer to drag a source table to a target connection to open the [Create Target Table wizard](#), and drag a source table to a target table to open the [Replication wizard](#).

#### Creating a Target Table

1. In the Metadata Explorer, expand the source and target connections to display both the source table and the target connection.
2. Click and hold the mouse button down on the source table and drag it down to the target connection, then release the mouse button.

The Create Target Table wizard is launched with the source table and target connection information already selected for you.

#### Creating a Replication

1. In the Metadata Explorer, expand the source and target connections to display both the source table and the target table.
2. Click and hold the mouse button down on the source table and drag it down to the target table, then release the mouse button.

The Replication wizard is launched with the source table and target table information already selected for you.



## Wizards

### Add Source Connection Wizard

[Select Provider](#)>[Set Connection String](#)>[Select Tables](#)>[Actions](#)>[Summary](#)

Use this wizard to create a connection to the database that contains the data you want to replicate, tables that hold all the information needed for your replication.

#### Select Provider

##### Source Name

Type a name to identify the source connection. This name appears in the Metadata Explorer as a way to group connections for a specific replication.

##### Database

Select the database for which you want to create a connection.

##### Provider

Select a .NET data provider from the list. If no .NET provider is available for the database, select either an OLE DB provider or an ODBC driver. By default, only default providers are listed in this dialog. If you want to see all providers for a specific database, uncheck the **Use Only Default Providers** option in the [Options dialog](#).

.NET providers are preferred over ODBC drivers or OLE DB providers because, with DBMoto being a .NET application, it can take advantage of the use of native .NET providers based on the same architecture. If you use ODBC or OLE DB connections, the .NET framework will need to add a software layer for every call you make to the driver, adding overhead. Besides, using 100% managed .NET providers, there is no risk of memory leaks because the .NET framework cleans up memory automatically using an internal garbage collector.

##### Assembly

This field is displayed only if using a .NET data provider, and the provider has not been registered. Provide a path to the .NET data provider DLL. Check the [HiT Software knowledge base article on data providers](#) before entering a value in the **Assembly** field.

#### Set Connection String

##### Connection Properties

Edit at least the **Required** connection properties by clicking in the property value field and typing a new value.

##### 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.

#### **For SQL Server Replications:**

If you are replicating from SQL Server using mirroring or synchronization, for the user ID and password fields, provide a login that does not have system administrator privileges. This login ID should also be used exclusively for DBMoto (i.e., no other applications should use the login ID that you specify for DBMoto). You will need to provide a second login with system administrator privileges in the [Setup Info](#) screen. The second login is used to create the distributor and to access the transaction log.


#### **For Db2 for i Replications:**

If you are connecting to IBM Db2 for i (iSeries/AS400) as your source database, see [IBM Db2 for i Connection Recommendations](#).

#### **For Oracle Replications:**

If you are replicating from Oracle using mirroring or synchronization, enter a user ID which will be exclusively used by DBMoto and has the authority to read the database transaction log (redo log.)

The list of **Optional** properties for .NET and OLE DB providers contains the most commonly used properties for the providers. Edit these as needed. Note that some properties are displayed with default values (no bold text.) Any values that you add or edit are displayed in bold text. Check the documentation for your provider for a complete list of properties. You can set the value of the ExtendedProperties property to define additional property-value pairs. The syntax for defining property-value pairs is: prop1=val1;prop2=val2;....

If you are using an ODBC driver, click in the ConnectionString value field and then click  to open the ODBC Driver Connection dialog. The contents of this dialog depend on the ODBC driver that you are using. Provide values for the dialog fields, then click OK to make a test connection and set the values in the DBMoto wizard.

#### **Edit**

Click **Edit** to open a text entry window where you can paste or type a connection string for your provider. This is offered as an alternative to the Connection Properties grid, but should be used with great care because an error in the connection string can cause a connection to fail or to have unexpected properties. This window displays any connection information that you have already entered in connection string format. Note that default values are not displayed as part of the connection string.


#### **Test**

Click **Test** to make sure that the connection correctly opens a database connection.



#### **Select Tables**

Choose one or more objects from the tree to associate with the source connection. Objects include tables, views and aliases. When creating a replication, you will be able to select an object for replication from those that you have chosen in this wizard. If you create multiple replications, you can select an object for each replication that you are defining.

## Filtering the Table View

Use the filter field to limit the number of tables to view at any time. The field expects standard SQL syntax. Either type in the full name of the object you want to retrieve, or use ‘%’ as search pattern. For example, if you type AB%, all objects beginning with the letters ‘AB...’ are displayed. To apply the filter, type the filter in the textbox and click . If filtering is not available for a specific node, or connection, the filter icon is inactive.



- You can apply a filter on any node in the tree, as long as you provide the correct SQL syntax.
- If you apply incorrect SQL syntax, the error dialog reports a SQL error. You should be familiar with the SQL syntax for your database to analyze and fix the error.
- DBMoto stores the filter so that if you select another node, then go back to the previous one, the filter is retrieved.
- When a node in the tree has been expanded with a filter, its icon is modified to show that the full contents of that node have been filtered.
- Note that some databases, such as IBM Db2 for i, may not support the % symbol on one or more levels in the tree (catalog, schema, etc.).
- If you expand a node applying one filter (for example, A%) and then select/remove some tables, then change the filter to B% and select/remove other tables, when clicking OK the dialog will remember all the chosen tables and will apply all filters.

To remove a filter, either click , or delete the filter text and click . The content for the node is reloaded without filtering information.

Use the **Hide System Tables** checkbox to limit the number of tables displayed.

Use the **Select All Tables** and **Deselect All Tables** buttons to work with multiple tables.

**Note:** When using SQL Server as a source in mirroring (or as a source or target in synchronization replications), SQL Server requires that tables have a primary key defined.

	<p>Select a database owner/schema, then click this button to check all tables under the owner/schema.</p>
	<p>Select a database owner/schema, then click this button to uncheck all tables under the owner/schema.</p>

## Actions

Choose any actions that you would like to perform after completing the wizard. Note that if you plan to set up mirroring or synchronization replications, you first need to complete the [Enable Transactional Replication wizard](#).

## Launch the Enable Transactional Replication wizard

Allows you to choose the appropriate source database log access for mirroring and synchronization replications. This wizard can also be accessed from the Metadata Explorer by selecting the connection, then choosing Connection Properties from the right mouse button menu. Set the Transactional Setup field to Enable to open the wizard.

## Launch the Target Connection wizard

Opens the Target Connection wizard to create a connection to the target database for replication

### Summary

#### Database name

The name of the database server

#### Database version

The database server version

#### Provider type

The description of the provider that you selected in the **Select provider** screen.

#### Provider name

Whatever connection information is available for the provider you have chosen. In some cases, this field will be blank. In other cases, the provider DLL may be displayed.

### Opening the Source Connection Wizard

1. In the DBMoto Management Center Metadata Explorer, select the label **Sources**.
2. From the right mouse button menu, choose **Add New Connection**.

### Add Target Connection Wizard

[Select Provider](#)>[Set Connection String](#)>[Select Tables](#)>[Actions](#)>[Summary](#)

Use this wizard to create a connection to a target database where replicated data will be stored.

### Select Provider

#### Target Name

Type a name to identify the source connection. This name appears in the Metadata Explorer as a way to group connections for a specific replication.

## Database

Select the database for which you want to make a connection. DBMoto also offers options to output data to text (Files) or CSV files (Comma Separated Value files). These options can be useful for auditing purposes. In addition, the Files option is used when replicating data with the DBMoto Cloud G Edition available under separate licensing. See [www.hitsw.com](http://www.hitsw.com) for more information.

## Provider

Select a .NET data provider from the list. If no .NET provider is available for the database, select either an OLE DB provider or an ODBC driver. By default, only default providers are listed in this dialog. If you want to see all providers for a specific database, uncheck the **Use Only Default Providers** option in the [Options dialog](#).

.NET providers are preferred over ODBC drivers or OLE DB providers because, with DBMoto being a .NET application, it can take advantage of the use of native .NET providers based on the same architecture. If you use ODBC or OLE DB connections, the .NET framework will need to add a software layer for every call you make to the driver, adding overhead. Besides, using 100% managed .NET providers, there is no risk of memory leaks because the .NET framework cleans up memory automatically using an internal garbage collector.

If you choose **Files** or **CSV files** in the **Database** field, the Provider field contains the value HiT Software File Provider to handle plain text output from the replication operation.

## Assembly

This field is displayed only if using a .NET data provider, and the provider has not been registered. Provide a path to the .NET data provider DLL. Check the [HiT Software knowledge base article on data providers](#) before entering a value in the **Assembly** field.

## Set Connection String

### Connection Properties

Edit at least the **Required** connection properties by clicking in the property value field and typing a new value.

If you will be creating target tables via DBMoto in preparation for replication, make sure that you choose a user ID with sufficient permissions to create tables in the designated target database.

#### **For Replications to SAP HANA and SAP Sybase IQ:**

After completing the Target Connection wizard, select the new connection and choose [Connection Properties](#) from the right mouse button menu. In the Connection Properties dialog, set values for the **FTP Server**, **FTP Port**, **FTP User**, **FTP Password** and **Import Path** properties. For optimal performance, the DBMoto connection uses FTP to transfer data from the source database to the target SAP HANA or SAP Sybase IQ server, then updates the SAP HANA or SAP Sybase IQ database using DBMoto's BulkInsert option. For replications to SAP Sybase IQ, [make sure that the allow read client file and allow write client file options are enabled on the Sybase IQ server](#).

#### **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.

**For SQL Server Replications:**

If you are replicating from SQL Server using synchronization, for the user ID and password fields, provide a login that does not have system administrator privileges. This login ID should also be used exclusively for DBMoto (i.e., no other applications should use the login ID that you specify for DBMoto). You will need to provide a second login with system administrator privileges in the [Setup Info](#) screen. The second login is used to create the distributor and to access the transaction log.

**For IBM Db2 for i Replications:**

If you are connecting to IBM Db2 for i (iSeries/AS400) for synchronization replications, see [IBM Db2 for i Connection Recommendations](#).


**For Oracle Replications:**

If you are replicating from Oracle using synchronization, enter a user ID which will be exclusively used by DBMoto and has the authority to read the database transaction log (redo log.)

**For File and CSV File Replications:**

If you are replicating data using DBMoto Cloud G Edition, check the documentation for the G Edition for information on the settings for this screen. If you are using the File or CSV File options for auditing, specify a directory path to save the output files and modify the default values for separators and additional columns as needed.

The list of **Optional** properties for .NET and OLE DB providers contains the most commonly used properties for the providers. Edit these as needed. Note that some properties are displayed with default values (no bold text.) Any values that you add or edit are displayed in bold text. Check the documentation for your provider for a complete list of properties. You can set the value of the ExtendedProperties property to define additional property-value pairs. The syntax for defining property-value pairs is: prop1=val1;prop2=val2;....

If you are using an ODBC driver, click in the ConnectionString value field and then click  to open the ODBC Driver Connection dialog. The contents of this dialog depend on the ODBC driver that you are using. Provide values for the dialog fields, then click OK to make a test connection and set the values in the DBMoto wizard.

**Edit**

Click **Edit** to open a text entry window where you can paste or type a connection string for your provider. This is offered as an alternative to the Connection Properties grid, but should be used with great care because an error in the connection string can cause a connection to fail or to have unexpected properties. This window displays any connection information that you have already entered in connection string format. Note that default values are not displayed as part of the connection string.


## Test

Click **Test** to make sure that the connection correctly opens a database connection.



## Select Tables

Choose one or more objects from the tree to associate with the source connection. Objects include tables, views and aliases. When creating a replication, you will be able to select an object for replication from those that you have chosen in this wizard. If you create multiple replications, you can select an object for each replication that you are defining.

## Filtering the Table View

Use the filter field to limit the number of tables to view at any time. The field expects standard SQL syntax. Either type in the full name of the object you want to retrieve, or use '%' as search pattern. For example, if you type AB%, all objects beginning with the letters 'AB..' are displayed. To apply the filter, type the filter in the textbox and click . If filtering is not available for a specific node, or connection, the filter icon is inactive.

- You can apply a filter on any node in the tree, as long as you provide the correct SQL syntax.
- If you apply incorrect SQL syntax, the error dialog reports a SQL error. You should be familiar with the SQL syntax for your database to analyze and fix the error.
- DBMoto stores the filter so that if you select another node, then go back to the previous one, the filter is retrieved.
- When a node in the tree has been expanded with a filter, its icon is modified to show that the full contents of that node have been filtered.
- Note that some databases, such as IBM Db2 for i, may not support the % symbol on one or more levels in the tree (catalog, schema, etc.).
- If you expand a node applying one filter (for example, A%) and then select/remove some tables, then change the filter to B% and select/remove other tables, when clicking OK the dialog will remember all the chosen tables and will apply all filters.



To remove a filter, either click , or delete the filter text and click . The content for the node is reloaded without filtering information.

Use the **Hide System Tables** checkbox to limit the number of tables displayed.

Use the **Select All Tables** and **Deselect All Tables** buttons to work with multiple tables.

**Note:** When using SQL Server as a source in mirroring (or as a source or target in synchronization replications), SQL Server requires that tables have a primary key defined.



	Select a database owner/schema, then click this button to check all tables under the owner/schema.
	Select a database owner/schema, then click this button to uncheck all tables under the owner/schema.

## Actions

Choose any actions that you would like to perform after completing the wizard. Note that if you plan to set up mirroring or synchronization replications, you first need to complete the [Enable Transactional Replication wizard](#).

### Launch the Enable Transactional Replication wizard

For use when setting up a [synchronization replication](#) only. Allows you to choose the appropriate target database log access for synchronization replications. This wizard can also be accessed from the Metadata Explorer by selecting the connection, then choosing **Connection Properties** from the right mouse button menu. Set the **Transactional Setup** field to **Enable** to open the wizard.

## Summary

### Database name

The name of the database server

### Database version

The database server version

### Provider type

The description of the provider that you selected in the **Select provider** screen.

### Provider name

Whatever connection information is available for the provider you have chosen. In some cases, this field will be blank. In other cases, the provider DLL may be displayed.

## Opening the Target Connection Wizard

1. In the DBMoto Management Center Metadata Explorer, select the label **Targets**.
2. From the right mouse button menu, choose **Add New Connection**.

## Create Target Table Wizard

[Select Source Connection](#)>[Select Target Connection](#)>[Define Columns](#)>[SQL Script](#)>[Summary](#)



Use this wizard to create a table in the database to which you are replicating if your target database does not already contain a table for the replicated data. Note that if source tables include constraints, and you want those same constraints on target tables, you need to create them manually at the target system once you have created the tables.

### Select Source Connection

This screen lists all the source connections that you have created in DBMoto.

#### Source Name

Choose a source connection name from the drop-down list.

#### Source Table

Choose a source table from the drop-down list.

#### Open Table

Opens the [Execute SQL Command dialog](#) that displays the contents of the source table for your review. Note that if you execute a SQL query from this dialog, it does not affect any source, target or replication settings.

### Select Target Connection

This screen lists all the target connections that you have created in DBMoto.

#### Source Name

Choose a target connection name from the drop-down list.

#### Source Table

Type a name for the table that will be created.

#### Open Table

Opens the [Execute SQL Command dialog](#) that displays the contents of the source table for your review. Note that if you execute a SQL query from this dialog, it does not affect any source, target or replication settings.

### Define Columns

Displays the columns that will be created on wizard completion. Edit column information by clicking on cells. You can also add and remove columns and primary keys using either the toolbar icons or the right mouse button menu. The available options are:

#### Insert Field Before

Select a field and choose this option to create a new field above this one.

#### Insert Field After

Select a field and choose this option to create a new field after this one.

### Delete Field

Select a field and choose this option to remove the column from the table.

### Set Primary Key

Select a field and choose this option to make the column a primary key. Note that primary keys can be set only on columns that cannot have a null value.

### Delete Primary Key

Select a field and choose this option to delete a primary key setting.

### SQL Script

Displays the SQL script that will be used to generate the target table based on the information from the source table and any changes you have made in the [Define Columns](#) screen. You can edit the SQL script as needed. Be sure to follow standard SQL syntax that can be used with your database system.

### Summary

Displays the settings for creating a target table.

### Create another table

Opens the Create Target Table wizard on completion of the current wizard.

### Proceed with the definition of a replication

Opens the [Replication wizard](#) on completion of the current wizard.

### Opening the Create Target Table Wizard

1. In the DBMoto Management Center Metadata Explorer, select a source table.
2. From the right mouse button menu, choose **Replication**, then **Create Target Table**.

### Disable Transactional Replication Wizard

[Setup Status](#) > [Uninstall Options](#)

Use this wizard to remove transactional replication settings either in DBMoto alone, or in DBMoto and the database associated with the connection.

### Setup Status

This screen allows you to review the current transactional replication settings for the connection. Use the information to check that you have selected the correct connection before processing to remove settings for the connection.

## Uninstall Options

Typically, two uninstall options are offered:

### Disable Transactional Replication

If no replications are enabled for this connection, you can disable the transactional replication settings without removing them.

### Complete Uninstall

If no replications are enabled for this connection, selecting this option will remove all DBMoto transactional settings for the connection and any programs that were installed on the system running DBMoto or the database to support transactional replications.

Click **Finish** to perform the disable and/or uninstall operations.

## Enable Transactional Replication Wizard

[Log Type](#)>[Log Settings](#)>[Agent Settings](#)>Actions>Summary

Use this wizard only if you are performing a mirroring or synchronization replication and you need to set up access to data changes on a source database (for mirroring and synchronization) or a target database (for synchronization).

### Log Type

For more information on log type options, see [Choosing a Log Type for Transactional Replications](#). Depending on the source database, available options are:

#### Log Reader

Reads the native database transaction log using a DBMoto reader thread for each replication. This option is the default choice and works well if you have a limited number of replications and do not want to install additional services on the system that is running the DBMoto Server Agent component.

Click **Next** to use this option and display the [Log Settings page](#) for your database:

#### Log Server Agent

Creates a Windows service on the system running the DBMoto Server Agent. The service autonomously reads the native database transaction log. Choose this option if you plan to add many replications and you need a scalable solution that optimizes access to the native database transaction log.

Select this option and click **Next** to use this option and display the **Log Server Agent Settings** page for your database.

### Triggers

Creates a trigger on the database for each replication to log data from committed transactions. This option is useful when neither the Log Reader or the Log Server Agent meet the needs of your environment. For example, use triggers when:

- There are no primary keys and the source is Microsoft SQL Server.
- All column data (before and after image) is needed. Some standard log readers only return record changed values
- Performance is better because only relevant information is logged and DBMoto does not read the entire database log
- The source database is MySQL versions prior to 5.1.5 or you are performing MySQL synchronization replications

Select this option and click **Next** to use this option and display the **Log Settings** page for trigger-based transactional replications.

### Log Settings/Trigger Settings

Select your source database from the options below for specific information on settings in the Log Settings screen:

- [Microsoft SQL Server](#)
- [Microsoft Azure SQL Database](#)
- [MySQL](#)
- [IBM Db2 for i](#)
- [IBM Db2 UDB or z/OS](#)
- [IBM PureData \(Netezza\)](#)
- [IBM Informix](#)
- [Oracle](#)
- [SAP HANA, SAP Sybase SQL Anywhere or SAP Sybase ASE](#)
- [Gupta SQLBase](#)
- [PostgreSQL](#)

If you are using triggers to replicate changed data, go directly to the [Trigger Settings](#).

### IBM Db2 for i (iSeries/AS400) Transactional Replication Settings

The information below explains how to configure the transactional replication settings for IBM Db2 for i in the [Enable Transactional Replication wizard](#) or the [Transactional Log Settings dialog](#) available from the **Transactional Setup** > **Manage** option on the connection in the Metadata Explorer.

When replicating from IBM Db2 for i using mirroring or synchronization, three transactional modes are available: [Log Reader](#), [Log Server Agent](#) and [Log Reader API](#). The Log Reader API is required when replicating tables with LOB values. Note that the Log Reader API option is available for Db2 for i V6R1 and above.

For all three modes, you first need to set up a library on your Db2 system. This step sets 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*

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.

### *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. If you are using operating system V5R4 or above, and choose either the DBMLIB54.SAVF file or the DBMLIBAPI61.SAVF file, a checkbox is displayed to allow you to install the procedure JRNSQMAPI. The box should be checked if you are planning to create more than 30 replications from a Db2 for i source database.

### *Verify (appears only in the Transactional Log Settings dialog)*

Click **Verify** first to see if the library that DBMoto will use (specified in **Library Name**) is already installed on the server. Depending on the results of this operation, the **Install** button will become active. Click **Install** to create the library on the Db2 system. To later remove the DBMoto library from the server, use the [Disable Transactional Replication wizard](#) available from available from the **Transactional Setup > Disable** option on the connection in the Metadata Explorer. .

### *FTP Status*

This area reports progress after you have entered all the information above and clicked **Install**.

### *Errors during Installation*

The FTP status area may report errors when attempting to restore the DBMOTOLIB library. Errors usually involve:

- A user ID with read-only permissions on the Db2 server or no QSECOFR privileges. This can be addressed by running the process again with a login that has write permissions and QSECOFR privileges.
- Library or Savefile are already present on the Db2 server, and can not be overwritten or deleted. This can usually be addressed by using a login that has privileges to delete/overwrite existing files.

In general, if errors occur during the process of automatically restoring the savf file, you can [restore the DBMOTOLIB library manually](#).

### [Log Server Agent Settings](#)

#### **IBM Db2 LUW Transactional Replication Settings**

The information below explains how to configure the transactional replication settings for IBM Db2 LUW i in the [Enable Transactional Replication wizard](#) or the [Transactional Log Settings dialog](#) available from the **Transactional Setup > Manage** option on the connection in the Metadata Explorer.

When replicating from IBM Db2 LUW using mirroring or synchronization, two transactional modes are available: [Log Reader and Triggers](#).

### Log Reader Settings

To define a mirroring or synchronization replication, you first need to set up stored procedures to access the log on your Db2 system. Before completing this screen, make sure that you have followed all the requirements for your Db2 connection described in [IBM Db2 Connection Requirements](#).

Note that LOB datatypes are not supported when using Db2 log-based transactional replications.

#### *Stored Procedures Schema Name*

The default value provided is the ID of the user specified in the Db2 connection. The value can be edited to reflect whichever schema you want to use for the log reading stored procedures. There are a total of three procedures created:

The library/dll has 3 stored procedures:

- LOGVER03 is used to identify the version of the stored procedures/library so that a mismatched version is not used
- CURLSN03 is used to find and retrieve the current LSN or Transaction ID in the Db2 log
- LOGREC03 is used to read and retrieve changed records from the Db2 log

#### *Log Reader Library Version*

A read-only field that displays the version of the file you copied from the DBMoto ServerFiles/Db2UDB folder to your Db2 Server function folder. See [IBM Db2 Connection Requirements](#) for information on the files to copy and additional setup requirements.

#### *Verify (available only in the Manage Transactional Log Settings dialog)*

Click **Verify** to check that:

- Log-based transactional replication is supported for the Db2 version that you are using
- The stored procedures that DBMoto will use are installed on the server.  
READDB2UDB\_LOGVER03  
READDB2UDB\_CURLSN03  
READDB2UDB\_LOGREC03

To later remove the stored procedures from the server, use the [Disable Transactional Replication wizard](#) available from the **Transactional Setup** > **Disable** option on the connection in the Metadata Explorer.

## [Triggers Settings](#)

### IBM Informix Transactional Replication Settings

The information below explains how to configure the transactional replication settings for Informix in the [Enable Transactional Replication wizard](#) or the [Transactional Log Settings dialog](#) available from the **Transactional Setup > Manage** option on the connection in the Metadata Explorer.

When replicating from Informix using mirroring or synchronization, two transactional modes are available: [Log Server Agent and Triggers](#).

## [Log Server Agent Settings](#)

## [Triggers Settings](#)

### Additional Step for Synchronization Replications

For synchronization replications only, you also need to obtain the user ID number for the user specified in the source connection. This number will be used to edit the `_config.ini` file for the Windows service that reads the Informix log. To obtain and use the user ID number:

1. Create the Windows service using the Install button as described above.
2. From the command line, run the program `InformixGetUserIDNumber.exe` in the DBMoto Install directory:  
`InformixGetUserID.exe <informix-server-name> <userid> <password>`

where:

`<informix-server-name>` is name of the informix server as defined in the source

`<userid>` the name of the user defined in the source connection

`<password>` the password for the user ID

The application outputs a number which must be manually inserted into the `_config.ini` file in the folder for the Windows service.

3. In the Windows Explorer, from the DBMoto install directory, go to the Windows service folder.
4. Edit the `_config.ini` file and locate the field `USER_ID_NUMBER`.
5. Set the value to the number returned from your call to `InformixGetUserIDNumber` above.
6. Save the file.
7. Start the Windows service.

### IBM PureData (Netezza) Transactional Replication Settings

The information below explains how to configure the Log Settings screen for IBM PureData in the [Enable Transactional Replication wizard](#) or the [Manage Transactional Log Settings dialog](#) available from the **Transactional Setup > Manage** option on the connection in the Metadata Explorer.

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.

## Log Reader Settings

There are no specific settings required for transactional replications from IBM PureData databases.

*Verify (available only in the Manage Transactional Log Settings dialog)*

Click **Verify** first to check that DBMoto is able to access and set up logging for the IBM PureData server.

## Microsoft SQL Server Transactional Replication Settings

The information below explains how to configure the transactional replication settings for SQL Server in the [Enable Transactional Replication wizard](#) or the [Manage Transactional Log Settings dialog](#) available from the **Transactional Setup > Manage** option on the connection in the Metadata Explorer.

When replicating from SQL Server using mirroring or synchronization, three transactional modes are available: [Log Reader, Log Server Agent and Triggers](#).

## Log Reader and Log Server Agent Settings

If you plan to define a mirroring or synchronization replication, you need to define two connections to the database: a non system administrator connection and a system administrator connection. The system administrator connection is used to set up the distributor and access the transaction log. However, a system administrator connection does not provide sufficient user information in the log to identify transactions for replication, so you also need the non system administrator connection.

### Notes:

1. When using SQL Server as a source in mirroring (or as a source or target in synchronization replications), SQL Server requires that tables have a primary key defined. If tables do not have a primary key, you can set up [trigger-based replication](#).
2. When using SQL Server for synchronization, the database should be set up with a recovery model other than Simple - either Full or Bulk-Logged recovery models are acceptable. This is because the Simple recovery model, the transaction log that is used to retrieve user ID information can be truncated, causing the synchronization algorithm to fail.

### *Publisher*



The name of the database server that you are using for the connection.

### *Distributor*

Initially, this field displays the name of the database server you are using for the connection. When you click **Verify** to see if a Distributor is installed for this database server, the value changes as follows:


- If a distributor exists for the server, the server name is displayed and two buttons are active



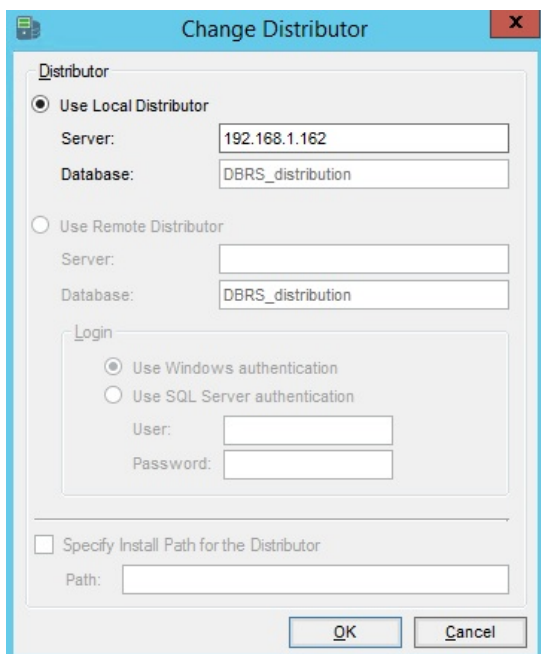
- The  Ellipsis button to the right of the Distributor field  
Click here to open the Change Distributor dialog and view the distributor values. You cannot edit values in the dialog unless you first exit the dialog, remove the installed distributor, then re-open the dialog.
- The **Remove** button is active so that you can remove the distributor in cases where you prefer to use a distributor defined for another accessible installation of SQL Server.
- If no distributor exists, the server name is displayed, and two buttons are active:
  - The  Ellipsis button to the right of the Distributor field  
Click here to open the Change Distributor dialog and set or create a distributor on a different SQL Server installation from the one containing the table(s) to be replicated (remote distributor).
  - The **Install** button is active so that you can install a distributor. If you choose this option to install a distributor in the current SQL Server installation, DBMoto creates a distributor called DBRS\_Distribution.

Be sure to install a distributor before closing the dialog.

### Change Distributor Dialog

This dialog can be accessed by clicking  next to the **Distributor** field.

- If you have a distributor installed in the SQL Server to which you are connecting for replication, the **Use Local Distributor** option is selected and values in the **Server** and **Database** fields reflect the SQL Server installation and distributor database. These values cannot be changed from the dialog. To change the distributor, you need to close the dialog, remove the local distributor using the Microsoft SQL Server console and reopen the dialog.



- If you do not have a distributor installed and want to set or create a distributor on a remote installation of SQL Server, select the **Use Remote Distributor** option and enter the SQL Server name (or an established alias for the SQL Server system), then the distributor database name. If the distributor does not exist, one will be created using the specified name. Provide a user name and password with appropriate privileges for creating the distributor in the remote installation of SQL Server.

NOTES: The following are limitations on the use of remote distributors:

- If problems arise when DBMoto attempts to create the remote distributor, you may need to enter the full pathname to the location where the distributor should be created. This is usually in the SQL Server folder that contains the Data and Log folders, for example, "C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL." The path has to be the complete path up to the SQL Server folder that contains the Data and Log folders.
- If problems arise because DBMoto is not able to locate the server name specified for a remote distributor, you can change the value of the Server field to use an established alias for the server.
- While you can set up a distributor on a remote installation of SQL Server, it is not currently possible to set a remote publisher for that distributor using the dialogs available in DBMoto. For example, if SQL Server installation A is used as a remote distributor for SQL Server installation B, it is not possible to set SQL Server installation C as a publisher for the remote distributor in server A. It is possible to configure this situation manually from the SQL Server management console.
- When a distributor is defined as a remote distributor for a SQL Server installation, if an attempt is made to use that distributor in its local installation, the operation will fail.

#### *Use Windows Authentication*

Select this option if you (or your system administrator) have set up your environment to use your Windows login ID to access SQL Server.

#### *Use SQL Server Authentication*

Select this option to use a SQL Server login ID and password. Provide a system administrator login and password.

#### *Verify (available only in the Manage Transactional Log Settings dialog)*

If you have made changes in the dialog, click Verify to make sure that the distributor is available and set up correctly.

#### [Log Server Agent Settings](#)

#### [Triggers Settings](#)

#### **MySQL Transactional Replication Settings**

The information below explains how to configure the transactional replication settings for MySQL in the [Enable Transactional Replication wizard](#) or the [Manage Transactional Log Settings dialog](#) available from the **Transactional Setup > Manage** option on the connection in the Metadata Explorer.

When replicating from MySQL using mirroring or synchronization, three transactional modes are available: [Log Reader](#), [Log Server Agent and Triggers](#).

### Log Reader Settings

Make sure that you have configured your MySQL server as explained in [MySQL Server Setup](#). You will not be able to complete the transactional setup without these settings.

#### *Case Sensitive Table Names*

Check this option if the MySQL database contains case-sensitive table names for tables that you wish to replicate.

#### *Verify (appears only in the Transactional Log Settings dialog)*

Click **Verify** first to see that the log options are correctly set up on the MySQL server. The **Verify** button checks to see if the binary log is enabled on the server and if the user has the right permissions to access to it.

### [Log Server Agent Settings](#)

### [Triggers Settings](#)

### Oracle Transactional Replication Settings

The information below explains how to configure the transactional replication settings for Oracle in the [Enable Transactional Replication wizard](#) or the [Manage Transactional Log Settings dialog](#) available from the **Transactional Setup > Manage** option on the connection in the Metadata Explorer.

When replicating from Oracle using mirroring or synchronization, three transactional modes are available: [Log Reader](#), [Log Server Agent and Triggers](#).

### Log Reader Settings

#### *Log Settings*

The Log Settings field displays the Oracle data source where the log is stored. This field is read-only and by default matches the data source specified in the source connection. If you are using the multitenant database architecture introduced in Oracle 12, and connecting to a pluggable database, you may need to modify the log location to match the Oracle root, or container, database. To modify the log location, click the [...] button.

Click [...] to open the [Change Log Settings dialog](#) to:

- Change the Oracle server where the log is stored (for Oracle 12 multitenant database installations).
- Set a path to an Oracle dictionary file (primarily for Oracle versions up to 9)
- Change the archived log settings

#### *Supplemental Log Minimal Level/Supplemental Log Database Level*

These fields are inactive until you have clicked **Verify** to check what type of supplemental logging is currently set for your Oracle source connection. For transactional replications where Oracle is a source database, DBMoto

requires supplemental logging to be enabled at least at the minimal level (and later at the table level) to record sufficient information about changes to the data in the table.

Click **Verify** to determine which type of logging is currently set. If you want to make changes to the current setting:

Choose **Minimal Level** to establish minimal supplemental logging at the database level, then, during replication setup, add logging for the specific table(s) involved in the replication.

Choose **Database Level** to establish supplemental logging for the entire database, including primary key and index information for all tables. If you have a large number of tables in your database, and they are not involved in DBMoto replications, you should be aware that supplemental logging can add unnecessary processing time/log information.

Click **Install** to make changes to the supplemental log settings.

Note that versions of DBMoto prior to 7.0.4 used database-level supplemental logging only.

#### *Verify (appears only in the Transactional Log Settings dialog)*

Click **Verify** to see if supplemental logging has been set up either at the database level or at the minimal level. DBMoto requires supplemental logging to be set at minimum at the table level for the table(s) involved in replication.

If supplemental logging has not been established, or you wish to make changes, select the type of logging then click **Install** to set it. To later remove the log settings from the server, use the [Disable Transactional Replication wizard](#) available from the **Transactional Setup > Disable** option on the connection in the Metadata Explorer.

### Oracle-Specific Log Server Agent Settings

#### *Use Log Container*

If your data source is an Oracle version 12 pluggable non-root database, you also need to complete the root database connection information:

#### *Server, User, Password*

The connection string for the root database. This is required because Oracle 12 does not keep redo log records for the entire database instance separately.

#### *Use Remote LSA*

This option is available for environments where it is preferable to read the Oracle log directly and involves the installation and configuration of an additional component on the Oracle server. Contact Hit Software via [the Help Center](#) for additional information.

## [Standard Log Server Agent Settings](#)

### [Triggers Settings](#)

### PostgreSQL Transactional Replication Settings

The information below explains how to configure the transactional replication settings for PostgreSQL in the [Enable Transactional Replication wizard](#) or the [Manage Transactional Log Settings dialog](#) available from the **Transactional Setup > Manage** option on the connection in the Metadata Explorer.

When replicating from PostgreSQL using mirroring or synchronization, one transactional mode is available: [Log Server Agent](#).

### Log Settings

#### *Replication Slot: Use Existing Slot*

Replication slots are channels that you open on the database to allow DBMoto to receive data changes from the log. While the default is to use a new slot specifically for a DBMoto connection, it is possible to reuse an existing slot if it is not consumed by any other connection. A replication slot has to be uniquely assigned to a single DBMoto connection.

#### *Replication Slot: Add New Slot*

DBMoto needs to define a “replication\_slot” for every connection that it uses to replicate from PostgreSQL. This option allows you to set up a new slot from DBMoto. However, the new slot will be added only if the maximum number of slots has not been exceeded. This value is set in the postgresql.conf file. Be aware that if slots are created then left unused, they still consume resources, so you should always manage replication slots carefully.

For more information on replication slots and the settings needed for transactional replications from PostgreSQL, see PostgreSQL Database Setup

#### *Verify (available only in the Manage Transactional Log Settings dialog)*

If you have made changes in the dialog, click Verify to make sure that the replication slots are set up correctly.

## [Log Server Agent Settings](#)

### Log Server Agent Settings

This screen is active only when you have selected Log Server Agent as the log type. It allows you to establish details for the Windows service and log files. It is displayed from [Enable Transactional Replication wizard](#) or the [Manage Transactional Log Settings dialog](#) available from the **Transactional Setup > Manage** option on the connection in the Metadata Explorer.

#### *Log Files Folder*

An existing folder where all the intermediate binary logs files are written. The folder is also used to contain trace files.

### *Prefix*

A prefix for all files created in the folder. This provides easy identification and management of files associated with your connection. The primary purpose for the prefix is to support the case when you configure more than one connection to use the same folder. In general, however, it is advisable to use different folders for different connections.

### *Log File Size*

The maximum size of each binary log file in megabytes.

### *Keep Max Files*

The maximum number of binary log files to keep. Combined with the log file size, this number needs to be large enough to make sure that all the files are read by DBMoto before they get deleted.

### *Trace*

Unchecked by default. When checked, enables tracing for diagnostics and problem reporting. Trace files are saved in the Log Server folder.

### *Windows Service Name*

#### **Prefix:**

This value is provided and cannot be changed. It allows you to easily identify the service in the Microsoft Windows Services tool.

#### **Name:**

Specify a unique name for the Windows service. Each connection runs its own instance of the DBMoto Log Server as a Windows service.

#### **Start service after completing the wizard:**

When checked, starts the service automatically after you click **Finish** to complete the wizard.

### *Start, Stop, Reset (appears only when managing the service via the [Manage Transactional Log Settings dialog](#))*

The Service can be started and stopped here or in the Windows Services tool. If stopped and then re-started, the Log Server resumes reading the database's binary log from the point where it was stopped.

If the database or binary log is reset, or if the Log Server has been stopped for too long and the database's binary logs are no longer available, first stop the service, then click **Reset** to reset the binary log position to where the Log Server can read again. The log position is reset to the current position in the database. Also reset the last transaction ID for all replications, or run a refresh replication to make sure they are up to date.

### **Trigger Transactional Replication Settings**

The information below explains how to configure the transactional replication settings for trigger-based replications in the [Enable Transactional Replication wizard](#) or the [Manage Transactional Log Settings dialog](#) available from the **Transactional Setup > Manage** option on the connection in the Metadata Explorer.

The information in this topic applies to **trigger-based replications** using any one of the following databases as a source in mirroring replications or as a source/target in synchronization replications:

IBM Db2 LUW, Db2 z/OS ([Transaction log-based replication for Db2 LUW](#) also available)

IBM Informix ([Transaction log-based replication](#) also available)

Microsoft SQL Server ([Transaction log-based replication](#) also available)

Microsoft Azure SQL Database

MySQL ([Transaction log-based replication](#) also available)

Oracle ([Transaction log-based replication](#) also available)

SAP HANA

SAP Sybase IQ

SAP Sybase SQL Anywhere

SAP Sybase ASE

Gupta SQL Base

If you plan to define a mirroring replication or a synchronization replication with any of the databases listed above, you need to fill out information in the Setup Info screen or dialog so that triggers can be created to log table changes for replication. Your system administrator needs to create and define appropriate table spaces and databases to hold the log tables. They should be large enough to handle the expected amount of replication data. The login you are using for the source connection should have appropriate authorizations to create tables. [More about trigger-based replication.](#)

**Note:** If using **IBM Db2** as a source database, the following versions are supported:

1. Db2 UDB v. 7.2 or higher
2. Db2 for OS390 v6 or higher

**Note:** If using **SQL Server triggers** option, the following restrictions apply for transactional replications:

- Functions with SQL Server 2000 or higher (except MSDE), including SQL Server 2005 Express
- Data types "image", "text" and "ntext" are not supported. If you create a transactional replication on a table with fields where these data types defined, the DBMoto Management Center warns you that the fields will not be replicated. In SQL Server 2005 and SQL Server Express, these types have been replaced by "varbinary(max)", "varchar(max)", and "nvarchar(max)". Microsoft recommends replacing the old data types with the new ones. The new data types are supported when using the Triggers option.
- The source table does not need a primary key set in SQL Server but it must have a primary key defined in DBMoto. See [Primary Key Settings for Mirroring and Synchronization](#) for an explanation on how to create a primary key in DBMoto.

**Note:** If using **SAP Sybase SQL Anywhere triggers** option, and you have triggers already defined on your source table, you can use the **Trigger Order** option to set the value that DBMOTO passes to the ORDER clause in the CREATE TRIGGER statement. In most cases, the default value of 100 works appropriately.



### *Master Table*

Either specify an existing qualified table name, or click **Change** to create a new table to hold general information about replication transactions including user name, timestamp, table name for each transaction.

There are two tables associated with each replication: a Master table, common to all replications using that connection, and a Log table for each replication source table. The Master table keeps track of all the transactions affecting the source tables and it records general transactional information.

Master and Log tables are created in the schema specified when you set the Master table name. You can choose a Master table name, or use the default `_DBM__MASTERLOG`. Log tables are automatically generated by DBMoto and the names are `_DBM__LOG_#`, where # is a number. The selected schema for the Master and Log tables must not contain other non-DBMoto tables with names `_DBM__LOG_#`. HiT Software recommends that you create a new schema to use specifically for the DBMoto Master and Log tables.

### *Tablespace*

HiT Software recommends that you assign a tablespace for the Master table and Log Tables so that it is easier to control log table sizes. If you leave this field blank, the default tablespace value for your login ID will be used. Your system administrator should be able to provide you with the appropriate value for this field.

If you are using IBM Db2, the value can be entered as `dbname.tablespace`.

### *Retention Time*

The amount of time in hours that a transaction is kept in the log tables. The default value is 72 hours. When the amount of time a transaction resides in the log exceeds the retention time, the transaction is permanently removed from the log tables. Tuning the retention time provides control over the size of the log tables.

### *Delete Block Size*

Based on the retention time, DBMoto deletes items from the log. This field specifies the maximum number of records to delete from the DBMoto log tables with a single SQL statement. The default value is 10,000 records. You do not typically need to edit this value.

### *Trigger Order*

For Sybase SQL Anywhere users only. See note above.

### *Lower-case Trigger Identifiers*

When checked, DBMoto creates objects for the trigger-based replications with lower case names. When unchecked, mixed case is used. This option should be checked if the database default case for identifiers is lower-case.

## **Export Replications Wizard**

[Existing Source Connection](#)>[Existing Target Connection](#)>[New Source Connection](#)>[New Target Connection](#)>[Set](#)



Use this wizard to create a new replication based on settings in an existing replication. The Export Replications wizard provides a way for you to use an existing replication definition with new source and/or target connections. For example, if you have created a replication that mirrors data from Oracle to SQL Server, and you wish to mirror the same data to a MySQL database, use the Export Replications wizard. First create the target connection for MySQL, then run the Export Replications wizard to copy the replication using the new target connection.

### **Existing Source Connection**

This screen lists all the source connections that you have created in DBMoto.

#### **Connection Name**

Choose a source connection name from the drop-down list. This source connection should match the source connection for the replication that you want to export. Fill out any additional fields required to identify the connection (e.g. Schema Name.)

