

Documentation Addendum:

New Features in StarQuest Data Replicator 3.60

May 2010

© 2010 StarQuest Ventures, Inc. All rights reserved.

StarQuest Ventures, Inc.

PO Box 1076

Point Reyes Station, CA 94956

Telephone: 415-669-9619

FAX: 415-669-9639

Sales & Support: contact@starquest.com

URL: <http://www.starquest.com/>

What's New in this Release

This release of the StarQuest Data Replicator introduces the following new features and improvements:

- New and Enhanced DBMS Support:
 - Support for MySQL as a destination or a snapshot source
 - Support for Informix as a destination or a snapshot source
 - Support for SQL Server Native Client
 - Enhancements to Oracle support
- Configuration Enhancements:
 - Support configuration of locale used by the SQDR Service
 - Enhancements to the Columns Property dialog of a subscription:
 - Ability to synchronize subscription column information with the source schema without losing previous customizations
 - Ability to sort columns
 - Support multiple select/change on destination columns data types
 - Support macro-based derived column creation based on source columns
- Incremental Replication Configuration Enhancements:
 - Enhanced incremental subscription destination options ("Append replicated rows to existing data" and "Manual Synchronization")
 - New incremental subscription options ("Ignore Deletes", "Relaxed Apply Rules", and "Use Unique Constraints")
 - Ability to move an incremental subscription member between replication groups (requires SQDR Plus 3.60 or later)
 - Ability to copy an incremental subscription member to another replication group
 - Ability to specify a Commit Limit for Incremental Apply
- Subscription Processing Enhancements:
 - Ability to resume an interrupted baseline or snapshot
 - Enhanced recovery of orphaned subscriptions (ability to reset an incremental group)
- New administrative tools TargetChecker and CatalogTest

This document also contains the following documentation additions:

- Copy DDL-only replication considerations
- Incremental Subscriptions and Where Clauses
- Description of the ir_keylog control table
- Destination Column Names and Reserved Names

New and Enhanced DBMS Support:

Support for MySQL as a destination or a snapshot source

Refer to the release notes for system requirements and considerations.

Support for Informix as a destination or a snapshot source

Refer to the release notes for system requirements and considerations.

Support for SQL Server Native Client

The SQL Server Native Client ODBC driver can be used for the SQDR control database as well as for either a source or destination. It provides access to newer data types introduced in SQL Server 2005 and later.

Enhancements to Oracle support

When replicating from DB2 to Oracle, DATE fields are now mapped to Oracle DATE fields rather than CHAR(10).

Configuration Enhancements:

Support configuration of locale used by the SQDR Service

To specify the locale used by the SQDR Service, run the Data Replication Configuration application and proceed to the Service Type panel. This is useful when the native locale of the data that you are processing is different than the locale of the Windows machine on which the SQDR Service is running.

The screenshot shows a dialog box titled "Data Replicator Configuration - Service Type". The dialog contains the following elements:

- Instructions:**
 - Choose whether to start the Data Replicator service when the computer is started (Automatic startup type) or manually when the service is needed (Manual startup type).
 - Also specify whether the Data Replicator service should run as a System Account or log on under another Windows account.
 - Specify Service Locale. This sets the ANSI CodePage of the Replicator Service.
- Service Startup Type:** Two radio buttons: Manual and Automatic.
- Log On As:** A group box containing:
 - System Account
 - This Account: followed by a text input field.
 - Password: followed by a text input field.
 - Confirm Password: followed by a text input field.
- Service Locale:** A list box with the following items: English (South Africa), English (Trinidad), English (United Kingdom), and English (United States). The "English (United States)" item is selected and highlighted.
- Navigation:** Three buttons at the bottom: "< Back", "Next >", and "Cancel".

Figure 1

Columns Property dialog: Ability to synchronize subscription column information with the source schema without losing previous customizations

When creating a subscription, especially between unlike database systems, you may have performed extensive column mapping customization. This new feature allows you to preserve existing customizations if the table on the source system is altered (e.g. columns are added or dropped).

A new button "Merge with Source" has been added to the Columns Property dialog:

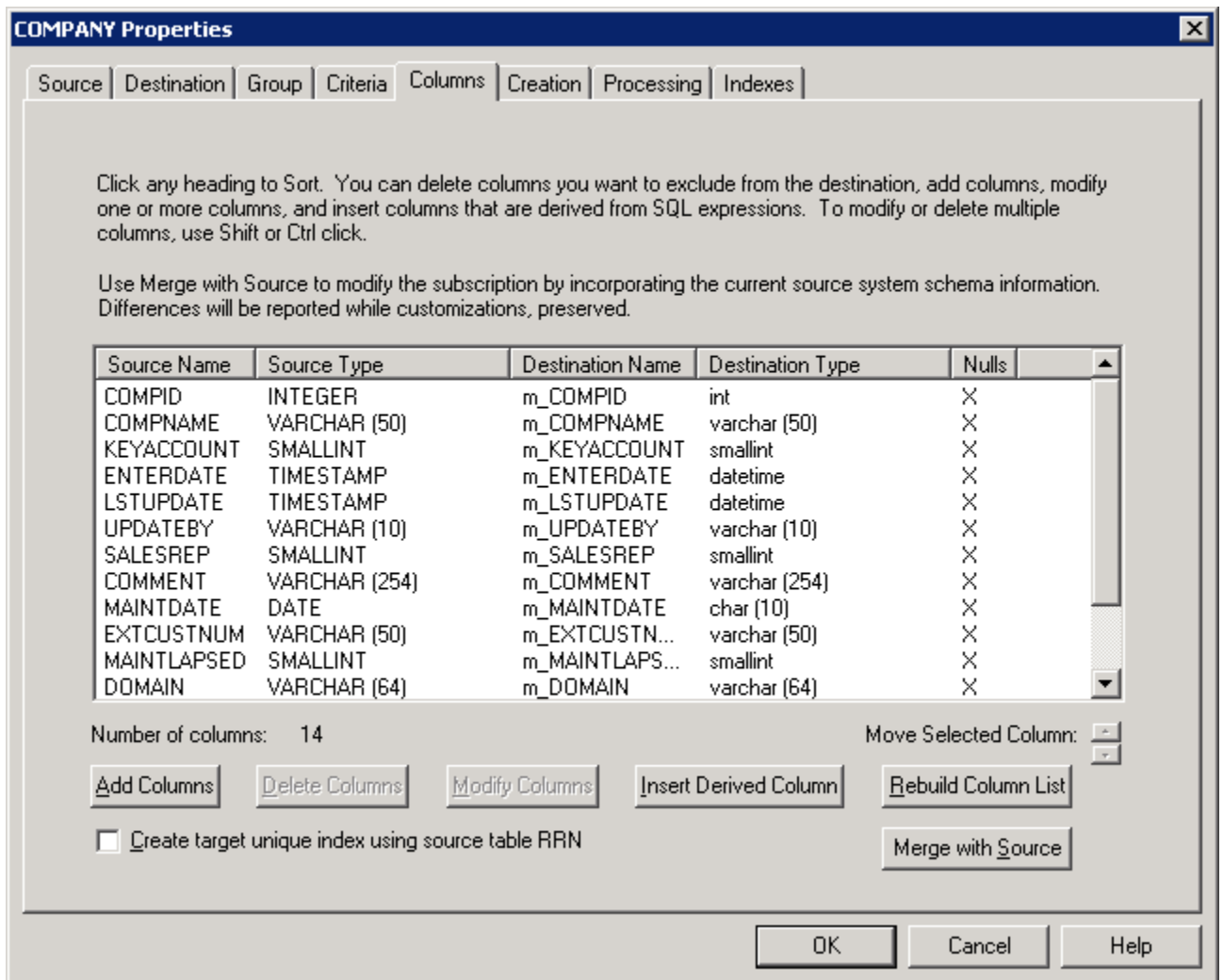


Figure 2

After pressing the "Merge with Source" button, you will see a dialog such as this:

