



White Paper

Data Replication from Progress RDBMS to MS SQL Server

Version 3.1

BravePoint, Inc.

www.BravePoint.com

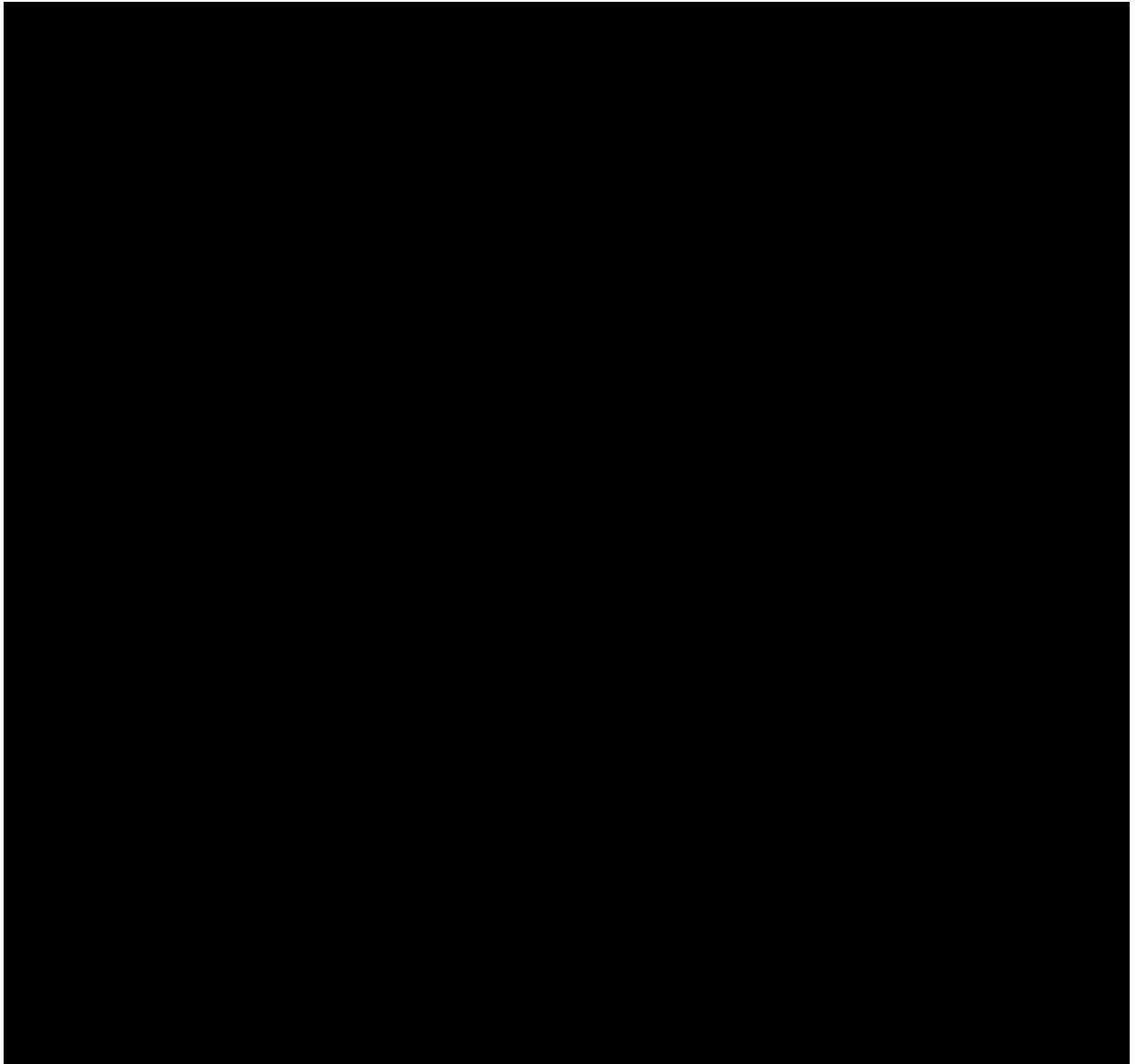
The Need for Replication

The Pro2SQL product was created to provide a lightweight, configurable utility for replicating data from a Progress OpenEdge database to Microsoft SQL Server. Replication requirements arise for a variety of reasons. Heterogeneous application integration, disaster recovery, reporting server, data archival, and business intelligence are just a sample of the business solutions that may require near-real time replication to a foreign database (in this case MS SQL Server).