### **Existing Target Connection**

This screen lists all the source connections that you have created in DBMoto.

#### **Connection Name**

Choose a target connection name from the drop-down list. This target connection should match the target connection for the replication that you want to export. Fill out any additional fields required to identify the connection (e.g. Schema Name.)

### **New Source Connection**

This screen lists all the source connections that you have created in DBMoto.

#### **Connection Name**

Choose a source connection name from the drop-down list. This source connection will be used as the source connection in the new replication that the wizard creates. Supply values for any additional fields required to identify the connection.

### **New Target Connection**

#### **Connection Name**

Choose a target connection name from the drop-down list. This target connection will be used as the target connection in the new replication that the wizard creates. Supply values for any additional fields required to identify the connection.

## Set Replications

This screen displays all replications matching the source and target connections you specified in the earlier [Existing Source Connection](#) and [Existing Target Connection](#) fields. Select the replications that you want to export. If the table names in the new replication will be different from the table names in the existing replication, click **Tables** to set the new table names.

## Summary

The summary lists the original replication name, the type of replication and the automatically-generated new replication name. You can rename the newly-created replication after the export process is complete by opening the [Replication Properties dialog](#).

## Metadata Connection Wizard

[Select Provider](#)>[Set Connection String](#)>[Select Qualifier](#)>[Define Metadata](#)>[Summary](#)

This wizard is optional and allows you to specify a connection to create metadata tables that hold all the information needed for your replication. By default, metadata is created in a Microsoft SQL Server CE database that is distributed with DBMoto.

### Select Provider

#### Metadata Name

Type a name to identify the metadata. This name appears in the Metadata Explorer as a way to group connections for a specific replication.

#### Database

Select the database for which you want to create a connection.

#### Provider

**Note:** Db2 running on mainframe and i (iSeries/AS400) cannot be used for the metadata connection. Choose a different database when defining your metadata connection.

Select a .NET data provider from the list. HiT Software recommends that you use the default Microsoft SQL Server CE provider so that the metadata tables are created for the included SQL Server CE database. However, you are free to use any database you want for metadata, with the exception of IBM Db2 for i or mainframe. If no .NET provider is available for the database, select either an OLE DB provider or an ODBC driver. By default, only default providers are listed in this dialog. If you want to see all providers for a specific database, uncheck the **Use Only Default Providers** option in the [Options dialog](#) available from the **Tools** menu.

.NET providers are preferred over ODBC drivers or OLE DB providers because, with DBMoto being a .NET application, it can take advantage of the use of native .NET providers based on the same architecture. If you use

ODBC or OLE DB connections, the .NET framework will need to add a software layer for every call you make to the driver, adding overhead. Besides, using 100% managed .NET providers, there is no risk of memory leaks because the .NET framework cleans up memory automatically using an internal garbage collector.

### Assembly

This field is displayed only if using a .NET data provider, and the provider has not been registered. Provide a path to the .NET data provider DLL.


### Set Connection String

#### Connection Properties

Note: If you are using the default SQL Server CE .NET Provider, the Data Source field provides a drop down list where you can select an existing local SQL Server CE database or create a new local database. SQL Server CE files have a .sdf extension, and you can find a default database in your DBMoto/SqlServerCE folder (metadata.sdf).

For all other providers, edit at least the **Required** connection properties by clicking in the property value field and typing a new value. The list of **Optional** properties for .NET and OLE DB providers contains the most commonly used properties for the providers. Edit these as needed. Note that some properties are displayed with default values (no bold text.) Any values that you add or edit are displayed in bold text. Check the documentation for your provider for a complete list of properties. You can set the value of the ExtendedProperties property to define additional property-value pairs. The syntax for defining property-value pairs is: prop1=val1;prop2=val2;...

If you are creating the metadata tables in SQL Server, note that each login has a default initial catalog. For example, the default initial catalog for a system administrator login is "master." If you do not want to use the default catalog, be sure to enter a catalog to use in the **Initial Catalog** field.

If you are using an ODBC driver, click in the ConnectionString value field and then click  to open the ODBC Driver Connection dialog. The contents of this dialog depend on the ODBC driver that you are using. Provide values for the dialog fields, then click OK to make a test connection and set the values in the DBMoto wizard.

### Edit

Click **Edit** to open a text entry window where you can paste or type a connection string for your provider. This is offered as an alternative to the Connection Properties grid, but should be used with great care because an error in the connection string can cause a connection to fail or to have unexpected properties. This window displays any connection information that you have already entered in connection string format. Note that default values are not displayed as part of the connection string.

### Test

Click **Test** to make sure that the connection correctly opens a database connection.

## Select Qualifer

This screen is not available if the database you have specified does not allow table qualifiers such as catalogs and schemas. If you do not select values for these fields, the default qualifiers will be used based on the connection properties provided in the [Set Connection String](#) screen.

## Define Metadata

### Create a new metadata database

Select this option to create DBMoto metadata tables using the connection properties specified in the [Set Connection String](#) screen,

### Restore the metadata set from a backup file

Select this option if you have previously saved a metadata set to a file using the DBMoto [Backup Metadata option](#).

### Open existing metadata database

Select this option to use an existing metadata set available from the connection properties specified in the [Set Connection String](#) screen,

## Summary

### Database name

The name of the database server

### Database version

The database server version

### Provider type

The description of the provider that you selected in the [Select provider](#) screen.

### Provider name

Whatever connection information is available for the provider you have chosen. In some cases, this field will be blank. In other cases, the provider DLL may be displayed.

### Selected Qualifier

Qualifier names (if any) selected in the [Select Qualifier](#) screen.

### Proceed with the definition of a source connection

When checked, the [Source Connection wizard](#) is displayed on completion of the Metadata Connection wizard. Use the Source Connection wizard to define a connection to the database that contains the tables or views you want to replicate.

## Opening the Metadata Connection Wizard

In the DBMoto Management Center, from the **Metadata** menu, choose **Add New Metadata**.

## Multiple Replications Wizard

[Define Replication Type](#)>[Select Source Connection](#)>[Source Log Info](#)> [Select Target Connection](#)>[Target Log Info](#)>[Set Replications](#)>[Scheduling](#)>[Summary](#)>[Processing](#)

This wizard is useful when you want to replicate all or many tables in the same database without defining specific replication needs for each table. It allows you to define multiple replications with a common replication type, common source and target connections and common scheduling needs.

### Define Replication Type

#### Replication Name

Type a base name for each replication. This name will be stored in the metadata and used to identify replications in the Replication Browser. For each replication that you create, a sequential number is added to the base name you have chosen. For example, if you type Pubs-SQLSrv-Access as the base name, the first replication that is created at the end of the wizard will be Pubs-SQLSrv-Access0, the second will be Pubs-SQLSrv-Access1, and so on. It is helpful to include information about the source and/or target tables in the replication name.

#### Description

Optional. Type a description for the replication. This information will be stored in the metadata and can be accessed when you need more information about the replication.

#### Use Group

Check this option to include the current set of replications as part of a group (recommended). You must [create the group](#) before running this wizard. Grouped replications do not have their own properties. Properties are set for the entire group so that all replications in the group are executed together. When using the Multiple Replications wizard to create replications for all or most tables in a database, it is recommended that you include them in a group so that all replications are executed together. This optimizes performance by reusing connections and limiting the need to access a transaction log (for mirroring and synchronization replications).

#### Replication Mode

Choose a replication mode from the options below.

##### *Refresh*

Mass replication is executed once based on the time setting specified in the [Scheduling](#) screen. This type of replication is sometimes referred to as a snapshot replication.

### *Mirroring*

Mass replication (refresh) can be executed once in order to fill the target records. Real-time mirroring based on recorded database transactions begins as soon as the refresh changes have taken place. The incremental replication is executed at user pre-determined intervals.

### *Synchronization*

Mass replication (refresh) can be executed once in order to fill the target records. Real-time bi-directional mirroring begins as soon as the refresh changes have taken place. The incremental replication is executed at user pre-determined intervals.

### **Select Source Connection**

This screen lists all the source connections that you have created in DBMoto.

#### **Connection Name**

Choose a source connection name from the drop-down list.

#### **Database Name**

Choose a database from the drop-down list.

#### **Owner Name**

Depending on the RDBMS that you are using as a source, this field may display schemas or libraries. Select an owner (or schema or library) from the drop-down list. Note that if you are replicating tables from the same database, but using tables with different owners, you will need to run the wizard multiple times, one for each owner.

### **Source Log Info**

This screen is displayed when you have set mirroring as your replication type. It allows you to specify transaction log information that is specific to the database from which you are replicating. For an explanation of fields in this screen, click on the link for your source database below.

[Microsoft SQL Server](#)

[IBM Db2 for i](#)

[Oracle](#)

MySQL

Trigger-based Replications

## Select Target Connection

### Connection Name

Choose a target connection name from the drop-down list. If no target connections are available, you need to exit the wizard and [create a target connection](#).

### Target Log Info

This screen is displayed when you have set synchronization as your replication type. It allows you to specify transaction log information that is specific to the database from which you are replicating. For an explanation of fields in this screen, click on the link for your database below.

[SQL Server](#)

[Db2](#)

[Oracle](#)

Trigger-based Replications

## Set Replications

### Replication List

The Replication List displays a list of possible source table-target table replication pairs, based on information gathered from the source connection. Note that tables in the Target Table column may not yet exist in the target database. By default, all replications in the list are selected for creation. If you proceed from this screen without making any changes, a replication object will be created for each source table-target table replication pair that is displayed. You can:

- Deselect a pair (i.e. the replication will not be created) by clicking in the checkbox to the left of the Source Table column.
- Deselect all pairs by clicking in the checkbox to the left of the column headings
- Select multiple pairs by holding down the <Control> or <Shift> key, then clicking the mouse on pairs, or a range of pairs. Use the **Check Selected Tables** button to create replications for those pairs.



- Deselect multiple pairs by holding down the <Control> or <Shift> key, then clicking the mouse on pairs, or a range of pairs. Use the **Uncheck Selected Tables** button to remove the pairs from the list of replications.



### Source Table Column

The Source Table column lists all the table objects found according to the source connection information you provided (database and owner/schema/library.)


### Source Type Column

The Source Type column displays the type of table object for the listed table. For example, if your environment consists of tables and views, this column allows you to distinguish between them, and even sort the list based on the type of table object.


### Target Table Column

The Target Table column displays all table names matching the source table names, even when the tables do not yet exist in the database. If tables exist in the target database but do not have names that match source table names, you can access these tables using a drop-down list for each replication pair.

Examine each source table-target table pair to determine if you want to proceed with creating a replication for the pair. Your options are:

- Deselect the pair so that no replication is created.
- Change the target table by clicking on the target table name and choosing the appropriate table from the drop-down list.
- Modify the field mappings for a source table-target table pair by clicking  to open the [Fields Mapping dialog](#). You can specify which fields to replicate, and whether to map values directly from source to target or to create an expression that modifies the target value.

### Mapping Column

Provides a button so that you can set up custom mapping for each pair. In cases where a table has not yet been defined on the target database, you are asked if you want to create the table before proceeding. If the target table already exists, click  to open the [Fields Mapping dialog](#).

### Mapping Rule

The default mapping rule used by DBMoto when creating replications is to map fields in source and target tables by name. However, you can modify this approach using the options available in the Mapping Rule field. The options are:

**Mapping By Name:** Maps each source field name to a matching target field name. If the target table exists and field names do not match, no mapping is performed and no replication for these fields will occur.

**Mapping By Ordinal:** Maps each source field column to a target field column based on its position in the table.

**Clear All Mapping:** Removes all established mappings, allowing you to create custom mappings for each table.



## Scheduling

### Enable Replication

Checked by default. When checked, the replication will begin at the scheduled time once [the Replicator has been launched](#). If you uncheck this option, and run the Replicator, the replication will not start at the specified time. A replication must be enable before it can be run.

### Start Time

Select a start date and time. When the Replicator is running, this time will be used to determine when to schedule the first replication.

#### Selecting a Date

You can select a date either by typing in the month, day and year, or by clicking the down arrow to view a scrollable monthly calendar.

To type in a date, first select the month, then type in a new value, then select the day and type in a new value and finally select the year and type in a new value.

If you click the down arrow, it displays the date that corresponds to the value currently set in the field. Use the left and right arrows to locate the month you want, then select a date by clicking on it. The window closes when you select a date and the date is displayed in the Start Time field.

#### Selecting a Time

To select a time, first select the hours, minutes or seconds, then type a new number or use the up and down arrows to increase or decrease the number.

### Execute Initial Refresh

Check this option to run a complete replication immediately the replication is enabled and the start time has been reached. Subsequent replications occur only as scheduled below.

For a synchronization replication, the initial refresh is always performed from the source connection to the target connection. Note that any transaction submitted during the time that the refresh is running might not be replicated. It is strongly suggested that you avoid updating the designated source and target tables until the refresh is done.

### Refresh Schedule

#### Run One Time Only

Select this option if you want to replicate data only once.

#### Run Recurrently

Select this option if you want to replicate data on a regular schedule. When you select this option, the Schedule button is activated. Click Schedule to open the [Scheduler dialog](#).

## Mirroring Schedule

### Run Continuously

Select this option to continuously mirror activity between the source and target database. The start of mirroring is determined by settings in the Scheduling screen. If you have enabled replication and set the start time, then mirroring will begin at the start time specified. If you have also checked the option to execute an initial refresh, then mirroring will begin once the database refresh is complete. If **Enable Replication** is not checked, neither the initial refresh or continuous mirroring will begin at the time specified in **Start Time**.

### Schedule Interruptions

Select this option if you need to schedule one or more interruptions to the mirroring schedule. When you select this option, the Schedule button is activated. Click **Schedule** to open the [Scheduler dialog](#).

## Summary

Displays the replication details that you have entered. Click **Back** to go back and change settings. Click **Start** to accept the replication settings and create replications for each selected source table-target table pair. Note that if you have installed the DBReplicator as a service and Enable Replication is checked in the Scheduling screen, replication will begin based on your settings in the Scheduling screen. If you did not install DBReplicator as a service, the replication will not run until you [start the Replicator](#) (and only if the Enable Replication option is checked.)

## Processing

This screen allows you to observe the creation of replication objects. When complete, you can review the replications in the Metadata Explorer by expanding the Replications node and selecting the [Replication Browser tab](#). You can also review the progress of running replications in the [Replication Monitor tab](#).

## Replication Group Wizard

[Define Group](#) > [Select Replications](#) > [Summary](#)

Use this wizard to create a group of replications. Groups allow you to set and change properties for all replications in the group from a single dialog. This is useful if you need to run replications for multiple tables at the same time but do not want to enter replication settings for each table involved.

Group replication properties are determined by the first replication that you add to the group. If you need to change the replication properties for a group, select the group then, from the right mouse button menu, choose **Edit Replications**.

### Define Group

#### Name

Type a name for the group. The group name is used in the [Replication Wizard](#) for adding a replication to a group.

## Description

(Optional) Add a description of the group to help you identify its contents.

## Select Replications

### Replication List

Displays the list of replications associated with the meta data where the group is being created. Check each replication that you want to include in the group. If you have not yet created any replications, click **Next** to go to the Summary screen. You can include replications in the group as you create them in the [Replication wizard](#).

When you add the first replication to a newly-created group, the properties of the replication become the properties of the group. Any subsequent replications added to the group take on the properties of the group. If you need to change the replication properties for a group, select the group then, from the right mouse button menu, choose **Edit Replications**.

## Summary

Displays the group details that you have entered. Click **Back** to go back and change settings. Click **Finish** to accept the settings, create the group and close the wizard.

## Replication Wizard

[Define Replication Type](#)>[Source Connection](#)>[Source Log Info](#)>[Target Connection](#)>[Target Log Info](#)>[Set Mapping Info](#)>[Scheduling](#)>[Summary](#)

## Define Replication Type

### Replication Name

Type a name for the replication. This name will be stored in the metadata and used to identify the replication in the Replication Browser. It is helpful to include information about the source and/or target tables in the replication name.

### Description

Optional. Type a description for the replication. This information will be stored in the metadata and can be accessed when you need more information about the replication.

### Use Group

Check this option to include the current replication as part of a group. You must [create the group](#) before running this wizard. Grouped replications do not have their own properties. Properties are set for the entire group, allowing you to quickly and easily define replications for multiple tables.

## Replication Mode

Choose a replication mode from the options below.

### *Refresh*

Mass replication is executed once based on the time setting specified in the [Scheduling](#) screen.

### *Mirroring*

Mass replication (refresh) can be executed once in order to fill the target records. Real-time mirroring begins as soon as the refresh changes have taken place. The incremental replication is executed at user pre-determined intervals.

### *Synchronization*

Mass replication (refresh) can be executed once in order to fill the target records. Real-time bi-directional mirroring begins as soon as the refresh changes have taken place. The incremental replication is executed at user pre-determined intervals.

## Source Connection

This screen lists all the source connections that you have created in DBMoto.

### Source Name

Choose a source connection name from the drop-down list.

### Source Table

Choose a source table from the drop-down list.

### Open Table

Opens the [Execute SQL Command dialog](#) that displays the contents of the source table for your review. Note that if you execute a SQL query from this dialog, it does not affect any source, target or replication settings.

## Source Log Info

This screen is displayed when you have set mirroring as your replication type. It allows you to specify transaction log information that is specific to the database from which you are replicating. For an explanation of fields in this screen, click on the link for your source database below.

[Microsoft SQL Server](#)

[IBM Db2 for i](#)

[Informix](#)

[Oracle](#)

[MySQL](#)

## [Trigger-based Replications](#)

### Target Connection

#### Target Name

Choose a target connection name from the drop-down list. If no target connections are available, you need to exit the wizard and [create a target connection](#).

#### Target Table

Choose a target table from the drop-down list. If no tables are displayed, you can type in a table name. If the table does not yet exist in the target database, you are asked if you want to create the table. Alternatively, you can exit the wizard and either [select tables](#) for the target connection, or [create a target table](#). Creating a target as part of the Replication wizard can save time, but creating a table using the Create Target Table wizard gives you more control over the target table.

#### Open Table

Opens the [Execute SQL Command dialog](#) that displays the contents of the target table for your review. Note that if you execute a SQL query from this dialog, it does not affect any source, target or replication settings.

### Target Log Info

This screen is displayed when you have set synchronization as your replication type. It allows you to specify transaction log information that is specific to the database from which you are replicating. For an explanation of fields in this screen, click on the link for your database below.

[SQL Server](#)

[Db2](#)




[Oracle](#)

[Trigger-based Replications](#)






### Set Mapping Info

By default, source fields are mapped to target fields with the same column names. Use this screen to modify the mappings, either by removing mappings, setting mappings to different fields, or mapping fields to expressions. You can also leave the default mappings at this stage and edit them later using the [Replication Properties dialog](#).

The top part of the screen shows source and target table fields. Arrows indicate the mappings that have been created automatically. When editing the mappings using the toolbar, you can map a source field to zero, one or more target fields, and map a target field to only one source field. If you select a source column already mapped, the new mapping will be added to the existing mappings. If you select a target column that has already been mapped, that mapping will be overwritten.

The lower part of the screen lists the target fields on the left side, with an icon that shows if the column is mapped to a field (, unmapped  or mapped to an expression . Scrolling to the right, you can see the exact mapping for the target field.

The toolbar displays the following icons.

	<p><b>Define Mapping</b></p> <p>Select a source or target field in the list and choose one of the following options from the drop-down list:</p> <p><b>Map to field:</b> Opens the Edit Mapping dialog, displaying the current mapping, if any. If you selected a source field, choose the target field from the drop-down list on the right side of the dialog. Note that a source column can be mapped to zero, one or more target columns (see <b>Remove Mapping</b> below to unmap a source column.) If you selected a target column, choose the source field from the drop-down list on the left side of the dialog. A target column can be mapped to only one source column. Note that if a target column is already mapped and you select another mapping, the new mapping overwrites the existing mapping.</p> <p><b>Map to Expression:</b> Available when a target field is selected. Opens the <a href="#">Expression Generator</a> so that you can map the selected target field to an expression.</p> <p><b>Remove Mapping:</b> Available when a target field is selected. Removes a mapping to the selected field.</p>
	<p><b>Show/Hide Arrows Panel</b></p> <p>A toggle to display or hide the arrows between source and target fields. The arrows provide a graphical representation of mappings between fields. No arrows are displayed when a field is mapped to an expression.</p>
	<p><b>Show/Hide Extended Columns in Fields Lists</b></p> <p>A toggle to display or hide column information for source and target fields. The default column information consists of field name, position and type. Additional column information consists of size, precision, scale, description, null values, identity and default value.</p>
	<p><b>Mapping Direction</b></p> <p>Active when defining a synchronization replication. Allows you to view the mapping with arrows reversed from target to source, thereby demonstrating mappings that are not necessarily symmetrical with the mappings from source to target.</p>
	<p><b>Automatic Mapping</b></p> <p>By default, fields in the source and target tables are mapped according to the field name, where all matching names are matched. Use this menu to select instead to map by column ordinal numbers, or using a custom script function. The Automatic Mapping menu displays 3 entries: Mapping by Name,</p>

Mapping by Ordinal and Custom Mapping. The Custom Mapping option is inactive unless you have defined one or more [mapping functions](#) as part of a [global script](#).

## Scheduling

### Enable Replication

Checked by default. When checked, the replication will begin at the scheduled time once [the Replicator has been launched](#). If you uncheck this option, and run the Replicator, the replication will not start at the specified time. A replication must be enable before it can be run.

### Execute Initial Refresh

Checked by default. When checked, DBMoto will first perform a complete snapshot replication of the specified source data to the target table. This option is important for situations where the data on the target does not yet exist, or may be out of date. The progress of the Initial Refresh operation can be tracked in the [Replication Monitor](#).

### Start Time

Select a start date and time. When the Replicator is running, this time will be used to determine when to schedule the first replication.

#### Selecting a Date

You can select a date either by typing in the month, day and year, or by clicking the down arrow to view a scrollable monthly calendar.

To type in a date, first select the month, then type in a new value, then select the day and type in a new value and finally select the year and type in a new value.

If you click the down arrow, it displays the date that corresponds to the value currently set in the field. Use the left and right arrows to locate the month you want, then select a date by clicking on it. The window closes when you select a date and the date is displayed in the Start Time field.

#### Selecting a Time

To select a time, first select the hours, minutes or seconds, then type a new number or use the up and down arrows to increase or decrease the number.

### Execute Initial Refresh

Check this option to run a complete replication immediately the replication is enabled and the start time has been reached. Subsequent replications occur only as scheduled below.

For a synchronization replication, the initial refresh is always performed from the source connection to the target connection. Note that any transaction submitted during the time that the refresh is running might not be

replicated. It is strongly suggested that you avoid updating the designated source and target tables until the refresh is done.

## Refresh Schedule

### Run One Time Only

Select this option if you want to replicate data only once.

### Run Recurrently

Select this option if you want to replicate data on a regular schedule. When you select this option, the Schedule button is activated. Click Schedule to open the [Scheduler dialog](#).

## Mirroring Schedule

### Run Continuously

Select this option to continuously mirror activity between the source and target database. The start of mirroring is determined by settings in the Scheduling screen. If you have enabled replication and set the start time, then mirroring will begin at the start time specified. If you have also checked the option to execute an initial refresh, then mirroring will begin once the database refresh is complete. If **Enable Replication** is not checked, neither the initial refresh or continuous mirroring will begin at the time specified in **Start Time**.

### Schedule Interruptions

Select this option if you need to schedule one or more interruptions to the mirroring schedule. When you select this option, the Schedule button is activated. Click **Schedule** to open the [Scheduler dialog](#).

## Summary

Displays the replication details that you have entered. Click Back to go back and change settings. Click Finish to accept the replication settings and close the wizard. Note that the replication will not run until you [start the Replicator](#) (and only if the Enable Replication option is checked.)

## Oracle Redo Log Information

### Service Name

This is a unique identifier for your Oracle Server. The value is obtained automatically from your Oracle database server and cannot be modified.

### Dictionary File

If using Oracle 9 or later, you can leave this field blank to use the online dictionary. If you prefer to supply a dictionary file, provide a path and dictionary file name on your Oracle database server. The online dictionary/dictionary file contains table information that is used in replication.

### Transaction ID

The ID for the transaction at which you want to start replication. If you want to change the transaction ID, click ... to open the Read Point dialog. In this dialog, you can either retrieve the current transaction or the transaction for a



specified date and time. If you enter a date and time, DBMoto retrieves the first transaction after the time entered. This information is available and can be changed in the [Replication Properties dialog](#) after the wizard is completed.

### Transaction Timestamp

The timestamp for the transaction above.

### 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.

### IBM Db2 for i Journal Information

#### 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 [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 journal during replication. For example, if the setting is 90 seconds, DBMoto will check the journal 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.

## SQL Server Transaction Log Information

### Distributor

A distributor, DBRS\_Distributor, is created by DBMoto to manage a log that keeps track of transactions for the source table. This is typically done when setting up the connection in the [Setup Info screen](#), although you can also set or modify the distributor or system administrator login information in the [Connection Properties dialog](#).

To create the distributor, your source connection should have system administrator privileges. If you encounter connection problems at this stage, you need to cancel the Replication Wizard and change the [Source Connection Properties](#) by selecting the source connection in the Metadata Explorer and choosing **Connection Properties** from the right mouse button menu.

### Publication

If this field is left unedited, DBMoto creates a publication in SQL Server called DBMotoPUB\_ followed by the source database name. The publication is used to identify the tables involved in replication. One publication per database is created, so that several replications may use the same publication. To create the publication, your connection should have system administrator privileges as described above.

You can choose to edit the field and provide an existing SQL Server Publication Name. This approach is useful if you want to reuse replication objects which are already defined in the database and avoid creating additional work for the server in generating log information on specific tables.

### Transaction ID

The ID for the transaction at which you want to start replication. If you want to change the transaction ID, click ... to open the Read Point dialog.

In the Transaction Read Point dialog, you can either retrieve the current transaction or the transaction for a specified date and time. If you enter a date and time, DBMoto retrieves the first transaction after the time entered. This information is available and can be changed in the [Replication Properties dialog](#) after the wizard is completed.

### Transaction Timestamp

The timestamp for the transaction above.

### 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.

### Informix Transaction Log Information

#### Log File Name

The log file name is provided automatically and cannot be changed. Click **Read** in the Transaction ID field to open the Transaction Read Point dialog. When you select how to set the transaction ID and click **OK**, the log file name is automatically added.

#### Log File Number

The log file number is provided automatically and cannot be changed. Click **Read** in the Transaction ID field to open the Transaction Read Point dialog. When you select how to set the transaction ID and click **OK**, the log file number is automatically added.

#### Transaction ID

The ID for the transaction at which you want to start replication. If you want to change the transaction ID, click **Read...** to open the Read Point dialog. In this dialog, you can either retrieve the current transaction or the transaction for a specified date and time. If you enter a date and time, DBMoto retrieves the first transaction after the time entered. This information is available and can be changed in the [Replication Properties dialog](#) after the wizard is completed.

#### Transaction Timestamp

The timestamp for the transaction above.

#### 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.

### Restore Metadata Wizard

[Select backup file](#)>[Select Replications](#)>[Map Connections](#)>[Set Replication Details](#)>[Map Groups](#)>[Summary](#)>[Processing](#)



Use this wizard to create a new replication based on settings in an existing replication. The Export Replications wizard provides a way for you to use an existing replication definition with new source and/or target connections. For example, if you have created a replication that mirrors data from Oracle to SQL Server, and you wish to mirror the same data to a MySQL database, use the Export Replications wizard. First create the target connection for MySQL, then run the Export Replications wizard to copy the replication using the new target connection.

### Select Backup File

Locate the metadata backup file you previously saved. You can save a metadata backup file using the Backup Metadata option on the Metadata menu in the DBMoto Management Center. See [.Backing Up and Restoring Metadata](#) for more information.

### Select Replications

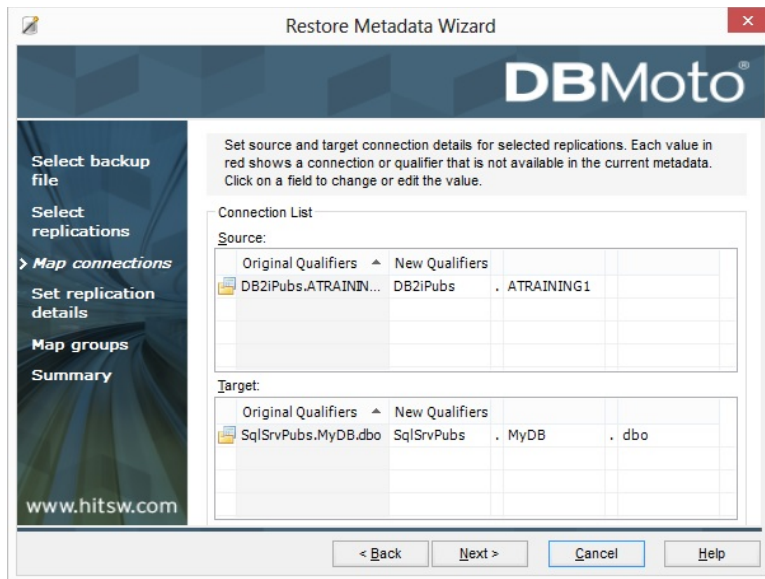
This screen lists all the replications defined in the metadata that you have selected. All replications in the list are checked by default. All checked replications will be restored. You can perform the following operations on the list of replications:

- Click in the checkbox in the title bar to remove the checkmark for all replications (if the box is already checked.)
- Click a checked replication to remove the checkmark for that replication only
- Click in the checkbox in the title bar to check all replications (if the box is not already checked.)
- Click an unchecked replication to set a check mark
- Select a group of unchecked replications using the Control or Shift keys with the mouse, then click the  **Check Selected** icon to check all selected replications
- Select a group of checked replications using the Control or Shift keys with the mouse, then click the  **Uncheck Selected** icon to remove the checkmark for all selected replications

When you have checked all the replications that you want to restore, click **Next** to set connection details for each one.

### Map Connections

This screen displays a list of all source and target connection details used by the original replications you selected in the [Select Replications](#) screen.



All source and target connections used in your selected replications are listed.

### Original Qualifiers

This column displays the connection information for the source and target tables obtained from the metadata backup file you identified in the **Select backup file** step. The data in this column cannot be modified.

### New Qualifiers

This set of columns initially displays the same information as the **Original Qualifiers** column. However, you can modify values in these columns to identify new connections for the replications. Values in red show that the connection does not exist in the current metadata. You should modify these values to successfully complete the restore operation.

To accommodate the different sets of qualifiers required by each database, the first field shows the name of the connection in DBMoto, and subsequent fields show required qualifiers for that connection, each separated by a period.

To modify values, click in the field and choose a new value from the drop-down list. In all fields, except for the connection name, you can also type in a new value. If a blank field is preceded by a period, you should supply a qualifier value in that field.

When you have selected new connection details, click **Next** to apply those connection details to the replications you have selected to restore.

### Set Replication Details

This screen displays replications with source and target connections modified based on your changes in the [Map connections](#) screen. You can make further changes for each replication in this screen by using the drop-down lists in each field. The Tables button provides access to source and target table qualifier information.

## Map Groups

This screen allows you to set groups for restored replications.

## Summary

The summary lists all the replications that will be restored, including the original replication name and its replication details as well as the restored replication name and its replication details. Review the summary before clicking **OK** to begin restoring the metadata information to your current metadata.

## Processing

Click **Start** to begin restoring metadata information. The progress bar shows the current status of the operation and messages are reported in the text area. When the restore operation is complete, review messages and click **Close** to exit the wizard.

## Scripts

### Expression Generator

The Expression Generator lets you build an expression to use when mapping a target field. It can also be used from the [Global Script Editor](#) and the [Replication Script Editor](#) to help write scripts to use during replication. Expressions can be written using Visual Basic (the default) or C# by setting the scripting language in the [Global Script Editor](#), accessible from the right mouse button menu on the metadata.

### Using the Expression Generator for Mapping Target Fields

You can map a target field to a source field or an expression when setting up your replication. Only fields that are mapped will be replicated.

You might want to map a target field to an expression when the source table does not contain a column that exactly matches the data you want to see in the target column. For example, in a source table where FirstName and LastName are two separate columns, but in the target table, there is a single column, Name, that should contain both the FirstName and LastName values. Use an expression to concatenate values from the two columns.

Either type in your expression or navigate through the trees to build your expression.



The functions and operators available in the Expression Builder are those typically available for most .NET providers. If you are not sure what functions and operators to use, check your .NET provider and RDBMS documentation. If you use a function or operator that is not supported by your .NET provider or RDBMS, the mapping will fail during replication. DBMoto additionally provides some [predefined values](#) that help you to define an expression quickly.

### Using the Expression Generator to Help Write Scripts


In both the [Global Script Editor](#) and the [Replication Script Editor](#), the Expression Generator button opens the Expression Generator so that you can use it to find functions and operators to use in the scripts. You can build parts of the script using the Expression Generator. Then, when you close it, the information is displayed in the Script Editor.

## Opening the Expression Generator


To write an expression to use when mapping fields from source to target tables:

- In the Replication wizard **Mapping info** screen, click  **Define Mapping** and choose **Map to Expression** to open the Expression Generator.
- On the [Replication Properties dialog](#) General tab, click **Mapping** to open the [Fields Mapping dialog](#). Click  **Define Mapping** and choose **Map to Expression** to open the Expression Generator.

To write an expression to use as part of a replication script:

- In the [Replication Properties dialog](#), check the Use Script option. .Click Script to open the [Table Script Editor](#). Click  to open the Expression Generator.

To write an expression to use as part of a global script:

- In the DBMoto Management Center Metadata Explorer, select the metadata for which you would like to write a script.
- From the right mouse button menu, choose **Global Script**.
- In the Global Script Editor, click .

## Replication Script Editor

Use this editor to write a script that is run when a specific event occurs during replication. The script should consist of one or more functions selected from the drop-down lists at the top of the Editor. The drop-down list on the left displays the categories for which you can define an event handler. The Editor provides [editing and formatting features](#) that include automatic indentation, automatic coloring for syntax, numeric values, comments, “code folding”, and keyboard shortcuts.

Category	Events	Event Type	Explanation
Refresh	<a href="#">onBeforeTruncate</a> <a href="#">onAfterTruncate</a> <a href="#">onBeforeRefresh</a> <a href="#">onAfterRefresh</a> <a href="#">onPrepareRefresh</a>	<a href="#">Writer</a> <a href="#">Writer</a> <a href="#">Writer</a> <a href="#">Writer</a> <a href="#">Reader</a>	Available events apply to refresh replications only.
LogReader	<a href="#">onPrepareMirroring</a> <a href="#">onBeforeMirroring</a>	<a href="#">Reader</a> <a href="#">Reader</a>	Available events apply only to mirroring and synchronization replications. onReceiverChanged applies



	<a href="#">onAfterMirroring</a> <a href="#">onReceiverChanged</a>	<a href="#">Reader</a> <a href="#">Reader</a>	only to replications involving System i/iSeries/AS400.
Record	<a href="#">onBeforeMapping</a> <a href="#">onAfterMapping</a> <a href="#">onBeforeExecute</a> <a href="#">onAfterExecute</a>	<a href="#">Reader</a> <a href="#">Reader</a> <a href="#">Writer</a> <a href="#">Writer</a>	Available events apply to each record that is considered for replication. In the case of refresh replications, the events would apply to every record in the source/target table. In the case of mirroring and synchronization replications, the events would apply to records found in the transaction log.
Replication	<a href="#">onCriticalError</a> <a href="#">onConflict</a> <a href="#">onLateConflict</a>	<a href="#">Writer</a> <a href="#">Reader</a> <a href="#">Reader</a>	The onCriticalError event applies to all replication types. It has been replaced by global events <a href="#">Record_onExecuteError</a> and <a href="#">Replication_onError</a> . onConflict and onLateConflict should be used to <a href="#">resolve conflicts</a> only for synchronization replications.

The right drop-down list displays a list of appropriate [events](#) for which you can write a script. If no function was selected from the left side, the list remains empty.

The default Visual Basic template content of the Replication Script Editor looks as follows. However, it is possible to write the script using C# instead of Visual Basic by setting the scripting language in the [Global Script Editor](#), accessible from the right mouse button menu on the metadata.

```
Imports Microsoft.VisualBasic
Imports DBMotoPublic
Imports DBMotoScript
Imports DBRS.GlobalScript
```

```
Namespace DBRS
    Public Class ReplicationScript : Inherits IReplicationScript
    End Class
End Namespace
```

When you choose an event to add to the script, it is always added directly below the ReplicationScript class as in the example below.

```
Imports Microsoft.VisualBasic
Imports DBMotoPublic
Imports DBMotoScript
Imports DBRS.GlobalScript
Namespace DBRS
    Public Class ReplicationScript : Inherits IReplicationScript
        Public Overrides Sub Record_onBeforeMapping(recSource As IRecord, ByRef
AbortRecord As Boolean)
```








```

End Sub
End Class
End Namespace

```

Add code to manage the event. You can include functions from other libraries in the script code, but be sure to add the library to the list of Imports, add the library location using the References dialog and set up a namespace for the library.

You can access the References dialog by clicking  in the toolbar below the script.

	Checks the script syntax. If you do not do this in the script editor, you may encounter errors when the replication is running.
	Opens the <a href="#">Expression Generator</a>
	Opens the References dialog so that you can add the location of any dlls that you are using from your script. You typically need to add dlls to the References dialog and to the script Imports when they do not belong to the default Microsoft .NET environment.
	Creates a comment for selected text
	Uncomments selected text

### Editing and Formatting Features

1. Automatic indentation based on syntax
2. Colors to identify specific code elements: syntax in blue, numeric values in pink and comments in green
3. Code folding where sections of code can be collapsed or expanded for ease of reading. The editor displays a green bar on the left side with +/- signs. Click - to collapse a section of code, and click + to expand a section of code.
4. Common editing operations using keystroke combinations:

<SHIFT>Left, Right, Up, Down, Home, End, PageUp, PageDown	Move caret with selection
<CTRL>C <CTRL>V <CTRL>X	Copy, paste, cut
<CTRL>A	Select all text

<CTRL>Z, <CTRL>Y	Undo/Redo
Tab <SHIFT>Tab	Increase/decrease left indent of selected range
<CTRL>Home <CTRL>End	Go to first/last char of the text
<SHIFT><CTRL>Home, <SHIFT><CTRL>End	Go to first/last char of the text with selection
<CTRL>- <SHIFT><CTRL>-	Navigate backward/forward
<CTRL>U <SHIFT><CTRL>U	Convert selected text to upper/lower case
<INSERT>	Switch between Insert Mode and Overwrite Mode
<CTRL>Backspace <CTRL>Del	Remove word left/right
<ALT>Mouse, <ALT><SHIFT>Up, Down, Right, Left	Enable column selection mode
<ALT>Up <ALT>Down	Move selected lines up/down
<SHIFT>Del	Remove current line
<ESC>	Close all opened tooltips, menus and hints
<CTRL>Wheel	Zoom
<CTRL>Up <CTRL>Down	Scroll Up/Down

<CTRL>NumpadPlus, <CTRL>NumpadMinus <CTRL>0	Zoom in, zoom out, no zoom
---	----------------------------

### Opening the Replication Script Editor

On the [Replication Properties dialog](#) General tab, check the Use Script option. Click **Script** to open the Replication Script Editor.



### Global Script Editor

In this dialog, you can write a script that defines one or more functions. Typically, global scripts define general user functions that need to be called from inside a [replication script](#) or an [expression script](#). See [Writing a Function to use in Scripts and Expressions](#) for details on how to write and call a global function. For additional examples of useful global script functions, check [Global Script Function Examples](#).


The Global Script Editor provides the following features.





5. Choice of languages for all scripts between VB and C# using the Script Language list at the bottom of the editor.
6. [Editing and formatting features](#) that include automatic indentation, automatic coloring for syntax, numeric values, comments, “code folding”, and keyboard shortcuts.
7. Access to the [Expression Generator](#) to identify functions and build expressions.
8. [Script Compiler](#)

To use the Global Script Editor:

1. Select the programming language for the script. The default language is Visual Basic, but you can choose C# using the Script Language list at the bottom of the Script Editor.
2. Type the script into the editor pane.
3. Use the [editor formatting and navigation features](#) to aid in writing and correcting the script.
4. Use the [Expression Generator](#)  as needed to identify functions and syntax that you can use in the script.
5. Use [Global Script Functions](#) in your script if needed.
6. Use the Compile button  to check your script syntax. Any script problems are displayed in a separate dialog.
7. Correct any syntax errors before attempting to run a replication.

### Editor Commands

	Checks the script syntax. If you do not do this in the script editor, you may encounter errors
---	--

	when the replication is running.
	Opens the <a href="#">Expression Generator</a>
	Opens the References dialog so that you can add the location of any dlls that you are using from your script. You typically need to add dlls to the References dialog and to the script Imports when they do not belong to the default Microsoft .NET environment
	Creates a comment for selected text
	Uncomments selected text
Script Language	Provides a drop-down menu to choose between VB and C# as the language for all scripts and expressions using the Global Script Editor, <a href="#">Replication Script Editor</a> and <a href="#">Expression Generator</a> . The selected language is set in the DBMoto metadata and used for all scripts associated with the metadata.

### Editing and Formatting Features

- Automatic indentation based on syntax
- Colors to identify specific code elements: syntax in blue, numeric values in pink and comments in green
- Code folding where sections of code can be collapsed or expanded for ease of reading. The editor displays a green bar on the left side with +/- signs. Click - to collapse a section of code, and click + to expand a section of code.
- Common editing operations using keystroke combinations:

<SHIFT>Left, Right, Up, Down, Home, End, PageUp, PageDown	Move caret with selection
<CTRL>C <CTRL>V <CTRL>X	Copy, paste, cut
<CTRL>A	Select all text
<CTRL>Z, <CTRL>Y	Undo/Redo
Tab <SHIFT>Tab	Increase/decrease left indent of selected range

<CTRL>Home <CTRL>End	Go to first/last char of the text
<SHIFT><CTRL>Home, <SHIFT><CTRL>End	Go to first/last char of the text with selection
<CTRL>- <SHIFT><CTRL>-	Navigate backward/forward
<CTRL>U <SHIFT><CTRL>U	Convert selected text to upper/lower case
<INSERT>	Switch between Insert Mode and Overwrite Mode
<CTRL>Backspace <CTRL>Del	Remove word left/right
<ALT>Mouse, <ALT><SHIFT>Up, Down, Right, Left	Enable column selection mode
<ALT>Up <ALT>Down	Move selected lines up/down
<SHIFT>Del	Remove current line
<ESC>	Close all opened tooltips, menus and hints
<CTRL>Wheel	Zoom
<CTRL>Up <CTRL>Down	Scroll Up/Down
<CTRL>NumpadPlus, <CTRL>NumpadMinus <CTRL>0	Zoom in, zoom out, no zoom

## Opening the Global Script Editor

- In the DBMoto Management Center Metadata Explorer, select the metadata for which you want to create a global script.
- From the right mouse button menu, choose **Global Script**.

