Introduction

The SQL Replication environment contains an example of how to implement the replication of a Synergy applications SDMS or RMS data to a SQL Server relational database in near-to-real-time.

The techniques demonstrated in this example are based in large part on code that is automatically generated using CodeGen. It is therefore a requirement that the data structures and files that are to be replicated, including key definitions, are accurately described in a Synergy repository.

Once the bulk of the code that is required to achieve the replication of data has been generated, the underlying application is modified by the addition of an I/O hooks object to any channels that are opened for update to files that are to be replicated. If your application already uses one or more subroutines to open its data files then those subroutines will be the only places that you need to alter, and the required change will typically be the addition of just a few lines of code to those routines. The code in the generated I/O hooks class detects and records changes to the ISAM and relative files that are being replicated.

Once this change information is being recorded a single process called the "replicator" is used to cause those changes to be mirrored to corresponding tables in the relational database.

There are several advantages to taking this kind of approach, some of the major ones being:

- You don't need to re-design your Synergy applications to store the actual application data in an SQL database. To do this properly would be a very major re-write of any application.
- You don't put the overhead of writing the data to both ISAM and SQL Server into your actual user applications, the performance overhead of which would be very significant.
- You don't make your user applications directly dependent on the database

being started. If the database, or replication server are not started then the transactions will simply build up in the log file until such time as they are started.

Following the code in this examples will mean that you can implement data replication with only minimal changes to the original Synergy applications. This is not a project that will happen overnight. In order to be successful with this type of project, the major requirements are:

- Each of the ISAM files that you wish to replicate to the database must have at least one unique key.
- In the extremely rare situation where a data file does not have any unique key then you will need to add a new field and unique key in order to replicate the data from that file to a relational database.
- You will need to add a couple of lines of code each time you open (for update) a data file that is being replicated. Typically this change takes place in one or more external subroutines that are already used to open channels.

License Agreement

All software included in this sample environment is distributed under the terms of the "Simplified BSD License" (also known as the "FreeBSD License").

License Agreement

Copyright (c) 2016, Synergex International, Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT. INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Change Log

Bulk Load via HTTP (2/15/2018)

We have removed the previous bulk load mechanism that used xfServer to upload data to the database server for subsequent BULK INSERT and have replaced it with a better and more flexible solution. The environment now includes a Synergy .NET project named FileService which is the code for a Windows Service that can be installed on the Windows database server host. The Windows service in turn hosts a Web API 2 RESTful web service that allows the delimited data files that are used during bulk load operations to be uploaded to the server via an HTTP post. This capability can be supported from any Synergy environment, including OpenVMS systems. The bilk load code that replicator uses has been updated to use this new HTTP mechanism instead of using xfServer on the database server system. If you wish to use this new bulk load mechanism then you must have a Synergy runtime (RUN10) license available on the database server system, in addition to the SQL Connection API (SCSQ10) license that is also required.

Replication of Relative Files (1/19/2018)

We introduced a mechanism for relative files to be replicated in addition to ISAM files. The code generated for relative files will include a "RowNumber" column that is populated with the associated record number from the underlying relative file. This row number is then used to synchronize the data in the file and table.

Initial Bulk Load Support (1/19/2018)

We introduced a new "bulk load" mechanism which in some circumstances can substantially reduce the time taken to load initial data into a table. The mechanism exports the data to a delimited file in the Synergy environment, then uses xfServer to copy the file to the database server where it is subsequently processed with a BULK INSERT statement. The sample EMPLOYEE file (which has 25,000 records) loads 20x faster using the new mechanism in my test environment (2 seconds vs 20 seconds). This mechanism requires an instance of xfServer to be running on the SQL Server system, and so can't be used when replicator is running on OpenVMS. This capability is intended to be a temporary

solution for use on Windows and UNIX / Linux until a better and more generic solution can be developed.

Requirements

Synergy Development Environment

This example environment is currently developed using Synergy/DE V10.3.3 on a Windows system and should work on any higher version without modification. Our policy is to develop with and take advantage of the features of the latest and greatest development tools; there is no guarantee that the code will work with older development tools, however the bulk of the code should be fairly compatible with older versions.

In addition to Windows, the software has also tested and implemented on Linux and OpenVMS systems.

Synergy License Requirements

All aspects of the SQL Replication environment are built using Synergy/DE, and therefor are subject to normal Synergy/DE licensing requirements. These requirements vary slightly depending on the platform that your Synergy application is hosted by.