Why is Pro2SQL lightweight? The entire architecture of Pro2SQL was designed with simplicity in mind. The product was designed using a small footprint with proven technology. The product is written using the OpenEdge Advanced Business Logic (ABL). The process also uses the OpenEdge DataServer as a conduit bridging the gap between OpenEdge fields and data types and the corresponding SQL Server fields and data types. The DataServer also includes some useful utilities enabling the two heterogeneous databases to “play nicely” with each other.

How is Pro2SQL configurable? Built into the design of Pro2SQL is the ability to replicate one or more OpenEdge databases to one or more SQL Server databases. You can duplicate entire databases identically or pick and choose the tables and fields where replication is desired. The source table and field names do not have to coincide with the target table and field names. It should be noted that automatic mapping will occur when table and field names match.

What is ‘near’ real time? Pro2SQL allows you to configure the time interval between the replication sweeping mechanisms. You can do it every “x” seconds, hourly, twice a day, or daily depending on your specific business problem. It is totally configurable by you and the timing interval can be changed in real time at your will. In addition, the entire replication process can be suspended within the administration tool temporarily. You may also discontinue specific table replication temporarily if needed. This may be done to delay overhead on your production system in a peak period such as end of quarter processing. Once the peak time has passed, simply restart the suspended replication. The only inconvenience in suspending or disrupting the replication process is that users of the SQL Server system will not have access to the records that have been updated in your OpenEdge database until the replication process is restarted and the replication queue is completely swept.



Replication Overview

The data replication process uses a single replication table consisting of minimal data and indexing and a single sequence for process control. No raw data is captured during database events, i.e. record Write and Delete. As create, update and delete events take place, the replication triggers fire and capture the updated record information to a Pro2SQL replication table (queue). By only capturing database, table, transaction, and event type information production (replication) overhead is greatly reduced. All replication data captured is maintained in a Pro2SQL database that contains 6 tables. A single batch process will cycle through the replication records periodically, based on user configuration, and replicate the table data directly to the MS SQL Server database via the OpenEdge DataServer.

Replication Triggers

Replication schema triggers are used with the production database. These triggers function as follows:

1. Create the replication record based on the database, table, and ROWID
2. Record the type of event, i.e. create, write, delete and the transaction number
3. Set the Applied flag to false (default)