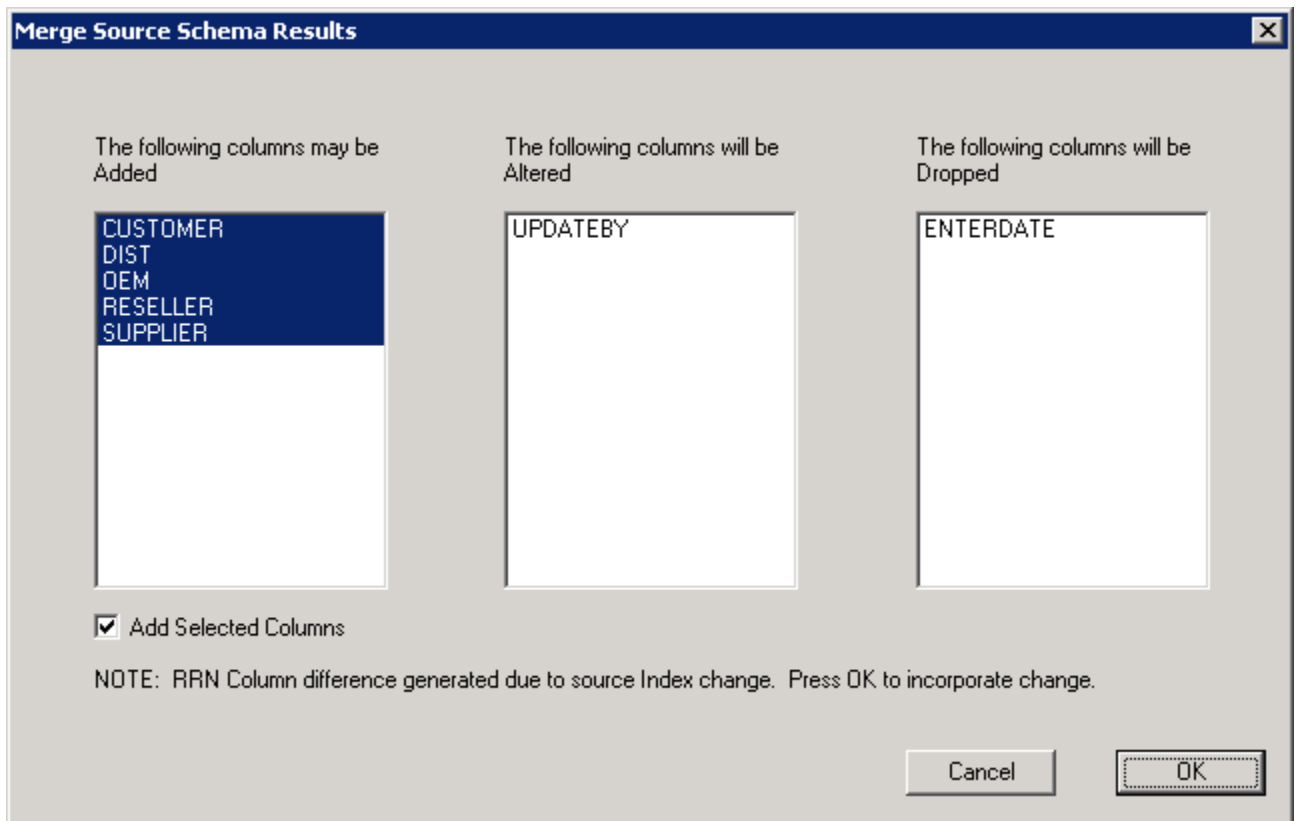


Figure 3

Note:

- The columns that may be added can be individually deselected, or the check box may be unchecked.
- The Altered columns are only those that have source column differences.
- The Columns to be dropped are those that were originally in the subscription but are no longer part of the source schema.
- The text message "RRN column differences..." only appears when this special column (which is not part of the source schema, but may be affected by source schema INDEX changes) has been altered.

If you press cancel, no action is taken.

Below are the results after clicking "OK". In our example, we now have 18 columns, where previously 14 were defined. Five new columns were added, and one existing column was dropped.

Also, note that the destination column names that start with "m_" have been preserved during the refresh operation, so you do not have to re-enter customizations for existing columns.

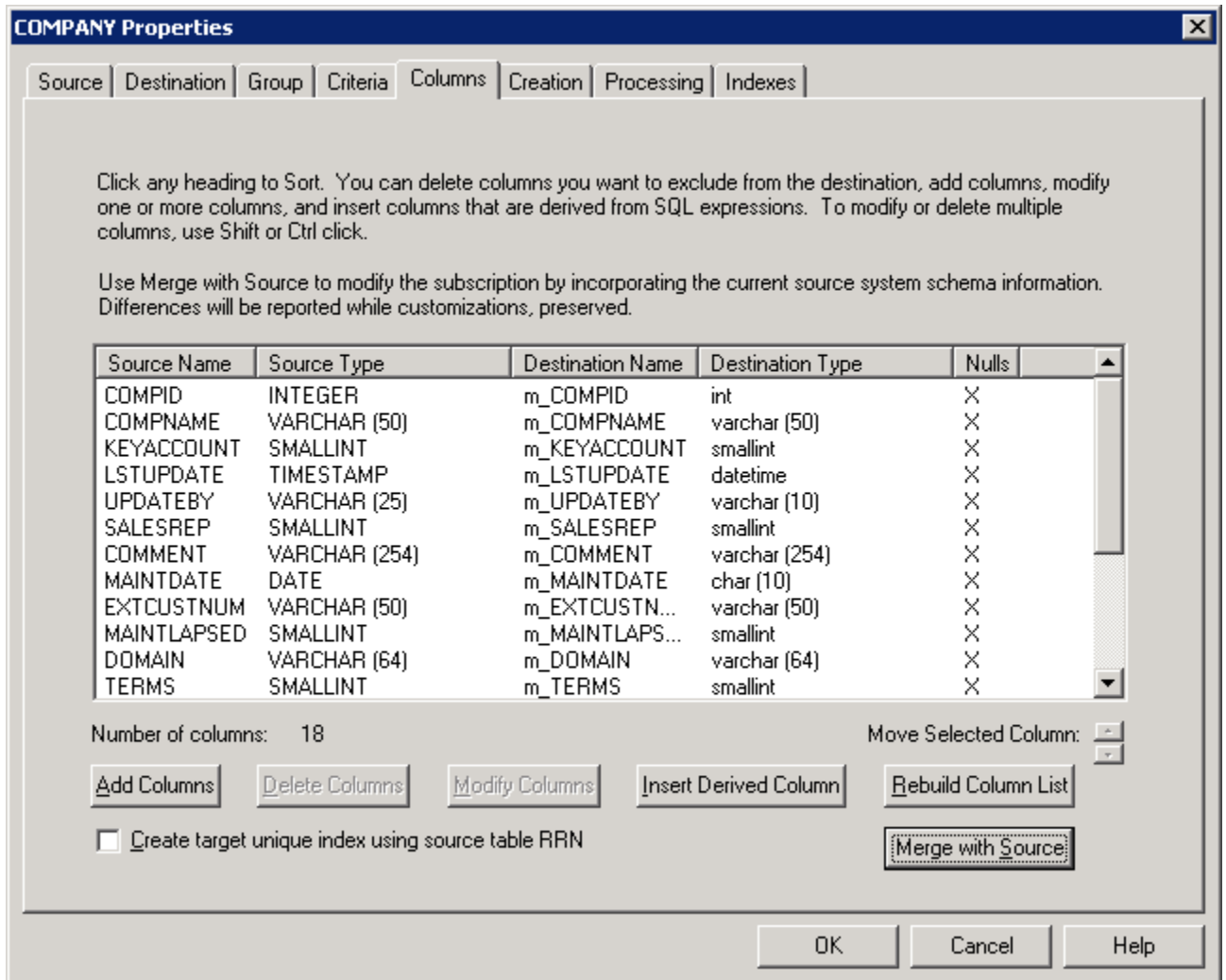


Figure 4

If the user presses "Merge with Source" button and no changes are detected, then the following dialog is presented:

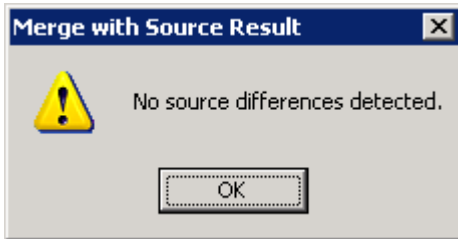


Figure 5

Columns Property dialog: Ability to sort columns

To view the Columns Property dialog of a subscription, select the subscription, right-click and choose **Properties**. Then select the Columns tab.

The column headers are now buttons. Clicking a heading button will sort the list in ascending sequence. Click the same heading again, and the list is resorted in a descending sequence.

Columns Property dialog: Support for multiple changes and Macro substitution

You may now select multiple columns for modification. Note that the ability to select multiple items to delete has been available in previous versions of SQDR.

Use the control or shift keys to select multiple list elements and click the Modify Column button.

