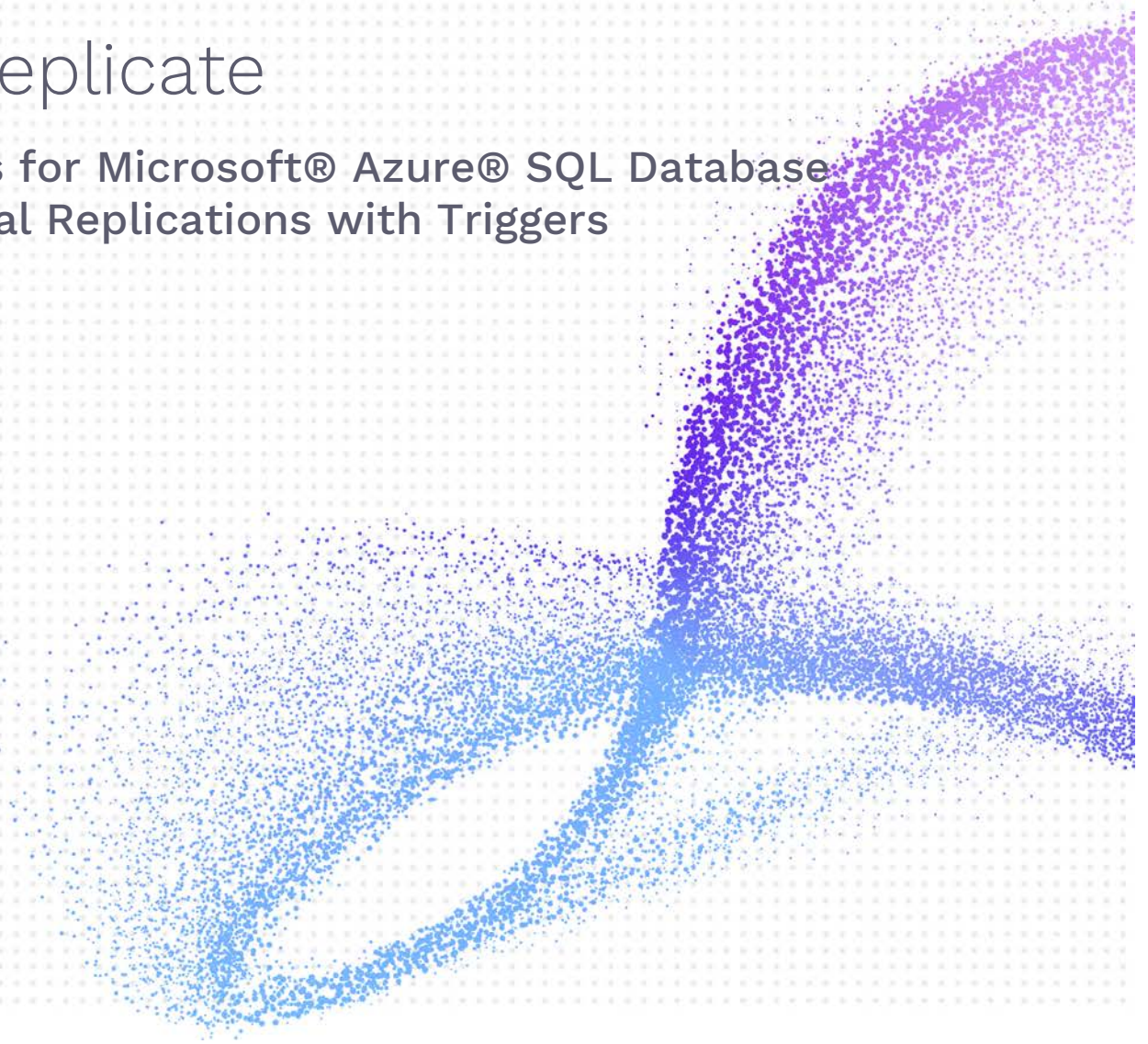
Syniti

Syniti Replicate

Setup Notes for Microsoft® Azure® SQL Database Transactional Replications with Triggers

Version 10.2



Syniti

Syniti Replicate

Table of Contents

Syniti Replicate

These notes provide essential information for setting up replications using **Microsoft Azure SQL Database** cloud database service.

