Introduction to SQL Replication

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

License Agreement

Change Log

7/9/2018

- We changed the sample environment to place SQL Server into "auto-commit" mode at all times except when executing the old-style table load mechanism (pre bulk load). By doing so we can avoid having to explicitly execute start transaction / end transaction statements, each of which is a network operation in client / server environments. If you wish to revert to the previous behavior then you can use the new replicator command line option **-autocommit NO** or alternatively set the environment variable **REPLICATOR_AUTO_COMMIT=NO**.

5/31/2018

- We addressed an issue when a bulk load operation was taking more than 60 seconds to process the BUILK INSERT statement. There was a default 60 second timeout on the execution of the SQL statement, which is inappropriate for large BULK INSERT operations. We suppressed the timeout for such operations.
- We enhanced the replicator_menu program by adding some basic support for replicator instance names. We also added an option to view running replicator processes on Linux systems.
- We made several enhancements to the Linux replicator management scripts.

5/17/2018

- We changed XCALL REPLICATE so that it adds shutdown requests to the beginning of the instruction queue instead of the end. This will mean that shutdown requests will be actioned as soon as possible, i.e. as soon as the replicator has completed its current task. Previously a shutdown request would bot be processed until all currently queued tasks had been completed.
- We fixed the Load and Bulk Load operations for multi-record-format (tagged) structures. These were inadvertently broken when we switched

from the custom IO routines to regular IO statements.

2/22/2018

- We changed the bulk load mechanism to now use a new feature in FileService called "chunked upload". This mechanism breaks down the upload of large files into multiple synchronous operations, in turn reducing the memory requirements and making is possible to upload very large files.
- The REPLICATOR_REMOTE_EXPORT environment variable has been removed and replaced with two new environment variables REPLICATOR_FILESERVICE_HOST and REPLICATOR_FILESERVICE_PORT.

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.

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.

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.

If you wish to develop in Visual Studio then you must have Visual Studio 2015 or later, and have the latest version of Synergy DBL Integration for Visual Studio installed.

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.

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.

Synergy Repository Database

Because of the dependency on CodeGen, the record layouts for the files that you wish to replicate must be declared in a Synergy repository, as must the data files.

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

Downloading the Code

There are two ways to obtain the code for the environment:

1. Clone the GIT repository.
2. Download the code in a ZIP file.

Cloning the GIT Repository

1. Ensure that you have GIT for Windows command line tools installed.
2. Open a Windows command prompt and move to the folder where you would like to download the code (a new sub-folder will be created automatically when you clone the repository).
3. Execute the following GIT command: git clone https://github.com/SteveIves/SqlReplicationIoHooks.git

Downloading the Code in a ZIP File

1. Use a web browser and go to https://github.com/SteveIves/SqlReplicationIoHooks
2. Click the green Clone or Download button ans select Download Zip.
3. Extract the zip fine in your chosen location.

Setting up the Environment

After obtaining the code there is a one-time setup step that you will need to follow. The commands for this are in a batch file named **setup.bat** that you will find in the root folder of the development environment. Open a command prompt, go to the root folder and execute the batch file. The batch file does the following things:

1. Loads the repository schema into a repository database in the RPS folder.
2. Loads two sample ISAM files in the DAT folder.
3. Creates the replicator queue ISAM file in the DAT folder.

Setting Up SQL Server

You will need administrative access to the SQL Server instance that you intend to use in order to complete the following initial setup tasks:

1. Create a SQL Server Login for use by the replicator and set the password to the account.
2. Allow the new login to perform bulk operations.
3. Create a new SQL Server database.
4. Set the owner of the new database to the name of the new login.

To assist with these steps you will find an SQL script in a file named **ConfigureDatabase.sql** in the root folder of the development environment. You can execute this script via SQL Server Management Studio in order to perform the steps listed above. Unless you edit it to do something different, the script creates a login called **SqlReplicationUser** with the password set to **SqlReplicationPassword**, and creates a database named **SqlReplication**.