## Add New Server Dialog

In addition to the local installation of DBMoto, you can view and manage remote installations of DBMoto from the Management Center. The Add New Server dialog allows you to add a remote DBMoto server to the Management Center.

### Server Name

Type the network name of the server you wish to add. The system should already have an installed copy of DBMoto with the DBMoto Server Agent running.

### Server Address

Type the IP address for the server.

### Port

In most cases, you do not need to change the default setting. If you have specific requirements for a different port number for the DBMoto Server, enter a new value here. You will also need to edit a DBMoto configuration file and set the port number there.

1. In your DBMoto installation directory, edit the file `dbmoto.server.config`.
2. Scroll to the entry for `AgentIPPort`:

```
...  
<optionList>  
...  
  <AgentIPPort>58361</AgentIPPort>  
...
```

3. Edit the port value.

### Authentication

Select the option below that matches your user settings for the new server. The options are:

#### AnonymousAccess

No login details are requested from the user. This is the default setting for installations where DBMoto security has not been enabled. To enable DBMoto security, go to the [User Settings dialog](#).

#### DBMoto Authentication

If DBMoto security has been enabled via the [User Settings dialog](#), and you have been directed by your DBMoto

administrator to use DBMoto Authentication, select this option, then type your user name and password in the fields below. DBMoto authentication requires the use of a X.509 certificate. Verify that a [certificate has been installed correctly](#) before completing this dialog. If you do not have a correctly installed certificate, you may not be able to use the DBMoto Management Center.

### Windows Authentication

If DBMoto security has been enabled via the [User Settings dialog](#), and you have been directed by your DBMoto administrator to use Windows Authentication, select this option. DBMoto will verify that your Microsoft Windows login and ID are legitimate before allowing you to continue working in the DBMoto Management Center.

### Certificate Authentication

If Certificate has been enabled via the [User Settings dialog](#), and you have been directed by your DBMoto administrator to use Certificate Authentication, select this option. Certificate authentication requires the use of a X.509 certificate. Verify that a [certificate has been installed correctly](#) before completing this dialog. If you do not have a correctly installed certificate, you may not be able to use the DBMoto Management Center.

### User Name

Active only when DBMoto Authentication has been selected. Enter the user name assigned to you by the DBMoto administrator. If you do not know your user name, verify login information with your DBMoto administrator or try setting the Authentication value to AnonymousAccess (no security) or Windows Authentication (uses your Microsoft Windows login and password.)

### Password

Active only when DBMoto Authentication has been selected. Enter the password assigned to you by the DBMoto administrator. DBMoto will verify that your DBMoto login and ID are legitimate before allowing you to continue working in the DBMoto Management Center.


If you do not know your password, verify login information with your DBMoto administrator or try setting the Authentication value to AnonymousAccess (no security) or Windows Authentication (uses your Microsoft Windows login and password.)

### Opening the Add New Server Dialog


In the Metadata Explorer, select the DBMoto root node, then choose **Add New Server** from the right mouse button menu.

## Connection Properties Dialog

This dialog reports connection properties for source and target connections. Some [source and target connection properties](#) are editable. These properties are typically set in the [Source](#) or [Target](#) connection wizards when setting up a replication.

NOTE: The Management Center window contains a View Properties pane. If you select a source or target connection in the Metadata Explorer, connection properties are displayed in the View Properties pane. However, to edit properties, you must open the Connection Properties dialog using the  **Show Properties** icon in the View Properties toolbar.

## Source and Target Connection Properties

- **Connection Name**  
The name of the connection that you provided when creating the connection.
- **Provider Assembly**  
If using a .NET data provider, and the provider has not been registered, provide a path to the .NET data provider DLL. Otherwise, this field should display the DLL for your registered provider.
- **Use Local Assembly**  
Default value is false. Set to true if you have not registered your provider and you are providing a pathname to the provider DLL.
- **DBVersion**  
The database version number retrieved from the provider.
- **Connection Class**  
The class for your provider.
- **DBType**  
The DBMS name from an internal list provided by DBmoto
- **Connection String**  
Enter a connection string to use in connecting to the database. The connection string should follow the format required by your .NET data provider, ODBC driver or OLE DB Provider. Click  to open a dialog where you can enter connection string values or edit the connection string in an input dialog.

## Dynamic Properties

- **Max Number of Concurrent Connections**  
Restricts the number of connections that DBMoto can make to the database server. This is a safeguard in situations where you have many replications using the same database server, but do not want to overwhelm the database server with concurrent connections from DBMoto, possibly affecting connections from other applications using the database server.
- **Convert CCSID 65535 (IBM Db2 for i/iSeries/AS400 only)**  
False by default. When True, automatically converts binary field values to varchar type so that they can be processed as strings. See also the Database Specific Settings dialog.
- **CCSID for Conversion (IBM Db2 for i/iSeries/AS400 only)**  
The language code to use in converting binary fields to varchar fields.



- **Service Name (Oracle only)**  
A read-only property that provides the service name of the system where Oracle is installed.
- **Max Size Mirroring Block (IBM Db2 for i/iSeries/AS400, Oracle, SQL Server and all databases that support transactional replications via triggers)**

#### **IBM Db2 for i/iSeries/AS400**

Set to 10000 by default. This field is useful if you expect a huge number of transactions between mirrorings. DBMoto reads the journal and stores records that need to be replicated in a temporary file. If the file contains a large number of records or a smaller number of very large records, it can impact performance. In this case, you can set a limit on the number of records to handle at one time. For example, if you set the value of the Block Size field to 250, only 250 records will be replicated at a time.

#### **SQL Server**

Set to 10000 by default. This field is useful if you expect a huge number of transactions between mirrorings. DBMoto reads the log and stores records that need to be replicated in a temporary file. If the file contains a large number of records or a smaller number of very large records, it can impact performance. In this case, you can set a limit on the number of records to handle at one time. For example, if you set the value of the Block Size field to 250, only 250 records will be replicated at a time.

## Oracle

Set to 0 (disabled) by default. This field is useful if you expect a huge number of transactions between mirrorings. DBMoto reads the log and stores records that need to be replicated in a temporary file. If the file contains a large number of records or a smaller number of very large records, it can impact performance. In this case, you can set a limit on the number of records to handle at one time. For example, if you set the value of the Block Size field to 250, only 250 records will be replicated at a time.

## Connections for Trigger-based Replications

For trigger-based replications, this property instructs the DBMoto engine to break the query that reads the transactions into multiple queries, each one returning not more than the specified number of records. For example, if you have 1000 transactions to retrieve and the property is set to 250, DBMoto runs 4 queries to retrieve all the transactions. This approach reduces the load on the database server and reduces the locks on the log tables which are also used by the triggers. The default value for this property is 1000.

- **Enable Transactional Objects**

For transactional replications, this property allows you to enable or disable the creation and validation of any transactional objects required by DBMoto in the source database. For example, triggers and log tables are created for trigger-based replications, and articles and publications are created for Microsoft SQL Server log-based replications. By default, the property is set to true and DBMoto will attempt to create objects required for transactional replications in the database as needed. **In some cases, the DBMoto user does not have enough permissions to make changes in the DBMS, so the property can be set to false to allow external creation of the needed objects.** When the property is set to false, DBMoto creates the replication definition without altering properties in the database. This property works in conjunction with the Validate Replications feature where it is possible to check if the replication has been set up correctly.

- **Temporary Files Library**

The default value is DBMOTOLIB. You can modify this value if you prefer to keep temporary files associated with DBMoto in a different library. Temporary files are automatically created and deleted by DBMoto. You do not need to maintain these files.

- **Command Timeout**

Default value is -1, which means that no timeout is set for DBMoto and the default timeout determined by the .NET data provider is used. For example, the default value for Microsoft SQL Server is 15 seconds. To set a different timeout value, enter a value in seconds. This property determines the amount of time before any command from DBMoto times out while waiting for a response from the database server.

- **Use Primary Key Extended (IBM Db2 for i/iSeries/AS400 only)**

When set to True, retrieves keys for tables that support this feature such as DDS tables.

- **Verifier Sort Sequence Table (IBM Db2 for i/iSeries/AS400 only)**

Designate a table that contains ordering rules for the data in the result set to be used by the DBMoto Verifier to sort character values.

**Default Commit Mode:** Default is AutoCommit where each statement is replicated as a standalone transaction. This means that only statements that fail will not be replicated. The CommitmentControl option applies all the statements associated with a transaction as a whole to the target. CommitmentControl is supported on Oracle (Log Reader and Log Server Agent), Microsoft SQL Server, IBM Db2 for i, IBM Db2 LUW, IBM Informix.

### Refresh Options (Target Connection Properties Only)

- **Default Insert Mode**

Allows you to set a default insert mode for all replications using this target connection. The default setting can be overridden using the [Replication Properties](#) for a specific replication. The options are:

- SingleInsert--inserts record by record using SQL INSERT statements
- BulkInsert--inserts blocks of records. Block size is determined by the Block Size option. Choose this option for increased performance if using one of the data access providers which currently support bulk inserts:

HiT Software Ritmo/i (IBM Db2 for i) .NET Provider  
 SAP HANA ODBC driver  
 SAP HANA .NET Provider  
 SAP Sybase ASE .NET Provider  
 SAP Sybase IQ .NET Data Provider  
 Microsoft SQL Server .NET Provider (using SQL Server's SqlBulkCopy)  
 MySQL .NET Data Provider  
 PostgreSQL .NET Data Provider  
 Oracle .NET provider.

For SQL Server, note that BulkInsert is the default option.

Record details are not reported in the DBMoto log when BulkInsert is set.

- SimulatedBulk--For databases that do not support bulk insertion, DBMoto simulates the effect by grouping records into blocks before performing an insert. This is a way to optimize performance.

- **Default Bulk Type**

Allows you to set a default bulk type for all replications using this target connection. The default setting can be overridden using the [Replication Properties](#) for a specific replication. Depending on the database type and provider type, some of the following options may not be available.

- ArrayBinding--inserts multiple rows at a time using an array of parameters
- Native--uses the native functionality of the provider, if available
- FTP--bulk insert implemented through FTP

- **Default Block Size**

Allows you to set a default block size for all replications using this target connection. The default setting can be overridden using the [Replication Properties](#) for a specific replication. Set the value for the number of rows to insert in a single operation using the Bulk option above.

- **Default Isolation Level**

Allows you to set a default isolation level for all replications using this target connection. The default setting can be overridden using the [Replication Properties](#) for a specific replication. This option allows you to choose a specific isolation level on the Refresh operation. Choose from the following options available through your .NET Framework data provider:

Unspecified	No level is determined.
Chaos	The pending changes from more highly isolated transactions cannot be overwritten.
ReadUncommitted	A dirty read is possible, meaning that no shared locks are issued and no exclusive locks are honored.
ReadCommitted	Shared locks are held while the data is being read to avoid dirty reads, but the data can be changed before the end of the transaction, resulting in non-repeatable reads or phantom data.
RepeatableRead	Locks are placed on all data that is used in a query, preventing other users from updating the data. Prevents non-repeatable reads but phantom rows are still possible.
Serializable	A range lock is placed on the DataSet, preventing other users from updating or inserting rows into the dataset until the transaction is complete.

**Mirroring Options (Target Connection Properties Only)**

- **Default Mirroring Insert Mode**

Allows you to set a default insert mode for all replications using this target connection. The default setting can be overridden using the [Replication Properties](#) for a specific replication. The options are

- SingleInsert--inserts record by record using SQL INSERT statements
- BulkInsert--inserts blocks of records. Block size is determined by the Block Size option. Choose this option for increased performance if available

- **Default Mirroring Block Size**

Allows you to set a default block size for all replications using this target connection. The default setting can be overridden using the [Replication Properties](#) for a specific replication. Set the value for the number of rows to insert in a single operation when using the BulkInsert option above.

- **Default Commit Isolation Level**


Allows you to set a default isolation level when the Commit Mode is set to CommitmentControl for all replications using this target connection. The default setting can be overridden using the [Replication Properties](#) for a specific replication. Choose from the following options.

Unspecified	No level is determined.
Chaos	The pending changes from more highly isolated transactions cannot be overwritten.
ReadUncommitted	A dirty read is possible, meaning that no shared locks are issued and no exclusive locks are honored.
ReadCommitted	Shared locks are held while the data is being read to avoid dirty reads, but the data can be changed before the end of the transaction, resulting in non-repeatable reads or phantom data.
RepeatableRead	Locks are placed on all data that is used in a query, preventing other users from updating the data. Prevents non-repeatable reads but phantom rows are still possible.
Serializable	A range lock is placed on the DataSet, preventing other users from updating or inserting rows into the dataset until the transaction is complete.
Snapshot	No locks on underlying data in a snapshot transaction, so other transactions can execute.


- **Default Mapping Rule**

Allows you to set a default [mapping rule](#) for all replications using this target connection. The default setting can be overridden when setting the mapping for a specific replication.

### Transactional Support Properties


These properties are relevant when performing transactional replications (mirroring or synchronization). Properties differ depending on the type of database used in the connection. To modify properties in this section, click the value for the Transaction Log Type property and then click  to open the [Setup Info dialog](#).

- **Transaction Log Type**

Always set to either the database name, where the database log is used to detect new transactions, or Triggers, where triggers are used to identify transactions in the database. Click the value for the Transaction Log Type property and then click  to open the [Setup Info dialog](#).


### SQL Server Source and Target Connections Only

- **Server Name**


Always set to the name of the server that you specified in the Source/Target connection. Select this field and click  to create the distributor required for mirroring and synchronization replications, or modify the user ID and password.

- **Database Name**


Set to <None> if no distributor has been created. A distributor is required for mirroring and synchronization replications that use the transaction log.

Set to DBRS\_DISTRIBUTION if the distributor has been created. To create the distributor, click in the Server Name field and then click  to open the Distributor Settings dialog.


- **Use Trusted Connection**

When False, you need to supply a user ID and password for SQL Server. When True, the connection uses your Windows user ID and password. Set this field to **True** using the Distributor Settings dialog available by clicking  in the Server Name field.

- **User ID**

Blank when no distributor has been created. Set this field using the Distributor Settings dialog available by clicking  in the Server Name field. The user ID should have system administrator privileges to create the distributor and access the transaction log.

- **Password**

Blank when no distributor has been created. Set this field using the Distributor Settings dialog available by clicking  in the Server Name field.

- **Log Options**


This property allows the user to add options to the query that DBMoto uses to open the SQL Server Log Reader, and thereby optimize query execution. Use the syntax specified for the SQL Server OPTION clause. For instance, for the following query:

```
SELECT ...
FROM ...
WHERE ...
OPTION (HASH GROUP, FAST 10);
```

put the value “HASH GROUP, FAST 10” in the Log Options property.

### IBM Db2 for i/iSeries/AS400 Source and Target Connections Only

- **Library Name**

By default, this is set to DBMOTOLIB when this has been installed. Click in the field and then click  to open the Install i5/AS400 Library dialog and install the library. This library is required when performing mirroring and synchronization replications that access the journal.

- **Version**

Read-only property. The version of the DBMOTOLIB library installed on the IBM Db2 for i (iSeries/AS400) system. This value is generated by HiT Software.

### Oracle Source and Target Connections Only

- **Use Online Dictionary**

Set to **True** if you are planning to run mirroring or synchronization replications for Oracle versions 9i or higher. Set to **False** for Oracle versions 8.1.7 up to 9i. You must also specify a dictionary file name if you set this value to false.

- **Dictionary File**

If you are using mirroring or synchronization and set **Use Online Dictionary** to **False**, type the path and file name for an Oracle dictionary file. The Oracle dictionary file should be defined in your Oracle environment. See [Setup for Different Oracle Versions](#) for more information.

- **Max Number of Log Files in Mirroring Block**

This property is used only when **Use CONTINUOUS\_MINE Option** is disabled. It sets the number of log files that can be loaded into memory per transaction-log reading session (default is 5). In cases where there is a long list of log files to load, processing them all at the same time would significantly increase system memory consumption. Controlling the number of files in memory reduces the impact on system performance.

### SAP HANA and SAP Sybase IQ Target Connection Only

In the Dynamic Properties section, you need to set FTP properties to transfer data from the source database to the SAP HANA server. With FTP file transfer information, DBMoto can use BulkInsert mode for replication speed and efficiency.

- **FTP Server, FTP Port, FTP User, FTP Password**

The FTP connection details for the server where SAP HANA is running.

- **Import Path**

The complete path for the file location on the target machine where SAP HANA is installed.

### Netezza Target Connection Only

- **Max Commit Interval**

Indicates the maximum waiting time (in seconds) to reach the block size before sending records processed so far to the target, even if records to fill the block size have not been collected yet. Works in conjunction with the Default Block Size property. The default value is 0, indicating that there is no maximum limit on the time to wait before sending records to the target--only the block size value will be used.

When processing a mirroring, the records changed are sent to the target in blocks as indicated in the Default Block Size field. If the time to process a block reaches the specified value, the records that have been processed so far are sent to the target even if the records count is less than the specified block size.

### Trigger-based Log Properties (Source and Target Connections for Db2 UDB, Db2 z/OS or Gupta SQL Base Only)

#### Use Transactional Replication

Check this option if you are planning to define a mirroring or synchronization replication.



## Master Table

Either specify an existing qualified table name, or click Change to create a new table to hold general information about replication transactions including user name, timestamp, table name for each transaction.

There are two tables associated with each replication: a Master table, common to all replications using that connection, and a Log table for each replication source table. The Master table keeps track of all the transactions affecting the source tables and it records general transactional information.

Master and Log tables are created in the schema specified when you set the Master table name. You can choose a Master table name, or use the default `_DBM__MASTERLOG`. Log tables are automatically generated by DBMoto and the names are `_DBM__LOG_#`, where # is a number. The selected schema for the Master and Log tables must not contain other non-DBMoto tables with names `_DBM__LOG_#`. HiT Software recommends that you create a new schema to use specifically for the DBMoto Master and Log tables.

## Tablespace

HiT Software recommends that you assign a tablespace for the Master table and Log Tables so that it is easier to control log table sizes. Your system administrator should be able to provide you with the appropriate value for this field.

## Retention Time

The amount of time in hours that a transaction is kept in the log tables. The default value is 72 hours. When the amount of time a transaction resides in the log exceeds the retention time, the transaction is permanently removed from the log tables. Tuning the retention time provides control over the size of the log tables.

## Delete Block Size

Based on the retention time, DBMoto deletes items from the log. This field specifies the maximum number of records to delete from the DBMoto log tables with a single SQL statement. The default value is 10,000 records. You do not typically need to edit this value.

## Opening the Connection Properties Dialog

In the Management Center Metadata Explorer, expand the tree to show a source or target connection. Select the connection. From the right mouse button menu, choose **Connection Properties**.

## Create DBMoto Login Dialog

The Create DBMoto Login dialog allows you to set up login and profile details for a new user.

## User Definition Tab

### Authentication

Choose one of the following options in the drop-down list:



### **DBMoto Authentication**

DBMoto will manage the user ID and password information supplied in the fields below.

**NOTE:** If you set up a user ID with DBMoto Authentication, you will not be able to use DBMoto until the certificate is correctly installed. [Generate and install the certificate](#) before setting up authentication.

### **Windows Authentication**

DBMoto uses your Microsoft Windows login and ID to control access to the DBMoto Management Center.

### **DBMoto Login**

Active only when DBMoto Authentication has been selected. Enter a user name.

### **Password**

Active only when DBMoto Authentication has been selected. Enter a password for the user.

### **Confirm Password**

Re-enter the password.

## **Permissions Tab**

### **User Role**

Select a user role from the drop-down list. When you select a role, the appropriate permissions are also set. You can modify the permission settings for a specific user by changing the True/False values, but note that the user role changes to **Custom** as soon as you change a permission setting.

#### **Administrator**

Permission for all operations.

#### **Auditor**

Permission to audit and monitor replications

#### **Custom**

Permissions selected manually.

#### **Operator**

Permission to set up and change replication settings but cannot work on administrative tasks (such as change license, change user permissions)

#### **Public**

Permission to audit and monitor replications, back up metadatas and change metadata settings (such as starting/stopping traces or changing email parameters for email notification).

#### **SecurityAdmin**

Permission to manage security. Does not normally need to change replication settings or work on creating new metadata.

The table below summarizes the different permissions for each user role.

Permission	Description	Admin.	Security Admin.	Auditor	Operator	Public
Connect to Server	Permission to connect to a DBMoto server	X	X	X	X	X
Set Metadata Default	Permission to set the default metadata. Can only be performed on local installation of DBMoto.	X			X	
Change License	Permission to change product license. Can only be performed on local installation of DBMoto.	X	X			
Add Metadata	Permission to add a metadata connection.	X			X	
Remove Metadata	Permission to remove a metadata connection	X			X	
Change Metadata Properties	Permission to edit metadata properties. Can only be performed on local installation of DBMoto.	X			X	X
Load	Permission to load	X		X		X


Metadata	metadata					
Browse Metadata	Permission to browse metadata tables through the SQL Command panel	X	X			
Change Data Replicator Settings	Permission to change Data Replicator Settings	X				
Change Security	Permission to change security settings. Can only be performed on local installation of DBMoto.	X	X			
Change Global Script	Permission to edit the current metadata global script	X				
Drop Metadata	Permission to drop the tables of the current metadata when they are removed from DBMoto metadata.	X				
Backup Metadata	Permission to execute a metadata backup of the current metadata.	X			X	X
Restore Metadata	Permission to execute a metadata restore of the	X			X	

	current metadata from a backup file.					
Custom Restore Metadata	Permission to execute a custom metadata restore of the current metadata from a backup file.	X			X	
Add Connection	Permission to add a connection to the current metadata.	X			X	
Change Connection	Permission to edit a connection in the current metadata.	X			X	
Remove Connection	Permission to remove a connection in the current metadata	X			X	
Add Replication	Permission to add a replication in the current metadata	X			X	
Change Replication	Permission to edit a replication in the current metadata	X			X	
Remove Replication	Permission to remove a replication in the current metadata	X			X	
Add Group	Permission to add group in the current metadata	X			X	


Change Group	Permission to edit a group in the current metadata.	X			X	
Remove Group	Permission to remove a group in the current metadata.	X			X	
Move Group Replication	Permission to move replications in and out of a group in the current metadata.	X			X	

## Opening the Create DBMoto Login Dialog

In the Metadata Explorer:

1. Select the server for which you want to create a user login.
2. Click  **Manage Users** in the toolbar or choose **Manage Users** from the right mouse button menu.
3. In the User Settings dialog, check the **Enable Security** option.
4. Click **Add** to open the Create DBMoto Login dialog.

## Define Primary Key Dialog

This dialog allows you to set a primary key on a table for use during transaction-based replications (mirroring or synchronization.) The primary key is for use within DBMoto only. It does not affect any primary key settings in your database. You can either set a primary key for a table that has no key, or you can override a primary key that has been set in the database. When you display a table in the Management Center Object Browser, primary key fields are displayed with a key icon . Fields in grey denote a primary key that has been set in the database, but not used in DBMoto.

**NOTE:** If you choose to set one or more table columns as primary key in DBMoto, it is critical that each record contains a **unique value**. If the values are not unique for each record, the results of any replication using the table are unpredictable. The primary key columns defined in DBMoto are used to match source and target records based on their values. DBMoto does not enforce primary key constraints internally, and is not able to determine whether a transaction will be applied to multiple records or whether it will generate duplicate values.

## Fields

This column displays all fields that are eligible to be a primary key. Fields that are nullable cannot be used as primary keys.

## Keys

This column displays fields that are currently set as a primary key.

Select a field and use the arrow keys to move the field between columns.

1. The left column of the Define Primary Key dialog, there is a list of fields eligible to be primary keys. It does not include any nullable fields. The right column of the dialog shows any primary keys that have already been set. If a primary key is set in the database, it is displayed in the right column of the dialog.
2. To add a new primary key, select a field name in the left column, then click the right-facing arrow to move the field into the right column of the dialog. You can use any number of fields (to the limit set by the database you are using) to create a primary key, as long as they can uniquely identify any record.
3. To remove an existing primary key, select the field name in the right column, then click the left-facing arrow to move the field into the left column of the dialog.
4. When you have set the primary key to your liking, click **OK** in the Define Primary Key dialog.

The Management Center table display updates to show your new primary key settings. Note that if you have overridden a primary key set in the database, the field is displayed in grey to show the change. In the example below, the primary key set in the database, **au\_id**, is displayed in grey to show that it will not be used as a primary key for replication, and the fields **au\_lname** and **au\_fname** have been set as primary keys.

## Opening the Define Primary Key Dialog

1. In the DBMoto Management Center Metadata Explorer, select a source or target table.
2. From the right mouse button menu, choose **Set Primary Key**.

## Data Replicator Options Dialog

[General](#)

[Conversion Rules](#)

[Log](#)

[Trace](#)

[Alerts](#)

[Mail](#)

## General

### Max Number of Concurrent Threads

Sets the maximum number of replications that can run concurrently. The default value is 5. Use the [Preferences tab](#) in the [Replication Properties dialog](#) to set priorities for the different replications. Use the **Defaults** button to set ALL **Execution Settings** back to the default values. See also [Managing Performance Using Thread Settings](#)

### Thread Delay

The default value is zero. Use the **Defaults** button to set ALL **Execution Settings** back to the default values. See also [Managing Performance Using Thread Settings](#)

### Thread Execution Factor

The default value is 10. Use the **Defaults** button to set ALL **Execution Settings** back to the default values. See also [Managing Performance Using Thread Settings](#)

### Replication Settings: Perform an INSERT if the UPDATE doesn't find the record

Checked by default so that records are inserted in the target table even if the source table log specifies that the operation performed in the source table was an UPDATE.

### Transaction Latency Threshold

The default value for this option is 1 minute. It is used with the Transaction Latency Chart in the [Replication Activity Viewer](#). The chart compares the current time with the latest processed transaction time (or the last scheduled mirroring time if there are no transactions). If the lag between the two exceeds the **Transaction Latency Threshold** setting, the shaded area in the graph is displayed in red and the **Latency Status column** in the [Replication Monitor](#) displays the value **Threshold Warning**. This can indicate that the current settings for the Data Replicator and the replication are not allowing the Data Replicator to keep up with the number of transactions. Adjust the Transaction Latency Threshold value as needed to provide a useful warning for your environment, depending on the number of transactions and the mirroring interval.

### Application Paths: Temporary Files Folder

Blank by default. In this case, DBMoto temporary files are saved to the DBMoto SyncCache folder. Type a new path to save temporary files to a different location.

## Conversion Rules

### Date/time conversion errors

Errors may occur when replicating date/time fields from one database to another. The accepted formats for this data type vary from DBMS to DBMS. For instance, IBM Db2 for i allows 1/1/1 as a valid date but Microsoft SQL Server does not. The options below allow you to control how such errors would be handled.

### Don't Trap Error

No attempt is made to trap the error in the replication code so an exception is generated and added to the DBMoto log (DBMoto.log.) No value is entered in the field where the error occurred and replication continues.

### Insert NULL Value

NULL will be inserted in fields where the conversion generated an error. No information is reported in the DBMoto log.

### Insert the Minimum/Maximum Allowed

Depending on the database and the column data type, DBMoto uses the minimum date allowed (if the date value was earlier than the minimum date setting for the database) or maximum date allowed (if the date value was later than maximum date setting for the database.)

### Insert this Date

The selected date is inserted in place of the data that is causing an error. In general, the date is substituted, and the time (if any) is left unchanged. For example, if you have 1/1/1 11:30 it becomes 1/1/1900 11:30.

## Log

The settings on this tab apply to both the DBMoto log (.log extension, viewable in the [Log Viewer](#)) and the DBMoto history files (.his extension, viewable in the [History Viewer](#).) This tab allows you to specify log type, location (if a database), size and generation frequency.

### Log Settings

Click this button to open the [Log Settings dialog](#).

### Write Log To

You can save the DBMoto log and history to one of the following. Your selection will be applied to both the log and history files.

- [A text file](#)
- [A database](#)
- The Microsoft Windows [Event Log](#)
- Apache [log4net](#)<sup>™</sup>

### Activate Log History

Check this option to start logging replication status information for each replication session. All information is recorded in a file with a .his extension in the DBMoto Logs folder. This information can then be displayed and reviewed in the [History Viewer](#).




You may want to leave this option unchecked if you are concerned about disk space and processing load for the Data Replicator. The process stores one record for each replication during each refresh/mirroring session. In the case of mirroring, this typically means that the .his file is updated every 60 seconds.

## Database Log

### Use Default Metadata

Select this option to create the log tables in the default (selected) metadata database.

### Use Specific Connection

Select this option to create the log tables in an alternative specified connection. Click  to open the [Log Database Connection Properties dialog](#) and set up a connection to the database in which you plan to write the log.

### Keep Max Log Messages

Check this option to set a specific number of messages to keep in the log database. This field allows you to control the size of the log database and ensure that it does not get too large. The default value is 0, which means that all messages are kept in the database. Note that the value is in thousands. So, if you want to keep 5,000 records in the log, set the field value to 5.

## Text File Log

Use the [Log Viewer](#) to access the log and save it to a file.

### Only one file

Generates a single log file that includes all log messages occurring during replications.

### One file every X days

If checked, and a number entered, generates a new log file after the number of days specified. All old log files are kept in the log directory. The convention used to name the log files is DBMoto\_XXXX, where XXXX is a number starting at 0001. If you do not set the "Keep Max X Log Files" option, remember to clean up your log directory regularly.

### One file with size limited to X Mb

If checked, and a number entered, creates a new log file when size X is reached. All old log files are kept in the log directory. The convention used to name the log files is DBMoto\_XXXX.log, where XXXX is a number starting at 0001. If you do not set the "Keep Max X Log Files" option, remember to clean up your log directory regularly.

### Keep Max X Log Files

If unchecked, all old log files are kept in the log directory. If checked, keeps only the specified number of old log files in the log directory. Use file dates (not file names) to determine the most recent log files.

## Windows Event Log

Events are saved to the Microsoft Windows Event Log and can be viewed using the Windows Event Viewer, available from your Windows desktop by choosing **Start**, then **Settings**, then **Control Panel**. In the Control Panel, double-click **Administrative Tools**, then **Event Viewer**.

## log4net

When you select the log4net option, by default DBMoto writes events to a file using Apache log4net. You can configure log4net features by editing the log4net section in the DBReplicator.exe.config file located in the DBMoto install folder. HiT Software recommends that you make a backup copy of the DBReplicator.exe.config file before editing. For additional information about log4net features and settings, see <http://logging.apache.org/log4net/release/features.html>.

## Trace

### Activate Server Agent Tracing

For debug purposes only. Should be enabled only when requested by HiT Software technical support staff. When checked, this option traces all activity related to the selected server. Stop and restart the Server Agent using the [DBMoto Service Monitor](#) in the Windows System Tray to enable tracing.

### Activate Data Replicator Tracing

For debug purposes only. Should be enabled only when requested by HiT Software technical support staff. Click **Settings** to open the Data Replicator Trace Settings dialog. This dialog allows you to select the type of information to include in the trace. The following options are available:


Trace binary info read from the transaction log (selected by default)

Trace low level database connectivity info (selected by default)

Trace metadata operations


Trace operations during replication refresh phase (selected by default)

Trace replication history info

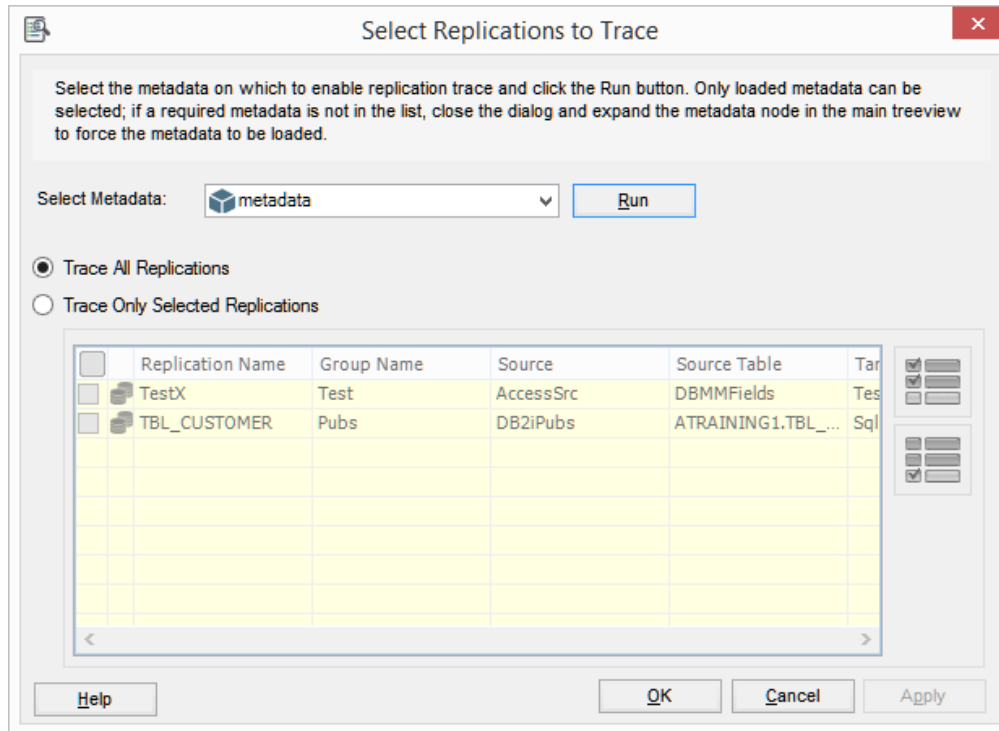
Trace selected replications only (Click in the values field then on  to open the Select Replications to Trace dialog)

Trace threads and tasks info

### Select Replications to Trace Dialog

The "Trace selected replications only" option, available from the Trace Settings dialog, allows you to limit the number of replications to trace. Click in the values field, then on  to open the Select Replications to Trace dialog. You can either select specific replications, or select all replications in a group to enable tracing for the group.

If you select all the replications in a group, then the whole group is traced. If you later add replications to the group, those replications will be traced as well. If you remove a replication from the group, the replication will no longer be traced.



Typically, you can leave the default settings. The DBMoto support staff will let you know if they need specific settings during tracing.

### Only one file

Generates a single trace file that includes all trace messages occurring during replications.

### One file every X days

If checked, and a number entered, generates a new trace file after the number of days specified. All old trace files are kept in the log directory. The convention used to name the Data Replicator trace files is DBMoto\_XXXX.trc, where XXXX is a number starting at 0001. If you do not set the "Keep Max X Log Files" option, remember to clean up your log directory regularly.

### One file with size limited to X Mb

If checked, and a number entered, creates a new trace file when size X is reached. All old trace files are kept in the log directory. The convention used to name the trace files is DBMoto\_XXXX, where XXXX is a number starting at 0001. If you do not set the "Keep Max X Log Files" option, remember to clean up your log directory regularly.

### Keep Max X Log Files

If unchecked, all old trace files are kept in the log directory. If checked, keeps only the specified number of old log files in the log directory. Use file dates (not file names) to determine the most recent files.

## Alerts

Use this tab to create and manage email alert messages for the selected server (usually **local**). It provides a way to define email messages for specific events that apply to all replications that are enabled and/or to the status of the Data Replicator itself.

### Replication Alerts Grid

The grid displays alerts that have been defined using the **Add Alert** button. Select an alert to edit or remove it.

### Add Alert

Click **Add Alert**, then choose **Add Event Alert** to define email message alerts for specific events in the [Server Alert Properties dialog](#). Click **Add Alert**, then choose **Add Activity Report** to create general email message alerts for Data Replicator activity in the [Server Alert Properties dialog](#). Note that newly defined email alert messages will only be sent after [stopping and restarting the Data Replicator](#).

### Remove Alert

Select an alert in the grid, then click **Remove Alert** to delete the email alert.

### Edit Alert

Select an alert in the grid, then click **Edit Alert** to open the [Server Alert Properties dialog](#) and edit the email alert.

## Mail

This tab is used in conjunction with the [Alerts tab](#) for server-related email alerts, the [Alerts tab](#) in the [Replication Properties dialog](#) for a specific replication and the DBMoto [SendMail script function](#). If you have defined an email alert, or a global or replication script that uses one of the SendMail functions, then you need to configure your SMTP email server and email address coordinates. Use the **Test Email** button to send an email and check your configuration.

### SMTP Server

Type the SMTP server for the mail account from which you want to send emails.

### SMTP Port

Type the port for the SMTP server.

### From

Type your email address, or the email address from which you want to send emails.

### To(s)

Type the email address(es) for the recipients of the email. Email addresses should be separated using a semi-colon (;).

### Use Authentication

Check this option and provide a user name and password if the email server requires authentication.

## Use SSL Encryption

Check this option if you are using SSL encryption.

## Test Email

Click to send an email using the email addresses defined in the **From** and **To** fields above. The email will have the subject line "Test email from DBMoto" and contain the text "This is a test. Ignore."

## Opening the Data Replicator Options Dialog

In the Management Center **Metadata Explorer**, select the server for which you want to set options. From the right mouse button menu, choose **Data Replicator Options**.

## DBMoto Dashboard (Statistics Tab)

The DBMoto Dashboard allows you to analyze replication data over a period of time based on the contents of a specified DBMoto Metadata and history file or database. It can provides information for each table in a selected set of tables including:

- Total number of processed records
- Total number of failed records
- Total number of records
- Total number of replication sessions
- Total number of processed records by all replications per unit of time.
- Average time it took to replicate specified records.

If you are interested in obtaining this type of information about your replication environment, and plan to use the Dashboard regularly, [set both the DBMoto log and history to save to a database rather than a file](#). This speeds up the data loading process. The Dashboard works also when using log and history files, but data is only loaded from the current history file. Additionally, if the history has been enabled only for a short period of time, the results shown in the dashboard are limited and therefore less meaningful.

## Steps for Using the Dashboard

Use the following steps to gain a basic understanding of Dashboard functionality. For more information, see the description of each area of the Dashboard below.

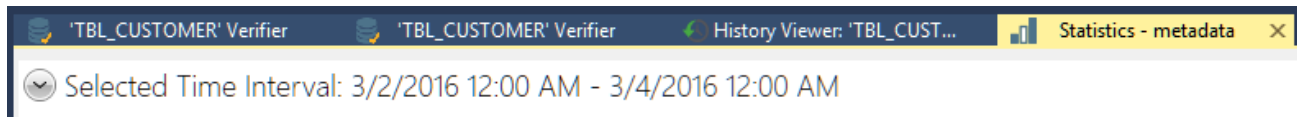
1. Select a time interval by expanding the time interval area and using the yellow scroll bar to adjust the default time.
2. Using the Replication Groups and Selected Replications areas, select which replications you would like to examine in the Dashboard.
3. Using the Dashboard Chart menu, or the [Overview](#) and [Throughput](#) tabs, choose which type of chart to view.

4. In the chart display, expand or reduce the amount of data displayed in the chart using the horizontal scrollbar or mouse wheel.
5. In the Overview Chart, the bar chart displays processed records for each table. Click on one of the options below the chart to change the line chart in the display. Note that the right y-axis scale for the line chart changes depending on the data displayed.
6. In the Throughput Charts, modify the time units displayed using the buttons at the top right of the chart area.

## Dashboard Layout

### Selected Time Interval

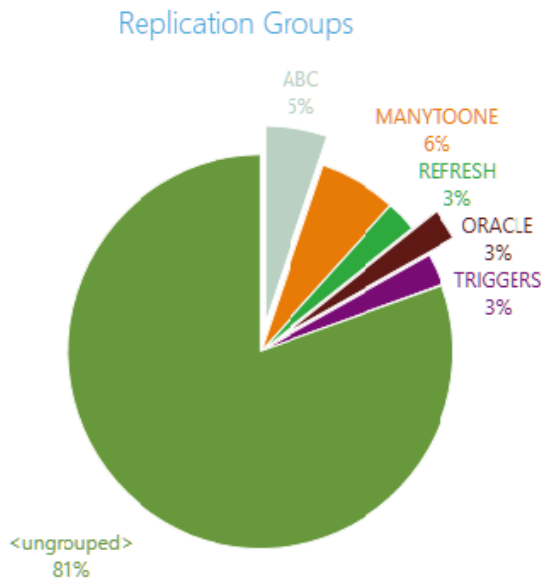
The dashboard automatically loads data from the last week of execution (if available in the history). In the example below, data from 11/1/14 to 11/8/14 because that is the most recent data available from the history.



To change the time interval displayed:

1. Click the down arrow on the left to expand the time selector.
2. Use the horizontal yellow scrollbar to adjust the time period displayed:
  - Use the mouse wheel or drag the end of the scroll bar to change the time period (days, weeks, months) of the whole axis
  - Use the left mouse button on the scrollbar to expand or contract its size

## Replication Groups



## Selected Replications

<input type="checkbox"/>	Name	Processed Records	Success Rate
▲ ABC			
<input type="checkbox"/>	EMPLOYEES	205,093,548	98.43 %
<input type="checkbox"/>	REGION	204,598,861	98.45 %
<input type="checkbox"/>	EMP50	206,148,759	98.44 %
<input type="checkbox"/>	CUSTOMERS	208,102,130	98.47 %
▲ ORACLE			
<input type="checkbox"/>	DEPT00001		100.00 %
<input type="checkbox"/>	TBL_CATEGORY		100.00 %

This area displays a pie chart with all groups belonging to the metadata. The chart legend shows the name of the group and a percent, which is the number of replications belonging to the group in relation to the total number of replications.

If a group has 0 replications, it is not shown in the chart because there is no historic data to analyze. A section of the pie chart called <ungrouped> collects together all replications that are not assigned to a group.

To select a group, click on the portion of the pie that belongs to it. The group section will be raised from the pie area to show selection. When no group is selected, all other charts show empty data. Charts are loaded only when at least one group is selected.

## Selected Replications

This area displays the list of replications from groups selected in the pie chart. Use the checkboxes next to each replication to select or deselect specific replications and obtain a more fine grained analysis of the processed data.

## Dashboard Charts

The menu list on the left of the Dashboard tab shows the options available for the performance charts. Select from the following:

- All sessions (both mirroring and refresh sessions)
- Refresh sessions
- Mirroring sessions

You can also choose which type of chart to view:

- [Overview Chart](#)  
Shows information on the number of processed records compared to failed records, total records and sessions for a specific time interval
- [Throughput Charts](#)  
Shows the processed records not related a replication but to a single time frame inside the selected time interval.

### Overview Chart

This is a multi-axis chart that shows multiple sets of data with a different y-axis, depending on the data displayed. The column bar chart displays a single bar for each selected replication. It shows the number of processed records. In addition, the chart can display total records, failed records or sessions in the selected time interval.

- Columns are colored to match the group as shown in the bar chart. If the processed records are different from the total records (i.e. some failed records), the column bar color is faded.
- The Y-axis on the left (green) shows the number of records. To see the actual number of records for table as a tooltip, move the mouse pointer over the column.
- Use the horizontal scrollbar to see replications beyond the current screen area.
- Use the mouse wheel, or drag the scroll bar, to enlarge or reduce the amount of data shown in the screen area.
- Choose an option below the chart to display the line chart (associated with the right side of the Y-axis) that shows the number of total records (in blue), the number of failed records (in red) or the number of sessions (purple) in the selected time interval. Check the scale on the right y-axis to compare data volumes.