License Requirements for a Windows Application Environment

In addition to your regular Windows Synergy/DE development licenses, you will also need:

- One Windows Synergy runtime license (RUN10) to support the operation of the replicator process on your Linux server.
- One Windows SQL Connection (SQL Server) license to allow connection to the SQL Server database.
- One (ideally two) Windows Synergy Runtime licenses to support code generation and the bulk upload web service.

License Requirements for a Linux Application Environment

In addition to your regular Linux Synergy/DE development licenses, you will also need:

- One Linux Synergy runtime license (RUN10) to support the operation of the replicator process on your Linux server.
- One Windows SQL Connection (SQL Server) license to allow connection to the SQL Server database.
- One (ideally two) Windows Synergy Runtime licenses to support code generation and the bulk upload web service.

License Requirements for an OpenVMS Application Environment

In addition to your regular OpenVMS Synergy/DE development licenses, you will also need:

- One OpenVMS Synergy runtime license (AXP-RUN10 or I64-RUN10) to support the operation of the replicator process on your OpenVMS server.
- One Windows SQL Connection for SQL Server license (SCSQ10) to allow connection to the SQL Server database.
- One (ideally two) Windows Synergy Runtime licenses (RUN10) to support code generation and the bulk upload web service.

We recently discovered that in Synergy versions up to and including 10.3.3d, I/O Hooks support was not functional on OpenVMS if the I/O hooks code was located in a shared image library. In order to configure

^{*} A Synergy/DE development environment (V10.3.3 or higher) on Windows

^{*} If you wish to use Visual Studio then you must have Visual Studio 2015 or higher) and have Synergy DBL Integration for Visual Studio installed.

a replication environment on OpenVMS using this sample code it is necessary to be running Synergy 10.3.3d and also have the support hotfix 37537 installed. Contact Synergex Support to obtain this hotfix.

Synergy Repository Database

CodeGen Code Generator

The majority of the code the is included in this example (and indeed the majority of the code that would be needed to implement replication in your own environment) is produced by CodeGen.

If you wish to regenerate the included code, or use the supplied templates to generate code for your own environment then you must have CodeGen 5.2.6 or higher installed. You can always download the latest version of CodeGen can always be downloaded from:

https://github.com/Synergex/CodeGen/releases

SQL Server Relational Database

This environment was originally created using Microsoft SQL Server 2012, and has been tested with SQL Server 2014, 2016 and 2017. The code should also work with later versions of SQL Server, and possibly some earlier versions.

Getting Started

Preparing Your Environment

Downloading the Code

Building the Code

You will find a batch file named setup.bat in the root folder. Execute it once to load the repository schema into a repository database and create various ISAM files required by the demo environment.

Setting Up Your Database Server

Creating a Database

A SQL Server 2012 or higher database and at least one Synergy/DE SQL Connection license available. If you wish to avoid editing the supplied example code then use a SQL Server database on the same computer that you intend to run the Synergy code on, and make sure that database is configured to accept Windows Authentication. If you do not use a local database then you will need change the value of the REPLICATOR_DATABASE environment variable that is defined in the project properties of the application.vpj project, or within the common properties of the Visual Studio environment.

Create a new SQL Server database and make sure that the account that you will be using to connect to the database server has access to the new database. If you wish to avoid editing the supplied example code then name the database "SqlReplicationIoHooks". If you do not use this default database name then you will need to change the value of the REPLICATOR_DATABASE environment variable that is defined in the project properties of the Workbench application.vpj project, or within the common properties of the Visual Studio environment.

Preparing the Database

The code in this example is configured for the following database configuration:

- Local default instance of a recent version of SQL Server (Express is OK)
- An empty database named "SqlReplicationIoHooks"
- Windows authentication based on your current windows login allows full access to the database (ability to create and drop tables, etc.)

If your SQL Server database is not local, is not a default instance (i.e.

it is a "named instance"), or does not accept Windows authentication, then you will need to change the SQL Connection "connect string" that is used to connect to the database.

Depending on how you start the replicator program, you need will to either change the value of the -database command line option, or change the value of the REPLICATOR_DATABASE environment variable.

If you are running replicator from the supplied Visual Studio project you will find that the REPLICATOR_DATABASE environment variable is set via the "common properties" tab in the Visual Studio projects; change the value and then restart Visual Studio.

If you are running replicator from the supplied Workbench projects you will find that REPLICATOR_DATABASE environment variable is set in the project properties of both the application and replicator projects.

If you are running replicator as a Windows Service using the supplied batch file RegisterReplicatorService.bat you will find that the - database command option is used.