Setting the Database Connect String

The Synergy SQL Connection API used by the replicator to connect to and interact with the database uses a "connection string" to identify the database server, database name and the database username and password. This connection string is stored in the environment variable REPLICATOR_DATABASE, or can be specified via the -database command line option. The default connection string is configured for my development environment:

net:SqlReplicationUser/SqlReplicationPassword/SqlReplication/SISQL2017///@

You will need to be changed it to match your environment.

- 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 project properties; 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**	**Connect via**	**Authentication**	**Connect string**
Local	Either	DSN	SQL Server Login	VTX12_SQLNATIVE:uid/pwd/dsn
Local	Default	Database name	SQL Server Login	VTX12_SQLNATIVE:uid/pwd/dbname/
Local	Default	Database name	Windows Login	VTX12_SQLNATIVE://dbname/.///Trus
Local	Named	Database name	SQL Server Login	VTX12_SQLNATIVE:uid/pwd/dbname/
Local	Named	Database name	Windows Login	VTX12_SQLNATIVE://dbname/.\\\insta
Remote	Either	DSN	SQL Server Login	net:uid/pwd/dsn@port:server_ip!VTX12
Remote	Default	Database name	SQL Server Login	net:uid/pwd/dbname/server_name///@pc
Remote	Default	Database name	Windows Login	net://dbname/server_name///Trusted_cor
Remote	Named	Database name	SQL Server Login	net:uid/pwd/dbname/server_name\\\insta
Remote	Named	Database name	Windows Login	net://dbname/server_name\\\instance///T

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_name	Name 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 SQL Connection documentation.

Setting Up the Synergy OpenNET Server

If the SQL Server database is on a different server than the replicator process then you will need to start and configure the Synergy OpenNET Server on the Windows system that hosts the SQL Server database. This requires you to install Synergy/DE "Core Components" and "Connectivity" on the database server. Assuming the server is running 64-bit Windows you should install 64-bit Synergy. In the unlikely event that your server is running 32-bit Windows, install 32-bit Synergy.

In this scenario the Windows server will need permanent access to two Synergy licenses:

1. A SQL Connection for SQL Server license (SCSQ10) to allow the remote replicator process to connect to the SQL Server database via the Synergy OpenNET server.
2. A Synergy runtime license (RUN10) to allow you to run the FileService server to support bulk uploads (more later).

Declaring the OpenNET protocol to Windows

After installing Synergy, start a copy of notepad **as administrator**, then edit the following file:

```
C:\WINDOWS\SYSTEM32\drivers\etc\services
```

Add the following line to the file:

```
vtxnet   1958/tcp    #Synergy/DE OpenNET Server
```

Close and save the file.

Registering and Starting the OpenNET Server

You can register and start the Synergy OpenNET server as a Windows service by using the Synergy Configuration Program, which you can access from the windows control panel.

1. Run the windows control panel (control.exe)
2. Search for the word Synergy, then click on the Synergy Control Panel icon to start the utility.
3. Click on the link for the Synergy Configuration program (64-bit).
4. In the Synergy Configuration Program, go to the Connectivity Series tab.
5. Click on the Add Service button to register the OpenNET server as a Windows Service.
6. Click on the Start Service button to start the service.

In the future the service will start automatically when Windows starts.

Verifying Access to the OpenNET Server

It is a good idea to verify that you can access the OpenNET service from the system that you intend to run the replicator process from. You can do this using the VTXPING utility. For example:

```
C:\> vtxping -p1958 SISQL2017

                         VORTEXping - Network Checker.
                        Version 4.0.0.17 - Production.
Copyright ⌐ 1989-2016, Trifox, Inc., California, USA.
                        All Rights Reserved Worldwide.



Pinging 'SISQL2017'.

vtxnetd is alive and kicking (1st attempt).

Host process ID       : 3808
Requires byte flipping: No
Requires SSL          : No
Character set - Client: ASCII
                Host  : ASCII
```

Check that you see the "alive and kicking" message.