### Throughput Charts

The Throughput Charts show the processed records in relation to a single time frame within the selected time interval. This display can be used to evaluate the number of records processed per unit of time and therefore the workload of the Data Replicator for all replications in the current selection.

The display shows two different charts aligned one above the other, but joined by a common X-axis showing the time period. The top chart counts the total number of processed records by all replications per unit of time within the time period. The bottom chart shows the average time it took to replicate all the records shown in the data above.

- Move the mouse over a section of either one of the charts to observe the change in the counters at the top to show the actual values for the section.
- Use the buttons at the top right of the display to select different time units: per hour, day, week or month.
- Use the mouse wheel, or drag the scroll bar to shift the charts horizontally or enlarge/reduce the area displayed.

### Opening the DBMoto Dashboard (Statistics Tab)

In the Management Center Metadata Explorer, expand the tree to show Metadata entries. Select a Metadata name. From the right mouse button menu, choose **Show Dashboard**.



## DBMoto Login Dialog

The DBMoto Login dialog allows you to change the user login details for a particular server. This dialog is useful only if you have set up user access control for a DBMoto installation via the [User Settings dialog](#). Note that if you change the DBMoto login from one user to another in the same DBMoto Management Center session, all open SQL Query, Log Viewer and History Viewer tabs are automatically closed to avoid leaving open tabs for which the new login may not have appropriate permissions.

### Server Name

Inactive. Reports the name of the server that you selected in the Metadata Explorer.

### Server Address

Inactive. Reports the IP address of the server you selected in the Metadata Explorer.

### Port

Inactive. Reports the port number that is being used for the server you selected in the Metadata Explorer.

## Authentication

The options are:

### AnonymousAccess

No login details are requested from the user. This is the default setting for installations where DBMoto security has not been enabled. To enable DBMoto security, go to the [User Settings dialog](#).

### Windows Authentication

If DBMoto security has been enabled via the [User Settings dialog](#), and you have been directed by your DBMoto administrator to use Windows Authentication, select this option. DBMoto will verify that your Microsoft Windows login and ID are legitimate before allowing you to continue working in the DBMoto Management Center.

### DBMoto Authentication

If DBMoto security has been enabled via the [User Settings dialog](#), and you have been directed by your DBMoto administrator to use DBMoto Authentication, select this option, then type your user name and password in the fields below. DBMoto authentication requires the use of a X.509 certificate. Verify that a [certificate has been installed correctly](#) before completing this dialog. If you do not have a correctly installed certificate, you may not be able to use the DBMoto Management Center.

### Certificate Authentication

If Certificate has been enabled via the [User Settings dialog](#), and you have been directed by your DBMoto administrator to use Certificate Authentication, select this option. Certificate authentication requires the use of a X.509 certificate. Verify that a [certificate has been installed correctly](#) before completing this dialog. If you do not have a correctly installed certificate, you may not be able to use the DBMoto Management Center.

## User Name


Active only when DBMoto Authentication has been selected. Enter the user name assigned to you by the DBMoto administrator. If you do not know your user name, verify login information with your DBMoto administrator or try setting the Authentication value to AnonymousAccess (no security) or Windows Authentication (uses your Microsoft Windows login and password.)

## Password

Active only when DBMoto Authentication has been selected. Enter the password assigned to you by the DBMoto administrator. DBMoto will verify that your DBMoto login and ID are legitimate before allowing you to continue working in the DBMoto Management Center.

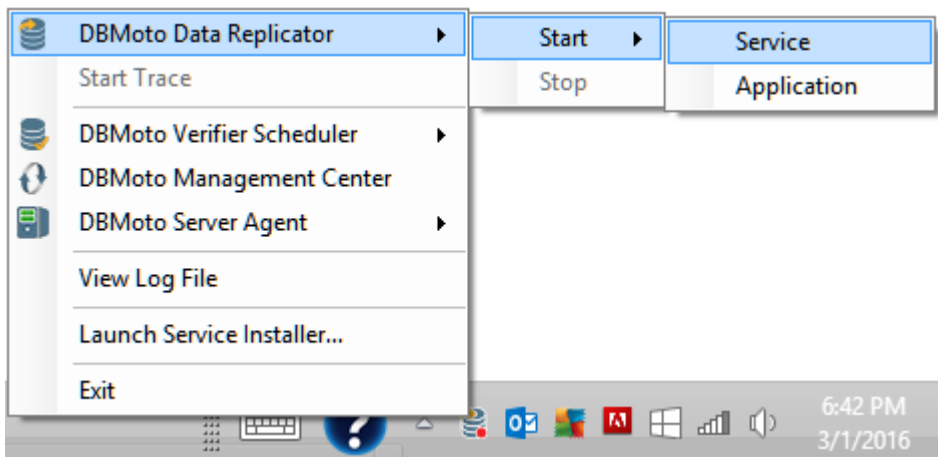
If you do not know your password, verify login information with your DBMoto administrator or try setting the Authentication value to AnonymousAccess (no security) or Windows Authentication (uses your Microsoft Windows login and password.)

## Opening the DBMoto Login Dialog

This dialog is displayed automatically when a user attempts to access a DBMoto server with established access control. In addition, it is possible to change a user login while using the Management Center. In the Metadata Explorer, select the server for which you want to change the login details, then click  **Change Login** in the toolbar or choose **Change Login** from the right mouse button menu.

## DBMoto Service Monitor

The DBMoto Service Monitor allows you to manage DBMoto applications and services from the Windows System Tray:



The Service Monitor is most commonly used to start and stop the DBMoto Data Replicator, and start and stop the DBMoto Server Agent:

- **DBMoto Data Replicator**

The engine that executes replications as set up in the DBMoto Management Center or via the DBMoto APIs. The

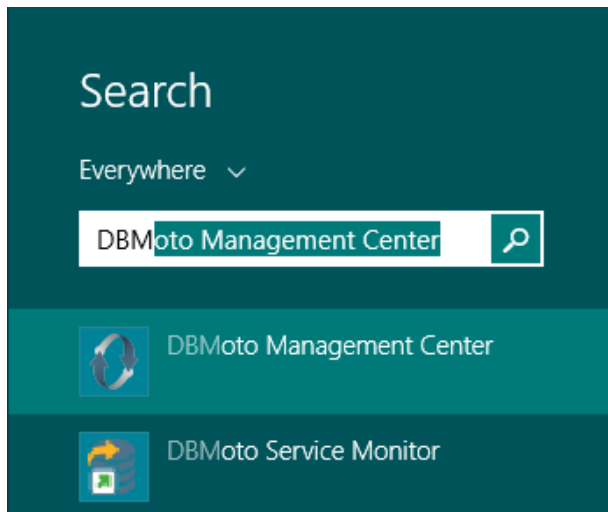
Data Replicator coordinates activities via the DBMoto Server Agent and uses the DBMoto metadata database to obtain replication settings. The Data Replicator can be run as a Microsoft Windows service (set up during DBMoto installation) or as an application.

- **DBMoto Server Agent**

A Microsoft Windows service that acts as a central manager for all operations done by all client applications in a DBMoto installation. DBMoto client applications are the Management Center, the Data Replicator, the Log Viewer (when used as standalone application) and any application built using the DBMoto APIs. These applications interact with the Server Agent to make requests and the Server Agent manages interaction to keep clients synchronized.

However, the Service Monitor can also be used to quickly start and stop the DBMoto Data Replicator trace without accessing the Management Center. Changes to the trace settings via the Service Monitor will be reflected in the Trace tab of the [Data Replicator Options dialog](#) in the Management Center.

After installing DBMoto, you can launch the Service Monitor from the Windows Start menu.



## DBMoto Verifier

The DBMoto Verifier tool can be used to verify replication results and reconcile data differences for replications that have been created and run in DBMoto. It allows you to check for differences between source and target tables and therefore helps with troubleshooting replication problems during your DBMoto replication setup and implementation phase.

The Verifier tab consists of three panes:

- **Source Table**

Displays the source table in the replication under verification.

### Target Table






Displays the target table in the replication under verification.

### Details

Displays a text summary of the verification process.

- In cases where the row appears in both the source and the target database, but one or more field values do not match, the source table row is displayed followed by the target table row. The differences are highlighted in the color selected for differences in the Options dialog (default is red), and you can scroll the rows to identify the fields with different values.
- In cases where the row appears in the source table, but not in the target table, the row is displayed using the color selected for the source table in the Options dialog (default is blue.)
- In cases where the row appears in the target table, but not in the source table, the row is displayed using the color selected for the target table in the Options dialog (default is yellow.)


The icon bar offers the following actions.

	<p><b>Run Verification</b> Compares source and target table to identify differences. Reports the results in table format and as text in the Details area of the tab.</p>
	<p><b>Stop</b> Halts the verification process. This is useful if verification is taking longer than expected.</p>
	<p><b>Reconcile Data</b> After confirmation, modifies the target table to match the source table based on information gathered during the verification process.</p>
	<p><b>Save Results</b> Saves information about table differences and messages displayed in the Details section of the tab to a text file.</p>
	<p><b>Options</b> Displays the <a href="#">DBMoto Verifier Options dialog</a>, allowing you to set a range of options to control how the verification is run.</p>

## Scheduling DBMoto Verifier Activity



During operation, the DBMoto Verifier accesses both the source and target tables, and on request also reconciles any mismatched data. This process can impact daily operations on production databases so DBMoto provides an option to

schedule verification. The schedule is set in the [Replication Properties dialog](#) and uses a Windows service called DBMoto Verifier Scheduler. To schedule verifications:

1. Select the replication in the Metadata Explorer.
2. From the right mouse button, choose Replication Properties.
3. In the [Replication Properties dialog](#), go to the Scheduler page.
4. On the Scheduler page, choose the Verifier Schedule tab.
5. Click the option **Run Recurrently**.
6. Check the **Reconcile Data** checkbox if you would like DBMoto to modify source and/or target tables when needed.
7. Click **Add** to open the [Scheduler dialog](#).
8. Set the frequency in the Scheduler dialog.
9. Click **OK** to close the Scheduler dialog.
10. Click **OK** to close the Replication Properties dialog.
11. In the Windows Notification Area, right click on the DBMoto Service Monitor .
12. From the menu, choose **DBMoto Verifier Scheduler**, then **Start**, then **Service**.

## Opening the DBMoto Verifier

In the DBMoto Management Center Metadata Explorer, Replication Browser, or Replication Monitor, select a replication and, from the right mouse button menu, choose **Verify Replication**.

Note that the verification process, by default, does not start automatically. You need to click  **Run Verification**. If you want to stop the process, click  **Stop** in the toolbar.


## DBMoto Verifier Options Dialog

Use the DBMoto Verifier Options dialog to determine the number of rows to process and display, and to customize the way that source and target tables are compared.

### Record Count Only

Unchecked by default. When checked, the verification process compares only the number of records in the source and target tables.

### Run Verification Automatically

Unchecked by default. Start the verification process using the  **Run Verification** button on the DBMoto Verifier tab. Check this option if you want to start the verification process automatically in the Verifier tab when you select a replication in the Metadata Explorer, and choose Verify Replication from the right mouse button menu.

### **Show Source and Target Table in the Grid**

Checked by default. When checked, the source and target table contents are displayed in the DBMoto Verifier tab for easy comparison of rows. However, if you know that your table contains large amounts of data, you may want to view only the summary in the Details pane in the Verifier.

### **Max Rows to Manage in Grid**

The default value is 10,000. This is the total number of rows to retrieve/view in the Verifier. This number can be modified in accordance with the memory and CPU configuration of your computer.

### **Show Differences Only**

Checked by Default. This option allows you to control if all records in the source and target tables are displayed in the Verifier tab, or only those records that contain differences. Uncheck the option to display all records.

### **Colors**

Use these three fields to select colors to be used for source table rows, target table rows and conflicts found during the comparison process.

### **Verify Primary Key Only**

Unchecked by default. When checked, the Verifier compares only the primary key field value(s) in source and target tables to determine differences between the tables.

### **Trim Chars**

Checked by default. This option provides a way for the tool to ignore blank characters in string values. Some databases set string values at a fixed length using blank spaces to fill the string, while others permit strings of different lengths. If you check this option, the DBCompare tool trims all strings with fixed string length, then compares the strings.

### **Skip/Ignore BLOB/CLOB**

Unchecked by default. Any BLOB/CLOB field values will be included when comparing source and target tables. This option is unchecked by default, but comparing BLOB/CLOB values can be very time consuming so you may want to change the setting.

### **Skip/Ignore Array of Bytes**

Unchecked by default. Any Byte Array field values will be included when comparing source and target tables.

### **Skip/Ignore Date Time**

When selected, any Date Time field values will be ignored when comparing source and target tables. This is because databases may store these values in different formats, so a verification might show a difference where there is no difference in the actual value, just the format in which it is stored.

### **Skip/Ignore Time**

When selected, any Time field values will be ignored when comparing source and target tables. This is because databases may store these values in different formats, so a comparison might show a difference where there is no difference in the actual value, just the format in which it is stored.

### **Number of Fractional Digits**

Selected by default, with the value 0. Specifically for time values, this option allows you to determine how many digits to the right of the decimal point you wish to use in comparing field values.


### **WHERE Condition**

Provides fields for both source and target tables to type in a condition to narrow the number of rows compared and reported. Type the SQL syntax that you would use following the "WHERE" condition in a SQL statement.

### **ORDER BY Clause**

The default ORDER BY clause in the Verifier is to order by primary key. However, this can be problematic when the primary key is a string because database providers can order strings differently. This field allows you to override the default ORDER BY clause by typing a single column name, or a list of comma-separated column names.

## **Opening the DBMoto Verifier Options Dialog**

1. In the Metadata Explorer, select a replication.
2. From the right mouse button menu, choose **Verify Replication**.
3. In the Verifier tab toolbar, click  **Options**.

## **Edit DBMoto Login Dialog**

The Edit DBMoto Login dialog allows you to modify login and profile details for the selected user.

### **User Definition Tab**

#### **Authentication**

Choose one of the following options in the drop-down list:

##### **DBMoto Authentication**

DBMoto will manage the user ID and password information supplied in the fields below.

##### **Windows Authentication**

DBMoto uses your Microsoft Windows login and ID to control access to the DBMoto Management Center.

#### **DBMoto Login**

Active only when DBMoto Authentication has been selected. Enter a user name.

## Password

Active only when DBMoto Authentication has been selected. Enter a password for the user.

## Confirm Password

Re-enter the password.

## Permissions Tab

### User Role

Select a user role from the drop-down list. When you select a role, the appropriate permissions are also set. You can modify the permission settings for a specific user by changing the True/False values, but note that the user role changes to **Custom** as soon as you change a permission setting.

#### Administrator

Permission for all operations.

#### Auditor

Permission to audit and monitor replications

#### Custom

Permissions selected manually.

#### Operator

Permission to set up and change replication settings but cannot work on administrative tasks (such as change license, change user permissions)

#### Public

Permission to audit and monitor replications, back up metadatas and change metadata settings (such as starting/stopping traces or changing email parameters for email notification).

#### SecurityAdmin

Permission to manage security, Does not normally need to change replication settings or work on creating new metadata.

The table below summarizes the different permissions for each user role.

Permission	Description	Admin.	Security Admin.	Auditor	Operator	Public
Connect to Server	Permission to connect to a DBMoto server	X	X	X	X	X
Set Metadata	Permission to set the default metadata. Can only be performed on	X			X	




Default	local installation of DBMoto.					
Change License	Permission to change product license. Can only be performed on local installation of DBMoto.	X	X			
Add Metadata	Permission to add a metadata connection.	X			X	
Remove Metadata	Permission to remove a metadata connection	X			X	
Change Metadata Properties	Permission to edit metadata properties. Can only be performed on local installation of DBMoto.	X			X	X
Load Metadata	Permission to load metadata	X		X		X
Browse Metadata	Permission to browse metadata tables through the SQL Command panel	X	X			
Change Data Replicator Settings	Permission to change Data Replicator Settings	X				
Change Security	Permission to change security settings. Can only be performed on local installation of DBMoto.	X	X			

Change Global Script	Permission to edit the current metadata global script	X				
Drop Metadata	Permission to drop the tables of the current metadata when they are removed from DBMoto metadata.	X				
Backup Metadata	Permission to execute a metadata backup of the current metadata.	X			X	X
Restore Metadata	Permission to execute a metadata restore of the current metadata from a backup file.	X			X	
Custom Restore Metadata	Permission to execute a custom metadata restore of the current metadata from a backup file.	X			X	
Add Connection	Permission to add a connection to the current metadata.	X			X	
Change Connection	Permission to edit a connection in the current metadata.	X			X	
Remove Connection	Permission to remove a connection in the current metadata	X			X	
Add Replication	Permission to add a replication in the current metadata	X			X	

Change Replication	Permission to edit a replication in the current metadata	X			X	
Remove Replication	Permission to remove a replication in the current metadata	X			X	
Add Group	Permission to add group in the current metadata	X			X	
Change Group	Permission to edit a group in the current metadata.	X			X	
Remove Group	Permission to remove a group in the current metadata.	X			X	
Move Group Replication	Permission to move replications in and out of a group in the current metadata.	X			X	

## Opening the Edit DBMoto Login Dialog

In the Metadata Explorer:


1. Select the server for which you want to edit a user login.
2. Click  **Manage Users** in the toolbar or choose **Manage Users** from the right mouse button menu.
3. In the User Settings dialog, select the user name.
4. Click **Edit** to open the Edit DBMoto Login dialog.

## Event Properties Dialog

This dialog shows details of an event recorded in the DBMoto log. You can set the log frequency and detail using the Log tab in the [Data Replicator Options dialog](#).

Use the up and down arrows to move from one event to the next.

## Opening the Event Properties Dialog

In the Metadata Explorer toolbar, click  to open the [Log Viewer](#). Double-click on an event listed in the Viewer, or select an event and, from the right mouse button menu, choose **View Details**.

## Expression Generator

The Expression Generator lets you build an expression to use when mapping a target field. It can also be used from the [Global Script Editor](#) and the [Replication Script Editor](#) to help write scripts to use during replication. Expressions can be written using Visual Basic (the default) or C# by setting the scripting language in the [Global Script Editor](#), accessible from the right mouse button menu on the metadata.

## Using the Expression Generator for Mapping Target Fields

You can map a target field to a source field or an expression when setting up your replication. Only fields that are mapped will be replicated.

You might want to map a target field to an expression when the source table does not contain a column that exactly matches the data you want to see in the target column. For example, in a source table where FirstName and LastName are two separate columns, but in the target table, there is a single column, Name, that should contain both the FirstName and LastName values. Use an expression to concatenate values from the two columns.

Either type in your expression or navigate through the trees to build your expression.



The functions and operators available in the Expression Builder are those typically available for most .NET providers. If you are not sure what functions and operators to use, check your .NET provider and RDBMS documentation. If you use a function or operator that is not supported by your .NET provider or RDBMS, the mapping will fail during replication. DBMoto additionally provides some [predefined values](#) that help you to define an expression quickly.

## Using the Expression Generator to Help Write Scripts


In both the [Global Script Editor](#) and the [Replication Script Editor](#), the Expression Generator button opens the Expression Generator so that you can use it to find functions and operators to use in the scripts. You can build parts of the script using the Expression Generator. Then, when you close it, the information is displayed in the Script Editor.

## Opening the Expression Generator


To write an expression to use when mapping fields from source to target tables:

- In the Replication wizard **Mapping info** screen, click  **Define Mapping** and choose **Map to Expression** to open the Expression Generator.
- On the [Replication Properties dialog](#) General tab, click **Mapping** to open the [Fields Mapping dialog](#). Click  **Define Mapping** and choose **Map to Expression** to open the Expression Generator.

To write an expression to use as part of a replication script:

- In the [Replication Properties dialog](#), check the **Use Script** option. Click **Script** to open the [Replication Script Editor](#).  
Click  to open the Expression Generator.

To write an expression to use as part of a global script:

- In the DBMoto Management Center Metadata Explorer, select the metadata for which you would like to write a script.
- From the right mouse button menu, choose **Global Script**.
- In the Global Script Editor, click .




## Fields Mapping Dialog

This dialog allows you to set up mappings between target and source tables. During replication, only those fields that have been mapped will be replicated. You can either create mappings using the [Replication wizard](#), the Fields Mapping dialog or [via a script](#) that is executed during the replication.


By default, source fields are mapped to target fields with the same column names. Use this screen to modify the mappings, either by removing mappings, setting mappings to different fields, or mapping fields to expressions. You can also leave the default mappings at this stage and edit them later using the [Replication Properties dialog](#).





The top part of the screen shows source and target table fields. Arrows indicate the mappings that have been created automatically. When editing the mappings using the toolbar, you can map a source field to zero, one or more target fields, and map a target field to only one source field. If you select a source column already mapped, the new mapping will be added to the existing mappings. If you select a target column that has already been mapped, that mapping will be overwritten. You can also

- Select a source or target column and set it as **Use Unmapped** from the right mouse button menu. This means that the column values are available for use within [scripting](#) even though the column is not mapped.
- Select a source column and set it as **Use Non Critical** from the right mouse button menu. During transactional replications, if only data in columns marked as non-critical have changed, the record will not be replicated. However, if other column values have changed, the record is replicated including the data in columns marked as non-critical.

The lower part of the screen lists the target fields on the left side, with an icon that shows if the column is mapped to a field , unmapped , mapped to an expression . Scrolling to the right, you can see the exact mapping for the target field. Note that if a field appears disabled in the list, it means that the data type for the field is not recognized by DBMoto (usually a user-defined data type) and the field cannot be mapped or replicated.

The toolbar displays the following icons.

	<p><b>Define Mapping</b></p> <p>Select a source or target field in the list and choose one of the following options from the drop-down list:</p>
---	--

	<p><b>Map to field:</b> Opens the Edit Mapping dialog, displaying the current mapping, if any. If you selected a source field, choose the target field from the drop-down list on the right side of the dialog. Note that a source column can be mapped to zero, one or more target columns (see <b>Remove Mapping</b> below to unmap a source column.) If you selected a target column, choose the source field from the drop-down list on the left side of the dialog. A target column can be mapped to only one source column. Note that if a target column is already mapped and you select another mapping, the new mapping overwrites the existing mapping.</p> <p><b>Map to Expression:</b> Available when a target field is selected. Opens the <a href="#">Expression Generator</a> so that you can map the selected target field to an expression.</p> <p><b>Remove Mapping:</b> Available when a target field is selected. Removes a mapping to the selected field.</p>
	<p><b>Show/Hide Arrows Panel</b></p> <p>A toggle to display or hide the arrows between source and target fields. The arrows provide a graphical representation of mappings between fields. No arrows are displayed when a field is mapped to an expression.</p>
	<p><b>Show/Hide Extended Columns in Fields Lists</b></p> <p>A toggle to display or hide column information for source and target fields. The default column information consists of field name, position and type. Additional column information consists of size, precision, scale, description, null values, identity and default value.</p>
	<p><b>Mapping Direction</b></p> <p>Active when defining a synchronization replication. Allows you to view the mapping with arrows reversed from target to source, thereby demonstrating mappings that are not necessarily symmetrical with the mappings from source to target.</p>
	<p><b>Automatic Mapping</b></p> <p>By default, fields in the source and target tables are mapped according to the field name, where all matching names are matched. Use this menu to select instead to map by column ordinal numbers, or using a custom script function. The Automatic Mapping menu displays 4 entries: Mapping by Name, Mapping by Ordinal, Custom Mapping and Clear All Mappings. The Custom Mapping option is inactive unless you have defined one or more <a href="#">mapping functions</a> as part of a <a href="#">global script</a>.</p>

## Opening the Fields Mapping Dialog

1. In the DBMoto Management Center Metadata Explorer, select the replication.
2. From the right mouse button menu, choose **Replication Properties**.

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.

3. On the **General** tab, check the **Automatic Mapping** option.
4. Click **Mapping** to open the Fields Mapping dialog.

## Group Properties Dialog

### General Tab

#### Group Name

Displays the group name. You can edit this field and change the name as needed.

#### Group Description

(Optional) Displays any information about the group that you choose to enter. You can edit this field as needed.

#### Available Replications

A list of replications within the current metadata. You can add replications to the group using the right arrow button. Select a replication and click the right arrow button. Note that if you add a replication to a group, the replication settings (scheduling and properties) will be modified to match those of the group. The first replication that is added to a group sets the replication properties for the group.

Note the following restrictions on adding replications to a group.

1. All replications in the group must use the same source connection.
2. If replicating from SQL server using mirroring, all replications in the group must use the same source database.
3. If replicating from IBM Db2 for i/iSeries/AS400 using mirroring, all replications in the group must use the same journal for source tables.

#### Selected Replications

Displays a list of replications in the group ordered by priority. Use the right and left arrow keys to add defined replications to this list of replications for the group. Use the up and down arrow keys to set priority for executing replications within the group. Note that this priority applies to Refresh replications only. When transactional replications (mirroring or synchronization) are grouped, DBMoto uses the database log entries to determine the order of replication. Changes are processed in the order they are found in the log.

#### Reorder Replications following Foreign Key Relationships

When set to True, this option works in conjunction with the Truncate Inverse property on the Preferences tab to ensure that tables in grouped refresh replications maintain data integrity during the truncation and insertion phases of the replication. See [Ordering Refresh Replications with Foreign Key Relations](#) for more information.

## Preferences Tab

### Refresh Recovery

When set to True, this option enables recovery from a problem that occurs while running a set of refresh replications for the group. Recovery starts from the first replication that failed rather than from the first replication in the group.

### Truncate Inverse

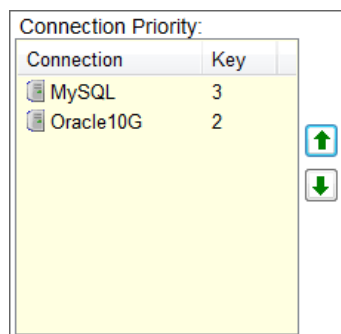
When set to True, this option works in conjunction with the "Reorder Replications following Foreign Key Relationships" button in the General tab to ensure that tables in grouped refresh replications maintain data integrity during the truncation and insertion phases of the replication. See [Ordering Refresh Replications with Foreign Key Relations](#) for more information.

### Conflict Resolver Priority

This property is used only for groups of synchronization replications (multi-table synchronization) where the **Conflict Resolver** property in the [Replication Properties dialog](#) is set to TargetServerWins. The value is an prioritized list of target connections, represented by connection keys obtained from the DBMoto metadata. To view the connection names and modify the priority of connections:

1. Click in the values field.
2. Use the down arrow to expand the field and display the editor.
3. Move connections up or down in priority by first selecting them, then using the up and down arrows.

Note that the numeric value associated with the connection name is a key used by the DBMoto metadata. It does not reflect the priority for that connection. In the example below, the MySQL connection is first in priority but its connection key value is 3.



### Opening the Group Properties Dialog

In the DBMoto Management Center Metadata Explorer, select a group and, from the right mouse button menu, choose **Group Properties**.



## History Viewer

The History Viewer provides a way for you to review the status of past iterations of a replication, whether they succeeded or failed, and the DBMoto Log messages associated with the replication.


The screenshot shows the History Viewer interface. The top pane displays a table of replication sessions with the following data:

Status	Date/Time	Time Elapsed (hh:mm:ss)	Session Type	Session Time	Processed Records	Failed Records
Success	3/4/2016 10:57:10 AM	--:--:--	Refresh	00:00:02.6728043	91	0
Success	3/4/2016 10:57:16 AM	00:00:06.02	Mirroring	00:00:00.0136065	0	0
Success	3/4/2016 10:58:16 AM	00:01:00.11	Mirroring	00:00:00.0315976	0	0
Success	3/4/2016 10:59:16 AM	00:01:00.11	Mirroring	00:00:00.0119974	0	0
Success	3/4/2016 11:00:16 AM	00:01:00.12	Mirroring	00:00:00.0322485	0	0
Success	3/4/2016 11:01:16 AM	00:01:00.13	Mirroring	00:00:00.0333997	0	0
Success	3/4/2016 11:02:16 AM	00:01:00.12	Mirroring	00:00:00.0140265	0	0
Success	3/4/2016 11:03:17 AM	00:01:00.13	Mirroring	00:00:00.0125122	0	0
Success	3/4/2016 11:03:32 AM	00:00:15.07	Refresh	00:00:00.3335115	91	0
Success	3/4/2016 11:04:17 AM	00:00:45.10	Mirroring	00:00:00.0129312	0	0

The bottom pane displays log messages with the following data:

Type	Date/Time	Description	Help
Information	2016-03-02 14:25:10.94	DBMoto Service version 9.0.0.6 started.	
Information	2016-03-02 14:25:10.98	Starting Replication Manager.	
Information	2016-03-02 14:25:12.04	Replication Manager started.	
Error	2016-03-02 23:45:37.13	Error in the Replication Manager thread: exception accessing the Ser...	Sy
Information	2016-03-02 23:45:44.39	The connection to the Server Agent has recovered from a critical error.	
Information	2016-03-03 16:22:08.77	Stopping Replication Manager.	
Information	2016-03-03 16:22:11.82	Replication Manager stopped.	
Information	2016-03-03 16:22:11.83	DBMoto Service version 9.0.0.6 stopped.	
Information	2016-03-04 10:56:58.89	DBMoto Service version 9.0.0.6 started.	

The top pane of the History Viewer shows past iterations of the selected replication. Select a row to view log details related to that iteration of the replication in the Log Messages tab below.

In the Log Messages tab, you can filter messages by clicking  **Show Filter Settings**. This pane provides all the functionality of the [DBMoto Log Viewer window](#), except that it applies to a single replication at a time.

One or more Replication tabs may also be available if specific records failed to replicate. The Replication tab displays a table containing records that failed to replicate for a specific replication interval.

## Opening the History Viewer

In the DBMoto Management Center Metadata Explorer, Replication Browser, or Replication Monitor, select a replication and, from the right mouse button menu, choose **Show History**.

## License Information Dialog

This dialog allows you to view current DBMoto license details, or update your DBMoto license.

### Import License...

Click **Import License** to add your license for DBMoto. You will need to supply the license file (dbmotolicense\_ *company*.txt) that HiT Software sends via email with your license key information. Note that any license file you supply in this field will replace previous license information.

### Export

Click **Export** to print all current license information to a text file.

### Show Pairs

Click **Show Pairs** to display the [Show Replication Pairs dialog](#) and review the source and target database pairs in the selected metadata.

If your license has expired, or you need to add connections, contact your HiT Software sales representative by phone at 408 345 4001 or via email at [ds.sales@boaweb.com](mailto:ds.sales@boaweb.com). If you obtained your copy of DBMoto from a local distributor or reseller, contact your reseller first. Check [www.boaweb.com](http://www.boaweb.com) for a current list of resellers and their contact information.

### Opening the License Information Dialog

In the DBMoto Management Center Metadata Explorer, select the server for which you want to view license information. From the right mouse button menu, choose **Manage Licenses**.

## Log Viewer

The Log Viewer tab displays the DBMoto log. The format depends on settings in the [Data Replicator Options dialog](#). It is possible to save the DBMoto log to a database, to a file or to the Windows Event Viewer. By default, the log is saved to a text file and contains a list of replication events.

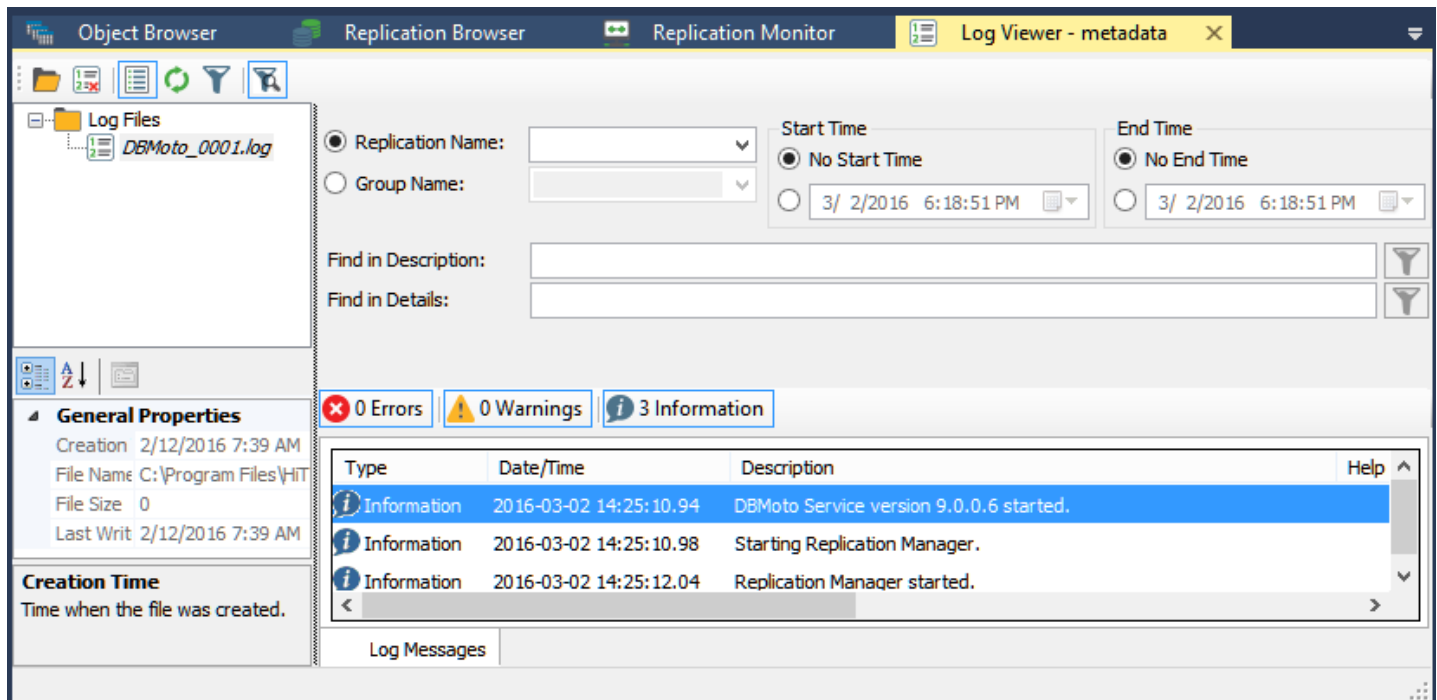
Double-click an event to view the details in the [Event Details dialog](#).

### Filtering Log Information

DBMoto provides tools to filter both text-based and database-based log messages. In the Log Viewer main toolbar, click




**Show Filter Settings** to display the options for searching and filtering log data.



Options for filtering log messages are:

- Select or deselect the Errors, Warnings and Information buttons to view any combination of these three categories of messages as they apply to all replications
- Select a specific group or replication to see messages which apply to that group/replication
- Set a specific time interval within which to list messages
- Apply a text search in either the Description or Details fields of the log to view log messages containing only the string you enter.


Be sure to click  **Apply Filter** in the Log Viewer toolbar when you have finished setting up the filter.

### Text-based Logs Only

For text files, you can set the log frequency and detail using the Log tab in the [Data Replicator Options dialog](#). If you choose to create multiple log files, note that the Log Viewer always displays the current log file by default. To open other log files, from the File menu, choose **Open File**. Note the following restrictions:

- If you choose to refresh the file, the refresh is performed on whichever file is currently open in the log. If, for example, you open a file other than the current file, the refresh is performed on the file that you specifically opened. The Log Viewer does not update to display the current file. You would need to either specifically open the current file, or exit the Log Viewer and re-run it to get the most recent file.
- If you open the Log Viewer to display the current file and, while the Viewer is open, a new log file is created, the Viewer does not automatically switch to the latest file. You would need to either specifically open the new file, or exit the Log Viewer and re-run it to get the most recent file.

## Database Logs Only

For database logs, you can either save the log information to the metadata database, or specify an alternative database connection. Log data is displayed in the Log Viewer, and can be saved in a text file format using the  **Export Log Database to File** button in the Log Viewer toolbar. Note that the entire database contents are saved to the specified file.

## Opening the Log Viewer

In the DBMoto Management Center Metadata Explorer, select the metadata for which you want to see the log.

From the right mouse button menu, choose **View Log**.

## Log Database Connection Properties Dialog

This dialog can be used to set up a database connection for storing DBMoto log information.

### Database

Select the database for which you want to make a connection.

### Provider

Select a .NET data provider from the list. If no .NET provider is available for the database, select either an OLE DB provider or an ODBC driver. By default, only default providers are listed in this dialog. If you want to see all providers for a specific database, uncheck the **Use Only Default Providers** option in the [Options dialog](#).


### Assembly

This field is displayed only if using a .NET data provider, and the provider has not been registered. Provide a path to the .NET data provider DLL.

### Connection String

Edit at least the **Required** connection properties by clicking in the property value field and typing a new value.

The list of **Optional** properties for .NET and OLE DB providers contains the most commonly used properties for the providers. Edit these as needed. Note that some properties are displayed with default values (no bold text.) Any values that you add or edit are displayed in bold text. Check the documentation for your provider for a complete list of properties. You can set the value of the ExtendedProperties property to define additional property-value pairs. The syntax for defining property-value pairs is: prop1=val1;prop2=val2;....

If you are using an ODBC driver, click in the ConnectionString value field and then click  to open the ODBC Driver Connection dialog. The contents of this dialog depend on the ODBC driver that you are using. Provide values for the dialog fields, then click OK to make a test connection and set the values in the DBMoto wizard.

### Edit



Click **Edit** to open a text entry window where you can paste or type a connection string for your provider. This is offered as an alternative to the Connection Properties grid, but should be used with great care because an error in

the connection string can cause a connection to fail or to have unexpected properties. This window displays any connection information that you have already entered in connection string format. Note that default values are not displayed as part of the connection string.

## Test

Click **Test** to make sure that the connection correctly opens a database connection.

## Opening the Connection Properties Dialog

1. In the DBMoto Management Center Metadata Explorer, click  **Data Replicator Options**.
2. In the Log tab, select **Write Log To Database**.
3. Select the Use Specific Connection option.
4. Click  to open the Connection Properties dialog.

## DBMoto Log Settings Dialog

### Write a warning on member reorganize

Relevant only when using IBM Db2 for i (AS/400) as a source database. False by default. When set to true, a warning will be recorded in the DBMoto log if RGZPFM (reorganized physical member) is executed on a table.

When an RGZPFM command is executed on an IBM Db2 for i table, the physical space used by the table is reorganized in order to compact the table and use empty slots. The consequence of this operation is that the RRN sequence is changed among records. If you rely on RRN for your replication, for example by using the RRN as source primary key, this warning would notify you of a reorganization so that you can run a full refresh from source to target table to keep your data consistent.


### Write a warning on multiple updates/deletes

True by default so that warnings are written to the log for update or delete operations that update more than one field. The warning is useful because it notifies you that there may be issues with primary key field settings which do not uniquely identify records. Disable this warning in cases where a trigger is attached to the field being updated or deleted, and that trigger also updates other fields. In this case, the log could quickly fill with warning messages for a behavior that is intentional.

### Write an error on insert attempt

True by default. Adds an error to the DBmoto log when an attempt is made to insert a record not found when trying to do an update.

## Opening the Log Settings Dialog

1. In the DBMoto Management Center **Metadata Explorer**, choose  **Data Replicator Options**.
2. In the Data Replicator Options dialog, switch to the **Log** tab.

3. Click **Settings** in the **Log** tab to open the **Log Settings dialog**.

## Manage Transactional Log Settings Dialog

This dialog consists of tabs that depend on the type of transactional replication you chose for the connection: Log Reader, Log Server Agent or Triggers. Select the replication type for your database from the options below for specific information:

<b>Gupta SQL Base</b>	<a href="#">Triggers</a>
<b>IBM Db2 LUW</b>	<a href="#">Log Reader/Log Server Agent</a> <a href="#">Triggers</a>
<b>IBM Db2 for i</b>	<a href="#">Log Reader/Log Server Agent</a>
<b>IBM Db2 for z/OS</b>	<a href="#">Triggers</a>
<b>IBM Informix</b>	<a href="#">Log Reader/Log Server Agent</a> <a href="#">Triggers</a>
<b>IBM PureData</b>	<a href="#">Log Reader/Log Server Agent</a>
<b>Microsoft SQL Server</b>	<a href="#">Log Reader/Log Server Agent</a> <a href="#">Triggers</a>
<b>Microsoft Azure SQL DB</b>	<a href="#">Triggers</a>
<b>MySQL</b>	<a href="#">Log Reader/Log Server Agent</a> <a href="#">Triggers</a>
<b>Oracle</b>	<a href="#">Log Reader/Log Server Agent</a> <a href="#">Triggers</a>
<b>PostgreSQL</b>	<a href="#">Log Reader/Log Server Agent</a>
<b>SAP Sybase ASE</b>	<a href="#">Triggers</a>
<b>SAP Sybase IQ</b>	<a href="#">Triggers</a>
<b>SAP HANA</b>	<a href="#">Triggers</a>

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.

## Opening the Transactional Log Settings Dialog

1. In the DBMoto Management Center Metadata Explorer, select a connection.
2. From the right mouse button menu, choose **Transactional Setup**.
3. Choose **Manage** from the drop-down menu.

## Metadata Properties Dialog

This dialog reports connection properties for your connection to the database containing your metadata tables. It also provides options for setting up primary and secondary backup metadata databases. DBMoto automatically keeps metadata synchronized between the main and backup metadata databases so that you can restore metadata tables from the backup in case of main metadata database failure.

### Authentication

#### Authentication Method

Read only property. Reports the type of security used to access the metadata

##### **AnonymousAccess**

No access control has been selected.

##### **DBMoto Authentication**

DBMoto manages the user ID and password information.

##### **Windows Authentication**

DBMoto uses your Microsoft Windows login and ID to control access.

#### User Name

Read only property. Reports the name of the user logged in at the current time.

#### User Role

Read only property. Reports the role of the user logged in at the current time.

### General

#### Connection Name

Read only property. The name of the connection that you provided when creating the connection.

#### Current

Not enabled for DBMoto 7.0. In future releases, when backups are enabled, it will show which metadata connection is current.

### **IsActive**

If true indicates that the information with the metadata database are constantly updated.

### **IsRunning**

If true, indicates that the Data Replicator (either locally or remotely) is executing with this set of metadata.


### **Status**

Provides status messages for this set of metadata.

## **Connection Properties**

Displays the connection details for the main metadata database.

### **Connection**

Displays the name of the data provider used to connect to the main metadata database. Click in this field, then click  to open the Connection Properties dialog and modify property values as needed. For example, to switch from using the main metadata to a backup metadata database, change the connection using this dialog.

#### **DriverName**

Read only property. The name of the data provider used to connect to the database.

#### **DBType**

Read only property. The DBMS name from an internal list provided by DBMoto

#### **DBVersion**

Read only property. The database version number retrieved from the provider.

#### **Provider Assembly**

Read only property. If using a .NET data provider, and the provider has not been registered, provide a path to the .NET data provider DLL. Otherwise, this field should display the DLL for your registered provider.