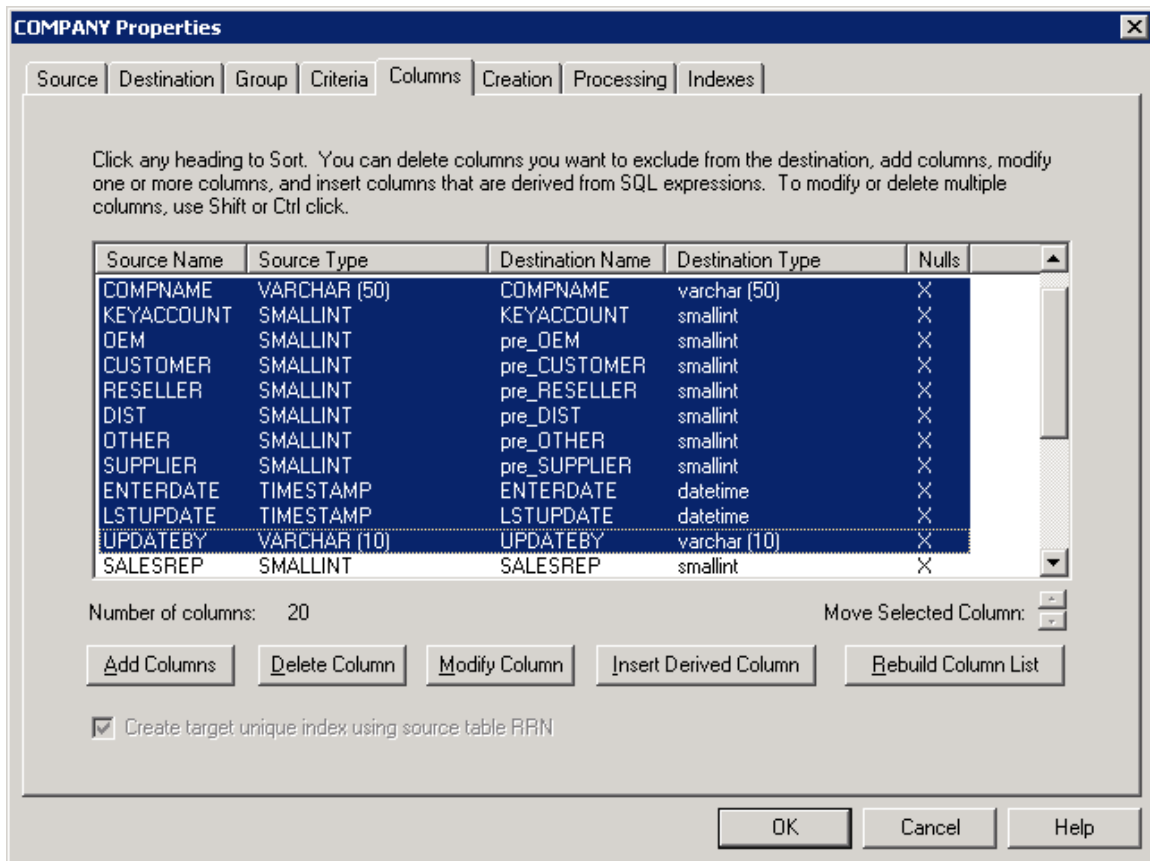


Figure 6

You will next see this dialog:

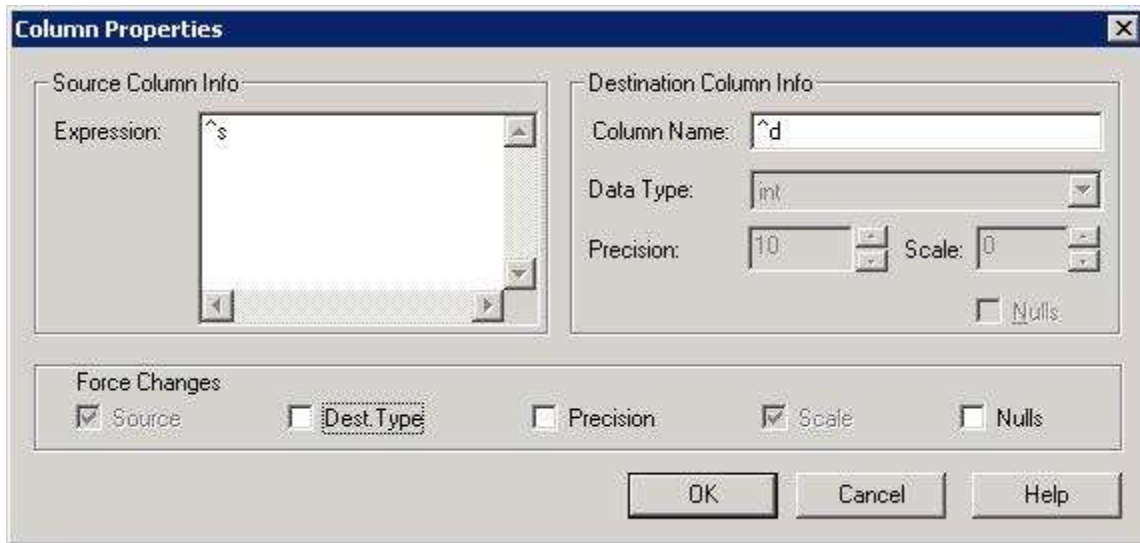


Figure 7

The source and destination column names are now showing special values of “^s” and “^d”, respectively. These are variables which will substitute the original “source” and “destination” names for each list element when OK is clicked. The variables may be combined with other text.

For example:

- “RTRIM(^s)” means specify a source expression using the scalar function RTRIM on the source column denoted by “^s” and apply the change to each row in the selection.
- “^d_01” will modify the destination name by adding a suffix of “_01” to the destination name.

In Figure 6, you can see the effect of a prior substitution, where the prefix string “pre_” was added to some of the destination names.

Columns Property dialog: Force Changes check boxes

Another change to the Columns Property dialog is the addition of a new set of check boxes under the caption "Force Changes". Up to five check boxes may be enabled, based upon whether or not the item's value is identical or dissimilar in all of the selections. If all the values are the same, then the user may enter a new value for that attribute without further confirmation. However, if the selections currently contain dissimilar values, then the user must first check the associated check box to enable data entry in the appropriate field. After this, any change made with "OK" will be applied to all selections, regardless of the original value. The five check boxes are "Source", "Dest.Type", "Precision", "Scale" and "Nulls". Some data types do not permit the direct specification of the Precision or Scale; in those cases the Data Type will force a value, whether or not the associated Force check box is checked.

This screen shot illustrates that the Data Type edit box became enabled for data entry after selecting the Dest. Type check box.

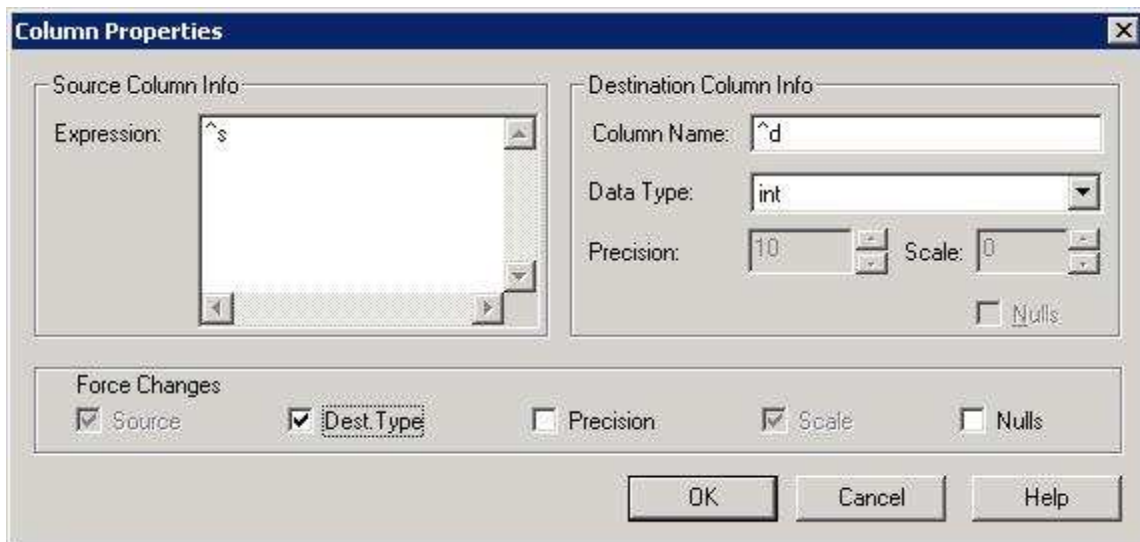


Figure 8

Incremental Replication Configuration Enhancements:

Enhanced incremental subscription destination options

The destination dialog for an incremental subscription contains the following new options:

Destination Option "Append replicated rows to existing data":

This option works in conjunction with the other Destination options and the Baseline Replication options.

Baseline Replication Option "Manual Synchronization":

This option signifies that no data copying or index replication activities will occur when the replication/baseline is run. Depending upon the other destination options selected, the run may still (re)create the destination table and/or delete/truncation destination rows and perform any specified pre-processing steps (post processing steps are ignored, if present).

To avoid ANY modification of the destination table, specify "use Existing" and "Append" destination options.