Here are a few examples of valid SQL Connection connect strings for use with SQL Server databases.

| Database Instance via | | | Authentication Connect string | | |
|-----------------------|--------|-----|-------------------------------|-----------------------------|--|
| Local | Either | DSN | SQL Server Login | VTX12_SQLNATIVE:uid/pwd/dsn | |

Local Default Database SQL Server name Login VTX12_SQLNATIVE:uid/pwd/dbname/

Local Default Database Windows name Login VTX12_SQLNATIVE://dbname/.///Trus

Database SQL Server

Connect

| Local | Named | name | Login | VTX12_SQLNATIVE:uid/pwd/dbname/ |
|--------|---------|------------------|-----------------------|--|
| Local | Named | Database name | e Windows Login | VTX12_SQLNATIVE://dbname/.\\\insta |
| Remote | Either | DSN | SQL Server Login | net:uid/pwd/dsn@port:server_ip!VTX12 |
| Remote | Default | Database name | e SQL Server Login | net:uid/pwd/dbname/server_name///@pc |
| Remote | Default | Database name | e Windows Login | net://dbname/server_name///Trusted_cor |
| Remote | Named | Database name | e SQL Server Login | net:uid/pwd/dbname/server_name\\\insta |
| Remote | Named | Database name | e Windows Login | net://dbname/server_name\\\instance///Ti |

A local connection should be used when the SQL Server database is hosted on the same system that the replicator is running on. A remote connection should be used when the SQL Server database is located on a different system, and requires that the Synergy OpenNET server is running on the database server system.

The various parts of the connect string are replaced as follows:

| Value | Replaced with |
|------------|--|
| uid | Username of the SQL Server login to use. |
| pwd | Password of the SQL Server login to use. |
| dsn | Name of an ODBC datasource to use. |
| dbname | Name of the SQL Server database to connect to. |
| port | TCP/IP port number that the Synergy OpenNet server is listening on (usually 1958) on the remote database server. |
| server_nam | eName of the remote SQL Server (Window server name). |
| server_ip | The DNS name or TCP/IP address of the remote database server. |
| instance | Name of the SQL Server named instance. |

If you are using an ODBC DSN to connect to the database then you

should:

- Create the DSN wherever the database is located. For local databases the DSN should be defined on the local system. For remote databases the DSN should be created on the remote server system.
- For local databases, create the DSN to match the bit-size that you are building the replicator application with. If you are building replicator for x86 then create a 32-bit System DSN. If you are building replicator for x64 then create a 64-bit System DSN.
- For remote databases, create a DSN to match the bit size of the server that is hosting the database, and running the Synergy SQL OpenNet server. For 32-bit servers (rare), create a 32-bit System DSN on the server. For 64-bit servers (usual), create a 64-bit System DSN on the server.

You can find additional information about SQL Connection connect strings in the <u>SQL Connection documentation</u>.

Starting the Synergy OpenNET Server

Installing and Configuring the FileService Server

Configuring the Replicator Environment

The replicator process can be configured either via command line options or by setting environment variables. In all cases command line options override the equivalent environment variables.

Command Line Options

The following command line options can be used:

| Command Line Option | Description |
|--|---|
| -datadir <data_location></data_location> | The location of the replication instruction file (REPLICATOR.ISM) |
| <pre>-database <connect_string></connect_string></pre> | SQL connection connect string identifying the database to connect to. |
| <pre>-erroremail <email_address></email_address></pre> | Email address that start, error and stop messages should be sent TO. |
| <pre>-localexport <export_path></export_path></pre> | The location where bulk export files will be created locally. |
| <pre>-instance <instance_name></instance_name></pre> | The name of this replicator instance. |
| -interval | The number of seconds the replicator should sleep if it finds no |
| <sleep_seconds></sleep_seconds> | instructions to process. |
| -keyvalues | Record the key values being used to relate ISAM records to SQL rows. |
| -loaderrors | Log failing records during a bulk load operation to a file. |
| -logdir | The location where the log file should be created. A full or relative |
| <log_location></log_location> | path, or an environment variable followed by a colon. |
| <pre>-mailfrom <email_address></email_address></pre> | The email address that replicator messages should be sent FROM. |
| -mailserver | The DNS name or IP address of the SMTP mail server that will be used |
| <smtp_server></smtp_server> | to send messages. |
| -maxcolumns <max_columns></max_columns> | The maximum number of columns in a database table. Default is 254. |
| -maxcursors | The maximum number of database cursors. Allow 4 per table. Default |
| <max_cursors></max_cursors> | is 128. |
| | The URI of the instance of FileService that can be used to upload bulk |
| -remoteexport | load files to the remote database server (e.g. |
| <export_path></export_path> | http:// <server_name_or_ip>:8080/FileService). On OpenVMS you</server_name_or_ip> |

must specify the URL within double quotes.