#### **Use Local Assembly**

Read only property. Default value is false. Set to true if you have not registered your provider and you are providing a pathname to the provider DLL.

#### **Connection Class**

Read only property. The class for your provider.

#### **Connection String**

Read only property. Connection string used to connect to the database. The connection string follows the format required by your .NET data provider, ODBC driver or OLE DB Provider.

### **Main Enabled**

Not enabled for DBMoto 7.0. Always set to true.



### **Qualifier**

Either reports the catalog and schema where the metadata tables were created (if this information was included in the connection) or allows you to specify a catalog and schema for the metadata tables.

### **Command Timeout**

Default value is -1, which means that no timeout is set for DBMoto and the default timeout determined by the .NET data provider is used. For example, the default value for Microsoft SQL Server is 15 seconds. To set a different timeout value, enter a value in seconds. This property determines the amount of time before any command from DBMoto times out while waiting for a response from the database server.


### **Restore**

Not enabled for DBMoto 7.0.

## **Primary Backup Properties**

For DBMoto V7.0, to use the backup metadata database in place of the main metadata database, you need to edit the connection properties for the main metadata database and set the connection to point to the backup metadata database.

### **Connection**

Allows you to set connection properties for a backup metadata database. Click in this field, then click  to open the Connection Properties dialog and set values as needed.

### **Backup Enabled**

False by default. When set to True, DBMoto backs up metadata to this database to keep it synchronized with the main metadata database.

### **Qualifier**

Either reports the catalog and schema where the metadata tables were created (if this information was included in the connection) or allows you to specify a catalog and schema for the metadata tables.

### **Command Timeout**

Default value is -1, which means that no timeout is set for DBMoto and the default timeout determined by the .NET data provider is used. For example, the default value for Microsoft SQL Server is 15 seconds. To set a different timeout value, enter a value in seconds. This property determines the amount of time before any command from DBMoto times out while waiting for a response from the database server.

### **Restore**


Copies data from the main metadata to the backup to ensure that the backup copy is synchronized with the main metadata. When you first set up a metadata backup connection and set its enabled status to True, you do not need to click Restore. When you click OK in the Metadata Properties dialog, DBMoto automatically calls the restore process to copy all data from the main connection to the backup connection. If you have a corrupted backup

metadata, or the backup metadata tables no longer match the main metadata (because the connection was down for some time, for example), you need to click in the Restore field to force the backup to refresh its data.

## Secondary Backup Properties

For DBMoto V7.0, to use the backup metadata database in place of the main metadata database, you need to edit the connection properties for the main metadata database and set the connection to point to the backup metadata database.

### Connection

Allows you to set connection properties for a backup metadata database. Click in this field, then click  to open the Connection Properties dialog and set values as needed.

### Backup Enabled

False by default. When set to True, DBMoto backs up metadata to this database to keep it synchronized with the main metadata database.

### Qualifier

Either reports the catalog and schema where the metadata tables were created (if this information was included in the connection) or allows you to specify a catalog and schema for the metadata tables.

### Command Timeout

Default value is -1, which means that no timeout is set for DBMoto and the default timeout determined by the .NET data provider is used. For example, the default value for Microsoft SQL Server is 15 seconds. To set a different timeout value, enter a value in seconds. This property determines the amount of time before any command from DBMoto times out while waiting for a response from the database server.


### Restore

Copies data from the main metadata to the backup to ensure that the backup copy is synchronized with the main metadata. When you first set up a metadata backup connection and set its enabled status to True, you do not need to click Restore. When you click OK in the Metadata Properties dialog, DBMoto automatically calls the restore process to copy all data from the main connection to the backup connection. If you have a corrupted backup metadata, or the backup metadata tables no longer match the main metadata (because the connection was down for some time, for example), you need to click in the Restore field to force the backup to refresh its data.

## Object Browser

The Object Browser displays information related to the item selected in the Metadata Explorer. For example, if you select a source table in the database tree, you can see the table column names and column information. Primary key and foreign key columns are preceded by icons to identify them:

 Primary key column.

-  Foreign key column. Scroll across the display to view the table/column for which this column is the foreign key.

## Opening the Object Browser

This is the default tab when you open the DBMoto Management Center. You can switch between the Object Browser, [Replication Browser](#) and [Replication Monitor](#) by clicking on the tab names in the Management Center right pane.

## Options Dialog

[General Tab](#)

[Preferences Tab](#)

[Verifier Tab](#)

[Trace Tab](#)

### General Tab

#### Use Only Default Providers

Checked by default. If checked, only .NET providers are displayed in connection wizards whenever possible. This is to guide users in selecting a .NET provider over an OLE DB Provider or ODBC driver for the same database. Typically a .NET provider will give better performance.

#### Current GDI Charset (Font)

The default value, Ansi, works in most cases for displaying database data in the Management Center. However, if your field values contain Japanese data, for example, you would need to set this field to Japanese.

#### IMEMode Setting for Script

Select a language to enable entry of complex characters and symbols, such as Japanese Kanji characters, in the Script Editor using a standard keyboard.

#### Default Read Interval

The default frequency in seconds with which the replicator checks to see if there are transactions to replicate when replication is set in mirroring mode. The **Read Interval** value can be changed for each replication either when creating the replication using the [Replication Wizard](#), or changing the [Replication Properties](#) for an existing replication.

#### Transaction Latency Threshold

The default value for this option is 1 minute. It is used with the Transaction Latency Chart in the [Replication Activity Viewer](#). The chart compares the current time with the latest processed transaction time (or the last scheduled mirroring time if there are no transactions). If the lag between the two exceeds the **Transaction Latency**

**Threshold** setting, the shaded area in the graph is displayed in red and the **Latency Status column** in the [Replication Monitor](#) displays the value **Threshold Warning**. This can indicate that the current settings for the Data Replicator and the replication are not allowing the Data Replicator to keep up with the number of transactions. Adjust the Transaction Latency Threshold value as needed to provide a useful warning for your environment, depending on the number of transactions and the mirroring interval.

## Preferences Tab

Each preference can have one of the following settings:

- **Ask Always**  
The default setting for all preferences. This setting always provides a confirmation dialog in the Management Center before proceeding with the operation
- **Yes**  
Select this option to avoid confirmation dialogs. However, read the description of the preference carefully to ensure that you are setting the desired behavior.
- **No**  
Select this option to avoid confirmation dialogs. However, read the description of the preference carefully to ensure that you are setting the desired behavior.

### Copy Properties from Group

When adding a replication to an existing group, you can decide whether the group properties should override the replication properties. If you select **Ask Always**, you will be prompted each time you add a replication to a group. If you select **Yes**, the replication properties will be set to the group properties without prompting. If you select **No**, the replication is not added to the group. Note that there is no message indicating that the replication has not been added. You can check which replications are in a group in the **Metadata Explorer** by viewing the **Group Properties** from the right mouse button menu on the group name.

### Copy TID from Group

When adding a replication to an existing group, you can decide whether the group transaction ID should be applied to the replication properties. If you select **Ask Always**, you will be prompted each time you add a replication to a group. If you select **Yes**, the transaction ID for the replication will be set to match the transaction ID for the group. If you select **No**, the replication is added to the group, but retains its transaction ID.

### Create Target Tables

When creating a replication, and the target tables do not currently exist in the target database, you can use this option to create target tables automatically. If you select **Ask Always**, you will be prompted each time you create a replication. If you select **Yes**, the table is created without prompting. If you select **No**, target tables are not created automatically, but you cannot complete the mapping from source to target table required for the replication.

## Add Created Tables

When creating a replication, and the target tables do not currently exist in the target database, you can use this option to add tables to the metadata when they are created. If you select **Ask Always**, you will be prompted each time you create a replication. If you select **Yes**, the tables are added to the metadata without prompting. If you select **No**, target tables are not added to the metadata and you are not offered an option to add them. You can add the tables manually in the Metadata Explorer by selecting the target connection name and using the right mouse button option **Select Tables....**

## Rename Replications with Same Name

If you are using the [Custom Restore](#) option for restoring metadata, and you import replications with the same names as existing replications, you can use this option to rename the replications automatically. If you select **Ask Always**, you will be prompted before importing the replications. If you select **Yes**, the replications are renamed without prompting. If you select **No**, and there is a naming conflict, DBMoto generates an exception.

## Create Replications to Same Target

When creating a replication, if a selected target table is already assigned to another replication, this option allows you manage the warning about a potential conflict of replicating from two different sources to a single target table. If you select **Ask Always**, you will be prompted with a warning each time you create a replication. If you select **Yes**, the warning will not be displayed, and the replication(s) will be created even though there may be conflicts when replications are running. If you select **No**, there is no warning, but DBMoto does not create the replication and generates an exception.

## Verifier

### Max Errors Allowed When Reconciling Data

When using the DBMoto Verifier to reconcile differences in data between source and target tables, you can set a limit on the number of errors encountered before stopping the process. The default value is 5 errors, so if the data reconciliation process encounters more than 5 errors, it stops. This option is useful if, for example, a database connection fails because the option would limit the number of attempts to connect to a database.

## Trace

This tab is provided for debug purposes only. It should be enabled only when requested by HiT Software technical support staff. Trace files are text files located by default in the Log folder under the DBMoto install folder, with a .trc extension.

Click **Settings** to open the **Trace Settings** dialog. This dialog provides a "Trace metadata operations" option which traces all commands via the metadata connection. It is set to False by default and should only be set to True on request by the HiT Software Technical Support staff.

### Activate Management Center Tracing

Select this option only if requested by HiT Software technical support staff to trace activity in the Management Center. Use the options described in **File Settings** below to determine the location and size of the text file that is generated when running the trace.

## Activate API Tracing

Select this option only if requested by HiT Software technical support staff to trace activity by the DBMoto API. Use the options described in **File Settings** below to determine the location and size of the text file that is generated when running the trace.

## Activate Service Monitor Tracing

Select this option only if requested by HiT Software technical support staff to trace activity by the Service Monitor. Use the options described in **File Settings** below to determine the location and size of the text file that is generated when running the trace.

## File Settings

### Only one file

If selected, a single file with no size limit is used for traces. This file can quickly become very large and should be monitored on a regular basis.

### One file every X days

If selected, and a number entered, generates a new trace file after the number of days specified. All old trace files are kept in the log directory. The convention used to name the Management Center trace files is DBMotoEM\_XXXX.trc, where XXXX is a number starting at 0001. If you do not set the "Keep Max X Log Files" option, remember to clean up your log directory regularly.

### One file with size limited to X Mb

If selected, and a number entered, creates a new trace file when size X is reached. All old trace files are kept in the log directory. The convention used to name the Management Center trace files is DBMotoEM\_XXXX.trc, where XXXX is a number starting at 0001. If you do not set the "Keep Max X Log Files" option, remember to clean up your log directory regularly.

### Keep Max X Log Files

If unchecked, all old trace files are kept in the log directory. If checked, keeps only the specified number of old log files in the log directory. Use file dates (not file names) to determine the most recent files.

## Trace File Path

The default location for traces is the Log folder in the DBMoto installation folder. However, you can set a different folder for trace files as needed.

## Opening the Options Dialog

In the Management Center, from the **Tools** menu, choose **Options**.

## Oracle Change Log Settings Dialog

The Oracle Change Log Settings dialog can be used to:

- Change the Oracle server where the log is stored (for Oracle 12 multitenant database installations).
- Set a path to an Oracle dictionary file (primarily for Oracle versions up to 9)
- Change the archived log settings

## Log settings

### Server

Specifies the Oracle data source where the log is stored. If you are using the multitenant database architecture introduced in Oracle 12, and connecting to a pluggable database, you need to modify the log location to match the Oracle root, or container, database.

### User

User ID for the Oracle data source where the log is stored.

### Password

Password for the Oracle data source where the log is stored.

## Dictionary

### Dictionary File

If the log is stored as a dictionary file (typically only for Oracle versions up to 9), type the path and file name for an Oracle dictionary file. The Oracle dictionary file should already be defined in your Oracle environment. See [Setup for Different Oracle Versions](#) for more information.

## Archived Logs

### Read Archived Logs

When checked, DBMoto accesses Oracle archived redo logs for transactions in addition to the online redo logs. Check with your Oracle system administrator to see if your database is configured to use ARCHIVELOG mode. If it is, you can check the **Read Archived Logs** option to ensure that all transactions between mirroring intervals are identified by DBMoto. For example, if your mirroring interval is set for 90 seconds, but within that time the online logs fill with transactions, the online logs are archived as needed to accommodate additional transactions. If you check the **Read Archived Logs** option, DBMoto will be able to locate the last transaction ID from the archived log. Note that the **Read Archived Logs** option is not useful when using the default setting on Oracle databases, NOARCHIVELOG mode.

### Use CONTINUOUS\_MINE Option

When **ReadArchivedLogs** is checked, the **Use CONTINUOUS\_MINE Option** is checked by default. The CONTINUOUS\_MINE option instructs the Oracle log reader to load archived log files continuously as they get created, instead of loading a batch of files statically when the log reader is initially instantiated. DBMoto can therefore replicate transactions as they are backed up. however, Oracle recommends that the



CONTINUOUS\_MINE option should not be used in a RAC (Real Application Cluster) environment where RAC nodes are continuously enabled/disabled.

### **Destination ID**

This option allows you to set the location from which DBMoto will pick up Oracle archived logs for replication. When the value is set to the default, 0, DBMoto searches for Oracle archived logs in the standard destination, ('FLASH RECOVERY AREA (FRA)'). A value between 1 and 31 specifies an alternative location to retrieve archived logs. Oracle users can retrieve a list of all archive locations by running the query: `SELECT * FROM V$ARCHIVE_DEST`.

This feature is useful when your environment includes a customized archive setup and you want to instruct DBMoto use a specific location for archived logs.

### **Opening the Change Log Settings Dialog**

From the [Enable Transactional Replication Wizard](#) Log Settings screen or the [Manage Transactional Log Settings dialog](#), click ... to the right of the **Log Settings** field.

## **Predefined Values in the Expression Generator**

The [Expression Generator](#) provides a way for you to access certain values related to your replication while writing an expression. Expand the list of expression building blocks below the expression editing area to view the options available under **Values**.

### **Replication**

**[FieldName]** - Provides a template that shows you how to use a field name in a script. Double-click to insert the template in the script, then edit the text "FieldName". Note that you can also use the field names from the source table by expanding the Field node and double-clicking any one of the listed fields.

**DBRS.SourceConnection** - Provides the connection for the source table, and allows you to open a connection to the source table. For an example of how to use this value, see [Global Script Function Examples](#).

**DBRS.InternalSourceConnection** - Returns the .NET connection used internally by DBMoto for accessing to the source and to the target database, unlike SourceConnection which returns as clone of the connection object used by the Data Replicator engine. This property is useful in cases where a statement must be executed using the same connection as the one used by the engine. **WARNING:** Use this property with great care because improper use may compromise the success of the replication. Any error generated by running a script using this connection will affect the Data Replicator and the replication process. For example if the connection is closed by the script, the engine will fail. Typically it is preferable to use the SourceConnection property.

**DBRS.TargetConnection** - Provides the connection for the target table, and allows you to open a connection to the target table.

**DBRS.InternalTargetConnection** - Returns the .NET connection used internally by DBMoto for accessing to the source and to the target database, unlike TargetConnection which returns as clone of the connection object used by the Data Replicator engine. This property is useful in cases where a statement must be executed using the same connection as the



one used by the engine. **WARNING:** Use this property with great care because improper use may compromise the success of the replication. Any error generated by running a script using this connection will affect the Data Replicator and the replication process. For example if the connection is closed by the script, the engine will fail. Typically it is preferable to use the TargetConnection property.

### Field

Displays the column names from the source table. Double-click a column name to add it to the expression. If you plan to use a column name in an expression, make sure that you have set the column name to **Use Unmapped** in the Fields Mapping dialog. To set the **Use Unmapped** option, select the column name in the list of source columns, then choose **Use Unmapped** from the right mouse button menu.

### Log Field

The following values represent values retrieved from the source database transaction log

IBM Db2 for i	Oracle	Microsoft SQL Server
TransactionID (STRING)	TransactionID (String)	TransactionID (String)
TransactionTS (DATETIME)	TransactionTS (String)	TransactionTS (String)
RecordID (LONG)	RowID (String)	
UserID (STRING)		
ReceiverLibrary (STRING)		
ReceiverName (STRING)		

## Replication Activity Viewer

The Replication Activity Viewer provides a graphical representation of records processed during a specific replication instance and over multiple replication instances. It shows mirroring latency and volume history to help analyze the transaction volume, allowing you to adjust settings for maximum performance by the DBMoto Data Replicator.

The Viewer consists of three tools:

### Transaction Latency Chart

This graph compares the current time with the latest processed transaction time (or the last scheduled mirroring time if there are no transactions). If the lag between the two exceeds the **Transaction Latency Threshold** (set in the General tab of the [Data Replicator Options dialog](#)), the shaded area in the graph is displayed in red and the **Latency Status column**

in the [Replication Monitor](#) displays the value **Threshold Warning**. This can indicate that the current settings for the Data Replicator and the replication are not allowing the Data Replicator to keep up with the number of transactions.

The left-hand column of the graph displays values in seconds based on the highest value visible in the displayed area. As the graph is updated, the range in the left-hand column may reset.

The graph is automatically updated every 3 seconds. This parameter is not currently configurable.

### Graph interpretation

Grey	Indicates that there have been no new transactions during a period of time greater than the <b>Transaction Latency Threshold</b> set in the <a href="#">Options dialog</a> .
Blue	Indicates progress since the last transaction time that was read from the log. The last transaction time is stored in the Replication Properties as a read-only property.
Green	Indicates progress since the last time mirroring was scheduled to start (based on the mirroring interval set for the replication in the <a href="#">Replication Properties dialog</a> .)
Red	Indicates that the limit set in the <a href="#">Options dialog</a> <b>Transaction Latency Threshold</b> field has been reached. It shows that the transaction log checking interval cannot keep up with the number of transactions that need to be processed.  If you enable a replication after a period of time, the chart will initially show high values in red to indicate the significant difference between the last transaction processed and the current time. As soon as mirroring starts, the display will update.

### Records Processed Dial

Displays the speed of records processed based on the average records processed per interval. Note that the display is updated every three seconds, so the number of transactions processed is averaged over 3 seconds.

### History Chart

Displays the number records processed based on the selected time interval. This chart is updated every ten minutes. Note that the information displayed in this chart is obtained from the DBMoto Log. The chart reads the log every 10 minutes, looking for replication information. Therefore, this chart can affect performance in the Management Center (not in replications.) Performance is typically better if the DBMoto Log is stored in a database rather than in text files. HiT Software recommends closing the Replication Activity tab if you are not actively using it to avoid affecting response time in the Management Center.

### Opening the Replication Activity Viewer

In the DBMoto Management Center Metadata Explorer, Replication Browser, or Replication Monitor, select a replication and, from the right mouse button menu, choose **Show Replication Activity**.

## Replication Alert Properties Dialog

This dialog allows you to set up email alerts for specific events that occur when processing the selected replication. Use the **Alert Settings** tab to choose which events to include in an email, and use the **Message Layout** tab to specify email recipients and email content. In addition to defining the email content in this dialog, you must set email server properties in the [Mail tab](#) of the [Data Replicator Options dialog](#).

As an alternative, or in addition to email alerts for a specific replication, you can [set up email alerts at the server level](#) for all replications defined on the server using the [Alerts tab](#) in the [Data Replicator Options dialog](#).

Note: It is also possible to create a custom email alert by writing a script that uses the [SendMail function](#).

## Filtering Alerts

Where the alert type allows, it is possible to set a filter to identify specific conditions for an alert. The filter field is displayed to the right of the alert name and can contain logical operators in addition to specific strings. The supported logical operators are AND, OR, NOT (uppercase with precedence as shown). Parentheses can be used to change operator precedence. Substrings to be matched with message content should be enclosed in quotes (') or double quotes(").

Examples:

- All messages for enabled replications except "EMP50" and "EMP51":

"enabled" AND NOT ("EMP50" OR "EMP51")

- All messages not containing the substring "warning 256":

NOT "warning 256"

- All messages containing "error" or "warning":

"error" OR "warning"

## Alert Settings Tab

Replication Events	
<b>Replication Info</b>	Reports all comments generated by DBMoto in the "Information" category for the selected replication. The filter option allows you to set a string to identify a subset of "Information" messages you want to receive.
<b>Replication Errors</b>	Reports all comments generated by DBMoto in the "Error" category for the selected replication. The filter option allows you to set a string to identify a subset of "Error" messages you want to receive.
<b>Replication Warnings</b>	Reports all comments generated by DBMoto in the "Warning" category for the selected

	replication. The filter option allows you to set a string to identify a subset of "Warning" messages you want to receive.
<b>Reached Max Replication Latency</b>	The <b>Transaction Latency Threshold</b> setting in the General tab of the <a href="#">Data Replicator Options dialog</a> has been exceeded by comparing the current time with the latest processed transaction time (or the last scheduled mirroring time if there are no transactions). If the lag between the two exceeds the <b>Transaction Latency Threshold</b> (set in the General tab of the <a href="#">Data Replicator Options dialog</a> ), an email message is generated. This can indicate that the current settings for the Data Replicator and the replication are not allowing the Data Replicator to keep up with the number of transactions.
<b>Reached Max Replication Downtime</b>	Sets a maximum number of hours that the selected replication could be stopped before sending an email notification. For example, if the number of hours is set at 3, an email would be send once the down time for the replication reaches 3 hours.

### Alert Schedule

Use this option to limit the number of email messages you receive for a defined set of alerts.

### Message Layout Tab

Use this tab to define the alert email recipients and the overall layout of the email message

### Recipients

A semi-colon separated list of email addresses.

### Title

A default email subject line is provided, but you can edit the text to provide your own subject line.

### Body

A default email body format is provided, but you can edit the text to customize the email format.

### Html Format

Check this option to send email messages in HTML format.

### Attachment

To attach the current DBMoto text file log to the email, select DBMoto Log from the drop-down list.

## Opening the Replication Alert Properties Dialog

In the Management Center **Metadata Explorer**, select a replication. From the right mouse button menu, choose **Replication Properties**. In the **Replication Properties** dialog, go to the Alerts tab. Click **Add Alert** to open the **Replication Alert Properties** dialog.

## Replication Browser

Once you have created a replication, it is displayed in the Replication Browser with enough information about the replication so that you can easily identify it. Use the right mouse button menu to edit [Replication Properties](#) and to enable or disable the replication.

Select a replication in the upper pane to view details of the columns to view details of the columns to be replicated in the lower pane.

## Opening the Replication Browser

In the DBMoto Management Center right pane, click the Replication Browser tab.

## Replication Monitor

The Replication Monitor tab provides current status information for all replications defined in the metadata that is selected in the Metadata Explorer.

Select a replication in the Replication Monitor tab to view some of its properties in the Properties pane. To view all replication properties, select the replication and, from the right mouse button menu, choose Replication Properties to open the [Replication Properties dialog](#).

Further information is available for:

[Replication Status Column](#)

[Initialization Status Column](#)

[Last Replication Status Column](#)

[History Status Column](#)

[Latency Status Column](#)

## Replication Status Column

The **Replication Status** column displays one of the following values:

<b>Idle</b>	The replication is active, or enabled, and ready to run but is not currently running.
<b>Disabled</b>	The replication is disabled. To enable or disable a replication, select the replication and use the right mouse button menu. When a replication is disabled, it cannot run. You must enable the replication so that it can be added to the Data Replicator task list.
<b>Refreshing</b>	The replication is currently running. A refresh operation is being performed.

<b>Mirroring</b>	The replication is currently running. A mirroring operation is being performed.
------------------	---

## Initialization Status Column

The **Initialization Status** column is used to display and manage the status of an Initial Refresh replication. It contains one of the following values

--	Uninitialized -- a Refresh operation has never run.
<b>Requesting</b>	A Refresh request has been submitted, but the DBMoto Data Replicator is not running, so the request cannot be processed.
<b>Pending</b>	A Refresh request has been submitted to the Data Replicator but the refresh operation has not completed.
<b>Executed</b>	A Refresh operation has successfully been executed

When creating a replication in the [Replication wizard](#), it is possible to set an Initial Refresh where all selected data is copied from the source table(s) to the target table(s) prior to running the regular replication schedule. In the [Scheduling screen](#), check the **Execute Initial Refresh** option.

To change the initialization status of a replication, right click on it and select 'Run Initial Refresh'. If the Data Replicator is not running, the status changes immediately, putting the replication into a temporary mode ("Requesting...") until the Data Replicator accepts the request. When the Data Replicator is running, once it has accepted the initialization request, the Initialization Status changes to Pending. The operation can also be performed on disabled replications, where the Refresh operation is executed once the Data Replicator is started and the replication enabled. A Pending status can be removed by selecting the replication and the same Run Initial Refresh menu item.

**NOTE:** It is possible to set the **Run Initial Refresh** option for an individual replication within a group, or for a whole group. To set **Run Initial Refresh** for a replication within a group, select the replication in the Replication Monitor and choose **Run Initial Refresh** from the right mouse button menu. To set **Run Initial Refresh** for a group, in the Metadata Explorer, select the group and choose **Run Initial Refresh** from the right mouse button menu.

## Last Replication Status Column




The **Last Replication Status** column can provide information about specific events that have recently occurred.

<b>Success (0)</b>	The replication was executed with no errors.
<b>Warning (1)</b>	The replication was executed through the end but some errors or warning were generated.

<b>Errors (2)</b>	The replication was stopped by the replication engine for some major problem. Check the <a href="#">DBMoto Log</a> for more information about the error.
<b>Aborted (3)</b>	Not currently implemented.
<b>Stopped (4)</b>	The replication was stopped by the user.

## History Status Column

The **History Status column** provides error information for both the most recent replication and for prior replications. When a replication error is flagged, select the replication and, from the right mouse button menu, choose **Show History** to open the [History Viewer](#) and explore the error(s). Note that, to record a replication status history, you need to set the **Activate Log History** option in the Log tab of the [Data Replicator Options](#) dialog.

	Replication(s) completed successfully without errors.
	One or more errors occurred during the replication(s). Select the replication and, from the right mouse button menu, choose <b>Show History</b> to open the <a href="#">History Viewer</a> and explore the error(s).
	The replication status reported by the History flag has been cleared. If the History Status column initially showed an error flag and you have examined the error, you can change the flag to show that the error has been investigated. Select the replication and, from the right mouse button menu, choose <b>Clear History</b> to display this flag.

## Latency Status Column

The **Latency Status** column provides information on the current status of the replication. It reports a warning when the latency period between the current time and the latest processed transaction time (or the last scheduled mirroring time if there are no transactions) exceeds the **Transaction Latency Threshold** value in the [Options dialog](#). When you see a warning in this column, open the Replication Activity tab by selecting the replication, then choosing **Show Replication Activity** from the right mouse button menu.

### Opening the Replication Monitor

In the DBMoto Management Center toolbar, select  **Replication Monitor**.

## Replication Properties Dialog

[General Tab](#)

[Scheduler Tab](#)

[Alerts Tab](#)

## [Preferences Tab](#)

### [Opening the Replication Properties Dialog](#)

## General Tab

### Replication Name

Type a name for the replication. This name will be stored in the metadata and used to identify the replication in the Replication Browser. It is helpful to include information about the source and/or target tables in the replication name.

### Description

Optional. Type a description for the replication. This information will be stored in the metadata and can be accessed when you need more information about the replication.

### Source Connection

Choose a source connection name from the drop-down list.

### Source Table

Choose a source table from the drop-down list.

### Target Connection

Choose a target connection name from the drop-down list.

### Target Table

Choose a target table from the drop-down list.

### Replication Mode

Choose a replication mode from the options below.

### Refresh

Mass replication is executed once based on the time setting specified in the [Scheduling](#) screen. All records in the target table are deleted before performing the refresh operation, although this behavior can be modified by [writing a script](#) for the [Refresh onBeforeTruncate](#) event.

### Mirroring

Continuous replication based on the source database transaction log, where the transaction log is checked according to the mirroring interval set in the Replication wizard or on the [Preferences tab](#) of this dialog.

Mass replication (refresh) can be executed once in order to fill the target records. Real-time mirroring begins as soon as the refresh changes have taken place. The incremental replication is executed at user-determined intervals.



## Synchronization

Mass replication (refresh) can be executed once in order to fill the target records. Real-time bi-directional mirroring begins as soon as the refresh changes have taken place. The incremental replication is executed at user-determined intervals.

### Automatic Field Mapping

Check this option to define mapping from target to source fields. Mappings created in this way are saved as part of the metadata. If you do not check this option, you will need to define scripts to create mappings when the replication runs.

Click Mapping to open the [Fields Mapping dialog](#). Use this dialog to customize the mapping between source and target tables.

### Use Script

Check this option if you want to [define a replication script](#) to run during replication. Define the script by clicking **Script** to open the [Replication Script Editor](#).

## Scheduler Tab

### Start Time

Select a start date and time. Click **Now** to set the current time and date.

- For the Refresh schedule, when the Replicator is running, this time will be used to determine when to schedule the first replication.
- For the Mirroring schedule, when the Replicator is running, this time will be used to determine when to schedule the first interruption to mirroring.
- For the Verifier schedule, when the Verifier Scheduler service is running, this time will be used to determine the first verification process.

### Selecting a Date

You can select a date either by typing in the month, day and year, or by clicking the down arrow to view a scrollable monthly calendar.

To type in a date, first select the month, then type in a new value, then select the day and type in a new value and finally select the year and type in a new value.

If you click the down arrow, it displays the date that corresponds to the value currently set in the field. Use the left and right arrows to locate the month you want, then select a date by clicking on it. The window closes when you select a date and the date is displayed in the Start Time field.

### Selecting a Time

To select a time, first select the hours, minutes or seconds, then type a new number or use the up and down arrows to increase or decrease the number.

## Refresh Schedule

### Run One Time Only

Select this option if you want to replicate data only once.

### Run Recurrently

Select this option if you want to replicate data on a regular schedule. When you select this option, the Schedule button is activated. Click Schedule to open the [Scheduler dialog](#).

NOTE: To run a complete refresh replication immediately the replication is enabled and the start time has been reached, check the Run Initial Refresh option on the right mouse button of the replication name in the Metadata Explorer, [Replication Browser](#) or [Replication Monitor](#). Subsequent replications occur as scheduled. Target table records are deleted prior to running the initial refresh operation. You can avoid truncation of target tables by [writing a script](#) for the [Refresh\\_onBeforeTruncate](#) event.

For mirroring replications which include an initial refresh, DBMoto handles all transactions which occur while the initial refresh was taking place as part of the first mirroring phase by considering all transactions between the initial refresh start and the initial refresh end as a special case and verifying that all these transactions are handled appropriately.

For a synchronization replication, the initial refresh is always performed from the source connection to the target connection. Note that any transaction submitted during the time that the refresh is running might not be replicated. It is strongly suggested that you avoid updating the designated source and target tables until the refresh is done.

## Mirroring Schedule

### Run Continuously

Select this option to continuously mirror activity between the source and target database. The start of mirroring is determined by settings in the Scheduling screen. If you have enabled replication and set the start time, then mirroring will begin at the start time specified. If you have also checked the option to execute an initial refresh, then mirroring will begin once the database refresh is complete. If **Enable Replication** is not checked, neither the initial refresh or continuous mirroring will begin at the time specified in **Start Time**.

### Schedule Interruptions

Select this option if you need to schedule one or more interruptions to the mirroring schedule. When you select this option, the Schedule button is activated. Click **Schedule** to open the [Scheduler dialog](#).

## Verifier Schedule

### Run One Time Only

Select this option if you want to perform a verification only once.

### Run Recurrently

Select this option if you want to run verifications on a regular schedule. When you select this option, the Schedule button is activated. Click **Schedule** to open the [Scheduler dialog](#).

## Alerts Tab

Use this tab to create and manage email alert messages for the selected replication. It provides a way to define email messages for specific events that occur when the replication is running.

### Replication Alerts Grid

The grid displays alerts that have been defined using the **Add Alert** button. Select an alert to edit or remove it.

#### Add Alert

Click Add Alert to define an email message alert in the [Replication Alert Properties dialog](#). Note that newly defined email alert messages will only be sent after [stopping and restarting the Data Replicator](#).

#### Remove Alert

Select an alert in the grid, then click **Remove Alert** to delete the email alert.

#### Edit Alert

Select an alert in the grid, then click **Edit Alert** to open the [Replication Alert Properties dialog](#) and edit the email alert.

## Preferences Tab

### General


**Thread Priority:** If running multiple replications concurrently (via the setting in the [Data Replicator Options dialog](#) General tab), you can set the priority for the selected replication. See also [Managing Performance Using Thread Settings](#).

**Record Pool Size:** This is an optimization feature that allows you to set the size of the buffer used to store records in memory while processing records for replication. The default size is 1000 records. For example, if you are performing a SELECT on a large table, you may need to increase the buffer size. However, the buffer size should only be increased where there is sufficient system memory. In general, increasing this number can enhance performance if the [reading task is faster than the writing task](#), i.e. operations involved in reading records from the source table are faster than operations involved in writing changes to the target table


**Stop on Schema Change:** False by default. When set to true, instructs the Data Replicator to disable a replication whenever a schema change operation is detected, either when validating the replication or when the Data Replicator finds a schema change record in the log. See [Managing Source/Target Database Schema Changes](#) for more detail on how to use this option.

**Override Identity Fields:** False by default. For use with mapped Identity fields in Microsoft SQL Server and IBM Db2 for i target tables only. When set to True, this allows writing of the source Identity value to the target using extra clauses to “overwrite” the ID. If the property is set to False, but the Identity field is mapped, the database generates an error on attempting to write to a target Identity field.

## Refresh Options

**Refresh Filter Source:** Allows you to specify a WHERE condition to apply during refresh applications to filter the records that are replicated. Click  to open the [Expression Generator](#) and write a SQL expression to apply during the refresh replication.

**Order By Clause Source:** Provides access to the Expression Generator to create an expression for additional control over processing records.

**Refresh Filter Target:** Allows you to specify a WHERE condition to apply during the deletion of records on the target table prior to a refresh operation. Click  to open the [Expression Generator](#) and write a SQL expression to apply during the refresh replication.

**Refresh Interval:** Default value is 60 seconds. Allows you to configure how long to wait before retrying a refresh in cases where the refresh replication was unable to start. The value must be expressed in seconds and greater than 0. A value of 0 means that there will be no attempt to retry the refresh replication.

**Refresh Recovery:** When checked, and a refresh replication is interrupted while running, the DBMoto Replicator is able to pick up the replication from the point at which it was interrupted. However, for this option to work, the source table involved in the replication must have one or more primary keys defined. If there are no primary keys, and the option is checked, a warning is printed in the DBMoto log and recovery from the interruption is not possible. Refresh will begin at the first record. If you are using a replication group, refresh recovery will start at the replication where the error occurred rather than starting at the first replication in the group. If you check this option, you may find that the replication runs more slowly. If performance is adversely affected, uncheck the option.

**Insert Mode:** The options are:

- SingleInsert--inserts record by record using SQL INSERT statements
- BulkInsert--inserts blocks of records. Block size is determined by the Block Size option. Choose this option for increased performance if using one of the data access providers which currently support bulk inserts:

HiT Software Ritmo/i (IBM Db2 for i) .NET Provider  
SAP HANA ODBC driver  
SAP HANA .NET Provider  
SAP Sybase ASE .NET Provider  
SAP Sybase IQ .NET Data Provider  
Microsoft SQL Server .NET Provider (using SQL Server's SqlBulkCopy)  
MySQL .NET Data Provider  
PostgreSQL .NET Data Provider  
Oracle .NET provider.

For SQL Server, note that BulkInsert is the default option. Record details are not reported in the DBMoto log when BulkInsert is set.

- SimulatedBulk--For databases that do not support bulk insertion, DBMoto simulates the effect by grouping records into blocks before performing an insert. This is a way to optimize performance.

**Bulk Type:** Depending on the database type and provider type, some of the following options may not be available.

- ArrayBinding--inserts multiple rows at a time using an array of parameters
- Native--uses the native functionality of the provider, if available
- FTP--bulk insert implemented through FTP

**Block size:** Set the value for the number of rows to insert in a single operation using the Bulk option above.

**Isolation level:** This option allows you to choose a specific isolation level on the Refresh operation. Choose from the following options available through your .NET Framework data provider:

Unspecified	No level is determined.
Chaos	The pending changes from more highly isolated transactions cannot be overwritten.
ReadUncommitted	A dirty read is possible, meaning that no shared locks are issued and no exclusive locks are honored.
ReadCommitted	Shared locks are held while the data is being read to avoid dirty reads, but the data can be changed before the end of the transaction, resulting in non-repeatable reads or phantom data.
RepeatableRead	Locks are placed on all data that is used in a query, preventing other users from updating the data. Prevents non-repeatable reads but phantom rows are still possible.
Serializable	A range lock is placed on the DataSet, preventing other users from updating or inserting rows into the dataset until the transaction is complete.

**Skip Record Count:** Default is False. Set to true to skip the SELECT COUNT(\*) operation during refresh replications. This command is used only to show replication progress in the Replication Monitor, and omitting the operation may improve performance.

**Fire Triggers:** When the target database is SQL Server and the Insert Mode is set to BulkInsert, this option can be set to True to fire any triggers defined on the target database. The default value is False to provide optimal performance during replication.

## Mirroring Options

**Read Interval:** 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 journal/log every 90 seconds to see if any transactions have occurred that need to be replicated to the target table.

**Command Pool Size:** The number of prepared SQL commands that are kept ready for execution during mirroring. Commands are added to the pool as they are used the first time on a last in, first out approach. Before executing a command, DBMoto verifies if the command is already prepared in the pool. If it is, the command from the pool is used (saving time, because it is already prepared), otherwise a new command is created, prepared and executed. After execution the command is moved to the top of the pool. When the pool is full and a new command is needed, the last command in the pool is closed and released to make space for the new command. The default pool size of 10 can be modified up to the number of prepared commands that your database will support. This is an optimization feature and can improve performance.

**Conflict Resolver:** This property applies only to synchronization replications and determines how [conflicts](#) will be resolved. If you have not defined a synchronization replication, the property and value are inactive and cannot be modified. If you have defined a synchronization replication, you can choose from the following values:

- **SourceServerWins:** This is the default value. Changes applied to the table defined in the source connection are also applied to the table defined in the target connection(s), overriding any changes that have occurred in the target connection table.
- **TargetServerWins:** Changes applied to the tables defined in the target connections are also applied to the table defined in the source connection, overriding any changes that have occurred in the source connection table. This option also requires you to set a priority for the target tables involved so that, if changes occur in more than one target table, DBMoto can determine which changed value to use. Set the priority for target connections as follows.
  1. In the Metadata Explorer, select the group.
  2. From the right mouse button menu, choose **Group Properties**.
  3. In the [Group Properties dialog](#), go to the **Preferences** tab.
  4. Click in the **Conflict Resolver Priority** field to view the target connections.
  5. Order the target connections by selecting the connection, then using the up and down arrows to determine its priority.
- **FirstComeWins:** The timestamps of the changes in the tables designated as source and target(s) are compared and the change that applied earliest is applied to all tables.
- **Use Script:** This option generates an event that can be handled by writing a function [Replication\\_onConflict](#). The script editor can be accessed from the General Tab Use Script option. You can [write a function](#) that handles the values from the tables in any way you wish using VB .NET.

**NOTE:** In the rare case that foreign key values are updated when synchronization has been defined, the synchronization algorithm may not work and may cause foreign key constraint violation issues. Please contact our support team via the [Help Center](#) if you need to modify foreign key values for replications with synchronization.

**Commit Mode:** Default is AutoCommit where each statement is replicated as a standalone transaction. This means that only statements that fail will not be replicated. The CommitmentControl option applies all the statements associated with a transaction as a whole to the target. CommitmentControl is supported on Oracle (Log Reader and Log Server Agent), Microsoft SQL Server, IBM Db2 for i, IBM Db2 LUW, IBM Informix.

### Mirroring Options - Source

**Commit Mode:** Default is AutoCommit where each statement is replicated as a standalone transaction. This means that only statements that fail will not be replicated. The CommitmentControl option applies all the statements associated with a transaction as a whole to the target. CommitmentControl is supported on Oracle (Log Reader and Log Server Agent), Microsoft SQL Server, IBM Db2 for i, IBM Db2 LUW, IBM Informix.

**Mirroring Insert Mode:** A default value can be assigned in the Connection Properties dialog for the target connection, but the value can be changed here for each replication. The options are

- SingleInsert--inserts record by record using SQL INSERT statements
- BulkInsert--inserts blocks of records. Block size is determined by the Block Size option. Choose this option for increased performance if available

**Mirroring Block Size:** Set the value for the number of rows to insert in a single operation when using the BulkInsert option above.

**Commit Isolation Level:** This option allows you to choose a specific isolation level when the Commit Mode is set to CommitmentControl. Choose from the following options.

Unspecified	No level is determined.
Chaos	The pending changes from more highly isolated transactions cannot be overwritten.
ReadUncommitted	A dirty read is possible, meaning that no shared locks are issued and no exclusive locks are honored.
ReadCommitted	Shared locks are held while the data is being read to avoid dirty reads, but the data can be changed before the end of the transaction, resulting in non-repeatable reads or phantom data.
RepeatableRead	Locks are placed on all data that is used in a query, preventing other users from updating the data. Prevents non-repeatable reads but phantom rows are still



	possible.
Serializable	A range lock is placed on the DataSet, preventing other users from updating or inserting rows into the dataset until the transaction is complete.
Snapshot	No locks on underlying data in a snapshot transaction, so other transactions can execute.

### Transaction Read Point - Source

This section of the dialog may contain database-specific fields in addition to the ones listed below.

**Transaction ID:** The ID for the transaction at which you want to start replication. If you want to change the transaction ID, click ... to open the Read Point dialog.

In the Transaction Read Point dialog, you can either retrieve the current transaction or the transaction for a specified date and time.

For SQL Server and trigger-based replications, if you choose the **Current Sequence** option, you can also check the **Refresh Transactional Replication Objects** option. This is useful, for example, when you know that SQL Server replication objects (article, publisher) have been manually removed or may need to be repaired or restored. A replication will not run if the database objects used for transactional replications are missing or corrupted.

If you enter a date and time, DBMoto retrieves the first transaction after the time entered. This information is available and can be changed in the [Replication Properties dialog](#) after the wizard is completed.

The transaction ID is set and transactional objects are refreshed (if checked) when you close the dialog.

**Commit TID:** A read-only property that indicates the location in the current command index for the transaction that is being processed.

**Transaction Timestamp:** The timestamp for the transaction above.

**Log Files Folder:** The location of the DBMoto log.

**Log File Name:** The name of the DBMoto log file.

**Master Table Name:** Appears only when replicating from [a source database that uses triggers](#). A read-only property that displays the name of the master table created by DBMoto in the source database and used to store information for trigger-based replications.

**Log Table ID:** Appears only when replicating from [a source database that uses triggers](#). A read-only property that displays the ID included in the name of the log table created by DBMoto in the source database and used to store information for trigger-based replications. The full name of the table would be `_DBM__LOG_#`, where # is the ID displayed in this field.