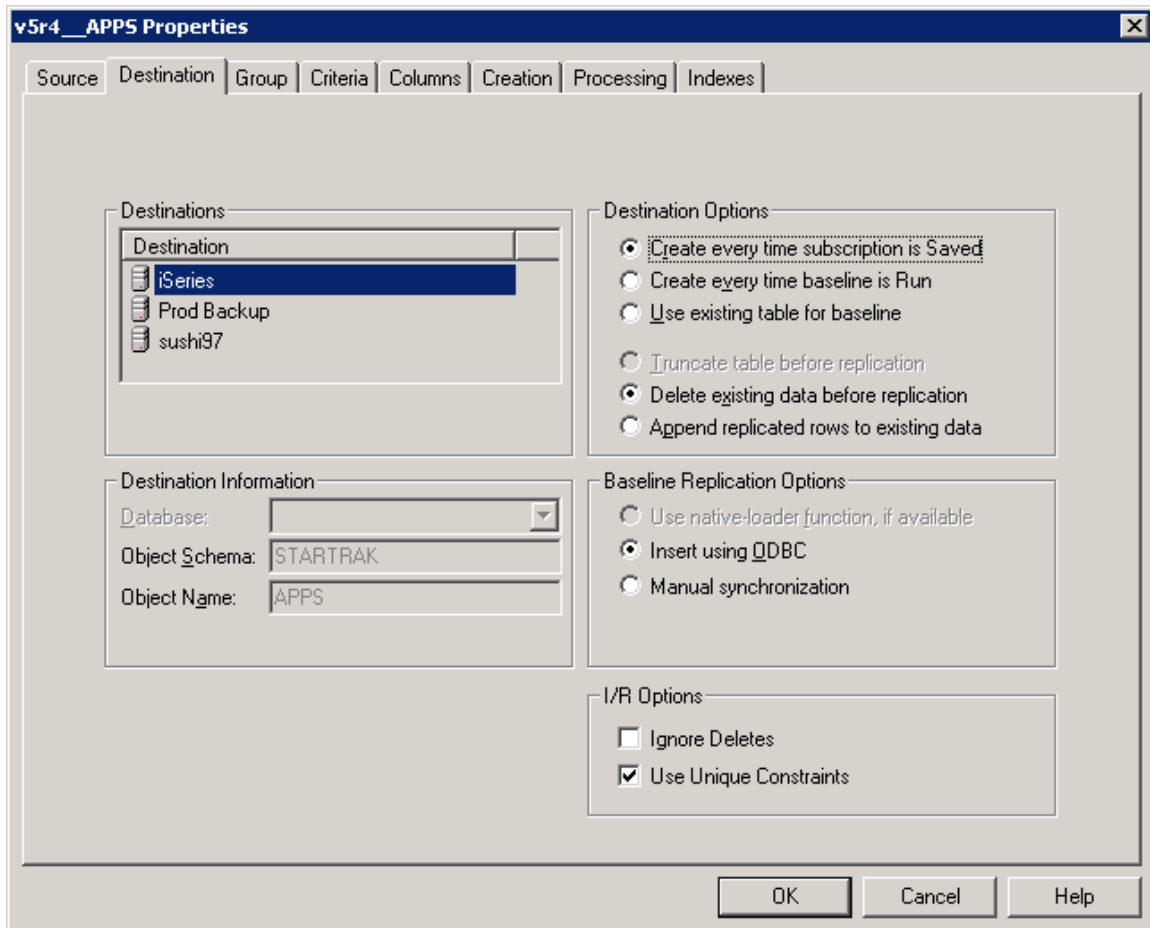


Figure 9

Considerations:

An incremental subscription's baseline will notify the Capture Agent on the SQDR Plus host that the position of the staging data has been updated to the moment that the subscription has been "run". It is incumbent upon the user to insure that the source and destination tables are in agreement at this time; otherwise change data be misapplied, leading to "row count" errors.

It is recommend that manual synchronization be undertaken under controlled circumstances, when the source data is not being modified, and no change data is being staged, and the containing group is paused. Upon completion of the manual operation to synchronize the destination table, the subscription must still be "run" to cause the Capture process to commence delivering changes. The containing I/R group may then be "resumed" to cause the subscription to become Active.

This new Baseline replication option is intended to provide an alternative method of utilizing incremental updates, without first using SQDR to establish a synchronized copy, when it may be impractical to use baselines. Care should be taken when using this option to insure that the initial conditions are met to resume incremental replication processing.

New incremental subscription options: "Ignore Deletes" and "Use Unique Constraints"

This screen shot illustrates the new default Subscription Options as specified on the Advanced property page of the containing incremental replication group:

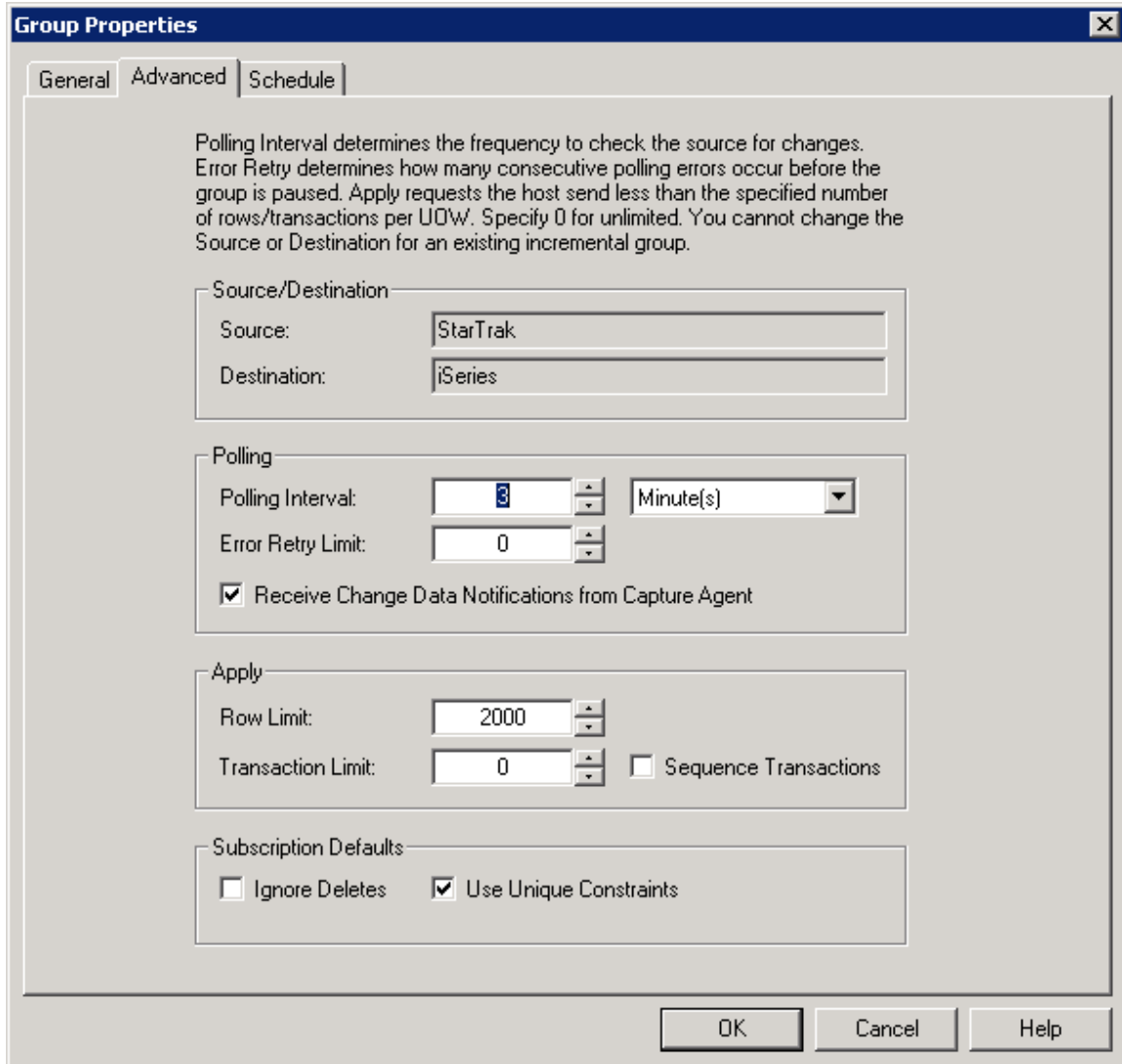


Figure 10

Two checkboxes have been added to the Subscription Defaults section. These are default values used when creating new subscriptions. The corresponding subscription values appear on the Destination page, and are only exposed for incremental subscriptions. Changing a group default has no effect upon existing subscriptions. The defaults are unchecked for all the options when a new incremental group is created.

NOTE: When copying a subscription from one group to another incremental group, the user may either elect to use the "To Group" defaults or the "From Subscription"-specific values for the newly copied subscription(s). The default behavior is to copy the existing subscription's values.

The Options are as follow:

- "Ignore Delete" processing: This is passed as a configuration option to the Capture Application, and requests that the host NOT deliver "DELETE" operations for the Apply processing on the destination table. Changes that are associated with INSERT or UPDATE operations continue to be delivered for application on the destination table. This option is intended for users who wish to treat the destination as an "archive" in addition to maintaining current values. This scenario is common in Data Warehouse environments: for example if the source is a 24-month view of Sales Activity, the destination may be designed to maintain a superset of current data and historical data (e.g. five years of history). This option permits the purging of rows on the destination to be independent of the current activity.
- "Use Unique Constraints" causes incremental subscriptions to copy the unique constraints on the source to the destination as unique constraints. When disabled (the default) Unique Indexes are mapped to Non-unique indexes and other constraints (Primary Keys, Foreign Constraints) are not copied to the destination.

Note: when using Unique Constraints, "Relaxed Apply Rules" are in force and row count errors will not be flagged. INSERT operations that fail due to unique constraints on the destination will be treated as "UPDATE" operations. Source UPDATE operations which fail will be treated as "INSERT" operations on the destination. DELETE operations that affect zero rows on the destination will be silently ignored and will not be flagged as row count errors.