-stoponerror Cause the replicator to stop if an error is encountered.

-syslog Log to the system log in addition to the log file.

-verbose Enable verbose logging.

Environment Variables

The following environment variables can be set:

| Environment Variable | Description |
|------------------------------------|--|
| REPLICATOR_DATA | The location of the replication instruction (REPLICATOR.ISM) |
| REPLICATOR_DATABASE | SQL connection database connectio identifying the SQL Server database to. |
| REPLICATOR_ERROR_EMAIL | The email address that start, error a messages should be sent TO. |
| REPLICATOR_EXPORT_LOCAL | The location where bulk export file created locally. |
| REPLICATOR_EXPORT_REMOTE | The URI of the instance of "FileSer can be used to upload bulk load file remote database server (e.g. http:// <server_name_or_ip>:8080/F On OpenVMS you must specify the within double quotes.</server_name_or_ip> |
| REPLICATOR_INSTANCE | The name of this replicator instance |
| REPLICATOR_INTERVAL | The number of seconds the replicate sleep if it finds no instructions to pr |
| REPLICATOR_LOG_KEYS | Set to YES to cause the key values to relate ISAM records to SQL rows |
| REPLICATOR_LOG_BULK_LOAD_EXCEPTION | S Set to YES to cause failing records bulk load operation to be logged to |
| REPLICATOR_LOGDIR | The location where the log file show created. A full or relative path, or an environment variable followed by a |
| REPLICATOR_EMAIL_SENDER | The email address that replicator meshould be sent FROM. |
| REPLICATOR_SMTP_SERVER | The DNS name or IP address of the mail server that will be used to send |

REPLICATOR_MAX_COLS

REPLICATOR_MAX_CURSORS

REPLICATOR_ERROR_STOP

REPLICATOR_SYSTEM_LOG

REPLICATOR_FULL_LOG

The maximum number of columns i database table. Default is 254.

The maximum number of database Allow 4 per table. Default is 128. Set to YES to cause the replicator to

error is encountered.

Set to YES to log to the system log to the log file.

Set to YES to cause more verbose lebe used.

Building the Sample Environment

Building the Sample Environment on Windows

Development Environment

This sample includes both Workbench and Visual Studio based development environments. If you wish to use use Workbench, you should start by opening the workspace file SQLReplicaitonIoHooks.vpw. If you wish to use Visual Studio, you should start by opening the solution file SqlReplicationIoHools.sln. Both of these environments contain several projects, as follows:

Library Project

Contains subroutines, functions and classes that are used both by the sample application, and by the replicator program. Note that the main code used to interact with ISAM files (EmployeeIO.dbl) and the relational database (EmployeeSqlIO.dbl) are in this library. These files, and others, were codegenerated by using CodeGen.

Replicator Project

Contains the replicator program as well as several utility programs.

One of the utility programs is named ReplicatorMenu and can be used to interact with and control a running replicator process.

Another of the utility programs is named EmployeeMaintenance and can be used to maintain the contents of the employee data file, which this environment is configured to replicate to the SQL database.

Building the Code in Workbench

- Start Workbench and open the workspace called SQLReplicaitonIoHooks.vpw via the "Project > Open Workspace..." menu option.
- Make sure you can see the "Projects" window. If it is not active then make

- it active, if it is not displayed then display it by selecting "View -> Toolbars" from the menu and checking the "Projects" option.
- Right-click on the library.vpj project and select "Set Active Project"
- From the main menu, select "Build > Build" to build the library project
- Right-click on the replicator.vpj project and select "Set Active Project"
- From the main menu select "Build > Build" to build the replicator program.

Building the Code in Visual Studio

- Start Visual Studio and open the solution called SQLReplicaitonIoHooks.sln.
- From the Build menu, select "Rebuild Solution".

Building the Sample Environment on UNIX or Linux

Refer to the various shell script files in the LINUX directory.

Building the Sample Environment on OpenVMS

Refer to the various shell script files in the VMS directory.