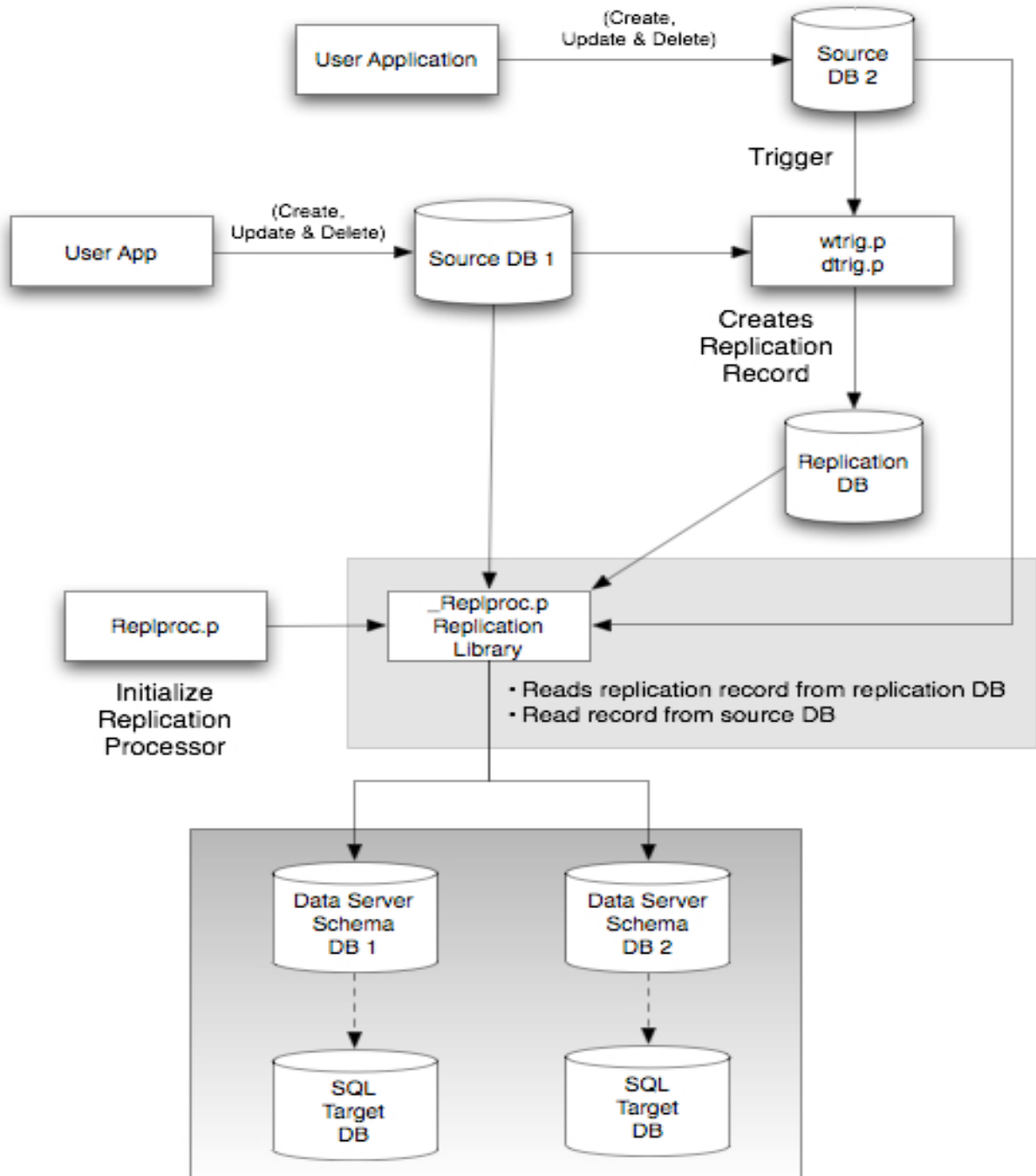
The batch process cycles through the replication records periodically based on the current value of the ReplControl setting. The current value of the setting represents the number of seconds to wait between cycles allowing the process to be tuned while it is running. Additionally, by setting the current value to 0 the processor can be stopped. The following is the basic functionality of the replication processor.

Replication Processor Flow

The Processor Library is a set of procedures used to handle the actual replication of individual records in a given source table over to their corresponding target records. These procedures are generated automatically during installation or when the user goes to the Administration Screen and selects Generate Code \diamond Processor Library. The application evaluates the fields and tables that have been selected for replication and creates logic that will execute the desired behavior. Part of this process is to create an individual "assignment" procedure for each replicated table. This solution enables for extensible functionality because users may add customized business logic to these assignment procedures to solve particular business problems. For example, if the source records need to be accumulated in some way prior to replication, this can be done by editing these assignment procedures. Obviously, care must be taken if the assignment procedures are edited. If not done correctly, you risk the possibility of breaking replication. In addition, all customized assignment procedures should be backed up in the case that the Pro2SQL product needs to be reinstalled or reinitialized.

Architecture

The basis of the replication functionality consists of five primary programs that generate the replication triggers, generate the replication processor functionality, processes the replication records, and a user interface program for monitoring and administering the replication processor and configuration. *Note that you will need at least 1 OpenEdge graphical development license of to run the Pro2SQL Administration Tool and compile the code.*



Replication Processor

The Replication Processor (RunReplProc.p) should be set up to run in batch mode via a scheduler program such as cron (UNIX) or some Windows equivalent. The processor is set to run perpetually unless the ReplControl sequence is set to 0 (zero). Setting the ReplControl sequence value can be done from the Administration Tool or from the Progress editor (using the CURRENT-VALUE statement). The value of ReplControl represents the sleep or pause time in seconds that the Replication Processor will allow between successive cycles through the unapplied replication queue records. *Note that it is recommended to keep a low ReplControl value increasing the probability of replicating records that are still physically in the database buffer cache. Doing this will provide more optimal performance for the replication process.* If set to (0) zero the Replication Processor will require restarting as this will cause the Progress session to exit. If the replication process needs to be “paused” for but not “halted” then an adequately large value could be applied to the ReplControl sequence to allow enough time between cycles as needed.