Setting Up the FileService Server

If you with to be able to perform "bulk load" operations when new tables are created, or need to be reloaded for some reason, then you will need to install the FileService server on the Windows SQL Server system.

FileService is a Windows service that hosts an ASP.NET Web API 2 RESTful web service. This web service can be used by the replicator process to transfer delimited text files containing the data for a table to the database server, from where they can be bulk loaded into the database very quickly.

Running the FileService server requires permanent access to one Synergy runtime (RUN10) license.

If you do not install and use FileService, then bulk upload will not be available to you and the replicator will have to load tables row by row, which takes significantly longer.

You can obtain the installer for the latest version of FileService from the GitHub repository. After downloading the installer, simply run it to install and start the service.

Default Behavior

By default FileService listens for messages on TCP/IP port 8080:

http://<server>:8080/FileService

and manages files in a FileService folder below the public documents folder:

C:\Users\Public\Documents\FileService

Both of these options can be changed if necessary

Configuring FileService

If you wish to change the port number or directory used, you can do so by editing the applications configuration file:

C:\Program Files (x86)\Synergex\FileService\FileService.exe.config

The file contains an AppSettings section that looks like this:

```
<applicationSettings>
  <FileService.Properties.Settings>
    <setting name="StorageFolder" serializeAs="String">
      <value />
    </setting>
    <setting name="HttpListenerPort" serializeAs="String">
      <value>8080</value>
    </setting>
  </FileService.Properties.Settings>
</applicationSettings>
```

As you can see, the default port number is already included and can simply be edited. But there is no default value for the StorageFolder setting. If you wish to change the folder then you must edit the file, like this:

```
<applicationSettings>
  <FileService.Properties.Settings>
    <setting name="StorageFolder" serializeAs="String">
      <value>D:\Some\Other\Folder</value>
    </setting>
    <setting name="HttpListenerPort" serializeAs="String">
      <value>8080</value>
    </setting>
  </FileService.Properties.Settings>
</applicationSettings>
```

If you make any changes to the file you must restart the FileService server process. You can do this in one of three ways:

1. Using the Services tab in Task Manager.
2. Using the Services management utility (services.msc)
3. From an administrative command line, like this:

   ```
   net stop FileService
   net start FileService
   ```

If you use the latter option, you should see output like this:

```
C:\> net stop fileservice
The File Service service is stopping.
The File Service service was stopped successfully.

C:\> net start fileservice
The File Service service is starting.
The File Service service was started successfully.

C:\>
```

Verifying FileService Operation

You can verify that FileService is operational by using it's PING operation from a web browser. For example, from a browser on the SAME SYSTEM, use the following URL:

```
http://localhost:8080/FileService/ping
```

You should see s "FileService Ping Response" message in your browser.

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
-autocommit <option>	Should SQL Server auto commit be enabled. Option values are YES or NO. The default is YES.
-datadir <data_location>	The location of the replication instruction file (REPLICATOR.ISM)
-database <connect_string>	SQL connection connect string identifying the database to connect to.
-erroremail <email_address>	Email address that start, error and stop messages should be sent TO.
-fileservicehost <host>	The DNS name or TCP/IP address of the database server where an instance of FileService is running to support bulk uploads.
-fileserviceport <port>	The TCP/IP port number that FileService is listening on. If not specified the default port is 8080.
-localexport <export_path>	The location where bulk export files will be created locally.
-instance <instance_name>	The name of this replicator instance.
-interval <sleep_seconds>	The number of seconds the replicator should sleep if it finds no 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 <log_location>	The location where the log file should be created. A full or relative path, or an environment variable followed by a colon.
-mailfrom <email_address>	The email address that replicator messages should be sent FROM.
-mailserver <smtp_server>	The DNS name or IP address of the SMTP mail server that will be used to send messages.
-maxcolumns	