## Mirroring Options - Target

NOTE: The settings below are used only when performing synchronization.

**Commit Mode:** Default is AutoCommit where each statement is replicated as a standalone transaction. This means that only statements that fail will not be replicated. The CommitmentControl option applies all the statements associated with a transaction as a whole to the target. CommitmentControl is supported on Oracle (Log Reader and Log Server Agent), Microsoft SQL Server, IBM Db2 for i, IBM Db2 LUW, IBM Informix.

**Mirroring Insert Mode:** A default value can be assigned in the Connection Properties dialog for the target connection, but the value can be changed here for each replication. The options are

- SingleInsert--inserts record by record using SQL INSERT statements
- BulkInsert--inserts blocks of records. Block size is determined by the Block Size option. Choose this option for increased performance if available

**Mirroring Commit Isolation Level:** This option allows you to choose a specific isolation level when the Commit Mode is set to CommitmentControl. Choose from the following options.

Unspecified	No level is determined.
Chaos	The pending changes from more highly isolated transactions cannot be overwritten.
ReadUncommitted	A dirty read is possible, meaning that no shared locks are issued and no exclusive locks are honored.
ReadCommitted	Shared locks are held while the data is being read to avoid dirty reads, but the data can be changed before the end of the transaction, resulting in non-repeatable reads or phantom data.
RepeatableRead	Locks are placed on all data that is used in a query, preventing other users from updating the data. Prevents non-repeatable reads but phantom rows are still possible.
Serializable	A range lock is placed on the DataSet, preventing other users from updating or inserting rows into the dataset until the transaction is complete.
Snapshot	No locks on underlying data in a snapshot transaction, so other transactions can execute.

## Transaction Read Point - Target

**Transaction ID:** The ID for the transaction at which you want to start replication. If you want to change the transaction ID, click ... to open the Read Point dialog.

In the Transaction Read Point dialog, you can either retrieve the current transaction or the transaction for a specified date and time.

For SQL Server and trigger-based replications, if you choose the **Current Sequence** option, you can also check the **Refresh Transactional Replication Objects** option. This is useful, for example, when you know that SQL Server replication objects (article, publisher) have been manually removed or may need to be repaired or restored. A replication will not run if the database objects used for transactional replications are missing or corrupted.

If you enter a date and time, DBMoto retrieves the first transaction after the time entered. This information is available and can be changed in the [Replication Properties dialog](#) after the wizard is completed.

The transaction ID is set and transactional objects are refreshed (if checked) when you close the dialog.

**Commit TID:** A read-only property that indicates the location in the current command index for the transaction that is being processed.

**Transaction Timestamp:** The timestamp for the transaction above.

**Log Files Folder:** The location of the DBMoto log.

**Log File Name:** The name of the DBMoto log file.

**Master Table Name:** Appears only when replicating from [a source database that uses triggers](#). A read-only property that displays the name of the master table created by DBMoto in the source database and used to store information for trigger-based replications.

**Log Table ID:** Appears only when replicating from [a source database that uses triggers](#). A read-only property that displays the ID included in the name of the log table created by DBMoto in the source database and used to store information for trigger-based replications. The full name of the table would be `_DBM__LOG_#`, where # is the ID displayed in this field.

## Verifier Options

These options are also available in the [DBMoto Verifier Options dialog](#). Note that if you set an option below, you need to close and re-open the Verifier tab before the changes are visible in the Verifier Options dialog. Changes you make to Verifier settings in the Replication Properties dialog will be saved to the DBMoto metadata and available to any Verifier activity for the replication.

**Records Count Only:** False by default. When checked, the verification process compares only the number of records in the source and target tables.

**Compare Primary Keys Only:** False by default. When checked, the Verifier compares only the primary key field value(s) in source and target tables to determine differences between the tables. Note that **Reconcile Data** is not

enabled when Verify Primary Key Only has been selected because differences and column data are not recorded during the operation to compare primary keys only.

**Trim Chars:** True by default. This option provides a way for the tool to ignore blank characters in string values. Some databases set string values at a fixed length using blank spaces to fill the string, while others permit strings of different lengths. If you check this option, the DBCompare tool trims all strings with fixed string length, then compares the strings.

**Date Time Options:** Set to **UseFractionalDigits** by default. Because databases can store date time values in different formats, you can select a value in this field to choose how to deal with date time fields. One of three values is possible:

- **SkipDateTime** - when selected, any Date Time field values will be ignored when comparing source and target tables.
- **SkipTime** - when selected, any Time field values will be ignored when comparing source and target tables.
- **UseFractionalDigits** - Allows you to compare field values using a selected number of many to the right of the decimal point

**Fractional Digits:** 0 by default. Specifically for time values, and works in conjunction with the **UseFractionalDigits** value in the **Date Time Options** field. This option allows you to determine how many digits to the right of the decimal point you wish to use in comparing field values.

**Skip Clob Blob:** False by default. Any BLOB/CLOB field values will be included when comparing source and target tables. This option is unchecked by default, but comparing BLOB/CLOB values can be very time consuming so you may want to change the setting.

**Skip Array of Bytes:** Unchecked by default. Any Byte Array field values will be included when comparing source and target tables.

**WHERE Condition Source Table:** Provides a way to specify a condition to narrow the number of rows compared and reported. Click in the value area to open the [DBMoto Expression Generator](#) and write the WHERE condition.

**WHERE Condition Target Table:** Provides a way to specify a condition to narrow the number of rows compared and reported. Click in the value area to open the [DBMoto Expression Generator](#) and write the WHERE condition.

**ORDER BY Clause Source Table:** The default ORDER BY clause in the Verifier is to order by primary key. However, this can be problematic when the primary key is a string because database providers can order strings differently. This field allows you to override the default ORDER BY clause by typing a single column name, or a list of comma-separated column names. Click in the value area to open the [DBMoto Expression Generator](#) and write the ORDER BY clause.

**ORDER BY Clause Target Table:** The default ORDER BY clause in the Verifier is to order by primary key. However, this can be problematic when the primary key is a string because database providers can order strings differently. This field allows you to override the default ORDER BY clause by typing a single column name, or a

list of comma-separated column names. Click in the value area to open the [DBMoto Expression Generator](#) and write the ORDER BY clause.

### Opening the Replication Properties Dialog

In the Metadata Explorer, select the replication. From the right mouse button menu, choose **Replication Properties**.

In the Replication Browser or Replication Monitor, select the replication. From the right mouse button menu, choose **Replication Properties**.

## Replication Script Editor

Use this editor to write a script that is run when a specific event occurs during replication. The script should consist of one or more functions selected from the drop-down lists at the top of the Editor. The drop-down list on the left displays the categories for which you can define an event handler. The Editor provides [editing and formatting features](#) that include automatic indentation, automatic coloring for syntax, numeric values, comments, “code folding”, and keyboard shortcuts.

Category	Events	Event Type	Explanation
Refresh	<a href="#">onBeforeTruncate</a> <a href="#">onAfterTruncate</a> <a href="#">onBeforeRefresh</a> <a href="#">onAfterRefresh</a> <a href="#">onPrepareRefresh</a>	<a href="#">Writer</a> <a href="#">Writer</a> <a href="#">Writer</a> <a href="#">Writer</a> <a href="#">Reader</a>	Available events apply to refresh replications only.
LogReader	<a href="#">onPrepareMirroring</a> <a href="#">onBeforeMirroring</a> <a href="#">onAfterMirroring</a> <a href="#">onReceiverChanged</a>	<a href="#">Reader</a> <a href="#">Reader</a> <a href="#">Reader</a> <a href="#">Reader</a>	Available events apply only to mirroring and synchronization replications. <code>onReceiverChanged</code> applies only to replications involving System i/iSeries/AS400.
Record	<a href="#">onBeforeMapping</a> <a href="#">onAfterMapping</a> <a href="#">onBeforeExecute</a> <a href="#">onAfterExecute</a>	<a href="#">Reader</a> <a href="#">Reader</a> <a href="#">Writer</a> <a href="#">Writer</a>	Available events apply to each record that is considered for replication. In the case of refresh replications, the events would apply to every record in the source/target table. In the case of mirroring and synchronization replications, the events would apply to records found in the transaction log.

Replication	<a href="#">onCriticalError</a> <a href="#">onConflict</a> <a href="#">onLateConflict</a>	<a href="#">Writer</a> <a href="#">Reader</a> <a href="#">Reader</a>	<p>The onCriticalError event applies to all replication types. It has been replaced by global events <a href="#">Record onExecuteError</a> and <a href="#">Replication onError</a>. onConflict and onLateConflict should be used to <a href="#">resolve conflicts</a> only for synchronization replications.</p>
-------------	---	--	--

The right drop-down list displays a list of appropriate [events](#) for which you can write a script. If no function was selected from the left side, the list remains empty.

The default Visual Basic template content of the Replication Script Editor looks as follows. However, it is possible to write the script using C# instead of Visual Basic by setting the scripting language in the [Global Script Editor](#), accessible from the right mouse button menu on the metadata.

```
Imports Microsoft.VisualBasic
```

```
Imports DBMotoPublic
```

```
Imports DBMotoScript
```

```
Imports DBRS.GlobalScript
```

```
Namespace DBRS
```

```
    Public Class ReplicationScript : Inherits IReplicationScript
```

```
    End Class
```

```
End Namespace
```

When you choose an event to add to the script, it is always added directly below the ReplicationScript class as in the example below.

```
Imports Microsoft.VisualBasic
```

```
Imports DBMotoPublic
```

```
Imports DBMotoScript
```

```
Imports DBRS.GlobalScript
```

## Namespace DBRS

```

Public Class ReplicationScript : Inherits IReplicationScript

    Public Overrides Sub Record_onBeforeMapping(recSource As IRecord, ByRef AbortRecord As Boolean)


        End Sub






    End Class

End Namespace

```

Add code to manage the event. You can include functions from other libraries in the script code, but be sure to add the library to the list of Imports, add the library location using the References dialog and set up a namespace for the library.

You can access the References dialog by clicking  in the toolbar below the script.

	Checks the script syntax. If you do not do this in the script editor, you may encounter errors when the replication is running.
	Opens the <a href="#">Expression Generator</a>
	Opens the References dialog so that you can add the location of any dlls that you are using from your script. You typically need to add dlls to the References dialog and to the script Imports when they do not belong to the default Microsoft .NET environment.
	Creates a comment for selected text
	Uncomments selected text

## Editing and Formatting Features

1. Automatic indentation based on syntax
2. Colors to identify specific code elements: syntax in blue, numeric values in pink and comments in green

3. Code folding where sections of code can be collapsed or expanded for ease of reading. The editor displays a green bar on the left side with +/- signs. Click - to collapse a section of code, and click + to expand a section of code.
4. Common editing operations using keystroke combinations:

<SHIFT>Left, Right, Up, Down, Home, End, PageUp, PageDown	Move caret with selection
<CTRL>C <CTRL>V <CTRL>X	Copy, paste, cut
<CTRL>A	Select all text
<CTRL>Z, <CTRL>Y	Undo/Redo
Tab <SHIFT>Tab	Increase/decrease left indent of selected range
<CTRL>Home <CTRL>End	Go to first/last char of the text
<SHIFT><CTRL>Home, <SHIFT><CTRL>End	Go to first/last char of the text with selection
<CTRL>U <SHIFT><CTRL>U	Convert selected text to upper/lower case
<INSERT>	Switch between Insert Mode and Overwrite Mode
<CTRL>Backspace <CTRL>Del	Remove word left/right
<ALT>Mouse, <ALT><SHIFT>Up, Down, Right, Left	Enable column selection mode
<ALT>Up <ALT>Down	Move selected lines up/down


<SHIFT>Del	Remove current line
<ESC>	Close all opened tooltips, menus and hints
<CTRL>Wheel	Zoom
<CTRL>M <CTRL>E	Start/stop macro recording, execution
<ALT>F [char]	Find nearest [char]
<CTRL>Up <CTRL>Down	Scroll Up/Down
<CTRL>NumpadPlus, <CTRL>NumpadMinus <CTRL>0	Zoom in, zoom out, no zoom

## Opening the Replication Script Editor

On the [Replication Properties dialog](#) General tab, check the Use Script option. Click **Script** to open the Replication Script Editor.

## Restore Metadata Options Dialog

This dialog provides options for restoring metadata information that has previously been saved (see [Backing Up and Restoring Metadata](#).) The menu option to restore metadata information is available only when an existing metadata is selected in the Metadata Explorer.

The backup metadata information will be restored to the currently selected metadata. No new metadata is created in the DBMoto Mangement Center. If you wish to restore metadata information to a completely new, or empty, metadata, first create the metadata using the  **New Metadata** icon in the toolbar, then make sure that it is selected in the Metadata Explorer.

### Standard Restore

Choose this option to restore a complete metadata file, overwriting all information in your currently selected metadata.

### Custom Restore

Choose this option to open the [Restore Metadata wizard](#) and select information from a metadata backup file and add that information to your currently selected metadata.

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.



## Opening the Restore Metadata Options Dialog

In the DBMoto Management Center Metadata Explorer, select a metadata, and from the right mouse button menu, choose **Restore Metadata**.

## Scheduler Dialog

This dialog displays different information depending on whether you have defined a Refresh, Mirroring or Synchronization replication.

### Scheduler for Refresh

#### Recurrence

First select how often you want the refresh to occur. Depending on your choice, the right side of the dialog updates to show the different options associated with your choice. For example, if you choose to refresh Weekly, the right side displays the days of the week so that you can choose which day you want to run the refresh.

#### Schedule Times

Based on your choices in the Recurrence area, this area updates to display checkboxes allowing you to select times for running the refresh operation. Note that you can select multiple times. For example, if you have chosen to run the refresh weekly, you could then set two times so that the refresh runs at 6 a.m. and at 12 midnight once per week on Mondays.

#### Opening the Scheduler Dialog

This dialog is available from the [Replication wizard](#), or from the Schedule tab in the [Replication Properties dialog](#).


## Select Tables Dialog


This dialog allows you to specify the source or target tables, views and aliases to display in the DBMoto Management Center for use in defining replications. You can either [select tables interactively](#) or [load tables from information in a previously defined file](#). Note that when defining a replication, you can specify only one source table/view/alias and one target table for each replication.



### Selecting Tables Interactively

Choose one or more objects from the tree to associate with the source connection. Objects include tables, views and aliases. When creating a replication, you will be able to select an object for replication from those that you have chosen in this wizard. If you create multiple replications, you can select an object for each replication that you are defining.

#### Filtering the Table View



Use the filter field to limit the number of tables to view at any time. The field expects standard SQL syntax. Either type in the full name of the object you want to retrieve, or use '%' as search pattern. For example, if you type AB%, all objects beginning with the letters 'AB...' are displayed. To apply the filter, type the filter in the textbox and click . If filtering is not available for a specific node, or connection, the filter icon is inactive.

- You can apply a filter on any node in the tree, as long as you provide the correct SQL syntax.
- If you apply incorrect SQL syntax, the error dialog reports a SQL error. You should be familiar with the SQL syntax for your database to analyze and fix the error.
- DBMoto stores the filter so that if you select another node, then go back to the previous one, the filter is retrieved.
- When a node in the tree has been expanded with a filter, its icon is modified to show that the full contents of that node have been filtered: 
- Note that some databases, such as IBM Db2 for i, may not support the % symbol on one or more levels in the tree (catalog, schema, etc.).
- If you expand a node applying one filter (for example, A%) and then select/remove some tables, then change the filter to B% and select/remove other tables, when clicking OK the dialog will remember all the chosen tables and will apply all filters.


To remove a filter, either click , or delete the filter text and click . The content for the node is reloaded without filtering information.

Use the **Hide System Tables** checkbox to limit the number of tables displayed.

Use the **Select All Tables** and **Deselect All Tables** buttons to work with multiple tables.

	Select a database owner/schema, then click this button to check all tables under the owner/schema.
	Select a database owner/schema, then click this button to uncheck all tables under the owner/schema.

### Loading Table Selections from a File

If you have large numbers of tables in your environment, this option allows you to build up a text file containing a list of tables and load the information into DBMoto. To select tables from an external file instead of connecting to the database to scroll the catalog and schema objects, in the toolbar, click  **Load File List**.

The file used to store your table selections should consist of comma-separated values as follows:

CATALOG,SCHEMA,TABLE,[KEY1,KEY2,...]

- CATALOG is the catalog name, optionally enclosed by double quotes. Use an empty string if catalog is not supported for this connection type.

- SCHEMA is the schema name, optionally enclosed by double quotes. Use an empty string if schema is not supported for this connection type.
- TABLE is the table name, optionally enclosed by double quotes.
- [KEY1,KEY2,...] is a list of primary keys, each of them optionally enclosed by double quotes. This parameter is used to set a list of primary keys programmatically.

See below for an example CSV file.

```

"" , "" , "ADDRESS"
"" , "" , "ADDRESS1" , ["STREET" , "STATE" , "ZIP"]
"" , "" , "ADDRESS2" , ["APT"]
"" , "" , "ADDRESS3"
"" , "" , "T1"
"" , "" , "T2"
"" , "" , "T3"
"" , "" , "T4"

```

Note the following constraints.

- If the table structure does not follow the database catalog/schema structure (e.g., a file containing a table with catalog and schema information for a Microsoft Access database), the table will be ignored.
- If the table does not exist in the database, an error occurs and the operation is aborted.
- If any of the primary key fields in the key list do not exist in the table structure, an error occurs and the operation is aborted

### Opening the Select Tables Dialog

In the DBMoto Management Center Metadata Explorer:

- Select a source connection, then choose **Select Tables** from the right mouse button menu.
- Select a target connection, then choose **Select Tables** from the right mouse button menu.

### Server Alert Properties Dialog

This dialog allows you to set up email alerts for specific events that occur at the server level. Use the Alert Settings tab to choose which events to include in an email, and use the Message Layout tab to specify email recipients and email content. In addition to defining the email content in this dialog, you must set email server properties in the [Mail tab](#) of the [Data Replicator Options dialog](#).

As an alternative, or in addition to email alerts at the server level, you can [set up email alerts for a specific replication](#) using the [Alerts tab](#) in the [Replication Properties dialog](#).

Note: It is also possible to create a custom email alert by writing a script that uses the [SendMail function](#).

## Event Alerts

To define email alert messages for specific Data Replicator or replication events, on the [Data Replicator Options dialog Alerts tab](#), click **Add Alert**, then **Add Event Alert** to display the **Server Alert Properties - Event Alert** screen. In the **Alert Settings** tab, select the event or events for which you want to receive an email message, then use the **Message Layout** tab to define the email format. Finally, set up the email server details in the [Data Replicator Options Mail tab](#).

## Filtering Alerts

Where the alert type allows, it is possible to set a filter to identify specific conditions for an alert. The filter field is displayed to the right of the alert name and can contain logical operators in addition to specific strings. The supported logical operators are AND, OR, NOT (uppercase with precedence as shown). Parentheses can be used to change operator precedence. Substrings to be matched with message content should be enclosed in quotes (') or double quotes(").

Examples:

- All messages for enabled replications except "EMP50" and "EMP51":

"enabled" AND NOT ("EMP50" OR "EMP51")

- All messages not containing the substring "warning 256":

NOT "warning 256"

- All messages containing "error" or "warning":

"error" OR "warning"s

## Alert Settings Tab

Server Events	
<b>Start Data Replicator</b>	Reports when the Data Replicator starts.
<b>Stop Data Replicator</b>	Reports when the Data Replicator stops.
<b>Reached Max Quota for Traces</b>	Sets a maximum trace file size after which you want to be notified. For example, if the trace file size is set to 100 MB, an email would be sent once the 100 MB limit is reached.
<b>Reached Max Data Replicator Downtime</b>	Sets a maximum number of hours that the Data Replicator could be stopped before sending an email notification. For example, if the number of hours is set at 3, an email would be send once the down time reaches 3 hours.

<b>Data Replicator Info</b>	Reports all comments generated by DBMoto in the Data Replicator "Information" category. The filter option allows you to set a string to identify a subset of "Information" messages you want to receive.
<b>Data Replicator Error</b>	Reports all comments generated by DBMoto in the Data Replicator "Error" category. The filter option allows you to set a string to identify a subset of "Error" messages you want to receive.
<b>Data Replicator Warning</b>	Reports all comments generated by DBMoto in the Data Replicator "Warning" category. The filter option allows you to set a string to identify a subset of "Warning" messages you want to receive.
<b>Replication Events</b>	
<b>Replication Info</b>	Reports all comments generated by DBMoto in the "Information" category for each replication. The filter option allows you to set a string to identify a subset of "Information" messages you want to receive.
<b>Replication Errors</b>	Reports all comments generated by DBMoto in the "Error" category for each replication. The filter option allows you to set a string to identify a subset of "Error" messages you want to receive.
<b>Replication Warnings</b>	Reports all comments generated by DBMoto in the "Warning" category for each replication. The filter option allows you to set a string to identify a subset of "Warning" messages you want to receive.
<b>Reached Max Replication Latency</b>	the <b>Transaction Latency Threshold</b> setting in the General tab of the <a href="#">Data Replicator Options dialog</a> has been exceeded by comparing the current time with the latest processed transaction time (or the last scheduled mirroring time if there are no transactions). If the lag between the two exceeds the <b>Transaction Latency Threshold</b> (set in the General tab of the <a href="#">Data Replicator Options dialog</a> ), an email message is generated. This can indicate that the current settings for the Data Replicator and the replication are not allowing the Data Replicator to keep up with the number of transactions.
<b>Reached Max Replication Downtime</b>	Sets a maximum number of hours that any replication could be stopped before sending an email notification. For example, if the number of hours is set at 3, an email would be send once the down time for any replication reaches 3 hours.

## Alert Schedule

Use this option to limit the number of email messages you receive for a defined set of alerts.

## Message Layout Tab

Use this tab to define the alert email recipients and the overall layout of the email message

### Recipients

A semi-colon separated list of email addresses

### Title

A default email subject line is provided, but you can edit the text to provide your own subject line.

### Body

A default email body format is provided, but you can edit the text to customize the email format.

### Html Format

Check this option to send email messages in HTML format.

### Attachment

To attach the current DBMoto text file log to the email, select DBMoto Log from the drop-down list.

## Activity Report

Use the **Add Activity Report** option to create general email message alerts for Data Replicator activity in the [Server Alert Properties dialog](#). On the [Data Replicator Options dialog Alerts tab](#), click **Add Alert**, then **Add Activity Report** to display the **Server Alert Properties - Activity Report** screen. In the **Alert Settings** tab, choose how often you want to receive an email message, then use the **Message Layout** tab to define the email format. Finally, set up the email server details in the [Data Replicator Options Mail tab](#).

### Opening the Server Alert Properties Dialog

In the Management Center **Metadata Explorer**, select a server (usually **local**). From the right mouse button menu, choose **Data Replicator Options**. In the **Data Replicator Options dialog**, go to the Alerts tab. Click **Add Alert** then either **Add Event Alert** or **Add Activity Report** to open the **Server Alert Properties** dialog.

## Server Connection Properties Dialog

This dialog provides a way to view settings for a server. Note that properties cannot be edited from this dialog. To change the properties of a server, you would first need to remove the server in the Metadata Explorer, then add a new server with different connection properties.

## Authentication

### Server Name

The name of the system where the DBMoto Server Agent is running.

### Server Address

The IP address of the system where the DBMoto Server Agent is running.

### Port

The port used by the DBMoto Server Agent. The default value is 58361.

### Authentication Method

The method established by an administrator to authenticate DBMoto users. The options are Anonymous Access (no access control), Windows Authentication and DBMoto Authentication.

### User Name

The name of the user connected to the server.

### User Role

The role associated with the user name. The role represents the set of permissions enabled for the user. For more information, review the user roles and permissions described in the documentation for [Create DBMoto Login Dialog](#).

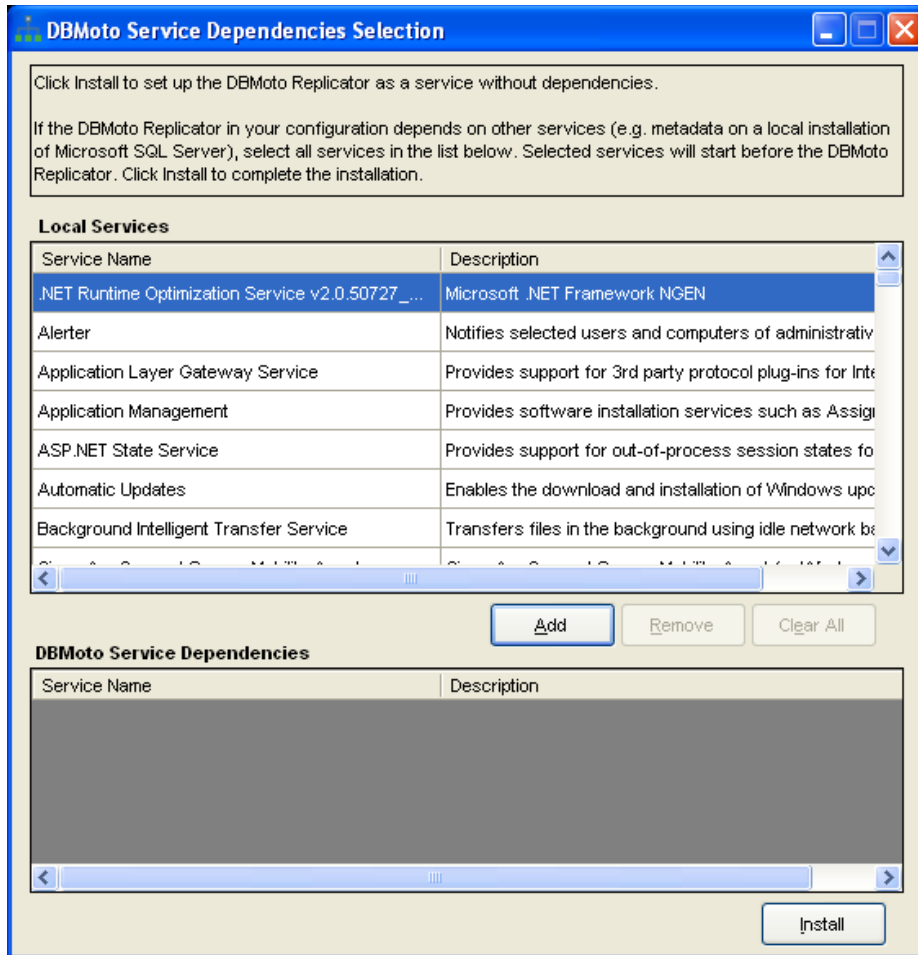
### Opening the Server Connection Properties Dialog

In the Management Center Metadata Explorer, select the server for which you want to view the properties. From the right mouse button menu, choose **Server Properties**.

## Service Dependencies Window

This window allows you to specify services needed by DBMoto prior to the DBMoto Data Replicator Service startup. If you are not running a metadata database locally (on the same system where DBMoto is installed), you do not need to set dependencies. Click **Done** to close the window.

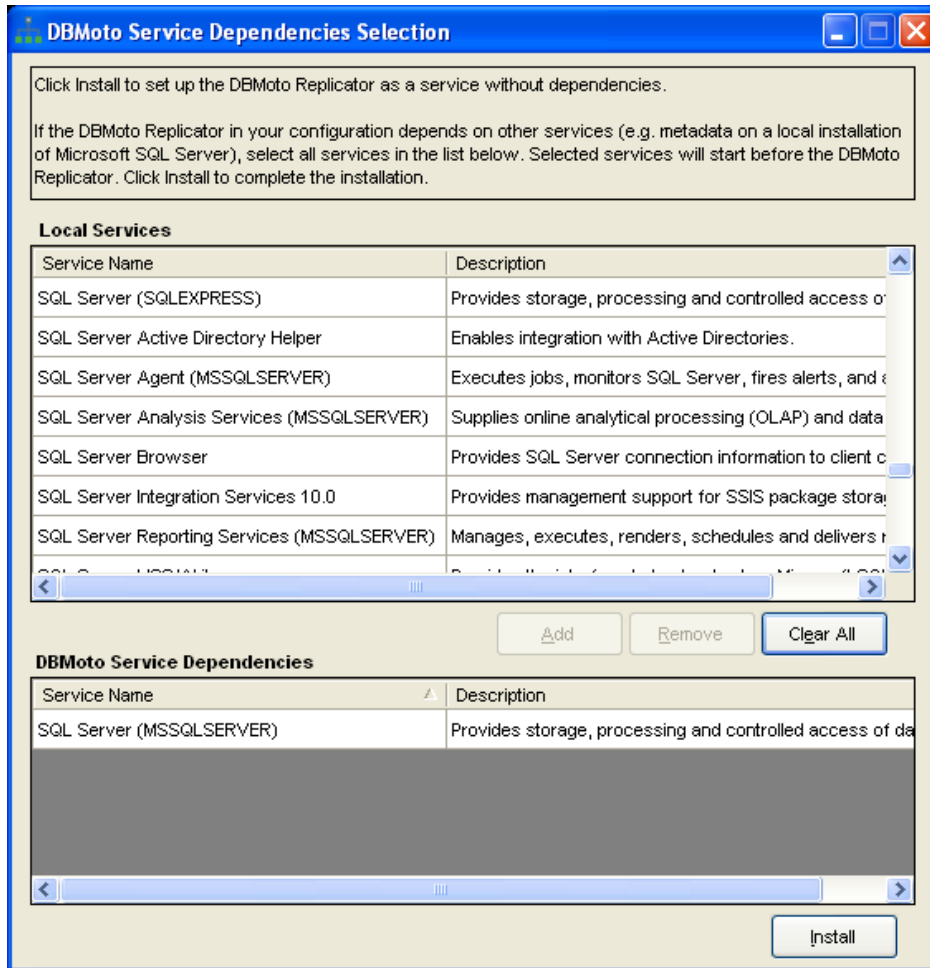
If you have a local database (installed on the same system as DBMoto) that is storing your DBMoto metadata, the database must be running before DBMoto to avoid errors when DBMoto checks the metadata.



- To add a dependency, select the service in the list, then click **Add**.

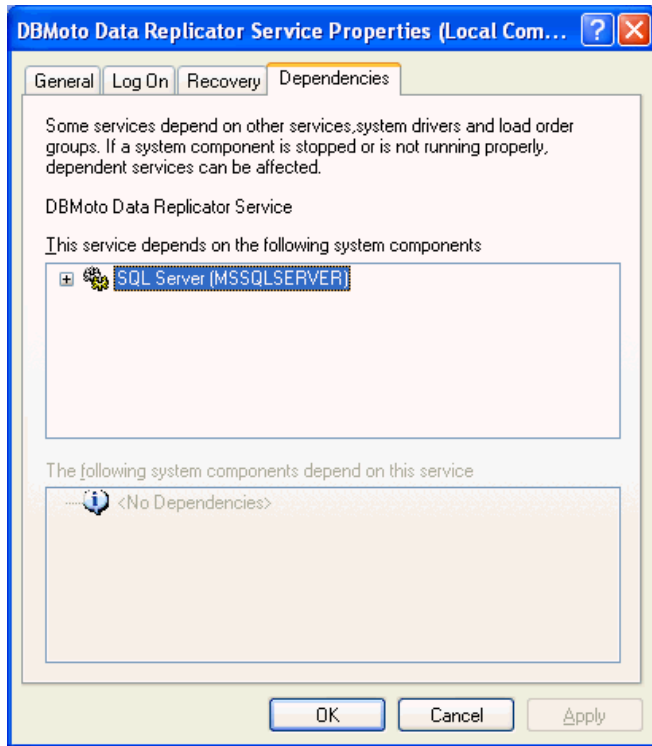
The DBMoto Data Replicator Service will not start until the selected service has started. For example, if you select SQL Server service and click **Add**, SQL Server is added as a dependency: the DBMoto service does not start until the SQL Server service starts.






- Select a dependency and click **Remove** if you no longer need that service to start first.
- To remove all dependencies, click **Clear All**.
- Click **Done** when you have finished editing dependencies, or if you have no dependencies

Dependencies are added to the Windows Service Properties for the DBMoto Data Replicator Service. Access the Windows Service Properties by opening **Control Panel > Administrative Tools > Services**, selecting the service, then right-clicking to open the properties dialog.




## Opening the Service Dependencies Window

The Services Dependencies window opens automatically following installation when you select to install the DBMoto Replicator as a service during the setup. You can also run the Service Installer application from the DBMoto Service Monitor in the Windows Notification Area. Move the mouse over the Service Monitor icon , then, from the right mouse button menu, choose **Launch Service Installer**.


## Service Installer Window

Use this window to install or uninstall the DBMoto Data Replicator service. Click **Install Service** to open the [Service Dependencies window](#) and install the DBMoto Data Replicator service.

If the service is already installed, and you want to remove it:

1. [Stop the DBMoto Data Replicator service](#).
2. Run the Service Installer application from the from the Service Monitor icon  in the Windows Notification Area. From the right mouse button menu, choose **Launch Service Installer**.
3. Click the active **Uninstall Service** button to remove the service from the Windows Services.

## Opening the Service Installer Window

Run the Service Installer application from the DBMoto Service Monitor in the Windows Notification Area. Move the mouse over the Service Monitor icon , then, from the right mouse button menu, choose **Launch Service Installer**.

## Show Replication Pairs Dialog

This dialog allows you to review the source and target database pairs defined in the selected metadata.

1. In the **Select Metadata** field, choose a metadata set from the drop-down list. Note that only metadata which have been successfully loaded into DBMoto will be available in the list.
2. Click **Run**.

The resulting grid shows one row per source-target pair with at least one replication defined between the source and target server.

- If one or more replications uses a script, the **Use Script** field is checked.
- If one or more replications is in synchronization mode, the **Use Synch** option is checked.
- If source or target connections exist in the Management Center without any associated replication, they are displayed in the list, using the appropriate source or target columns to describe the connection.

Click **Copy to Clipboard** to copy the list so that you can then paste it into an email or other document.





Click **Cancel** to close the dialog.










## Opening the Show Replication Pairs Dialog

1. In the DBMoto Management Center Metadata Explorer, choose the server for which you want to manage licenses.
2. From the right mouse button menu, choose **Manage Licenses**.
3. In the [License Information dialog](#), click **Show Pairs**.


## SQL Query Tab

This tab can be used to type SQL queries for a specific table and retrieve results. It contains two panes, the [Query pane](#) and the [Results pane](#). The toolbar and Query menu offer the following options.

Icon	Menu	Description
	Create New Query	Clears the Query pane in preparation for typing a new query.
	Open Query File	Displays an Open File dialog to open a file containing a query.
	Save Query File	Active only after you have saved a query file using  <b>Save Query File As</b> . Saves

		the current contents of the Query pane to the file.
	Save Query File As	Displays a Save File dialog so that you can save the query displayed in the query pane to a file.
	Change Login	Opens the <b>Change Login</b> dialog to allow you to set a different login for use in the SQL Query dialog. This is particularly useful when running synchronization replications where you should keep the login used for DBMoto operations (like reading logs) separate from the login used to perform transactions. See the note below for more information.
	Clear Query Panel	Clears the Query panel.
	Word Wrap	When selected, alters the display of a SQL command so that text which extends beyond the visible display area is automatically split and displayed across two or more lines.
	Execute Query	Executes the query you have typed in the Query pane.
	Stop Execution	Stops execution of the query in the Query pane.
	Get Next Rows	Based on the value of Returned Rows in the Query Settings dialog, retrieves the next set of rows.
	Export Results	Saves query results to a text file.
	Settings	Opens the Query Settings dialog. This dialog allows you to set how many rows to retrieve each time you choose Get Next Rows, and to set the maximum number of characters per column in the result set.

## Query Pane

This is an editable text area that either displays a query that has been generated for you, or allows you to type a query for execution. Once the query has been entered, use the toolbar icon  to execute the query. Query results are displayed in

the Results pane. If the query cannot be executed successfully, check the Messages tab in the Results pane for information.

**Note:** If you are planning a synchronization replication, be sure to change your user ID before performing insert, update or delete operations on either database involved in the synchronization. If you continue using the same user ID, the SQL operations will be executed using this ID which is also 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.

### Editing and Formatting Features

1. Automatic indentation based on syntax
2. Colors to identify specific code elements: syntax in blue, numeric values in pink and comments in green
3. Code folding where sections of code can be collapsed or expanded for ease of reading. The editor displays a green bar on the left side with +/- signs. Click - to collapse a section of code, and click + to expand a section of code.
4. Common editing operations using keystroke combinations:


<SHIFT>Left, Right, Up, Down, Home, End, PageUp, PageDown	Move caret with selection
<CTRL>F <CTRL>H	Show Find and Replace dialogs
F3	Find next
<CTRL>G	Show GoTo dialog
<CTRL>C <CTRL>V <CTRL>X	Copy, paste, cut
<CTRL>A	Select all text
<CTRL>Z, <ALT>Backspace, <CTRL>R	Undo/Redo
Tab <SHIFT>Tab	Increase/decrease left indent of selected range
<CTRL>Home <CTRL>End	Go to first/last char of the text

<SHIFT><CTRL>Home, <SHIFT><CTRL>End	Go to first/last char of the text with selection
<CTRL>- <SHIFT><CTRL>-	Navigate backward/forward
<CTRL>U <SHIFT><CTRL>U	Convert selected text to upper/lower case
<CTRL><SHIFT>C	Insert/remove comment prefix in selected lines
<INSERT>	Switch between Insert Mode and Overwrite Mode
<CTRL>Backspace <CTRL>Del	Remove word left/right
<ALT>Mouse, <ALT><SHIFT>Up, Down, Right, Left	Enable column selection mode
<ALT>Up <ALT>Down	Move selected lines up/down
<SHIFT>Del	Remove current line
<CTRL>B, <CTRL><SHIFT>B, <CTRL>N, <CTRL><SHIFT>N	Add, remove and navigate to bookmark
<ESC>	Close all opened tooltips, menus and hints
<CTRL>Wheel	Zoom
<CTRL>M <CTRL>E	Start/stop macro recording, execution
<ALT>F [char]	Find nearest [char]
<CTRL>Up <CTRL>Down	Scroll Up/Down

<CTRL>NumpadPlus, <CTRL>NumpadMinus <CTRL>0	Zoom in, zoom out, no zoom
---	----------------------------

## Results Pane

The Results pane consists of two tabs:

- **Grid** displays data retrieved. The number of rows retrieved at any time is set using the **Settings** option on the **Query** menu. You can retrieve additional rows by clicking  in the toolbar.
- **Messages** displays information, including error messages from the database, if the query could not be executed.

### Opening the SQL Query Tab

In the DBMoto Management Center Metadata Explorer:

- Select a source connection, then from the right mouse button menu, choose **Execute SQL Query**.
- Select a target connection, then from right mouse button menu, choose **Execute SQL Query**.

A SQL Query dialog is also available from the Replication wizard Select Source and Select Table screens by clicking **Open Table**.

## Table Properties Dialog


### General Tab

#### General Settings

Displays information about the table gathered from input in the [Source Connection Wizard](#) or [Target Connection Wizard](#). The available information depends on the database type. Listed below are properties for IBM Db2 for i (iSeries/AS400), SQL Server and Oracle.

#### IBM Db2 for i (iSeries/AS400)

Property	Read/Write	Description
<b>Connection Name</b>	R	The name of the connection that will be used to access this table. The connection name is defined when creating a connection.
<b>Library</b>	R	The name of the library selected when creating the connection.
<b>Table Name</b>	R	The currently selected table.

<b>System Name</b>	R	If an alias is used, original table name is provided. If the selected table has a long table name, both the generated internal name and user-defined name for the table are displayed.
<b>Description</b>	R	A value for this field is displayed if it can be retrieved from the database. This depends on the driver/provider that you are using to access the database.
<b>Type</b>	R	One of Table, Alias, View or System Table. Describes the type of database object that you have selected to use for your source or target connection.
<b>Character CCSID</b>	R/W	The language that is currently set for the table. You can edit this field to set a different CCSID for the table, or you can click  to open the Field CCSID Settings dialog. In this dialog, you can set the CCSID for each column in the table by clicking in the CCSID column and modifying the value. You will need to know the exact CCSID value for the language that you are using. This information should be available from your Db2 administrator or from the Db2 documentation.

## SQL Server

Property	Read/Write	Description
<b>Connection Name</b>	R	The name of the connection that will be used to access this table. The connection name is defined when creating a connection.
<b>Database</b>	R	The name of the library selected when creating the connection.
<b>Owner</b>	R	The owner of the currently selected table.
<b>Table Name</b>	R	The currently selected table.
<b>Type</b>	R	One of Table, Alias, View or System Table. Describes the type of database object that you have selected to use for your source or target connection.
<b>System Name</b>	R	If an alias is used, original table name is provided.



<b>Description</b>	R	A value for this field is displayed if it can be retrieved from the database. This depends on the driver/provider that you are using to access the database.
--------------------	---	--

## Oracle

Property	Read/Write	Description
<b>Connection Name</b>	R	The name of the connection that will be used to access this table. The connection name is defined when creating a connection.
<b>Database</b>	R	The name of the library selected when creating the connection.
<b>Owner</b>	R	The owner of the currently selected table.
<b>Table Name</b>	R	The currently selected table.
<b>Type</b>	R	One of Table, Alias, View or System Table. Describes the type of database object that you have selected to use for your source or target connection.
<b>System Name</b>		If an alias is used, original table name is provided.
<b>Description</b>	R/W	A value for this field is displayed if it can be retrieved from the database. This depends on the driver/provider that you are using to access the database.

## Replication Settings

Displays the following information gathered from input in the [Replication wizard](#) when the replication type is mirroring or synchronization.

- For IBM Db2 for i/iSeries/AS400 source tables: Journal Library and Journal Name.
- For Oracle: Service Name and Dictionary File.
- For SQL Server: Publication Name and Article ID.

## Fields Tab

Displays a grid that shows the table column details.

## Table Script Dialog

This dialog allows you to view and copy a SQL, JSON or XML script to represent the selected table.

When replicating to JSON format text files, this can be useful to set up the file structure for mapping between source table fields and JSON. See [Replicating Data to a JSON Format File](#) for more details.

### Opening the Table Script Dialog

1. In the DBMoto Management Center Metadata Explorer, select a table.
2. From the right mouse button menu, choose **Table Script**, *then choose SQL, JSON or XML*.

## User Settings Dialog

Use this dialog to control user access to DBMoto. You can select a security option, add and remove users, and set specific permissions or user roles for each user.

**NOTE:** If you set up a user ID with DBMoto Authentication, or Certificate Authentication, you will not be able to use DBMoto until the certificate is correctly installed. [Generate and install the certificate](#) before setting up authentication.

### Authentication

Select an option from the drop-down list. For more information on security options in DBMoto, see [DBMoto Server Client Security Options](#).

Anonymous Access	Leave this option if you do not want to restrict access to DBMoto in any way.
DBMoto Authentication	Allows you to establish login IDs and passwords for users, but also requires the <a href="#">installation of an X.509 certificate</a> on the client system (where the DBMoto Management Center is running).
Windows Authentication	DBMoto uses your Microsoft Windows login and ID to control access to the DBMoto Management Center and server agent.
Certificate Authentication	Requires the <a href="#">installation of an X.509 certificate</a> but does not allow specific user login IDs.