These options are displayed on the Destination page of the Subscription Wizard and Properties displays. Defaults for new subscriptions use the values specified at the group level. The equivalent group level defaults are considered "Apply" options on the Advanced property page of the incremental group.

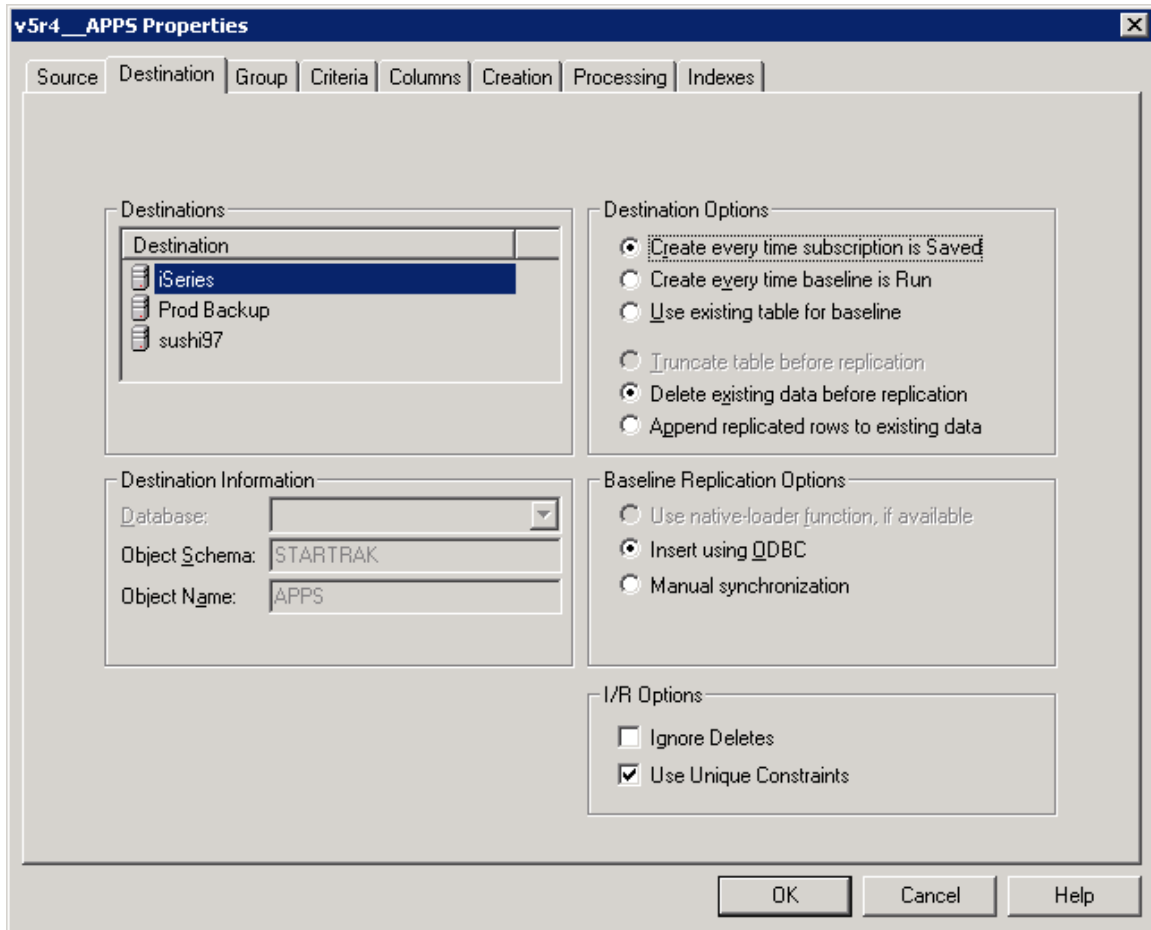


Figure 11

The values of the I/R options will use the Group Defaults if the "Insert Member" option was selected, else, no Group options are provided as a default. For an existing subscription, the values may be changed. Modifying the "Ignore Deletes" option will involve the subscription being dropped and re-added. Changing any I/R option will require a new baseline to be run.

Ability to move an incremental subscription member between replication groups

This feature requires that SQDR Plus 3.60 or later be installed on the database server.

You may move an incremental subscription between existing replication groups if the following criteria are met:

- SQDR Plus 3.60 or later is installed on the database server.
- Both groups use the same source and destination.
- Both groups are paused.
- The destination group must not already contain a subscription for the same source file.

To move a subscription:

1. Pause the TO and FROM groups with Pause Updates.
2. In the Data Replication Manager, display the members of the incremental group containing subscriptions that you wish to move.
3. Select the members (subscriptions) that you wish to move. Use the control or shift keys to select multiple members.
4. Right click and select Move Member from the drop down menu. If Move Member is disabled, then you need to pause the group.



Figure 12

You will see a list of eligible target groups (those that use the same source and destination, and are paused).

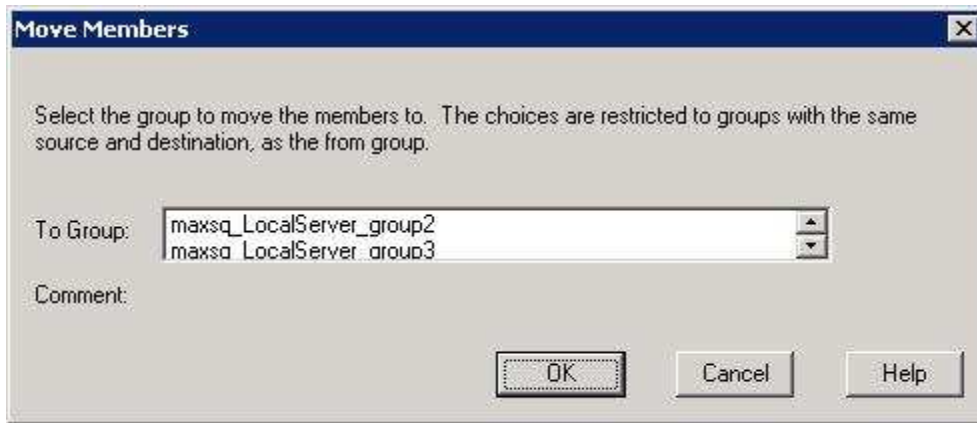


Figure 13

5. Choose the group that you wish to move the members to and select OK.

Ability to copy an incremental subscription member to another replication group

The ability to copy incremental subscription members allows you to set up multiple target locations using the same basic subscriptions for a given source.

You may copy an incremental subscription between existing replication groups if the following criteria are met:

- Both groups are using different SQDR Destinations. If you need to replicate to the same target database, create a new Destination.
- The destination group must not already contain a subscription for the same source file.
- Both the source and target groups must be paused.
- The Database and/or Object schema in the Advanced tab of the Source and Destination properties must be configured.

To copy a subscription:

1. Pause the TO and FROM groups with Pause Updates.
2. In the Data Replication Manager, display the members of the incremental group containing subscriptions that you wish to copy.
3. Select the members (subscriptions) that you wish to copy. Use the control or shift keys to select multiple members.
4. Right click and select Copy Member from the drop down menu. If Copy Member is disabled, then you need to pause the group.

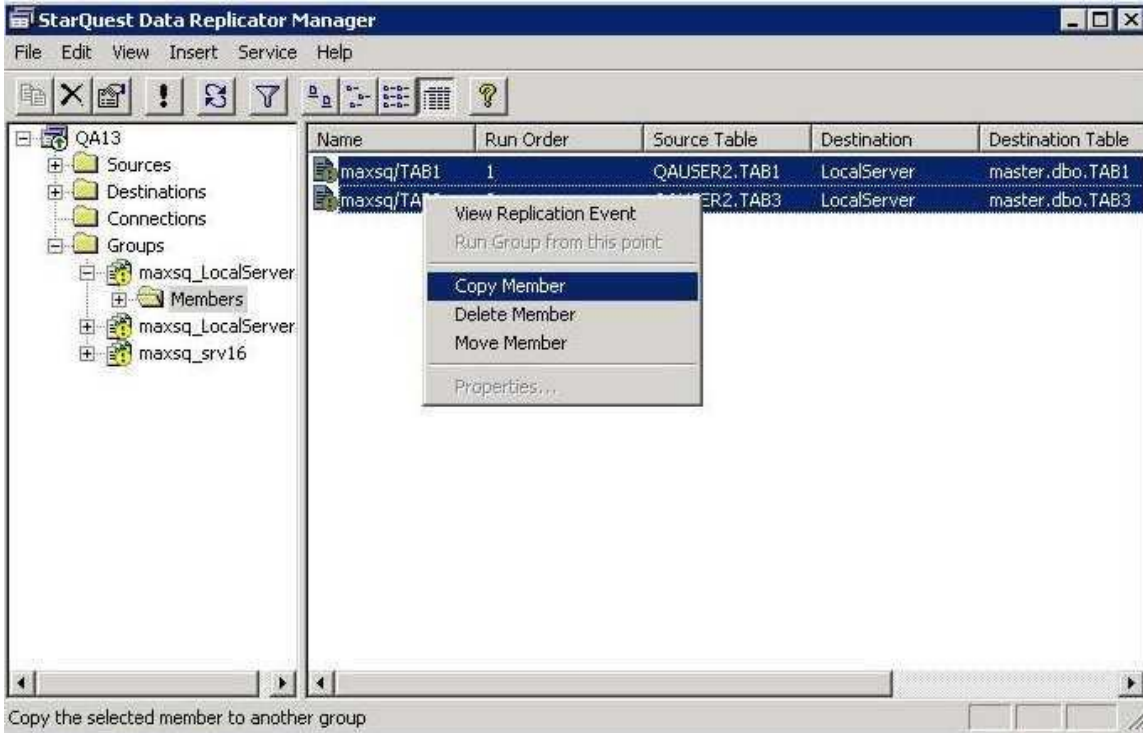


Figure 14

You will see a list of eligible target groups (those that use a different destination, and are paused).