This guide describes the setup process using the Triggers option for one-way mirroring and synchronization when replicating data from Azure SQL Database.

For mirroring and synchronization replications using Azure SQL Database as a source, Syniti Replicate offers:

- **Triggers**: Uses triggers installed on the Azure SQL Database to log changes.

Connection Type

Microsoft SQL Server .NET Data Provider (included with .NET Framework)

No value needed in the Assembly field when configuring the connection.

Trigger-Based Replication Overview

A database trigger is code that is automatically executed in response to certain events on a database table. To define a trigger-based replication (mirroring or synchronization), 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, Syniti Replicate 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 Syniti Replicate. 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 Syniti Replicate 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 Syniti Replicate does not create a tablespace. If you want to have a table space named SYNITIDR, you must create it beforehand using a SQL tool. Run the appropriate SQL statement for your environment. For example:

CREATE TABLESPACE SYNITIDR

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. Syniti Replicate may report errors about locked tables

during replication. Although Syniti Replicate 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.

A transaction ID is saved both on the master records and log record to maintain a link between the transaction and the data changes for that transaction. See “Syniti Replicate Log Tables” for more details on the structure of the Master and Log tables.

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.

Syniti Replicate Log Tables

Log tables are used to record all data changes made to the source tables. They are populated by triggers that are fired when source tables are modified. Currently, log tables must reside in the replication source database. Note that log tables are created by Syniti Replicate only if they do not already exist in the database.

There are two log tables associated with each replication: a master table, common to all replications using that connection, and a log table, one for each replication. The master table keeps track of all the transactions affecting the source tables and it records general transactional information.

Master table structure

- `TID DECIMAL(31,0) GENERATED BY DEFAULT AS IDENTITY`:
  Transaction ID number associated with each record data change (transaction)

- `SNAME VARCHAR(128)`:
  Name of the schema the transaction was applied to.

- `TNAME VARCHAR(128)`:
  Name of the table the transaction was applied to.

- `TTS TIMESTAMP`:
  Transaction timestamp indicating when the transaction was submitted to the system

- `TUSER VARCHAR(128)`:
  Name of the user who executed the transaction

Log Table Structure

The Log table contains the actual data changes for a specific source table. Its structure depends on the structure of the source table.

- `__TID DECIMAL(31,0):`

  Transaction ID, link to the corresponding record in the master table

- `__OP CHAR:`

  Indicates the type of operation:

  'I' INSERT
  'D' DELETE
  'B' UPDATE: values before the update
  'A' UPDATE: values after the update

- `<Field list>:`

  All the columns of the source table with their original data type. For example if the source table was created as:

```
CREATE TABLE SOURCET
(ID INTEGER,
 FNAME VARCHAR(30),
 LNAME VARCHAR(50))
```

  The log table will have the following structure:

```
CREATE TABLE _DBM__SOURCET
(__TID DECIMAL(31,0),
 __OP  CHAR,
 ID    INTEGER,
 FNAME VARCHAR(30),
 LNAME VARCHAR(50))
```

Reading From/Managing Log Table

Syniti Replicate reads the log tables using the SELECT SQL statement. First, it queries the master table to see if new transactions came in since the last processed __TID. If transactions are found, Syniti Replicate queries the corresponding log table to collect the data changes and apply them to the target table.

The SELECTs on the master and log tables are sorted by the unique column __TID which ensures that all records will be read in the order that they were written. Syniti Replicate also uses the unique __TID column to keep track of the point where the last record was read and processed from the log tables.

SYNITI REPLICATE provides options to manage log records that have been read and replicated. They can be deleted from the log table as soon as they are processed or a retention time can be set to leave this record in the log tables for the specified number of hours.