The Replication Processor writes to a log file that is located in the operating system directory specified by the LOG_DIRECTORY value in the properties table. Any errors encountered and general information such as cycle start and stop time and number of records processed will be captured here. This file should be reviewed periodically and will require periodic truncation/deletion or archiving.

Session Trigger Procedure Generator

The Session Trigger Procedure Generator will create REPLICATION-DELETE and REPLICATION-WRITE schema trigger procedures for each replication table. These procedures will be placed in the directories specified by the DELETE_TRIG_DIRECTORY and WRITE_TRIG_DIRECTORY values in the Properties table. The source database should be the only database connected to the session at the time this program is run.

Processor Procedure Generator

The Processor Procedure Generator creates a single procedure library that contains an internal procedure for each of the replication tables. These internal procedures encapsulate the replication logic specific to a specific table. The resulting procedure library is run persistently while the Replication Processor is running. If this procedure is run from the Progress editor all databases, i.e. source, schema holder, and target, must be connected. Additionally the following database aliases will need to be created:

- SourceDB for the source database
- SchemaDB for the Progress schema holder database
- TargetDB for the MSS target database.

Replication Administration User Interface

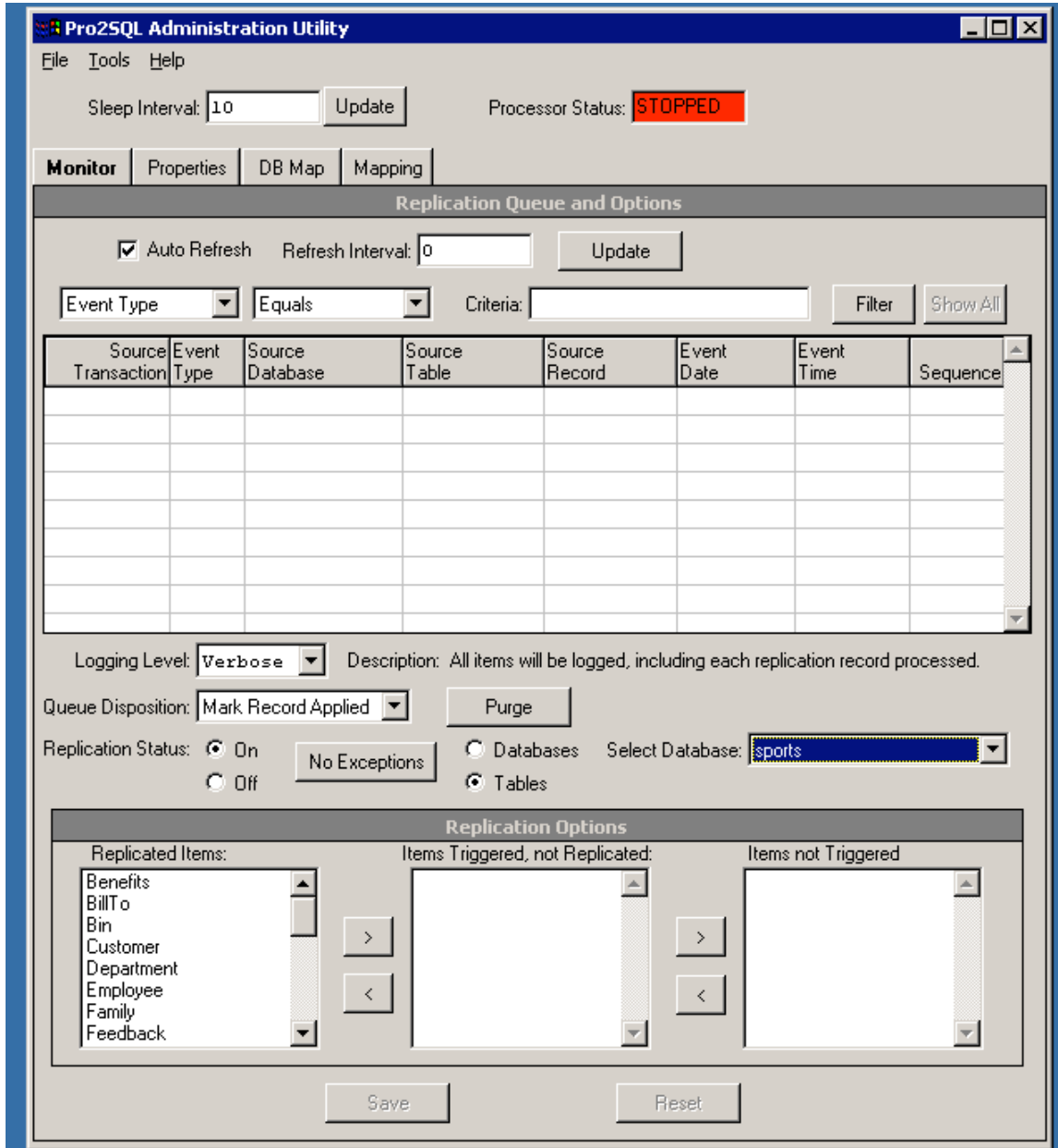


Figure 3 - Replication Administration Screen - Monitor Tab - with Exceptions expanded and the Purge Button in place.

Replication Administration User Interface

The functionality is divided between three tabs; Monitor, Properties, DB Tab and Mapping. Additionally, at the top of the screen the current processor state is displayed and the Processor Sleep Interval can be set.

The Sleep Interval value controls the number of seconds the Replication Processor pauses between cycles through the pending Replication queue records. Changing this value to 0 (zero) will cause the Replication Processor to end. In order to begin processing ReplQueue records again, simply restart the Replication batch process again.

The Processor Status fill-in shows the status of the Replication Processor as of the last check. The status will be “RUNNING” when the Replication Processor is either actively cycling through the queue or in a paused state between cycles. If the Processor is neither active nor sleeping, then the status will be “STOPPED.” An initial check is performed when the Administration program is first run. Thereafter the check is performed approximately every 3 seconds.