### Binding Protocol

Select an option from the drop down list. For more information on appropriate binding protocols for use with DBMoto, see [DBMoto Server Client Security Options](#).

Authentication	Binding Protocol Options
Anonymous Access	Inactive. Uses TCP/IP binding protocol
DBMoto Authentication	TCP/IP or HTTP binding protocol. TCP/IP is suitable for intranet access, HTTP is recommended for internet access.
Windows Authentication	Inactive. Uses TCP/IP binding protocol
Certificate Authentication	TCP/IP or HTTP binding protocol. TCP/IP is suitable for intranet access, HTTP is recommended for internet access.

**Add**

Enabled only when Windows Authentication or DBMoto Authentication have been selected. Click **Add** to add a new DBMoto user. This opens the [Create DBMoto Login dialog](#). From this dialog, you can add users and set passwords and permissions.


**Remove**

Select an existing user in the table, and click **Remove** to remove the user.

**Edit**

Select an existing user in the table and click **Edit** to open the [DBMoto Login Dialog](#). From this dialog, you can modify the password settings and permissions for the user (as long as you have appropriate authority.)

### Opening the User Settings Dialog

1. In the Metadata Explorer, select the server for which you want set up or modify user access.
2. In the toolbar, click  **Manage Users**, or choose **Manage Users** from the right mouse button menu.

### Validate Replications Dialog

The Validate Replications dialog can be used to:

- Check that a replication has the transactional objects required in the source database for the replication to run, generate a SQL script to create the objects as needed, or create missing objects in the database.
- Check that the actual source database schema match the schema saved in the DBMoto Metadata. This is useful if schema changes are suspected and may cause replications to fail. The schema saved in the DBMoto Metadata can be updated using the Rebuild option.

Click in the checkbox to the left of each replication you want to validate. All checked replications are validated for the above two purposes.

## Validate Replications for Transactional Objects

This option allows you to check that a replication has all the necessary transactional objects set up and running correctly in the source database. For example, if triggers for trigger-based replications, or articles and publications for replications from Microsoft SQL Server are missing or modified, or you set the EnableTransactionalObjects property to False to create transactional objects directly on the database (instead of from DBMoto), you can validate the replication using the right mouse button on the replication name in the Metadata Explorer.

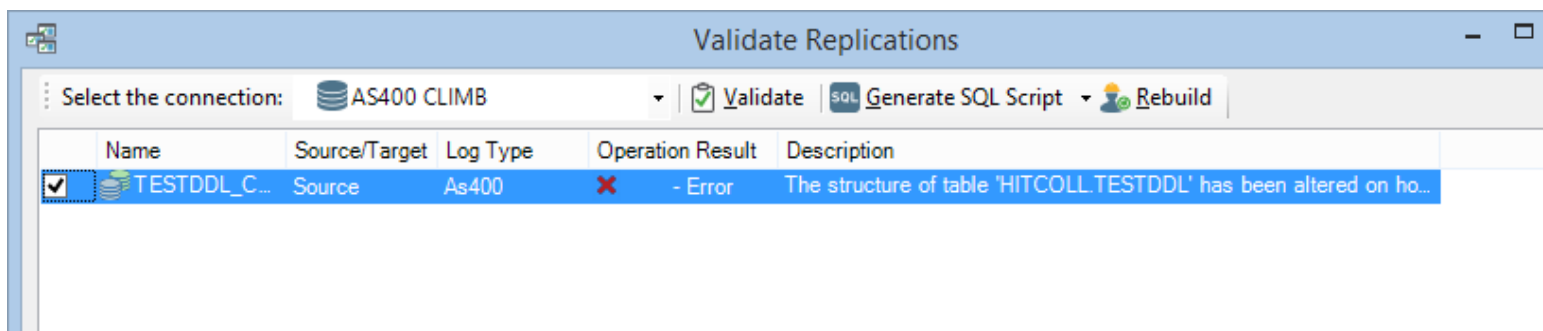
No transactional objects are created using this option, but if any of the transactional objects exist, and the replication does not have the association established in the metadata (e.g., the log trigger tag for the replication is blank even if a trigger and log file exist), then the correct association is made and saved in the metadata.

## Generate SQL Script

The Validate Replications dialog allows you to generate SQL statements to create or remove the transactional objects needed for a replication. This option is helpful when you need to modify the database directly (perhaps via your database administrator) instead of from DBMoto.

## Validate Replications for Schema Changes

This feature works in conjunction with the **Stop on Schema Change** property in the [Replication Properties dialog](#) for mirroring and synchronization. Click **Validate** to check if the schema stored in the DBMoto Metadata for both the source and target tables match the current table schema. If the schemas do not match, an error is displayed in the Validate Replications dialog:



If the **Stop on Schema Change** property in the [Replication Properties dialog](#) is set to True, in the Replication Monitor, the ValidationResult field of the replication is set to Fail. The validation fail status prevents the Data Replicator from enabling or executing the replication.

To repair the replication status, click Rebuild. This updates the table schema after prompting whether to proceed and sets the replication validation status to Success.

## Rebuild

This option allows you to:

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.

- Build transactional objects in the database if they do not exist or to re-create them if they are invalid. This option requires the [Connection property](#) **EnableTransactionalObjects** to be set to True.
- Update the source table schema saved in the DBMoto Metadata to match the actual source table schema.

The Rebuild confirmation message provides information on exactly which operations will be performed, and give details in the case of any schema changes. Rebuild **does not** re-enable a replication.

**NOTE:** Following schema updates, it is advisable to check the field mapping between source and target tables using the [Fields Mapping dialog](#) available in the [Replication Properties](#).

### Opening the Validate Replications Dialog


1. In the DBMoto Management Center Metadata Explorer, select a replication.
2. From the right mouse button menu, choose **Validate Replication**.

# Chapter 9: Running Replications

## Running a Replication

The Enable Replication option must be checked, either when creating the replication via the [Replication wizard](#), or through the [Replication Browser](#), by selecting the replication then, from the right mouse button menu, choosing **Enable Replication**.

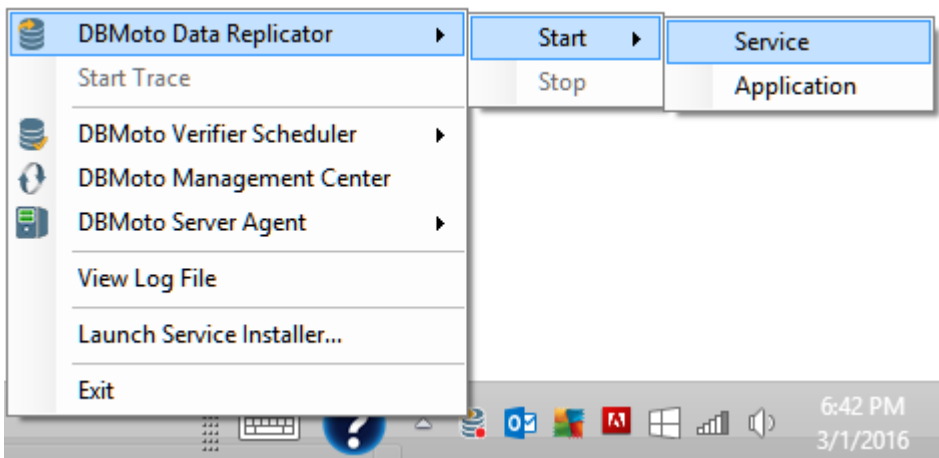
You can run a replication in the following ways:



- Interactively from the Windows Start menu.  
Choose **Start**, then **Programs**, then **HiT Software DBMoto**, then **DBMoto Service Monitor**.  
In the Windows Tool Tray, click the right mouse button over the Data Replicator icon   
Choose **DBMoto Data Replicator**, then **Start**, then either **Service** or **Application**, depending on how you installed the Data Replicator.
- Interactively, on your Windows desktop by running the program:  
C:\Program Files\HiT Software\dbmoto\DBReplicator.exe.
- [Programmatically from your .NET environment](#).
- By [setting up the DBReplicator as a service](#).

## Running the DBMoto Data Replicator as a Service

### Installing the Data Replicator Service During Setup

During DBMoto installation, you can specify that you want to install the DBMoto Data Replicator as a service. If you check this option, the service is installed at the end of the setup, but, because it requires a valid metadata connection to run, the service must be started manually. The DBMoto Service Monitor tool is added to the Windows **Start > Programs > Startup** menu, and is available from the Windows Notification Area:




	DBMoto Data Replicator started
	DBMoto Data Replicator stopped

Once you have created a metadata connection and metadata tables, start the service from the DBMoto Service Monitor icon in the Windows Notification Area or from **Control Panel > Administrative Tools > Services**. Note that if you leave the service on the manual setting, you will need to restart the service each time you reboot your system. You can set the service to start automatically via its properties in the Services window. When the service runs automatically, all scheduled replications will resume on system startup (after failure or reboot, for example.)

The DBMServiceMonitor executable is available in your DBMoto install folder. Make sure that you install the DBReplicator as a service before attempting to run the tool.

## Installing the DBMoto Data Replicator Service After Setup

If you did not install the DBMoto Data Replicator as a service during setup, you can install it as follows:

- From the Windows Desktop **Start** menu, choose **Programs** then **HiT Software DBMoto** then **Service Installer**.
- In the DBMoto Service Installation Utility window, click **Install Service**.
- In the DBMoto Service Dependencies Selection dialog, click **Done** if DBMoto is not dependent on any local databases containing DBMoto metadata.  
If you have metadata in a local database, the database must start before DBMoto begins replicating (otherwise an error may be generated). You should therefore add the local database service as a dependency. Select the service in the list of local services, then click **Add**. Click **Done** when you have finished managing dependencies.
- Manage the service from the Windows Control Panel. Choose **Administrative Tools**, then **Services**. You can also manage the service via the Data Replicator application available from the Windows Start menu. If you run this application, it adds an icon to the Windows Notification Area to start and stop the DBMoto Data Replicator service.  

- Use the [Replication Monitor](#) tab in the Management Center to track the progress of replications.

## Manually Uninstalling the DBMoto Data Replicator Service

To remove the DBMoto Data Replicator as a service:

- From the Windows Desktop **Start** menu, **Programs** then **HiT Software DBMoto** then **Service Installer**.
- In the DBMoto Service Installation Utility window, click **Uninstall Service**.
- Click **Exit**.

## Running DBMoto Data Replicator Interactively

To run the DBMoto replicator interactively:

- From the Windows Desktop **Start** menu, choose **Programs** then **HiT Software DBMoto** then **DBMoto Service Monitor**.
- In the Windows Notification Area, click the DBMoto Service Monitor icon and choose **DBMoto Data Replicator** then **Start**, then **Application** from the right mouse button menu.
- The replication that you have scheduled should start at the specified time.
- Use the [Replication Monitor](#) tab in the DBMoto Management Center to track the progress of the replication.

Note that you cannot run the DBMoto Data Replicator interactively if it is installed and running as a service.

## Disabling and Stopping Replications

### Disabling a Replication


To prevent a particular replication from running, in the DBMoto Management Center:

1. Display the Replication Browser tab.
2. Select the replication you want to modify.
3. On the right mouse button menu, uncheck the **Enable Replication** option.

If the replication is part of a group, all replications in the group are disabled.

### Stopping the Data Replicator

To stop all replications from running, on your Windows desktop:

1. Select the DBMoto Service Monitor icon  in the Windows Notification Area.
2. From the right mouse button menu, choose **DBMoto Data Replicator** then **Stop**.

Alternatively, you can manage the DBMoto Data Replicator service from the Windows Services dialog. On your Windows desktop, choose **Control Panel > Administrative Tools > Services**.

### Consequences of Disabling or Stopping Replication

If a Refresh replication is interrupted by disabling the replication or stopping the Data Replicator, it will restart from the beginning once the replication is enabled again.

If a transactional replication is interrupted by disabling the replication or stopping the Data Replicator, it will restart from the point at which it was interrupted once the replication is enabled again. The transaction ID for the last transaction executed is stored in the metadata, and retrieved once the replication resumes



## Chapter 10: Managing Replications Programmatically

**Note:** The API described here is available in DBMoto version 7.1 and above. If you have an application that contains code for running DBMoto replications in earlier releases, you should plan to modify your application to use the new API.

The code below provides a simple example of how to start the Data Replicator for a replication in the default metadata running on the "local" DBMoto server.

The .NET namespace for the DBMoto API is HitSoftware.DBMoto.Application. This namespace includes the class DBMotoApplication, the main singleton object of the DBOM, which can be instantiated through the static singleton method Instance:

```
using HitSoftware.DBMoto.ObjectModel; \\APIs
using HitSoftware.DBMoto.Common; \\ Type Definition for the APIs
...
public void Run(){
    try
    {
        // Static singleton constructor
        dbmApp = DBMotoApplication.Instance;
        // Retrieving the local server from the list available in the
        DBMotoApplication instance
        dbmServer = dbmApp.Servers["local"];
        // Connecting to the server agent using anonymous authentication
        dbmServer.Connect();
        // Define and load the current metadata
        currentMetadata = dbmServer.Metadata.DefaultMetadata;
        // Set synchronization between metadata and Data Replicator (TCP/IP)
        currentMetadata.IsSynchronized = true;
        currentMetadata.Load();
        if (currentMetadata.IsRunning())
        {
            Console.WriteLine("Data replicator is running on " +
currentMetadata.Name);
            Console.WriteLine("Stopping now ...");
            // Stopping the Data Replicator after a timeout of 10000 milliseconds
            currentMetadata.Server.StopReplicationManager(10000);
        }
        else
        {
            Console.WriteLine("Data replicator is stopped");
            Console.WriteLine("Starting now (as an application) ...");
            // Starting the Data Replicator as an application
            currentMetadata.Server.StartReplicationManager(false);
        }
    }
    catch(Exception e)
    {
        if (currentMetadata != null)
            currentMetadata.Unload();
        if (dbmServer != null)
```

```
        dbmServer.Disconnect();  
        Console.WriteLine(e.Message);  
    }  
...  
}
```

The DBMotoApplication object has access to a list of IServer objects, which is the list of server agents configured in the Management Center and stored in the local dbmoto.config file. DBMoto users typically define at least a "local" server in the Management Center. It is therefore possible to get a pointer of the local server by calling:

```
dbmServer = dbmApp.Servers["local"];
```

With a pointer to the local server, you can connect to the server:

- Using anonymous authentication (if allowed by the server agent):  
`dbmServer.Connect();`
- Using DBMoto authentication (by passing an application login):  
`dbmServer.Connect(txtUser.Text.Trim(), txtPassword.Text.Trim(), false);`
- Using Windows authentication (the Windows domain credentials):  
`dbmServer.Connect(null, null, true);`

The IServer class also contains the list of metadata defined for the server, and identifies the default metadata, which will be used by the Data Replicator:

```
currentMetadata = dbmServer.Metadata.DefaultMetadata;
```

For interaction that involves the Data Replicator and the metadata, the metadata should be 'synchronized' with the Data Replicator, meaning that the application using the DBMoto API should open a TCP/IP connection to the Data Replicator and receive notification from the Data Replicator:

```
currentMetadata.IsSynchronized = true;
```

should be called before loading the metadata.

The IsSynchronized property also allows you to receive monitor information via the API including how many records have been processed and the current transaction ID in a running replication.

Use the IServer StartReplicationManager and StopReplicationManager methods to start and stop the Data Replicator.

The simple code sample described above provides an introduction to the DBMoto API. Explore the full capabilities of the DBMoto API from Visual Studio, or from the DBMoto API Reference Guide.






# Chapter 11: Verifying Replication Results

## Verifying Replication Results

The DBMoto Verifier tool can be used to verify replication results for replications that have been created and run in DBMoto. It allows you to check for differences between source and target tables and therefore helps with troubleshooting replication problems during your DBMoto replication setup and implementation phase.

Here are the steps involved in using the DBMoto Verifier tool:

1. In the DBMoto Management Center, select the replication for which you want to compare source and target tables.
2. From the right mouse button menu, choose **Verify Replication**.  
The Verifier tab is displayed in the Management Center. It automatically starts the source and target table comparison. If the comparison is taking too long, you need to halt the process for any other reason, click  **Stop** at any time.
3. Review results in the Verifier tab.  
The default setting for the table comparison is to display only those records which contain differences between source and target. Differences are highlighted in red.
4. Optionally, click  **Options** to set options in the [Verifier Options dialog](#), including the number of rows to view at a time, specific data types to ignore and so on. This dialog also allows you to set WHERE conditions and override the default ORDER BY clause. You can also choose to compare only the number of records in the source and target tables.
5. To run the table comparison process with new options, click  **Run Verification** to start the comparison process. If you are comparing an active replication that is in mirroring mode, you should be aware that the results of the comparison process are not reliable because a replication process may run at any time.


## Verifying Results for Multiple Replications

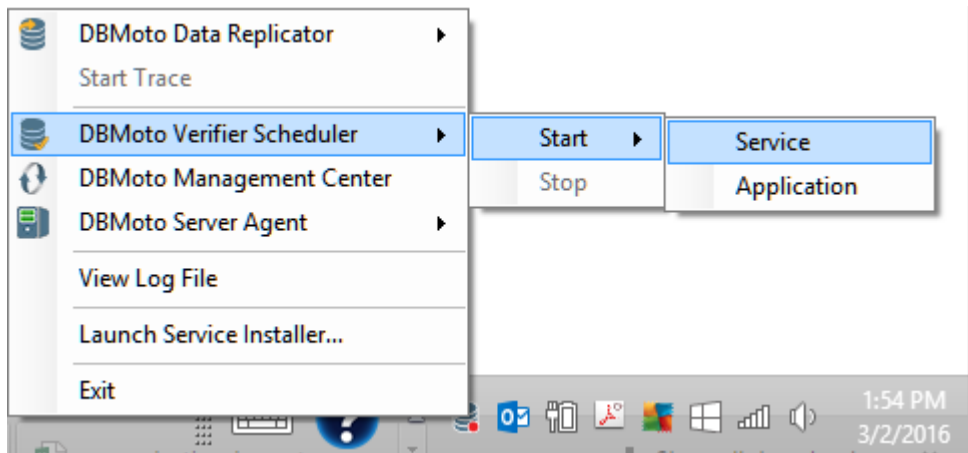
You can run DBMoto Verifier from the Replication Monitor tab in the Management Center. This tab allows you to select multiple replications, then, from the right mouse button menu, choose **Verify Replication**. In the DBMoto Verifier tab, a text summary is printed for each replication. To view source and target tables for a specific replication, note the name of the replication in the summary, select it in the Metadata Explorer and run the DBMoto Verifier for just that replication.

If you are verifying multiple replications, you can [save the results to a file](#) for easy reference.

## Scheduling Verifier Activity

DBMoto includes a Windows service to allow you to schedule verification and avoid peak database usage times. The schedule is set in the [Replication Properties dialog](#) and uses a Windows service called DBMoto Verifier Scheduler. The service is installed with DBMoto but is not started automatically.

To start the service, using the DBMoto Service Monitor program  in the Windows Notification Area. Note that if the service is stopped, scheduled verifications (and data reconciliations when enabled) will not be performed.



## Limitations

1. You must have a primary key defined on the target table, and the primary key must be defined as a field. For IBM Db2 for i users using a RRN as a primary key, the comparison fails because the RRN is implemented as an expression instead of a field in DBMoto.
2. Sometimes a Float is compared with a Double. Converting one to the other could cause a loss in precision and then the values will appear different. In this case, it is better to skip the comparison.
3. Comparison is not supported for XML data types.
4. Mapping expressions that contain special log values cannot be compared

"!TransactionID"

"!TransactionTS"

"!UserID"

"!RecordID"

"!ReceiverLibrary"

"!ReceiverName"

5. Custom scripts will not be included in the data comparison.

## Saving and Reviewing Comparisons

The DBMoto Verifier allows you to save out the results of a comparison in a text file format. The file can be viewed and edited from a text editor on your Windows desktop.

To save results:

1. Click  **Save Results** in the Toolbar.

2. In the Save As dialog, type a file name.

There are options to save the file as either .txt or RTF, but for DBMoto 7 the only difference between the files is the file extension.

3. Click **Save** to save the contents of the grids and the Details section of your results to a file.

## Analyzing Results

Results from running a comparison between source and target tables are reported in the DBMoto Verifier tab. You can use the [Verifier Options dialog](#) to specify the information that you want to see, but, by default, only records with differences are displayed.

- In cases where the row appears in both the source and the target database, but one or more field values do not match, the source table row is displayed followed by the target table row. The differences are highlighted in the color selected for differences in the Options dialog (default is red), and you can scroll the rows to identify the fields with different values.
- In cases where the row appears in the source table, but not in the target table, the row is displayed using the color selected for the source table in the Options dialog (default is blue.)
- In cases where the row appears in the target table, but not in the source table, the row is displayed using the color selected for the target table in the Options dialog (default is yellow.)

The Details pane shows a text output summary of the verification operation that has been performed. It includes the following information. The contents of the Details pane can be saved using the [Save Results button](#).

```
Comparing SAMPLES.DBM1 (18 records) to  
"TestCompare"."dbo".DBM1 (18 records)...
```

```
Results:
```

```
Number of records only on Source Table: 1
```

```
Number of records only on Target Table: 1
```



```
Number of different records: 3
```

## Reconciling Data Differences in Results

After running the DBMoto Verifier and identifying differences between source and target tables, you can use the Data Reconciliation feature to update the target table with differences found in the source table. For example, in the screenshot below, the source table does not contain a row with a "Product ID" value of 78. The Reconcile Data feature would modify

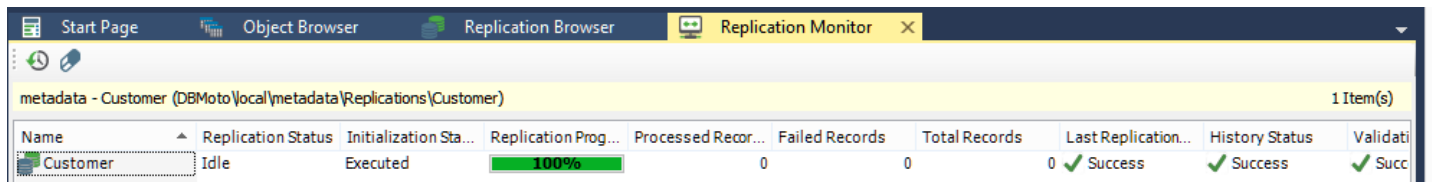
the target table to remove the row so that source and target tables match. DBMoto stores differences between source and target tables while running the Verifier and uses that information to modify the target table to match the source.

If the Verifier encounters errors while trying to modify the target table, the data reconciliation process stops after a default of 5 errors. You can change the limit on the number of errors encountered before stopping the process using the [Verifier tab in the Options dialog](#). This feature is useful if, for example, a database connection fails because the option would limit the number of attempts to connect to a database.

1. In the Metadata Explorer, select a replication to verify.
2. From the right mouse button menu, choose **Verify Replication**.
3. In the Verifier Tab, click  **Run Verification**.  
Following the verification, the Reconcile Data icon is active.
4. Click  **Reconcile Data**.
5. Confirm that you want to modify data in the target table.

## Monitoring and Reviewing Replications

You can monitor the progress of all your replications using the [Replication Monitor](#) in the DBMoto Management Center. Multiple replications can run concurrently so the Replication Monitor is useful in showing which replications are running and how each replication is progressing. For historical data on performance of replications over time, use the [DBMoto Dashboard](#).



Name	Replication Status	Initialization Sta...	Replication Prog...	Processed Recor...	Failed Records	Total Records	Last Replication...	History Status	Validati
Customer	Idle	Executed	100%	0	0	0	✓ Success	✓ Success	✓ Succ

The Replication Monitor provides access to:

- The [Replication Activity Viewer](#) for a more detailed view of performance for a specific replication.
- The [History Viewer](#) for a detailed account of the replication history for a specific replication.

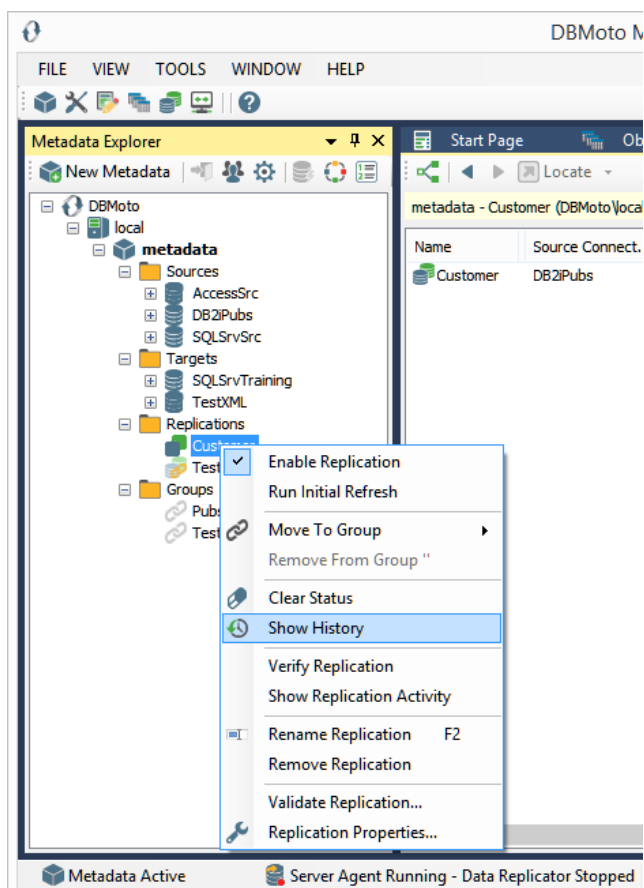
During replication, the Replication Monitor displays the progress of the replication (percentage of records/transactions already replicated, total number of records/transactions to replicate) and the time left for a replication to be completed. You can:

- Select a replication in the Monitor window to view the replication properties, including a history of past replication attempts.
- Review the Records columns (Processed Records, Failed Records, Total Records) for a quick overview of replication activity

- Scroll to the History Status column to see if the replication succeeded or failed. If it failed, use the right mouse button menu to open the [History Viewer](#) and identify errors. Note that the History Viewer only shows information on past replications when the **Activate Log History** option is checked in the Log tab of the [Data Replicator Options dialog](#).
- Review the **Transaction Latency Status** column to check that performance for a specific column is not lagging. A reg flag, with the label **Threshold Warning** indicates that records are not being processed

To identify where a replication failed:

1. Select a replication in the Metadata Explorer, Replication Browser or Replication Monitor.
2. Choose **Show History** from the right mouse button menu.



The [History Viewer tab](#) allows you to explore completed replication intervals and even view the DBMoto log to explore errors and warnings associated with the replication interval.




The screenshot shows the 'History Viewer: Customer' window. The top pane displays a table of replication sessions with columns: Status, Date/Time, Time Elapsed (hh:mm:ss), Session Type, Session Time, Processed Records, and Failed Records. The bottom pane displays a log of events with columns: Type, Date/Time, Description, and Help.

Status	Date/Time	Time Elapsed (hh:mm:ss)	Session Type	Session Time	Processed Records	Failed Records
Success	3/4/2016 10:57:10 AM	--:--:--	Refresh	00:00:02.6728043	91	0
Success	3/4/2016 10:57:16 AM	00:00:06.02	Mirroring	00:00:00.0136065	0	0
Success	3/4/2016 10:58:16 AM	00:01:00.11	Mirroring	00:00:00.0315976	0	0
Success	3/4/2016 10:59:16 AM	00:01:00.11	Mirroring	00:00:00.0119974	0	0
Success	3/4/2016 11:00:16 AM	00:01:00.12	Mirroring	00:00:00.0322485	0	0
Success	3/4/2016 11:01:16 AM	00:01:00.13	Mirroring	00:00:00.0333997	0	0
Success	3/4/2016 11:02:16 AM	00:01:00.12	Mirroring	00:00:00.0140265	0	0
Success	3/4/2016 11:03:17 AM	00:01:00.13	Mirroring	00:00:00.0125122	0	0
Success	3/4/2016 11:03:32 AM	00:00:15.07	Refresh	00:00:00.3335115	91	0
Success	3/4/2016 11:04:17 AM	00:00:45.10	Mirroring	00:00:00.0129312	0	0

Type	Date/Time	Description	Help
Information	2016-03-02 14:25:10.94	DBMoto Service version 9.0.0.6 started.	
Information	2016-03-02 14:25:10.98	Starting Replication Manager.	
Information	2016-03-02 14:25:12.04	Replication Manager started.	
Error	2016-03-02 23:45:37.13	Error in the Replication Manager thread: exception accessing the Ser...	
Information	2016-03-02 23:45:44.39	The connection to the Server Agent has recovered from a critical error.	
Information	2016-03-03 16:22:08.77	Stopping Replication Manager.	
Information	2016-03-03 16:22:11.82	Replication Manager stopped.	
Information	2016-03-03 16:22:11.83	DBMoto Service version 9.0.0.6 stopped.	
Information	2016-03-04 10:56:58.89	DBMoto Service version 9.0.0.6 started.	

3. Examine the replication intervals for errors in the upper pane of the History Viewer.
4. Select the replication interval that shows errors to display log messages in the lower pane of the History Viewer.

This pane provides the complete functionality of the [DBMoto Log Viewer](#). You can double-click a specific message to find out more about it, or you can click  **Show Filter Settings** to type in a specific search pattern for log messages.

5. If available, check the Replication tab for a table with the records which were not replicated.
6. Fix the errors.

You may need to [modify mappings in the Replication Properties dialog](#), or make changes in the source or target database tables.

7. Go back to the Replication Monitor to clear the History Status flag ready for future replications.
8. Try running the replication again to see that errors have been resolved.





## Chapter 12: Performance and Tuning

### Log Check Frequency in Mirroring and Synchronization Modes

If you are running replications in mirroring or synchronization modes, the transaction log on the table from which you are replicating is checked regularly to see if any transactions have occurred. You can modify the interval between log checks to adjust for best performance in your environment. For example, if you are using the default setting of 60 seconds, the source database log will be checked every 60 seconds for new transactions. If you want to keep your source and target tables synchronized as closely as possible, you might change the frequency to every 15 seconds. However, if the number of transactions between checks is very high, setting the frequency to 15 seconds will not help because updates to the target database could take longer than 15 seconds.

To change the frequency of log checks:

1. In the DBMoto Management Center Metadata Explorer, select the replication.
2. Using the right mouse button menu, make sure that there is no check mark next to **Enable Replication**.  
If **Enable Replication** is checked, select it to remove the checkmark. You have now disabled the replication.
3. From the right mouse button menu, choose **Replication Properties**.
4. In the [Replication Properties dialog](#), go to the **Preferences** tab.
5. Change the value of the **Read Interval** field. The default value is 60 seconds.
6. Click **OK** to close the Replication Properties dialog.
7. From the right mouse button menu, select **Enable Replication**.  
You should see a check mark next to the menu item to show that it is enabled.

If the replication is already running, for this change to take effect, you will need to [disable, then re-enable the replication](#).

### Performance and Efficiency

Processing performance for DBMoto varies widely from one configuration to another. DBMoto processing time depends on the resources available:

- Network bandwidth in connection between the DBMS servers and the server where DBMoto is running;
- Number of processors for the machines involved;
- Types of processors;
- Memory available;
- CPU usage.

The replication design in DBMoto needs to be carefully balanced to avoid bottlenecks during the replication process. Any measures you take depend on:

- Number of source and target connections;
- Number of tables in replication;
- Record size and record numbers for each table replicated;
- Replication mode (refresh, mirroring, synchronization);

- Number of transactions per second on the source database.
- Grouping of replications to optimize open database connections and query processing

For instance, the size of the table to be replicated and the number of transactions are important to evaluate whether it is more convenient to run continuous refreshes during the specified time, instead of a single full refresh and a continuous mirroring afterwards.

The DBMoto Management Center provides access to the [DBMoto Dashboard](#), a tool to analyze replication data and performance over a period of time based on the contents of a specified DBMoto Metadata and history file or database. It can provide information for each table in a selected set of tables including:

- Total number of processed records
- Total number of failed records
- Total number of records
- Total number of replication sessions
- Total number of processed records by all replications per unit of time.
- Average time it took to replicate specified records.

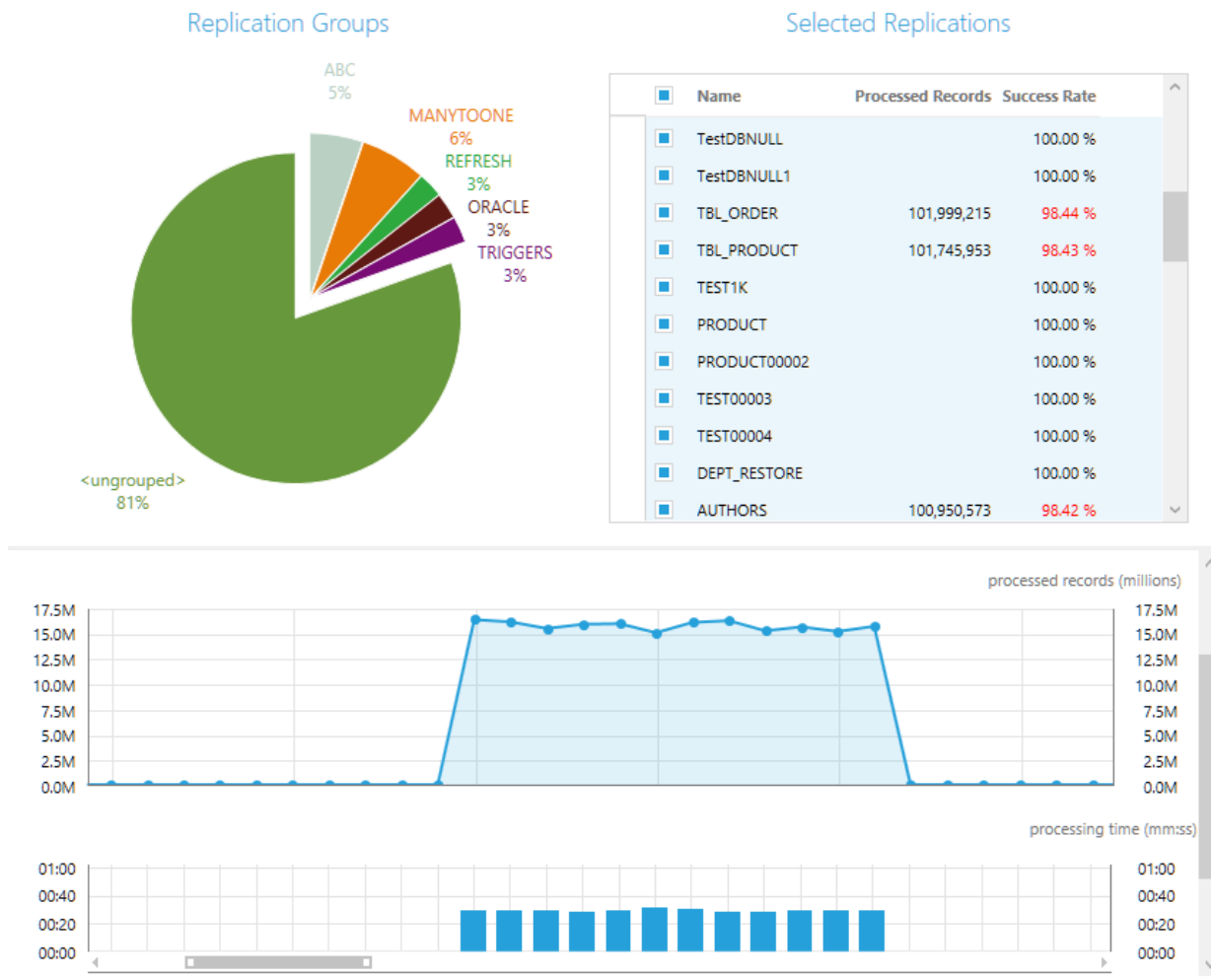
## Reviewing Replication Performance over Time

From the DBMoto Management Center, you can access the [DBMoto Dashboard \(Statistics Tab\)](#) to analyze performance of replications associated with a specific metadata. The Dashboard uses data stored in the Metadata and history files or database to provide a series of charts with performance information including:

- Total number of processed records
- Total number of failed records
- Total number of records
- Total number of replication sessions
- Total number of processed records by all replications per unit of time.
- Average time it took to replicate specified records.

The Dashboard provides tools for you to select replications to analyze and charts to review data on replication performance.

val: 11/1/2014 12:00 AM - 11/8/2014 12:00 AM



Take the following steps to set up and use the DBMoto Dashboard.

### 1. Enable history logging to a database

DBMoto Dashboard data is obtained from the history log, which is not enabled by default. Additionally, if you are planning to analyze large amounts of data, it is better to save history data to a database rather than a file (the default.) To set up the history log:

1. In the Management Center Metadata Explorer, select the server containing the metadata and replications you want to analyze. (The default server is called local.)
2. From the right mouse button menu, choose Data Replicator Options.
3. In the Data Replicator Options dialog, click the Log tab.

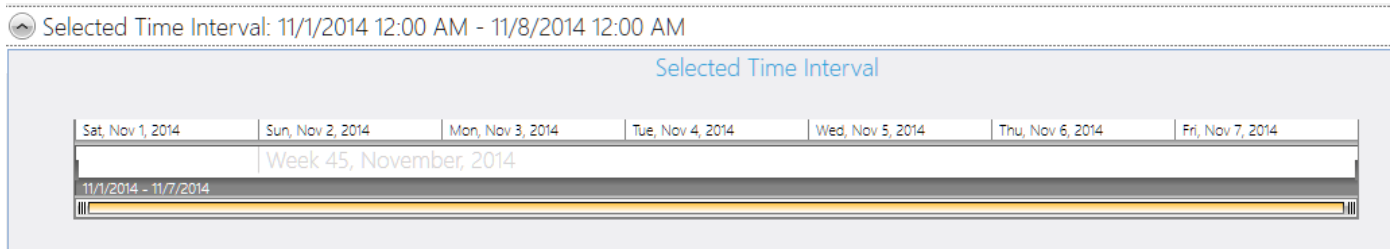
4. In the **Write Log To** field, from the drop-down list, choose **Database**.
5. Set up a connection to either the metadata database (the default is SQL Server CE, and the data is stored locally) or a database of your choice.
6. Set a limit on the number of messages saved, if desired. Bear in mind that the DBMoto Dashboard tools show performance over time, so limiting the number of messages reduces the amount of data you will be able to see.
7. Check the **Activate Log History** option.
8. Click **OK** to close the dialog.
9. Stop and restart the Data Replicator to begin logging history data.

## 2. Run replications for a period of time

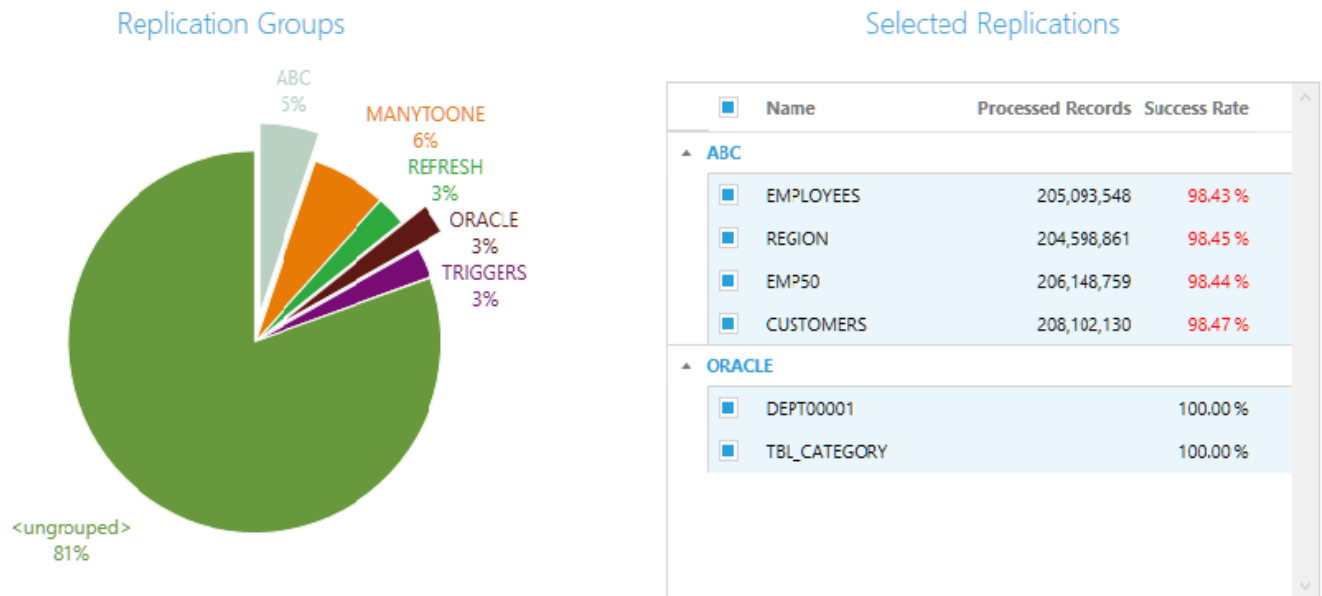
With the history log enabled, run replications for enough time to collect a useful amount of data for your needs. The DBMoto Dashboard charts can show records processed hourly, daily, weekly or monthly. 3. Review performance data in the Dashboard

## 3. Open the Dashboard and review the data

1. In the Management Center Metadata Explorer, select the metadata containing the replications you want to analyze. (The default metadata is called "metadata".)
2. From the right mouse button menu, choose **Show Dashboard**.  
A new tab named Statistics opens in the Management Center.
3. In the Selected Time Interval area, use the down arrow button and the horizontal scrollbar to adjust the time period you want to analyze.



4. In the Replication Groups area, click on the sections of the pie chart containing groups with replications you want to analyze. Note that replications not included in a DBMoto group are in the section <ungrouped>.



5. In the Selected Replications area, use the checkboxes to select all the replications to analyze.
6. Use the [Overview](#) and [Throughput](#) charts to analyze the replications. For details on these charts see the [DBMoto Dashboard](#) topic.

## Improving Performance Using Refresh with SQL Server, MySQL

If you are replicating to Microsoft SQL Server or MySQL using Refresh mode, you can improve performance by using the bulk insert option during replication. This option is selected by default in the [Preferences tab](#) of the [Replication Properties dialog](#).

1. In the Target Connection wizard, when [creating a target connection](#), select the Microsoft SQL Server .NET Data Provider or the MySQL .NET Data Provider.
2. In the Replication wizard, when [configuring the replication](#), choose **Refresh** as the Replication mode.
3. Select the replication in the Metadata Explorer.
4. From the right mouse button menu, choose [Replication Properties](#).
5. In the Preferences tab, check that the **Insert Mode** property is set to BulkInsert.
6. Adjust the **Block Size** property to a value between 50 and 100 for best performance.

If the replication is already running, for this change to take effect, you will need to [disable, then re-enable the replication](#).