Microsoft Azure SQL User Permissions

When setting up replications that use Azure SQL Database as either a source or target database, you need to be sure that the user ID used for making connections to the database has sufficient privileges to complete all the operations required for Syniti Replicate to perform a replication.

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

Refresh with Azure SQL as Either Source or Target Database

1. AUTHORITY TO CONNECT TO DATABASE
   This should already be granted when the user is created. However, here is the syntax, just in case:
   ```
   grant connect to <uid>;
   ```
   Example where sdruser is the user ID:
   ```
   grant connect to sdruser;
   ```

2. AUTHORITY TO SELECT CATALOG
   To display a list of tables and show fields in the table in the Management Center (for selecting a source or target table and for setting which fields to replicate), Syniti Replicate runs a SELECT command on the catalog. If the user ID has insufficient privileges, an error is generated on the Azure SQL system.
   The command below allows read access to any table in the database, including catalog information. It is actually broader than the necessary permission, since this step requires access only to the system tables.
   ```
   use <database>;
   exec sp_addrolemember 'db_datareader', '<uid>'
   ```
   Example where sdruser is the user ID:
   ```
   use test;
   exec sp_addrolemember 'db_datareader', 'sdruser'
   ```

3. AUTHORITY TO SELECT TABLES
   Syniti Replicate runs a SELECT statement to identify records to replicate. Therefore, the user ID used to make a connection must have adequate authority to run a SELECT statement for tables involved in replication.
   If you used the syntax in 2 above, you do not need to grant select authority to specific tables involved in replications because the above command covers all tables.

4. AUTHORITY TO CREATE TABLES
   To create a target table in the Management Center (as part of the Create Table Wizard), Syniti Replicate requires permissions to modify the database schema.
   ```
   use <database>;
   grant alter on schema::<schema> to <uid>;
   grant create table to <uid>;
   ```
   Example where sdruser is the user ID:
   ```
   use test;
   grant alter on schema::dbo to sdruser;
   grant create table to sdruser;
   ```

5. AUTHORITY TO UPDATE TABLES
   To create a target table in the Management Center (as part of the Create Table Wizard), Syniti Replicate requires write permissions to the database. The following command allows write access to any table in the database and is broader than the necessary permission. It is possible to grant more granular access to every single table in the database, if necessary, by changing permissions for a user.
   ```
   use <database>;
   exec sp_addrolemember 'db_datawriter', '<uid>'
   ```
   Example where sdruser is the user ID:
   ```
   use test;
   exec sp_addrolemember 'db_datawriter', 'sdruser'
   ```

6. AUTHORITY TO DROP TABLES, ALTER TABLES (optional)
   The use of these commands from within Syniti Replicate is entirely optional (i.e. not necessary for running a refresh replication.) They are used if you choose to remove a table from Azure SQL or change the table via the Management Center SQL Query tab. The following commands are broader than needed. Alter and drop can also be granted to specific tables.
   ```
   use <database>;
   grant alter on schema::<schema> to <uid>;
   grant create table to <uid>;
   ```
   Example where sdruser is the user ID:
   ```
   use test;
   grant alter on schema::dbo to sdruser;
   grant create table to sdruser;
   ```

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

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

7. AUTHORITY TO CONNECT TO DATABASE
   This should already be granted when the user is created. However, here is the syntax, just in case:
   ```
   grant connect to <uid>;
   ```
   Example where sdruser is the user ID:
   ```
   grant connect to sdruser;
   ```

8. AUTHORITY TO SELECT CATALOG
   To display a list of tables and show fields in the table in the Management Center (for selecting a source or target table and for setting which fields to replicate), Syniti Replicate runs a SELECT command on the catalog. If the user ID has insufficient privileges, an error is generated in the Azure SQL environment.
   The command below allows read access to any table in the database, including catalog information. It is actually broader than the necessary permission, since this step requires access only to the system tables.
   ```
   use <database>;
   exec sp_addrolemember 'db_datareader', '<uid>'
   ```
   Example where sdruser is the user ID:
   ```
   use test;
   ```

```
exec sp_addrolemember 'db_datareader', 'sduser'
```

9. AUTHORITY TO SELECT TABLES
   Syniti Replicate runs a SELECT statement to identify records to replicate. Therefore, the user ID used to make a connection must have adequate authority to run a SELECT statement for tables involved in replication.