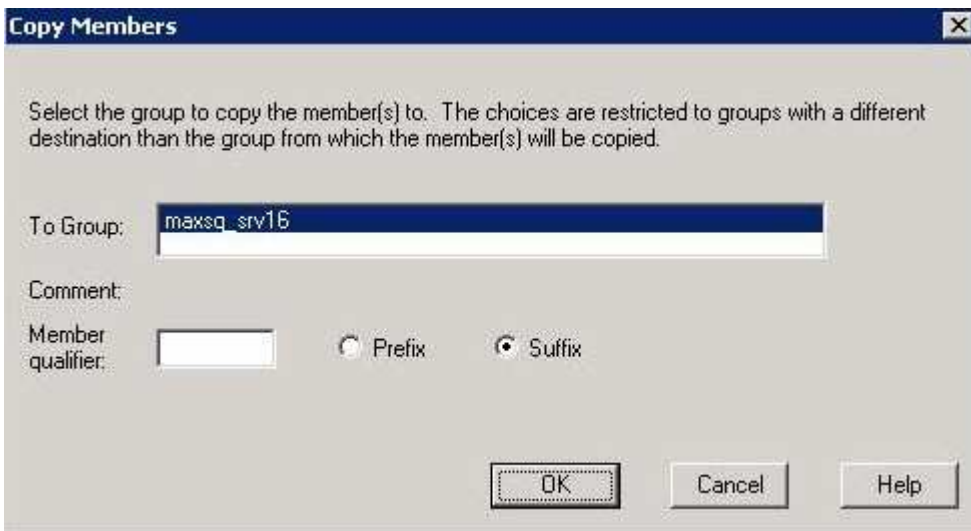


Figure 15

5. Choose the group that you wish to copy the members to and select OK.
6. Optionally provide a member qualifier to use as a prefix or suffix.
7. Click OK.

Because subscription names must be unique within a given source, this dialog allows you to specify a prefix and/or suffix member qualifier; the value is added to the name of the newly created subscription to insure uniqueness. If no value is provided, SQDR will name the subscriptions with automatically generated unique names of the form "TOGroupName_*subscription-name*" where "TOGroupName" is the name of the Group the subscriptions are copied to.

When attempting to copy a subscription to a group that already has a subscription of that name, that member will be skipped but copying will continue. After processing the remaining copy operations, a summary showing success and failures is displayed.

When an incremental member (subscription) is copied, the new member's qualifier (database) and owner (schema) values (for both source and destination) are updated to reflect the defaults in effect for the "TO-group's" source and destination. These values can be viewed and modified on the "Advanced" page of the definition of a source or destination. The values are evaluated at the time of the copy operation, and may be subsequently changed without effecting existing subscriptions.

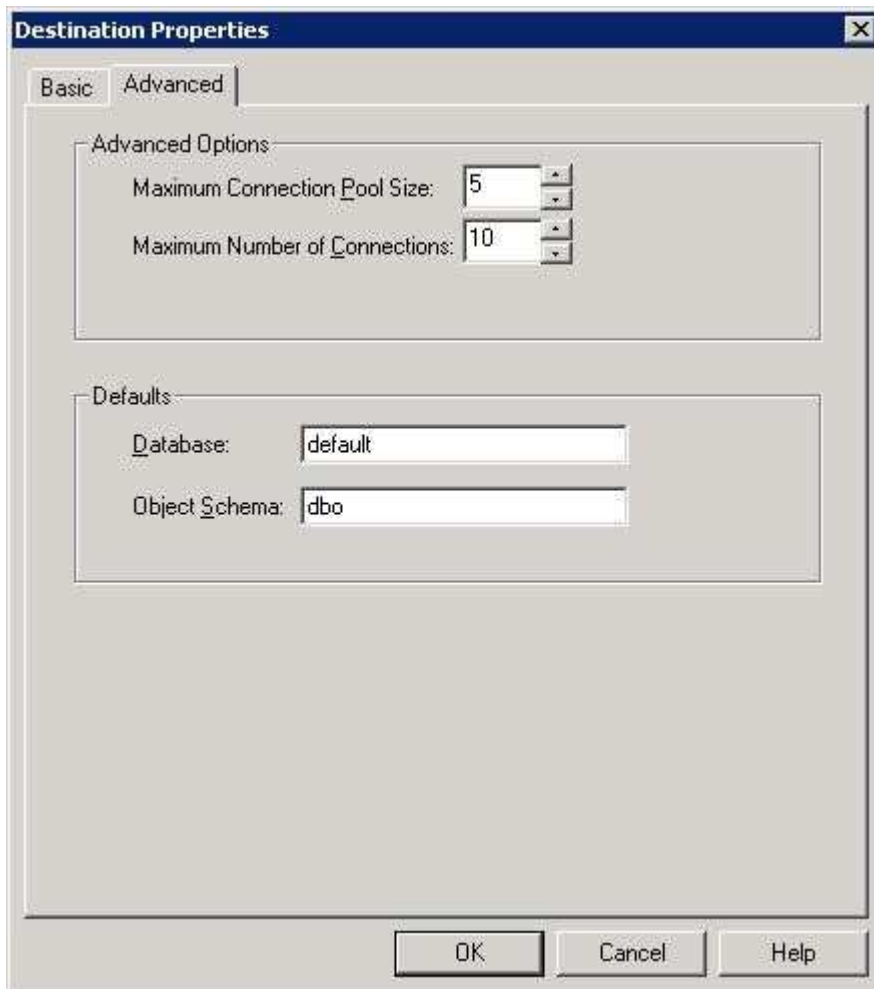


Figure 16

Caveats:

- This operation may be performed even when the groups use different source databases or different schemas, but this must be done with caution, as SQDR does not re-validate the schema on the source system.
- Subscription schedules are not copied. You may use the group scheduling instead.
- No verification is made when copying to a destination RDBMS of a different type.
- A subscription won't be copied if the table definition of the "FROM" source table does not match the "TO" source table in a different database and/or object schema.

Ability to Specify a Commit Limit for Incremental Apply

New options in the Advanced dialog for properties of an incremental group allow you to control the commitment limits (Row Limit and Transaction Limit) when applying changes to the target database in an incremental replication.

Background:

Incremental changes for incremental groups are returned as a consequence of calling the stored procedure "GetChanges" on the source system. GetChanges is designed to dynamically return multiple result sets per invocation, where each result set corresponds to the staged changes for a given table in the group. The maximum number of result sets that may be returned in a single invocation is determined by the source system. For i5/OS, the maximum is set to 24.

To increase the efficiency of the change data delivery, multiple source transactions may be grouped together within a given table's result set. The SQDR Service "applies" these transactions in the destination database as a single transaction. This approach reduces the overhead of obtaining changes from the source system but may end up transferring a very large number of rows for a given table, and potentially extending the time associated with completing any given transaction on the source system.

Transaction buffering may also result in a large amount of log space being used by the destination system. The V3.60 enhancement provides a mechanism for the user to manage this performance tradeoff. Under the "Apply" section of the Incremental Group Advanced Properties tab, there are two parameters to constrain the amount of buffering that may occur.

The Transaction Limit specifies the maximum number of source transactions that may be buffered. A value of 0 (the default) denotes an unlimited number of source transactions may be aggregated into a single destination transaction. A value of 1 specifies that a source transaction will be applied as a single destination transaction (i.e. no transaction buffering).

Since an unlimited number of transactions may result in a very large use of the destination database log resources, a second governing mechanism is also provided – the Row Limit (Commit Level) which works in conjunction with the Transaction Limit.

The Row Limit specifies a threshold of the total number of rows to be returned for a given invocation of GetChanges. Because at least one transaction must always be allowed, regardless of the number of source rows involved, the Row Limit value is honored on a "best-try" basis. The default value is 2,000; specifying "0" means "no row limit".

It is recommended that the defaults be used unless specific application requirements dictate otherwise (i.e. single transactions on the destination or a limited amount of log space.)