## Improving Performance Using Mirroring with IBM Db2 for i

If you are replicating from IBM Db2 for i (iSeries/AS400) using Mirroring mode, you can improve performance by setting a block size for reading data from the journal.

The block size setting is useful if you expect a huge number of transactions between mirrorings. DBMoto reads the journal and stores records that need to be replicated in a temporary file. If the file contains a large number of records or a smaller number of very large records, it can impact performance. In this case, you can set a limit on the number of records to handle at one time. For example, if you set the value of the **MaxSizeMirroringBlock** property in the [Connection Properties](#) to 250, only 250 records will be replicated at a time.

1. In the DBMoto Management Center, expand the tree to display your Db2 for i (iSeries/AS400) source connection.
2. From the right mouse button menu, choose Connection Properties.
3. Scroll down to the **MaxSizeMirroringBlock** property.
4. Set the value of the property according to the size of your records and the number of records you are likely to be mirroring.
5. Click **OK** to close the [Connection Properties dialog](#).

If the replication is already running, for this change to take effect, you will need to [disable, then re-enable the replication](#).

## Managing Performance Using Thread Settings

Here are some hints on how to manage performance using settings found in the [Data Replicator Options dialog](#) and [Replication Properties dialog](#).

### Data Replicator Options Dialog (General Tab)

#### Max number of concurrent threads

Increasing this number will boost the performance but the CPU and memory usage will also increase and other applications running on the same PC might be affected.

#### Thread Delay

When this value is greater than 0, it represents the number of time slices the thread executes before going into a sleep state. The purpose of this parameter is to release some CPU resources if the CPU usage is too high. Setting this value to 1 instructs DBMoto to suspend each thread for a fraction of time (Sleep(0)) at every time slice. 1 is the value that releases the maximum of the CPU usage but it's also the value that penalizes performance the most. Setting it to n indicates that a thread will sleep every n slices of time. If the thread delay is 0, the thread is never suspended if there are replications running.

#### Thread execution factor

This value is a factor indicating how much time a thread spends processing a task (time slice) before switching to the following task in the execution list. It is used to increase the time slice DBMoto assigns to each task in the following

way. The number specified multiplies the slice of time DBMoto assigns to each task (reader or writer). For example, if you double the value, DBMoto will double the time assigned for each task.

## Replication Properties Dialog

### Thread Priority

This value affects the time slice which DBMoto assigns to a thread when the thread is running the current replication. It's similar to the "Thread execution factor", but relative to a specific replication. Increasing the thread priority of the replication might increase the performance because it increases the slice of time which DBMoto dedicates for processing the replication. However the performance is also affected by the total number of replications running and the "Max number of concurrent threads" value set in the [Data Replicator Options dialog](#).



## Chapter 13: Maintenance and Recovery

### Identifying and Recovering From Errors

DBMoto is designed to recover from most errors in an automated and efficient manner so that you do not lose data or have unusual delays.

However, on occasion during continuous mirroring, you may encounter an error that requires a complete refresh of the target table. Before taking this step, verify that the errors you have observed in the log apply to the current replication and replication settings:

1. [Stop the DBMoto Replicator.](#)
2. Remove the DBMoto log file. (You might want to copy it to a different location so that you can use it for comparison later.)
3. [Start the DBMoto Replicator.](#)
4. Check the log to see if the error(s) occur in the most recent replication.

Depending on the error, you may need to change the source tables to avoid the errors, then do an Initial Refresh of the target tables to match the modified source tables:

1. [Stop the DBMoto Replicator.](#)
2. In your database environment, make changes in source tables as necessary to prevent further errors from occurring.
3. In the Management Center, select the replication.
4. From the right mouse button menu, choose **Replication Properties**.
5. On the **General** tab, click **Mapping**.
6. Check the mapping between source and target fields to see that this is not the cause of replication errors. For example, are source and target field data types compatible?
7. On the **Scheduler** tab, check **Execute Initial Refresh**.
8. In the **Start Time** field, click **Now** to set the execution time.
9. [Start the DBMoto Replicator.](#)

The refresh replication starts at the last transaction ID in the log and performs a truncation(removal) of the target table, then a complete replication of the source table. Subsequent mirroring operations use the transaction ID stored at the end of the refresh replication

### Email Alerts

You can configure DBMoto to send email alerts when certain events occur during replication using one of the following three options:

- [Define email alert messages for a specific replication](#) using the [DBMoto Replication Alert Properties dialog](#).

- [Define email alert messages for all replications and/or the Data Replicator](#) using the [DBMoto Server Alert Properties dialog](#).
- [Write a script using the SendMail function](#) to generate email alerts on any available replication event

## Defining Email Alert Messages for a Specific Replication

This option provides a way to quickly define email alert messages for a selected replication using the [Replication Properties dialog](#) and the [DBMoto Replication Alert Properties dialog](#).

1. In the Management Center **Metadata Explorer**, select a replication.
2. From the right mouse button menu, choose **Replication Properties**.
3. In the **Replication Properties** dialog, go to the **Alerts** tab.
4. Click **Add Alert** to open the **Replication Alert Properties** dialog.
5. Use the [Alert Settings tab](#) to choose which events to include in an email.
6. Set the Alert Schedule if you want to limit the number of emails you receive.
7. Go to the **Message Layout** tab.
8. Type recipient email addresses, separated by semi-colons.
9. Edit the email **Title** and **Body** text as needed.
10. Optionally specify HTML format and/or DBMoto log attachment.
11. Click **OK** to create the alert and close the Replication Alert Properties dialog.
12. Click **OK** to close the Replication Properties dialog.
13. In the DBMoto Management Center **Metadata Explorer**, select the server (usually **local**), then, from the right mouse button menu, select **Data Replicator Options...** to open the [Data Replicator Options dialog](#).
  1. Go to the **Mail** tab.
  2. Enter the SMTP Server and Port Number for your email server.
  3. Enter an email address in the **From** field.
  4. Enter an email recipients in the **To** field.
  5. Set authentication and SSL encryption as needed for your email server.
  6. Click **Test Email** to be sure that the email information you have entered is correct.
14. Stop and restart the Data Replicator to enable the email alert(s).

## Defining Email Alert Messages for Replications and/or the Data Replicator

This option provides a way to quickly define email alert messages for the Data Replicator and/or all replications associated with the server using the [Data Replicator Options dialog](#) and the [DBMoto Server Alert Properties dialog](#).

1. In the Management Center **Metadata Explorer**, select a server (usually **local**).
2. From the right mouse button menu, choose **Data Replicator Options**.

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.

3. In the **Data Replicator Options** dialog, go to the **Alerts** tab.
4. Click **Add Alert**, then **Add Event Alert** to open the **Server Alert Properties** dialog.
5. Use the [Alert Settings tab](#) to choose which events to include in an email.
6. Set the Alert Schedule if you want to limit the number of emails you receive.
7. Go to the **Message Layout** tab.
8. Type recipient email addresses, separated by semi-colons.
9. Edit the email **Title** and **Body** text as needed.
10. Optionally specify HTML format and/or DBMoto log attachment.
11. Click **OK** to create the alert and close the Server Alert Properties dialog.
12. In the Data Replicator Options dialog:
  1. Go to the **Mail** tab.
  2. Enter the SMTP Server and Port Number for your email server.
  3. Enter an email address in the **From** field.
  4. Enter an email recipients in the **To** field.
  5. Set authentication and SSL encryption as needed for your email server.
  6. Click **Test Email** to be sure that the email information you have entered is correct.
13. Stop and restart the Data Replicator to enable the email alert(s).

You can also use the Alerts tab in the Data Replicator Options dialog to generate an email Activity Report:

1. In the **Data Replicator Options** dialog, go to the **Alerts** tab.
2. Click **Add Alert**, then **Add Activity Report** to open the **Server Alert Properties - Activity Report** dialog.
3. Complete the dialog and set up your email server using the **Mail** tab.

## Writing a Script Using the SendMail Function

This option allows you to build a fully customized email message using any of the [replication events](#) made available in DBMoto.

In the DBMoto Management Center **Metadata Explorer**, select the server for which you want to set email alerts, then, from the right mouse button menu, select **Data Replicator Options...** to open the [Data Replicator Options dialog](#).

1. Go to the **Mail** tab.
2. Enter the SMTP Server for your email account.
3. Enter your email address in the **From** field.
4. Enter the email recipients in the **To** field.

5. Click **Test Email** to be sure that the email information you have entered is correct.

Use a [SendMail function](#) from a [script](#) to provide an email alert as in the example below.

```
SendMail ("Message from DBMoto",
        "An error occurred that requires the intervention of the system administrator.")
```

## Understanding Log Files

The DBMoto log file is an important source of information for details on any replication. Even though a replication process is typically automatic enough to determine choices based on the replication configuration, you sometimes need to examine the log file make adjustments to the replication configuration. For this reason is important to check the DBMoto.log file and understand the difference between:

<b>Information messages</b>	Useful information to follow the replication process as it progresses.
<b>Warning messages</b>	A warning usually indicates an issue that does not typically generate an error in the replication process. However, you should check the source of the message and assess whether it could affect the results of your replication.
<b>Error messages</b>	<p>An error message generally indicates an error in the replication. In some cases, DBMoto is able to resolve the issue automatically, but the information is still signaled as an error message because it is a source of possible asynchronism between source and target tables.</p> <p>In some cases, the error is an internal DBMoto error (this is determined by the presence of a single level error message.) In other cases, the error is generated from the .NET, ODBC or OLE DB provider and the native provider error is shown, together with the DBMoto error message. Sometimes errors appear simply because wrong assumptions are made during the replication setup. For instance, when replicating a table from an IBM Db2 for i source in mirroring mode, if the table is not journalized, an error will indicate the need to create a journal for that table. A duplicate primary key error could occur if inserting a record when the target table already contains that primary key value.</p>

These are some of the entries you may find in your DBMoto log file.

**Warning A replication conflict has occurred during replication (AddNew) of a record.**

A synchronization conflict error. For example, the replicator is trying to insert a new record when the target table already contains that record (identified by the primary key value) but there are some differences with the values in the other columns. Verify why the conflict is occurring and if the current values are valid. Often previous errors in the replication can generate synchronization conflict errors.

**Warning A replication conflict has occurred during replication (Delete) of a record.**

A synchronization conflict error. For example, the replicator is trying to insert a new record when the target table already contains that record (identified by the primary key value) but there are some differences with the values in the other columns. Verify why the conflict is occurring and if the current values are valid. Often previous errors in the replication can generate synchronization conflict errors.

**Replication Complete**

You will only see a message in the log file to indicate that a replication has completed for refresh and continuous refresh. For mirroring and synchronization, there is no end to the replication and therefore no such message in the log file.

**Date/Time Conversion Errors**

These messages are shown when the Replicator reads a null date on the Db2 system (null date set to 01/01/01) because SQL Server doesn't accept the year. The error is only a notification of the forced conversion and in this cases for every forced conversion an error is sent to DBMoto log; but replication is not stopped. However is possible manage date/time conversion errors from the [Data Replicator Options dialog](#), available from the **Metadata** menu.

**(-2147217900) ORA-01293: time or SCN range not fully contained in the listed logfiles ORA-06512: at "SYS.DBMS\_LOGMNR", line 42 ORA-06512: at line 1**

The range requested from the log is not present in the log files used. Because DBMoto opens the Oracle log based on the last SCN (transaction ID) processed during the last mirroring step (which is the number you see displayed in the "Redo Log" dialog in DBMoto EM), this error might indicate that, at some point, after a certain number of transactions processed, the Oracle log files have reached the maximum size available and are getting over-written from the beginning (in circular way).

If this is the problem, increasing the size of the log files or creating more log files could at least delay the problem if not solve it. Of course all depends on the number of transactions DBMoto has to analyze, if it is too huge chances are that DBMoto is not able to keep up with this increasing number.

**'OracleEllipsePrd' connection transactions have been deleted. Database will be temporarily removed from replicated databases. (-2147217900) ORA-01293: time or SCN range not fully contained in the listed logfiles. ORA-06512: at "SYS.DBMS\_LOGMNR", line 42.**

Use the [HiT Software Knowledge Base](#) to research issues or [contact customer support](#) for help on DBMoto.



## Creating and Deleting Text Log Files

When saving the log to a text file, DBMoto uses the file DBMoto.log in the **Log** folder to store information about each replication session. This file is created and maintained automatically, however, you can specify how often you want to create a new file and how large each log file should be in the [Data Replicator Options dialog](#) in the DBMoto Management Center.

1. In the **Metadata Explorer**, select the server for which you want to set log options.
2. From the right mouse button menu, choose **Data Replicator Options**, to open the Data Replicator Options dialog.
3. In the Data Replicator Options dialog, go to the **Log** tab.

To delete a log file, you must first stop and exit the DBMoto Data Replicator so that the log file is no longer used. When you delete the log file, a new file is created next time you run a replication.

Note that, in the [Data Replicator Options dialog](#), you can also choose to save the DBMoto log to a database or to the Windows Event Viewer.

## Monitoring Text Log File Size

When saving the log to a text file, DBMoto uses the file DBMoto.log in the DBMoto\Log folder to store information about each replication session. DBMoto Replicator performance decreases with the growth of the log file. To avoid this situation, plan to manage your log file by using the [Data Replicator Options dialog](#) in the DBMoto Management Center.

1. In the **Metadata Explorer**, select the server for which you want to set log options.
2. From the right mouse button menu, choose **Data Replicator Options**, to open the Data Replicator Options dialog.
3. In the Data Replicator Options dialog, go to the **Log** tab.

By default, the log file is a single file with a maximum size of 10MB, but you can change the maximum size or create a new file every certain number of days. If you are creating multiple log files, you can determine how many files you want to keep. DBMoto then manages that number of files, naming them by date and/or increment number).

Note that, in the [Data Replicator Options dialog](#), you can also choose to save the DBMoto log to a database or to the Windows Event Viewer.

## Generating Trace Files

The technical support staff at HiT Software sometimes request one or more of the following trace files.

Trace Type	Settings	Description
Data Replicator	<a href="#">Data Replicator Options</a> dialog on the right mouse button menu of a server.	Traces all activity associated with the Data Replicator running on a specific server.
Server Agent	<a href="#">Data Replicator Options</a>	Traces all activity associated with a specific DBMoto

	dialog on the right mouse button menu of a server.	server.
Management Center	<a href="#">Options dialog</a> on the Tools menu.	Traces all activity associated with a specific Management Center installation. Primarily tracks user interaction with the Management Center. Can include traces of commands to the metadata tables via the Settings dialog.
API	<a href="#">Options dialog</a> on the Tools menu.	Traces the activity that occurs when using the <a href="#">DBMoto API</a> to run DBMoto programmatically.
Service Monitor	<a href="#">Options dialog</a> on the Tools menu.	Traces all application and service activity associated with the <a href="#">DBMoto Service Monitor</a> located in the Windows System Tray.

## Generating a Data Replicator or Server Agent Trace

1. [Stop the Data Replicator](#).
2. In the Management Center, select the current DBMoto server (usually local), and, from the right mouse button menu, choose **Data Replicator Options...** to open the [Data Replicator Options dialog](#).
3. Go to the **Trace** tab.
4. Click in the checkbox **Activate Data Replicator Tracing** to enable the trace.
5. If requested by the support staff, click **Settings** to open the [Data Replicator Trace Settings dialog](#) and specify which traces to run, or which replications to include in the trace.
6. In the Data Replicator Options dialog, click **OK** to set the trace(s).
7. [Start the Data Replicator](#).

The default location for generated trace files is the DBMoto Log folder.

NOTE: It is possible to switch tracing on and off via the DBMoto Service Monitor icon  in the Windows Notification area, without restarting the Data Replicator.

## Generating a Management Center, API or Service Monitor Trace

1. In the DBMoto Management Center menubar, click **Tools**, then **Options**.
2. In the [Options dialog](#), go to the **Trace** tab.
3. Click in the appropriate check box to activate tracing.
4. Modify the trace file folder as needed. The default location for generated trace files is the Log folder where you installed DBMoto.
5. Click **OK** to close the dialog.

6. For Service Monitor and API traces, you need to stop and restart the applications before the trace settings take effect. Management Center trace setting take effect immediately after closing the dialog.

## Backing Up and Restoring Metadata

DBmoto offers two approaches to backing up your metadata, each with a different purpose in mind.

- **XML Backup and Restore**

User-driven backup of metadata to an XML file to record a specific metadata state, or to easily move metadata details from one system to another. This approach is described below.

- **Real-time Metadata Database Backup**

One or two established alternative databases are kept synchronized with the main metadata so that, in case of metadata corruption or failure, a backup database can be designated as the main metadata, with minimal interruption to the replication cycle. Alternative metadata database connections are established via the [Metadata Properties dialog](#).

### Backing Up Metadata to an XML File

It is a good habit to create a backup of your metadata tables every time you change the replication rules and settings. It can prevent loss of data and provide easy recovery of data and application settings, even if a new installation of the application is required.

A metadata backup is an export of the metadata tables in XML format. The backup does not contain any metadata connection details.

To save your existing Metadata database as a backup:

1. Shut down the DBMoto Data Replicator.
2. In the DBMoto Management Center, select the metadata for which you want to create a backup.
3. From the main menu bar, choose **Metadata**, then **Backup Metadata....**
4. In the Save DBMoto Metadata Backup dialog, choose a folder and type a file name.
5. Click **Save** to save the metadata in an XML file.

### Restoring Metadata

Use either the Restore Metadata wizard or the Metadata Connection wizard, depending on your needs:

- The [Restore Metadata wizard](#) allows you to selectively restore saved metadata in the existing metadata, effectively letting you add replications defined in a different metadata to the current metadata. This wizard also allows you to restore an entire metadata set in the existing metadata, thereby overwriting the existing metadata.
- The [Metadata Connection wizard](#) allows you to create a new metadata connection and specify an existing metadata backup file to use with the new connection.

### Using the Restore Metadata Wizard to Restore a Subset of Replications

Using the [Restore Metadata wizard](#), you can restore selected replications from a metadata backup file. The restored replications will be added to your current metadata.

1. Make sure that you have saved a copy of the metadata containing the replications you want to restore.
2. In the DBMoto Management Center, select the metadata into which you want to restore replications.
3. From the Metadata menu, choose Restore Metadata.
4. In the **Restore Metadata Options** dialog, choose **Custom Restore**.
5. Follow steps in the [Restore Metadata wizard](#) to specify the replications to restore and their associated source and target connections.

## Using the Restore Metadata Wizard to Replace the Current Metadata

Using the [Restore Metadata wizard](#), you can restore a metadata backup file using an existing metadata connection, in which case you'll overwrite your existing metadata.

1. Make sure that you have saved a copy of the metadata you want to restore.
2. In the DBMoto Management Center, select the metadata where you want to restore data.
3. From the Metadata menu, choose Restore Metadata.
4. In the **Restore Metadata Options** dialog, choose **Standard Restore**.
5. In the **Open DBMoto Metadata Backup** dialog, Select the metadata backup file that contains the metadata you want to restore.  
NOTE: This metadata will replace all existing metadata.
6. Click **Open** to begin the restoration process.

## Using the Metadata Connection Wizard to Restore a Saved Metadata File

Using the [Metadata Connection wizard](#), you can restore a metadata backup using an existing metadata connection, in which case you'll overwrite your existing metadata. You can also create a new metadata and import the metadata backup XML file. It is possible to restore a metadata backup on any database, effectively [copying the metadata](#) to another location.

To restore the Metadata backup file:

1. In the Management Center, from the main menu bar, choose **Metadata**, then **Add New Metadata...**
2. In the Metadata Connection wizard **Select Provider** screen, type a name for the new metadata. This name does not have to match the name of the metadata that you are restoring.
3. Specify the .NET provider that you plan to use for the new database server.
4. In the **Set Connection String** screen, provide the connection information for the database server where the metadata was originally installed.
5. In the **Define Metadata** screen, select the option **Create a new metadata database**.
6. Check the option **Restore the metadata set from a backup file**.
7. Enter the pathname to the file you saved in step 4 above.
8. Click **Next**, then **Finish** to complete the wizard.

9. Clear out the DBMoto.log file.
10. Start the Data Replicator and monitor the DBMoto.log file for errors.

## Moving a Metadata Database

If you need to move your DBMoto Metadata database to a different server, you can create a backup of the current metadata using the tools available in the Management Center. The metadata (without the current connection details) is saved to an XML file. You can then create a new metadata connection and use the backup file to create the metadata using the new connection.

To save your existing Metadata database as a backup:

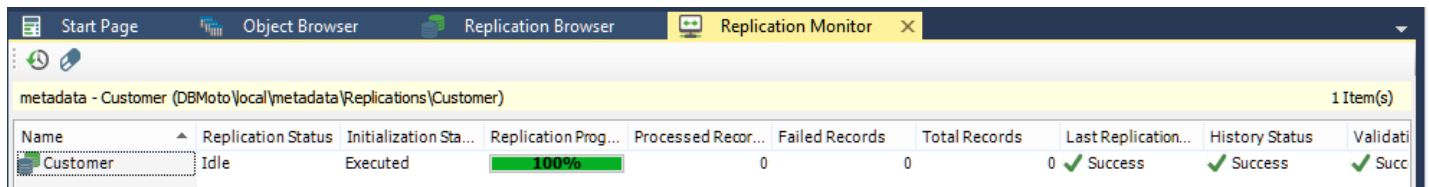
1. Shut down the DBMoto Data Replicator.
2. In the Management Center, select the metadata for which you want to create a backup.
3. From the main menu bar, choose **Metadata**, then **Backup Metadata....**
4. In the Save DBMoto Metadata Backup dialog, choose a folder and type a file name.
5. Click **Save** to save the metadata in an XML file.

To use the Metadata backup file for a new database server:

1. In the Management Center, from the main menu bar, choose **Metadata**, then **Add New Metadata....**
2. In the Metadata Connection wizard **Select Provider** screen, type a name for the new metadata. This name does not have to match the name of the metadata that you are moving.
3. Specify the .NET provider that you plan to use for the new database server.
4. In the **Set Connection String** screen, provide the connection information for the new database server.
5. In the **Define Metadata** screen, select the option **Create a new metadata database**.
6. Check the option **Restore the metadata set from a backup file**.
7. Enter the pathname to the file you saved in step 4 above.
8. Click **Next**, then **Finish** to complete the wizard.
9. Clear out the DBMoto.log file.
10. Start the Data Replicator and monitor the DBMoto.log file for errors.

## Monitoring and Reviewing Replications

You can monitor the progress of all your replications using the [Replication Monitor](#) in the DBMoto Management Center. Multiple replications can run concurrently so the Replication Monitor is useful in showing which replications are running and how each replication is progressing.



The Replication Monitor provides access to:

- The [Replication Activity Viewer](#) for a more detailed view of performance for a specific replication.
- The [History Viewer](#) for a detailed account of the replication history for a specific replication.

During replication, the Replication Monitor displays the progress of the replication (percentage of records/transactions already replicated, total number of records/transactions to replicate) and the time left for a replication to be completed.

You can:

- Select a replication in the Monitor window to view the replication properties, including a history of past replication attempts.
- Review the Records columns (Processed Records, Failed Records, Total Records) for a quick overview of replication activity
- Scroll to the History Status column to see if the replication succeeded or failed. If it failed, use the right mouse button menu to open the [History Viewer](#) and identify errors. Note that the History Viewer only shows information on past replications when the **Activate Log History** option is checked in the Log tab of the [Data Replicator Options dialog](#).
- Review the **Transaction Latency Status** column to check that performance for a specific column is not lagging. A reg flag, with the label **Threshold Warning** indicates that records are not being processed

To identify where a replication failed:

1. Select a replication in the Metadata Explorer, Replication Browser or Replication Monitor.
2. Choose **Show History** from the right mouse button menu.

The screenshot shows the History Viewer application with two main panes. The top pane displays a table of replication sessions, and the bottom pane displays a log of events.


Status	Date/Time	Time Elapsed (hh:mm:ss)	Session Type	Session Time	Processed Records	Failed Records
Success	3/4/2016 10:57:10 AM	--:--:--	Refresh	00:00:02.6728043	91	0
Success	3/4/2016 10:57:16 AM	00:00:06.02	Mirroring	00:00:00.0136065	0	0
Success	3/4/2016 10:58:16 AM	00:01:00.11	Mirroring	00:00:00.0315976	0	0
Success	3/4/2016 10:59:16 AM	00:01:00.11	Mirroring	00:00:00.0119974	0	0
Success	3/4/2016 11:00:16 AM	00:01:00.12	Mirroring	00:00:00.0322485	0	0
Success	3/4/2016 11:01:16 AM	00:01:00.13	Mirroring	00:00:00.0333997	0	0
Success	3/4/2016 11:02:16 AM	00:01:00.12	Mirroring	00:00:00.0140265	0	0
Success	3/4/2016 11:03:17 AM	00:01:00.13	Mirroring	00:00:00.0125122	0	0
Success	3/4/2016 11:03:32 AM	00:00:15.07	Refresh	00:00:00.3335115	91	0
Success	3/4/2016 11:04:17 AM	00:00:45.10	Mirroring	00:00:00.0129312	0	0

Type	Date/Time	Description	Help
Information	2016-03-02 14:25:10.94	DBMoto Service version 9.0.0.6 started.	
Information	2016-03-02 14:25:10.98	Starting Replication Manager.	
Information	2016-03-02 14:25:12.04	Replication Manager started.	
Error	2016-03-02 23:45:37.13	Error in the Replication Manager thread: exception accessing the Ser...	
Information	2016-03-02 23:45:44.39	The connection to the Server Agent has recovered from a critical error.	
Information	2016-03-03 16:22:08.77	Stopping Replication Manager.	
Information	2016-03-03 16:22:11.82	Replication Manager stopped.	
Information	2016-03-03 16:22:11.83	DBMoto Service version 9.0.0.6 stopped.	
Information	2016-03-04 10:56:58.89	DBMoto Service version 9.0.0.6 started.	

The [History Viewer](#) tab allows you to explore completed replication intervals and even view the DBMoto log to explore errors and warnings associated with the replication interval.

3. Examine the replication intervals for errors in the upper pane of the History Viewer.
4. Select the replication interval that shows errors to display log messages in the lower pane of the History Viewer.

This pane provides the complete functionality of the [DBMoto Log Viewer](#). You can double-click a specific message to find out more about it, or you can click  **Show Filter Settings** to type in a specific search pattern for log messages.

5. If available, check the Replication tab for a table with the records which were not replicated.
6. Fix the errors.

You may need to [modify mappings in the Replication Properties dialog](#), or make changes in the source or target database tables.

7. Go back to the Replication Monitor to clear the History Status flag ready for future replications.



8. Try running the replication again to see that errors have been resolved.

## Managing Source/Target Table Schema Changes

### Detection and Repair of Source Table Schema Changes

DBMoto is able to detect table schema changes on the source table for transactional replications in two ways:

- When the Data Replicator finds a schema change record (DDL) in the database transaction log
- When validating a schema using the Validate Replication dialog in the Management Center

If the **Stop on Schema Change** property in the [Replication Properties dialog Preferences tab](#) is also set to True, the following occurs:

1. The Data Replicator disables a replication when a schema change operation is detected.
2. The replication cannot be enabled or executed by the Data Replicator without updating the schema.
3. The Replication Monitor shows the ValidationResult field as Fail.

To repair the replication status, use the **Rebuild** feature in the Validate Replications dialog. After prompting with details on the steps to be performed, **Rebuild** updates the table schema and sets the replication validation status to Success. Rebuild **does not** re-enable a replication.

**NOTE:** Following schema updates, it is advisable to check the field mapping between source and target tables using the [Fields Mapping dialog](#) available in the [Replication Properties](#).

### Proactive Update of Source/Target Schemas

Alternatively, if you are aware of a change in the structure of source or target tables after setting up your connections and any replications, you can update the schema information stored in the DBMoto metadata using the **Refresh Schema Information** option available on all source and target table, schema and connection nodes in the Metadata Explorer. Please do not confuse this refresh with the replication mode or the initial refresh of a mirroring or synchronization.

1. Stop the DBMoto Data Replicator or disable the replication(s) accessing the affected table(s) by right clicking on the replications and clicking on the checked **Enable** entry.
2. Right click on the source/target table(s), schema(s) or connection(s) and choose **Refresh Schema Information**. If you have changed more than one table in a schema or on a connection you can right click on the schema or connection and choose **Refresh Schema Information** rather than separately doing this on each table.
3. Enable the replication(s) again.

Note that this operation only updates DBMoto metadata information about the table(s). It does not cause another initial refresh replication and does not change any properties for replications which use that table as a source or target table. If



you make changes to a source and/or target table already involved in a replication, you must manually [change the replication properties](#) to reflect these changes.

## Recovering from Interruptions to Refresh Replications

Refresh replications can take a significant amount of time, depending on the size of the tables involved in the replication. If a connection problem occurs during the refresh, the point at which the replication was interrupted is recorded by DBMoto. If the connection is re-established, replication starts automatically from the point at which it stopped. However, if you stop and restart DBReplicator, you will lose the information needed to continue replication and the replication will start from the beginning again.

## Running a Refresh Replication with Transactional Replications

If you have set up mirroring or synchronization replications, there may be times when you also want to execute a full refresh (snapshot) replication, as in the following examples:

- If the DBMoto replicator has been stopped for a period of time that exceeds the length of time that logs are kept available, a refresh replication can capture all the changes that are no longer available in the log.
- If a field value changes in every source record of a large table, it can be more efficient to run a refresh replication instead of capturing the change using mirroring or synchronization.
- When replicating from an IBM Db2 for i table which uses RRN (relative record numbers). If the source table is reorganized, the RRN are modified and the new information needs to be present on the target database for subsequent mirroring replications to work correctly.

The [Replication Properties dialog](#) for mirroring and synchronization replications allow you to specify that you want to perform an initial refresh (erase all records from the target database and complete copy from the source database based on the mappings in your replication definition.) You can avoid truncating all records on the target database by [writing a script](#) for the [Refresh\\_onBeforeTruncate](#).

You can either define an initial refresh to occur when setting up the replication (typically using the [Replication wizard](#)) or you can pause the replication schedule to add a refresh operation:

1. [Disable and/or stop the replication](#). The replication must be both stopped and disabled before you can modify its properties.
2. In the DBMoto Management Center, expand the metadata for the replication.
3. Select the replication that you want to refresh.
4. From the right mouse button menu, choose **Replication Properties**.
5. Click **Scheduler** in the left hand pane to view the Scheduler tab.
6. In the Start Time field, click **Now**.
7. Check **Execute Initial Refresh**, if it is not already checked
8. Click **OK** to close the Replication Properties dialog.
9. [Enable the replication or restart the Data Replicator](#)

10. Monitor the refresh operation in the Replication Monitor tab of the Management Center.

If your replication is part of a group, first remove the replication from the group, set the initial refresh, then add the replication back into the group.



## Chapter 14: DBMoto API Reference

### DBMoto API Overview

The DBMoto API, a .NET object model structure called the DBMoto Object Model (DBOM), provides developers with complete control over creating and running replications programmatically.

The [code below](#) provides a simple example of how to start the Data Replicator for a replication in the default metadata running on the "local" DBMoto server.

The .NET namespace for the DBMoto API is `HitSoftware.DBMoto.Application`. This namespace includes the class `DBMotoApplication`, the main singleton object of the DBOM, which can be instantiated through the static singleton method `Instance` `DBMotoApplication.Instance`.

When building a client application using the DBMoto API, be sure to follow directions below for the [AppConfig file](#).

### API Sample Code

```

.....
using HitSoftware.DBMoto.ObjectModel;
using HitSoftware.DBMoto.Application;
...
public void Run(){
    try
    {
        // Static singleton constructor
        dbmApp = DBMotoApplication.Instance;
        // Retrieving the local server from the list available in the
DBMotoApplication instance
        dbmServer = dbmApp.Servers["local"];
        // Connecting to the server agent using anonymous authentication
        dbmServer.Connect();
        // Define and load the current metadata
        currentMetadata = dbmServer.Metadata.DefaultMetadata;
        // Load and synchronize metadata and Data Replicator (TCP/IP)
        currentMetadata.Load(true);
        if (currentMetadata.IsRunning())
        {
            Console.WriteLine("Data replicator is running on " +
currentMetadata.Name);
            Console.WriteLine("Stopping now ...");
            // Stopping the Data Replicator after a timeout of 10000 milliseconds
            currentMetadata.Server.StopReplicationManager(10000);
        }
        else
        {
            Console.WriteLine("Data replicator is stopped");
            Console.WriteLine("Starting now (as an application) ...");
            // Starting the Data Replicator as an application
            currentMetadata.Server.StartReplicationManager(false);
        }
    }
}

```

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.

```
        catch(Exception e)
        {
            if (currentMetadata != null)
                currentMetadata.Unload();
            if (dbmServer != null)
                dbmServer.Disconnect();
            Console.WriteLine(e.Message);
        }
    }
    ...
}
```

The DBMotoApplication object has access to a list of IServer objects, which is the list of servers configured in the Management Center and stored in the local dbmoto.config file. DBMoto users typically define at least a "local" server in the Management Center. It is therefore possible to get a reference to the local server by calling:

```
dbmServer = dbmApp.Servers["local"];
```

With a pointer to the local server, you can connect to the server:

- Using anonymous authentication (if allowed by the server agent):  
`dbmServer.Connect();`
- Using DBMoto authentication (by passing an application login):  
`dbmServer.Connect(txtUser.Text.Trim(), txtPassword.Text.Trim(), false);`
- Using Windows authentication (the Windows domain credentials):  
`dbmServer.Connect(null, null, true);`

The IServer class also contains the list of metadata defined for the server, and identifies the default metadata, which will be used by the Data Replicator:

```
currentMetadata = dbmServer.Metadata.DefaultMetadata;
```

For interaction that involves the Data Replicator and the metadata, the metadata should be 'synchronized' with the Data Replicator, meaning that the application using the DBMoto API should open a TCP/IP connection to the Data Replicator and receive notification from the Data Replicator:

```
currentMetadata.IsSynchronized = true;
```

should be called before loading the metadata.

The IsSynchronized property also allows you to receive monitor information via the API including how many records have been processed and the current transaction ID in a running replication.

Use the IServer StartReplicationManager and StopReplicationManager methods to start and stop the Data Replicator.

The simple code sample described above provides an introduction to the DBMoto API. Explore the full capabilities of the DBMoto API from Visual Studio, or from the DBMoto API Reference Guide.

## AppConfig File References

Add the following references to libraries for the DBMoto API in your AppConfig file <runtime> section. Replace all items in italics with correct pathnames, cultures and version numbers. To get the correct version number for your DBMoto installation:

1. Open the DBMoto Management Center.
2. In the main menu bar, from the **Help** menu, choose **About DBMoto...**
3. Make a note of the exact DBMoto version number on the right-hand side of the dialog.
4. Click **OK** to close the dialog.

The options for the culture value are en, it or jp.

<runtime>

```

<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
  <dependentAssembly>
    <assemblyIdentity name="APILibrary" culture="neutral" publicKeyToken="ea88625a625249df"/>
    <codeBase version="7.0.0.0" href="FILE://G:/products/dbmoto/test/sut/APILibrary.dll"/>
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="DBUtil" culture="neutral" publicKeyToken="ea88625a625249df"/>
    <codeBase version="7.1.0.10" href="FILE://G:/products/dbmoto/test/sut/DBUtil.dll"/>
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="DBMetadata" culture="neutral" publicKeyToken="ea88625a625249df"/>
    <codeBase version="7.1.0.10" href="FILE://G:/products/dbmoto/test/sut/DBMetadata.dll"/>
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="DBConfig" culture="neutral" publicKeyToken="ea88625a625249df"/>
    <codeBase version="7.1.0.10" href="FILE://G:/products/dbmoto/test/sut/DBConfig.dll"/>
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="DBGate" culture="neutral" publicKeyToken="ea88625a625249df"/>
    <codeBase version="7.1.0.10" href="FILE://G:/products/dbmoto/test/sut/DBGate.dll"/>
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="DBUtil.resources" culture="en" publicKeyToken="ea88625a625249df"/>
    <codeBase version="7.1.0.10" href="FILE://G:/products/dbmoto/test/sut/en/DBUtil.resources.dll"/>
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="EMUtil.resources" culture="en" publicKeyToken="ea88625a625249df"/>
  </dependentAssembly>

```

```

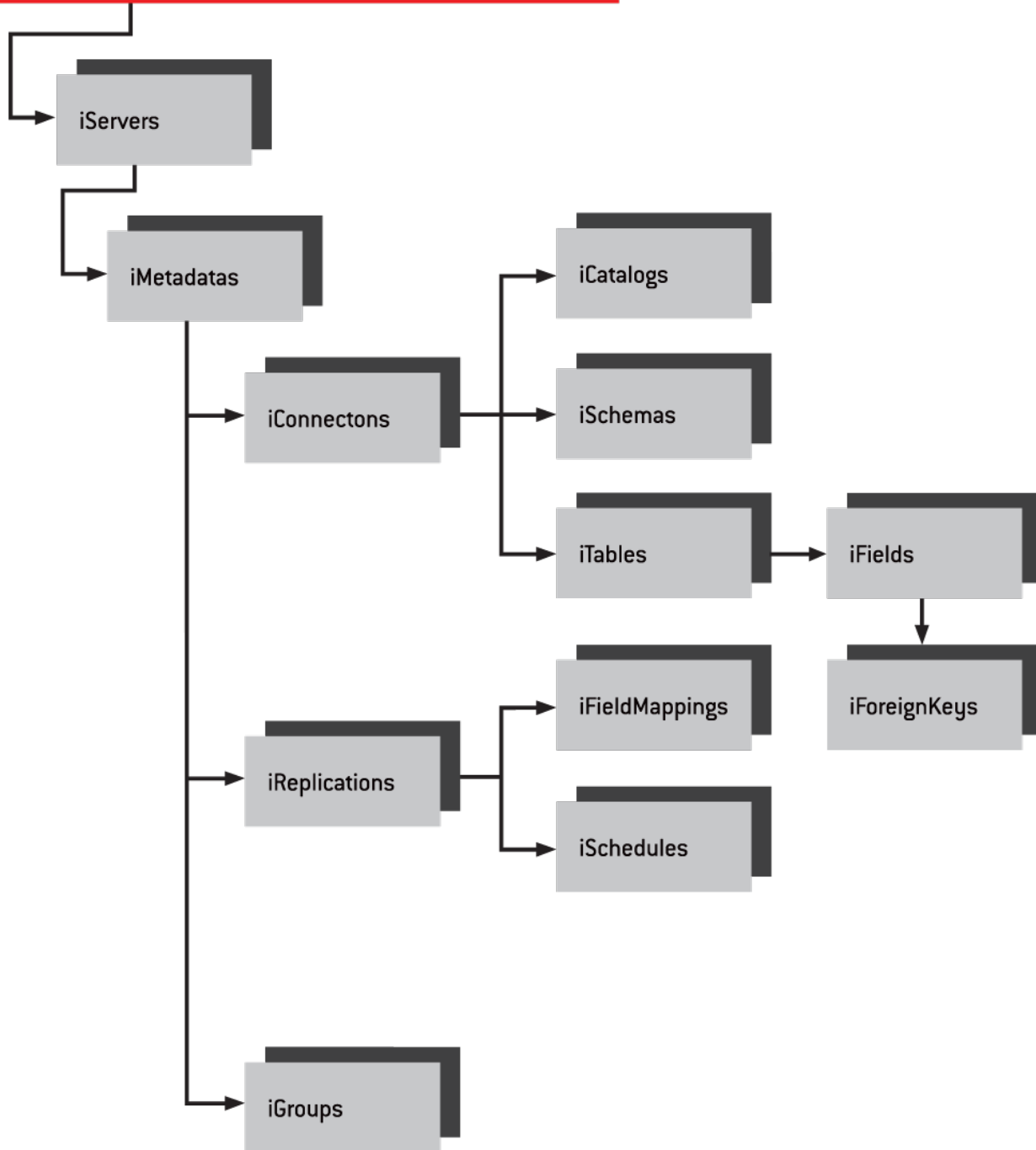
    <codeBase version="7.1.0.10" href="FILE://G:/products/dbmoto/test/sut/en/EMUtil.resources.dll"/>
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="EMUtil" culture="neutral" publicKeyToken="ea88625a625249df"/>
    <codeBase version="7.1.0.10" href="FILE://G:/products/dbmoto/test/sut/EMUtil.dll"/>
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="LogHandler" culture="neutral" publicKeyToken="ea88625a625249df"/>
    <codeBase version="7.1.0.10" href="FILE://G:/products/dbmoto/test/sut/LogHandler.dll"/>
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="DBMBusinessRules" culture="neutral" publicKeyToken="ea88625a625249df"/>
    <codeBase version="7.1.0.10" href="FILE://G:/products/dbmoto/test/sut/DBMBusinessRules.dll"/>
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="DBLogReader" culture="neutral" publicKeyToken="ea88625a625249df"/>
    <codeBase version="7.1.0.10" href="FILE://G:/products/dbmoto/test/sut/DBLogReader.dll"/>
  </dependentAssembly>
</assemblyBinding>
</runtime>

```

## DBMoto Object Model Diagram

The diagram below provides a graphical representation of the DBMoto Object Model. Note the one-to-many relations between the parent object and the children; and the one-to-one link from each child class to the parent (for instance, an IMetadata has a pointer to IServer).

## HiTSoftware.DBMoto.Application.DBMotoApplication





## Chapter 15: DBMoto Integration

### Installing DBMoto from a Windows Command Line

You can silently install HiT DBMoto from a Windows command prompt using the syntax described below. This approach enables you to run the setup without opening any of the install dialogs.

```
C:\Temp\setup.exe /s /z"LicenseKeyPath" /f1"AnswerFilePath"
```

where:

C:\Temp\setup.exe	The location of the DBMoto setup.exe file
/s	The option that runs setup without dialogs
/z"LicenseKeyPath"	The HiT DBMoto license file name (including the path) Example: /z"C:\Temp\dbmotolicense MYCOMPANY.txt"
/f1"AnswerFilePath"	The name of the answer file needed to run the silent installation (including the path) The answer file is distributed with the setup.exe file and differs depending on whether a new installation (setup.iss) or an upgrade installation (setupUpgrade.iss) is being performed. Examples: /f1"C:\Temp\setup.iss" for CLEAN INSTALLATION /f1"C:\Temp\setupUpgrade.iss" for UPGRADES

By default, after installation, a response file is created in the directory where the setup was run. The default response file name is SETUP.LOG.

Installation is successful when the response file contains the following:

```
[ResponseResult]
```

```
ResultCode=0
```

For any error that occurs during installation, the response file contains the following. If an error occurs, DBMoto is not installed.

```
[ResponseResult]
```

```
ResultCode=-3
```

To create the response file in a different folder or with a different name, use the option /f2 to specify a path and file name:



C:\Temp\setup.exe /s /z"LicenseKeyPath" /f1"AnswerFilePath" /f2"C:\setup.log

Last Updated on 4/24/2018

Copyright © 2018 HiT Software, Inc and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by HiT Software®. Other names appearing within the product manuals may be trademarks of their respective owners.