   If you used the syntax in 2 above, you do not need to grant select authority to specific tables involved in replications because the above command covers all tables.

10. AUTHORITY TO CREATE TABLES (Optional)
    To create a target table in the Management Center (as part of the Create Table Wizard), Syniti Replicate requires permissions to modify the database schema.
    ```
    use <database>;
    grant alter on schema::<schema> to <uid>;
    grant create table to <uid>;
    ```
    Example where sduser is the user ID:
    ```
    use test;
    grant alter on schema::dbo to sduser;
    grant create table to sduser;
    ```

11. AUTHORITY TO UPDATE TABLES (Optional)
    To create a target table in the Management Center (as part of the Create Table Wizard), Syniti Replicate requires write permissions to the database. The following command allows write access to any table in the database and is broader than the necessary permission. It is possible to grant more granular access to every single table in the database, if necessary, by changing permissions for a user.
    ```
    use <database>;
    exec sp_addrolemember 'db_datawriter', '<uid>'
    ```
    Example where sduser is the user ID:
    ```
    use test;
    exec sp_addrolemember 'db_datawriter', 'sduser'
    ```

12. AUTHORITY TO DROP TABLES, ALTER TABLES (Optional)
    The use of these commands from within Syniti Replicate is entirely optional (i.e. not necessary for running a refresh replication.) They are used if you choose to remove a table from Azure SQL or change the table via the Management Center SQL Query tab. The following commands are broader than needed. Alter and drop can also be granted to specific tables.
    ```
    use <database>;
    grant alter on schema::<schema> to <uid>;
    grant create table to <uid>;
    ```
    Example where sduser is the user ID:
    ```
    use test;
    grant alter on schema::dbo to sduser;
    grant create table to sduser;
    ```

13. AUTHORITY TO CREATE/DROP TRIGGERS

Add a Source Connection Wizard

Select Provider Screen

Database

Choose Microsoft Azure SQL Database

Provider

Microsoft .NET Driver (SQL Client)

Assembly

No value required.

Set Connection String Screen

Server

In the **Server** field, be sure to add the port number following the server name as in the following example:

```
tcp:hit.database.windows.net,1433
```

Database

Type in the name of the database.

Encrypt

Set to True.

Extended Properties

Set the properties Initial Catalog, Persist Security Info=False, MultipleActiveResultSets=False, and TrustServerCertificate=False, as in the example below:

```
Initial Catalog=sdruserazure;Persist Security
Info=False;MultipleActiveResultSets=False;TrustServerCertificate=False;
```

Enable Transactional Replication Wizard

For transactional replications (mirroring and synchronization), use the Enable Transactional Replication wizard after setting up a source connection.

Log Type Screen

Select the Triggers option.

Trigger Settings Screen

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 Syniti Replicate and the names are _DBM__LOG_#, where # is a number. The selected schema for the Master and Log tables must not contain other non-Syniti Replicate tables with names _DBM__LOG_#. It is highly recommended that you create a new schema to use specifically for the Syniti Replicate Master and Log tables.

Tablespace

It is highly recommended 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.

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. It is also possible to instruct Syniti Replicate to remove all the processed transactions at the end of each mirroring interval. Tuning the retention time provides control over the size of the log tables.

Delete Block Size

Based on the retention time, Syniti Replicate deletes items from the log. This field specifies the maximum number of records to delete from the Syniti Replicate log tables with a single SQL statement. The default value is 10,000 records. You do not typically need to edit this value.

Lower-case Trigger Identifiers

Check this option if your database installation uses lower-case trigger identifiers.

Trigger Order

Always inactive for AzureSQL Databases sources.