Basic steps to build on VMS:

- 1. Log into a privileged account (privilege is needed to start a detached process as SYSTEM)
- 2. Create an empty directory and copy the sample environment there.
- 3. Move to the VMS folder
- 4. Execute the BUILD.COM command procedure.
- 5. Edit REPLICATOR_SETUP.COM and check settings. In particular you will need to configure REPLICATOR_DATABASE based on the SQL Server database you intend to use.
- 6. Start replicator as a detached process by executing REPLICATOR_DETACH.COM
- 7. Or start the replicator as an interactive process by executing REPLICATOR_RUN.COM
- 8. Check the log file in the [.LOG] folder and ensure that replicator was able to connect to the database.
- 9. Run REPLICATOR_EXE:EMPLOYEEMAINANCE.EXE and use it to add instructions to the replicator queue.
- 9. Run REPLICATOR_EXE:REPLICATORMENU and test using the EMPLOYEE file.
- 10. Use the S command in REPLICATORMENU or run REPLICATOR_EXE:REPLICATORSSTOP.EXE to stop the replicator process.

Running the Sample Environment on Windows

In order to see the replication happening use SQL Server Management Studio to connect to the SqlReplicationIoHooks database and display the list of tables in the database - there aren't any at the moment. Run the replicator menu application by ensuring that replicator.vpj is the current project, opening the ReplicatorMenu source file, then selecting "Build > Execute" from the menu.

Start the replicator process by selecting the "SR" option; you will see it start in a different window. You should see some messages, including:

Next, pick the "M" option to start the employee maintenance program; again you will see it start in a new window. Enter employee number 1 to display that record, then select field number 1 and enter a new value for the employees fist name. Save the change by typing E to exit, then Y to confirm the change.

This will record an update operation in the replication servers transaction log. Within a few seconds the replicator process should pick up the change, realize there is an update to the database, and try to replicate the change. The first time this happens it will realize that the EMPLOYEE table doesn't exist in the database, so it should create the table, and then initiate a full load of the table from the ISAM file. You should see replicator messages similar to this:

First instruction checks for table EMPLOYEE

- Opening associated data file
- File opened
- Checking if table exists
- Table not found!

- Creating table
- Table created
- Bulk load starting at YYYY-MM-DD HH:MM:SS
- Bulk load complete at YYYY-MM-DD HH:MM:SS
- Adding indexes
- Indexes added
- Key 0 will be used to synchronize changes

Update row in table EMPLOYEE

- Key: 000001- Row updated

Check SQL Server Management studio, is the table and data there? If not then you probably got error messages from the replicator and need to debug the environment.

From now on, as you create, amend and delete employee records, those changes should be replicated to the EMPLOYEE table in SQL server. The example replicator goes to sleep for two seconds if there is nothing to do, so you should see any changes within that time frame. If you are sitting looking at the table in Management studio however, the table is not automatically refreshed, you you'll have to refresh it manually each time you want to see a change.

The replicator process would generally be run as a Windows service or detached process, and can be controlled by putting instructions into it's ISAM file. There are various options in the replicator menu to do this. For example, to stop the replicator, select the "S" option.

Running Replicator as a Windows Service

The replicator can be registered and started as a Windows Service, via the dbssvc.exe service runtime. An example of doing so can be round in the batch file RegisterReplicatorService.bat, and an example of un-registering the service can be found in UnregisterReplicatorService.bat.

Once registered the service may be stopped and started via the Windows Services application or via shell commands, e.g.:

net start SynergyReplicator

net stop SynergyReplicator

The connect string used in the example command assumes a SQL server database named SqlReplicationIoHooks running on the local system, and Windows Authentication is used to authenticate the user. The service that is registered by dbssvc will run under the context of the Local System account, so you must authorize that account to access the SQL Server database. To do this, use SQL Server Enterprise Manager:

- 1. Go to Secutiry / Logins
- 2. Right-click NT AUTHORITY\SYSTEM and select Properties
- 3. Go to "User Mapping" and check the Map check-box next to your database, and then check the "db_owner" role.
- 4. Click OK to save the change

You must also ensure that the EXE logical name is specified in a way that it is available to the replicator when it runs under the context of the Local System account. There are a couple of ways of doing that:

- Use the Windows System Properties dialog to set EXE as an actual system wide environment variable.
- Set the value in the [replicator] section of your synergy.ini file like this:

[replicator]EXE=C:\path\to\replicator\exe

IMPORTANT NOTE

Due to a bug (related to the maximum length of a windows command line) in the 10.3.3c and earlier versions of the dbssvc.exe runtime, if you wish to register replicator as a windows service in conjunction with using the command-line configuration options, you must be running Synergy 10.3.3c with the HotFix dated July 31st 2017 or later, or with a later version of Synergy/DE. You can obtain the required hot-fix from Synergex Support. As an alternative, use the environment variables method of configuring the service (in synergy.ini as shown above).