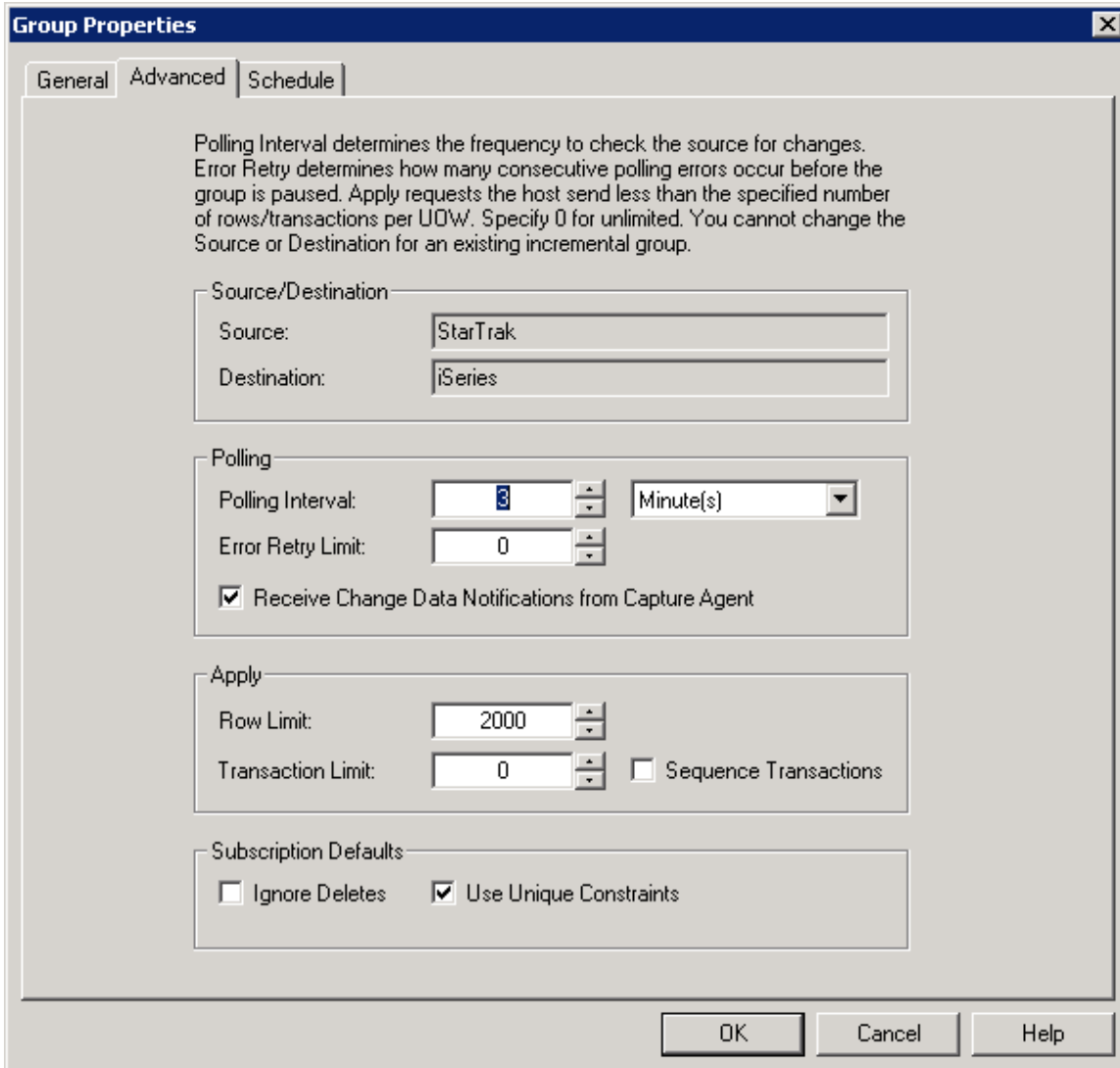


Figure 17

Subscription Processing Enhancements:

Ability to resume an interrupted baseline or snapshot

One advantage of using incremental subscriptions is most apparent with large database tables, since once the baseline has been run, only changes need to be transmitted. However, the execution of a baseline for a very large table may take a long time and can be subject to disruptions in the network or on the source or target system. This feature allows you to resume an interrupted baseline, appending new rows to the last committed row in the destination table, without having to start the baseline from the beginning.

This feature applies to both snapshot and incremental subscriptions.

The source system can be either an iSeries server or DB2 UDB 9.5 or later.

In the Global Service properties for the SQDR service, set the ODBC or BCP commit interval to a relatively low non-zero value, or use an ODBC commit interval value of one, which means "use autocommit" - each insert is checkpointed; this provides the most granular level of recovery at the expense of performance.

If a baseline fails to run, the icon associated with the subscription will be a yellow exclamation mark, indicating that a baseline is still required, and there will be an event with a red "X". If you view the details of the replication event, you may see an error such as:

```
Fetch failed at source.  
ODBC message: SQLSTATE 08S01, native error 0, [StarSQL][StarSQL  
ODBC Driver] Communications link failure.
```

When the baseline runs again (either started explicitly by selecting "Run Subscription", or started as a scheduled or "as-needed" triggered operation), the details of the replication event should contain an informational message:

```
Subscription restarted after <N> rows.
```

Viewing the properties of the subscription and saving it will reset the subscription and remove all checkpoints; the next baseline will run from the beginning.

Enhanced recovery of orphaned subscriptions

A new action related to an incremental group allows you to reset subscriptions on the database server without having to manually remove and re-add the incremental group and its subscriptions as described in step 8 of the Tech Note [SQV00PL004 Migrating SQDR Plus Subscriptions from a Test Environment to a Production Environment](#).

This is useful for recovery of orphaned subscriptions that have been accidentally deleted from the SQDR Plus control database on the database server, or when migrating subscriptions to a new database server.

1. Select the incremental group.
2. Choose **Reset I/R Group** from the File menu or right-click and Choose **Reset I/R Group** from the dropdown menu.
3. Answer Yes to the confirmation dialog.

Note that all member subscriptions will be flagged as needing to be re-baselined.

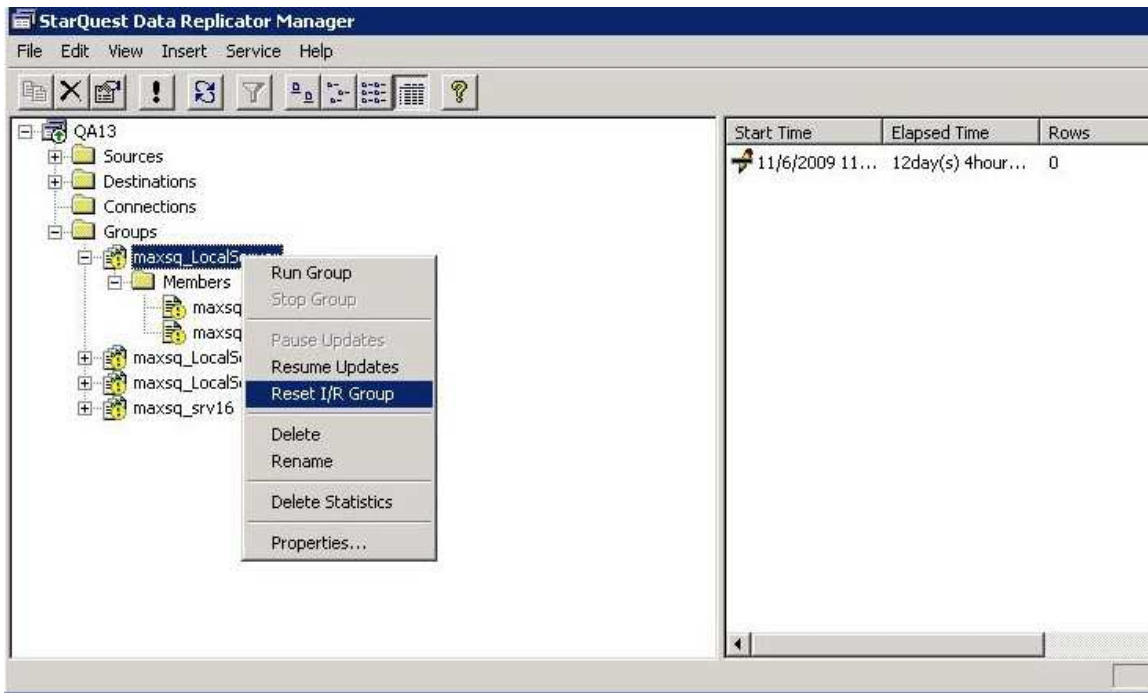


Figure 18

New administrative tools TargetChecker and CatalogTest:

Two new administrative tools are now included in the Tools subdirectory of the SQDR installation:

- TargetChecker compares row counts between the source and destination tables of an incremental group.
- CatalogTest verifies what SQDR Plus views as the index information, etc. of the Capture Agent's host tables.

These Java applications should be run on the SQL Server system where SQDR resides.

Make sure that this system has Java installed (from a command window enter "java -version). If not, download and install the Java run time from <http://www.java.com>.