Replication Queue Browser

The browser shows the records in the Replication Queue that are pending processing. Records are initially displayed by sequence number in chronological order. This is also the order in which the Replication Processor will cycle through the records. Note that it is possible for the same records to be present in the browser through multiple cycles of the processor. This occurs when the transaction that created the Replication Queue record is still open during the replication cycle and is expected behavior. Replication will only occur on records that have been committed to the Progress database that are no longer in a locked state. The replication will attempt to retrieve the source table record using the ROWID to verify that the update has completed. If the replication process encounters a record in a locked state, meaning that the update is not yet released or that the source record is being updated again, that replication record will be skipped and processed during the next replication cycle.

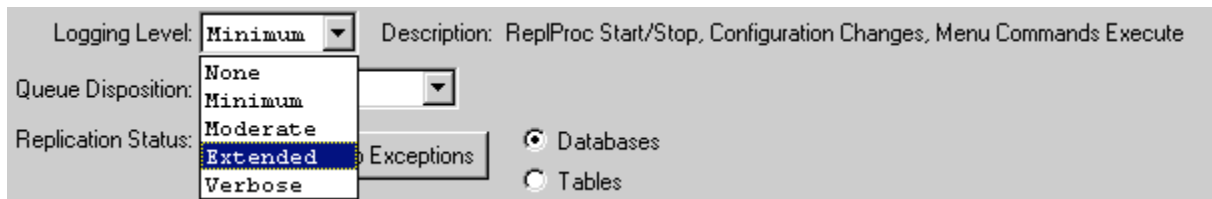


Figure 4 – Logging Levels

Summary

The Pro2SQL product was created to provide a cost effective way of replicating data from a Progress OpenEdge database to Microsoft SQL Server. Every organization typically has replication requirements. A sample of the business solutions that may require near-real time replication to a foreign database are application integration, disaster recovery, reporting servers, data archival, and business intelligence.

Using a proven technology and a simple, yet robust architecture Pro2SQL can fulfill your OpenEdge data replication needs. The graphical user interface is intuitive and controls everything from trigger generation to field mapping to customized program compilation based on your mapping needs. In addition, multiple database support is available at both the source as well as the target data sources. The timing of replication cycles is completely configurable and various levels of logging are supported.

If you are a Progress OpenEdge user struggling to solve business problems internally which require a replication solution, Pro2SQL may be the answer. It has a small footprint built using a proven technology platform with an architecture that can be implemented within weeks instead of months.

For more information contact BravePoint. View our Pro2SQL product pages which include a recorded demonstration at <http://www.BravePoint.com> .