<max_columns>	The maximum number of columns in a database table. Default is 254.
-maxcursors <max_cursors>	The maximum number of database cursors. Allow 4 per table. Default is 128.
-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_AUTO_COMMIT	Should SQL Server auto commit be enabled. Possible values are YES or NO. The default is YES.
REPLICATOR_DATA	The location of the replication instruction file (REPLICATOR.ISM)
REPLICATOR_DATABASE	SQL connection database connection string identifying the SQL Server database to connect to.
REPLICATOR_ERROR_EMAIL	The email address that start, error and stop messages should be sent TO.
REPLICATOR_EXPORT_LOCAL	The location where bulk export files will be created locally.
REPLICATOR_FILESERVICE_HOST	The DNS name or TCP/IP address of the database server where an instance of FileService is running to support bulk uploads. On OpenVMS if you use a TCP/IP address you must enclose it in quotes.
REPLICATOR_FILESERVICE_PORT	The TCP/IP port number that FileService is listening on. If not specified the default port is 8080.
REPLICATOR_INSTANCE	The name of this replicator instance.
	The number of seconds the

REPLICATOR_INTERVAL	replicator should sleep if it finds no instructions to process.
REPLICATOR_LOG_KEYS	Set to YES to cause the key values being used to relate ISAM records to SQL rows.
REPLICATOR_LOG_BULK_LOAD_EXCEPTIONS	Set to YES to cause failing records during a bulk load operation to be logged to a file.
REPLICATOR_LOGDIR	The location where the log file should be created. A full or relative path, or an environment variable followed by a colon.
REPLICATOR_EMAIL_SENDER	The email address that replicator messages should be sent FROM.
REPLICATOR_SMTP_SERVER	The DNS name or IP address of the SMTP mail server that will be used to send messages.
REPLICATOR_MAX_COLS	The maximum number of columns in a database table. Default is 254.
REPLICATOR_MAX_CURSORS	The maximum number of database cursors. Allow 4 per table. Default is 128.
REPLICATOR_ERROR_STOP	Set to YES to cause the replicator to stop if an error is encountered.
REPLICATOR_SYSTEM_LOG	Set to YES to log to the system log in addition to the log file.
REPLICATOR_FULL_LOG	Set to YES to cause more verbose logging to be 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 code-generated 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

Obtaining the Source Code from GitHub

- Ensure that you have the git client tools installed (type "git version" and make sure the git version number is displayed). If not then you'll need to install git using your operating system package manager (e.g. sudo apt-get install git).
- Move to a folder where you would like the SQL Replication environment to be downloaded (a new sub-folder will be created to contain the environment).
- Clone the git repository: git clone https://github.com/SteveIves/SqlReplicationIoHooks.git
- A new sub-folder named SqlReplicationIoHooks should have been created. Move into that folder, and then into the linux folder.
- dos2unix *
- chmod +x *
- build

Building the Sample Environment on OpenVMS

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

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

Basic steps to build on VMS:

- Log into a privileged account (privilege is needed to start a detached process as SYSTEM)
- Create an empty directory and copy the sample environment there.
- Move to the VMS folder
- Execute the BUILD.COM command procedure.
- 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.
- Start replicator as a detached process by executing REPLICATOR_DETACH.COM
- Or start the replicator as an interactive process by executing REPLICATOR_RUN.COM
- Check the log file in the [.LOG] folder and ensure that replicator was able to connect to the database.
- Run REPLICATOR_EXE:EMPLOYEEMAINANCE.EXE and use it to add instructions to the replicator queue.
- Run REPLICATOR_EXE:REPLICATORMENU and test using the EMPLOYEE file.

- 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:

```
SQL Replicator Log
Replicator startup
Processing interval is 2 seconds
Connecting to database...
ConnectedAuto-commit enabled
Maximum cursors: 128
Maximum columns: 254
--- Processing instructions -----------------
```

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).