TargetChecker:

1. Copy the contents of C:\Program Files\StarQuest\SQDR\Tools to a working directory e.g. C:\temp\Tools.
2. Open a "cmd" window (Start menu->Run->cmd)
3. CD to the tools directory:

C> cd "C:\temp\Tools"
4. Edit TargetChecker.bat, changing the appropriate values for DSN, user and password for the ODBC data source that connects to the source database system:

set DSN=MYDSN
set USER=MYUSER
set PWD=MYPWD
5. If you are using SQDR Plus for UDB, stop the Capture Agent.
6. Run TargetChecker.bat
7. You will be prompted for the name of the ODBC data source and user/password credentials to connect to the control database.
8. You will then be prompted to select the group to verify.

Results are written to a CSV file.

CatalogTest:

1. Copy the contents of C:\Program Files\StarQuest\SQDR\Tools to a working directory e.g. C:\temp\Tools.

2. Open a "cmd" window (start menu->run->cmd)

3. CD to the tools directory:

```
C> cd "C:\temp\Tools"
```

1. Edit CatalogTest.bat, changing the appropriate values for DSN, user, and password for the ODBC data source that connects to the source database system, and SCHEMA:

```
set DSN=MYDSN
set USER=MYUSER
set PWD=MYPWD
set SCHEMA=MYSHEMA
```

2. Run CatalogTest.bat

3. Enter Arguments for TableInfo (Schema Table Option):

This is a bitmap:

0 - tables (recommend using this)

8 - tables

Different versions

SQL wildcards or patterns

1 - primary keys

2 - unique indexes

6 - return all indexes

7 - get 2 result sets

one with primary keys

and one with all indexes

schemapattern tablepattern option

The output will contain results from calling the stored procedure TableInfo() (which is used by SQDR and SQDR Plus) and results from ODBC catalog calls.

Documentation Additions:

Copy DDL-only replication considerations

Enabling the Copy DDL Only option replicates only the Data Definition Language (DDL) CREATE statement that creates the object (e.g., view, alias, or synonym) on the destination database.

- This option is only available to snapshot replication subscriptions.
- If the Copy DDL Only option is enabled on the subscription Source panel, the DDL CREATE statement will be replicated to the destination either when the subscription is saved or every time the subscription is run.
- If the subscription replicates an object to a database with a different schema structure (i.e., different qualifier, schema, or table naming conventions), the default CREATE statement generated by the subscription wizard may not be valid. Modify the CREATE statement as needed while the subscription is being defined, as SQDR will not parse or attempt to validate the CREATE statement.

For example, the CREATE statement of a SQL Server view will typically refer to the object using a three-part naming format, as in [database].[owner].[object_name]. If the subscription replicates the object to a different type of database, such as DB2, the CREATE statement will fail because DB2 supports only a two-part naming convention (e.g., <library>.<object>).

- Copying DDL only between databases that use similar object naming conventions may also pose a problem if the dependent objects referred to in the CREATE statement are not in the same location on the destination database as on the source database.

For instance, a MySQL DDL CREATE statement may contain a reference to an object in the form of <qualifier>.<object>, where "qualifier" identifies the database name. A DB2 system, despite using a similar two-part naming convention, would interpret the "qualifier" as the library (schema) where the object exists, which may or may not be the correct location of the object. Modify the CREATE statement as needed to ensure that all referenced objects are qualified correctly.

- For best results, the CREATE statement should be carefully reviewed and modified accordingly to insure that the resulting statement is valid on the destination database and achieves the desired effect.

Incremental Subscriptions and Where Clauses

Using references to other tables to construct selection criteria for incremental subscriptions has two consequences:

- A View is created in the SQDR Schema referencing user tables on the source system. The views are created over the CT table (in SQDR) and any other user table that is referenced.
- Unintended loss of incremental replication fidelity may ensue.

For example if a source table for replication is being selected based upon criteria in another table (the "filter" table) and changes to the "filter" table result in existing rows in the source table either being newly "selected" in the view or "de-selected", these changes will not be detected in the logs (as no inserts/delete activities occurred on the source table). This effect may be mitigated to the extent that a Foreign Key (FK) relationship exists between the source table and the filter table - which specifies "CASCADE DELETES" - so that changes to the filter table will either cause deletes in the log to appear for the source table, OR, rows must only be inserted into the source table AFTER a valid FK exists. This approach is not guaranteed to insure fidelity, because changes are evaluated at the time that Apply requests data, not at the time the transactions occur, so under some circumstances staged rows might be omitted, when they should be included, and vice versa.

We recommend that you avoid this use of horizontal partitioning of source by using only static data for criteria – i.e., use a constant expression in a partition rule.

For SQDR Plus for iSeries users: you cannot specify "criteria" when configuring incremental subscriptions for tables whose journaling is configured with Record Images=*AFTER; *BOTH record images must be journaled in order to use criteria.

Ir_keylog Control Table

The ir_keylog control table can be used to drive downstream events instead of using triggers.

The ir_keylog file is a log of the communication between the host system (running SQDR Plus) and the client system (running SQDR) as it relates to the application of changes to the destination table (DDL and DDM.) The file is pruned based upon the number of statistics rows (a service level property.) The table refers to other SQDR Control Tables: subscriptions (identifying the source, destination ODBC DSNs, the source and destination tables), groups - identifying the containing "Group" and the statistics table which summarizes baseline (snapshot) results as well as summary incremental counters.

For a given subscription, the ir_keylog table will contain the details of the row to be inserted, deleted, or updated. The before, after key fields illustrate the source ODBC type, the local program data type and destination ODBC types used to effect the transfer, in addition to the string representation of the value(s) used to identify the row. The resulting number of rows affected by the operation is identified in the result_row_count column. If an ODBC error is reported, the "result_error" column contains the text message associated with the error condition. In the event that an unexpected number of rows (usually something other than "1") are affected by the operation, a subscription is "flagged". The "flagged_count" tracks the number of operations since the last baseline that resulted in an unexpected row_count_error.

The table has the following columns, where the associated text describes the source of the information (i.e. "SQDR Plus" denotes data supplied by the Host, "SQDR", data supplied by the client, etc.):

"COLUMN_NAME"	SQDR: FK: table.column relationships or attribute meaning
"run_id"	SQDR: FK: refers to statistic.run_id
"group_id"	SQDR: FK: refers to groups.id; statistic.group_id; ir_subscription.group_id
"subscription_id"	SQDR: FK: refers to ir_subscription.subscription_id
"resync_state"	
"transaction_id"	SQDR Plus: An enumerator used by SQDR Plus to order transactions
"change_row_counter"	
"change_row_sequence"	SQDR Plus: An enumerator used to order rows within a transaction
"change_row_timestamp"	SQDR Plus: Time data was logged on i5/OS host DBMS (in seconds - host time) (or staged for UDB host)
"change_row_nanosecond"	SQDR Plus: Time data was logged on i5/OS host DBMS (fractional part - in nanoseconds)
"change_row_type"	SQDR Plus: see below
"action_taken"	SQDR: see below
"result_row_count"	Destination DBMS: Number of rows affected by action_taken-associated operation as returned by destination
"result_error"	Destination DBMS: Any ODBC error message associated with action_taken on the destination table
"flagged_count"	SQDR: Current count of number of rows flagged for this subscription
"before_key"	
"before_key_edited"	SQDR: for "Delete" and "Update" operations - used by WHERE CLAUSE

"after_key"	
"after_key_edited"	SQDR: for some Update operations (partitioned subscriptions and open-window updates)
"keylog_sort"	SQDR: An increasing sequence number within the ir_keylog file
"keylog_ts"	SQDR: ir_keylog datetime of insertion (client platform datetime)
"insert_ts"	SQDR: ir_keylog row insertion timestamp

The details of the change_row_type field:

opcode	
"B"	Delete row - Horizontal Partition
"D"	Delete row
"U"	Update row
"I"	Insert row
"R"	Table reorganized
"S"	Snapshot required
"T"	Truncate table
"A"	Source table altered
"X"	Source table dropped
"C"	Close window signal

The result of the change_row_type is described in the "action_taken" column. The codes used for "action_taken" are a subset of the "change_row_type". A special value of 0x00 means the change_row_type was ignored. Some change_row_type only change the state of the subscription and do not result in the changes on the destination table and are NULL.

The details of the action_taken field:

action_taken	
D	Delete row
U	Update row
I	Insert row
T	Truncate table
S	Snapshot required
A	Source table Altered
X	Source table dropped
C	Close window signal
#	"Stale" Close window token
0x00	change_row_type ignored
?	Unrecognized change_row_type

Destination Column Names and Reserved Names

Because database systems vary on what they consider reserved names, one must be careful when replicating tables with columns that have names that are reserved names for the destination system, but not the source system. You may be able to handle this issue with quoted identifiers, or by editing the subscription and changing the name of the destination column.

An error such as:

```
[Informix][Informix ODBC Driver][Informix]DDL operations on "rowid" prohibited.
```

Or

```
ORA-00904: "ROWID": invalid character
```

may be the result of using a column name in the source system that happens to be a naming function on the destination system, and where the destination database does not support quoted identifiers to override the default naming. In the case of Informix or Oracle, one can use a different case for the destination column name to distinguish the name.