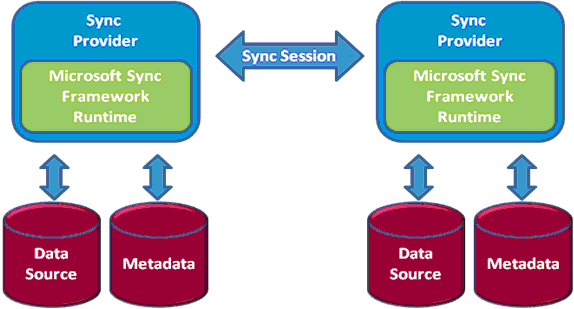
# Firebird Synchronisation Services

## Microsoft Sync Services Overview

“Microsoft Sync Framework is a comprehensive synchronization platform that enables collaboration and offline scenarios for applications, services and devices. Developers can build sync ecosystems that integrate any application, any type of data, using any protocol over any network.”

<http://msdn.microsoft.com/en-us/sync/bb887625>

The basic architecture of the Sync Framework:



Sync Providers for SQL Server and SQL Server CE are included with the Framework, but custom providers can be written (such as the one implemented here for Firebird).

More information on the Sync Framework can be found on MSDN here:

<http://msdn.microsoft.com/en-us/sync/default.aspx>

## Firebird Provider

This demonstration application was based on the sample provided by Microsoft for implementing an Oracle Provider. All the SQL routines and data access were ported to the equivalent Firebird objects.

The Oracle sample can be downloaded from here:

<http://code.msdn.microsoft.com/Database-Sync-Oracle-and-037fb083/sourcecode?fileId=19015&pathId=1409552837>

## Further Activities

The Sync Toolkit has been open sourced and there is a Framework for extending to other clients which can be found at:

<http://syncwinrt.codeplex.com/>

There is also a port of this Sync Framework to permit the use of the Sync Framework with clients on WinRT and Windows Phone 8 with SQLite as the datastore on those platforms.

<http://jtabadero.wordpress.com/2013/01/09/synchronizing-winrtsqlite-using-sync-framework-toolkit/>

## Requirements

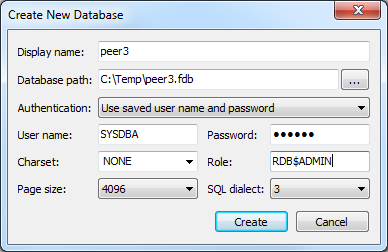
|  |  |
| --- | --- |
| File | Description / Location |
| Visual Studio 2013 | Development tool used for building the test application |
| Sync Services 2.1 SDK | Microsoft Sync Framework 2.1 Software Development Kit (SDK)  <http://www.microsoft.com/en-us/download/details.aspx?id=23217> |
| SQL CE 3.5 Desktop Edition | This should be included as part of the Visual Studio installation. |
| Firebird .NET Data Provider | FireBird .net data provider  Version 4.1.0 for dot net 4.0 (with Entity Framework Support – Jan 2 2014)  <http://firebirdsql.org/en/net-provider/> |
| Flame Robin | A good free tool for looking at a Firebird database in Flame Robin <http://www.flamerobin.org/> |
| SQL Server Management Studio | The SQL Server Management Studio provides the capability to edit the sqlce databases to test modifications being updated back into the firebird database  (optional) |

## Possible Sync Configurations



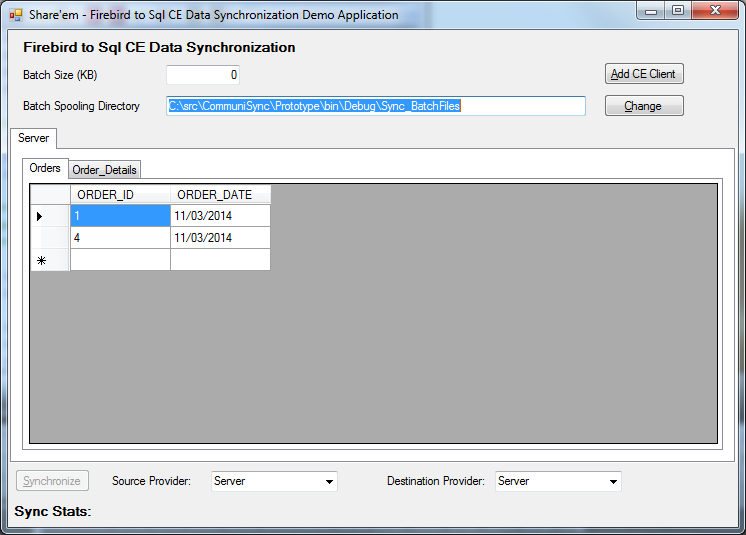
# Demonstration

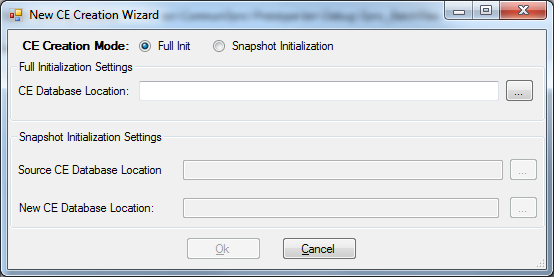
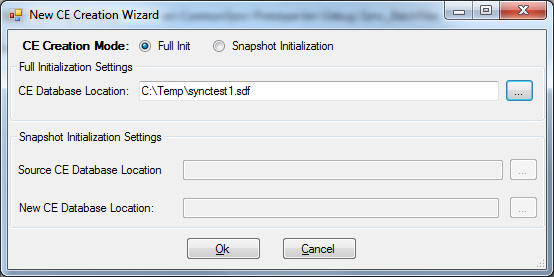
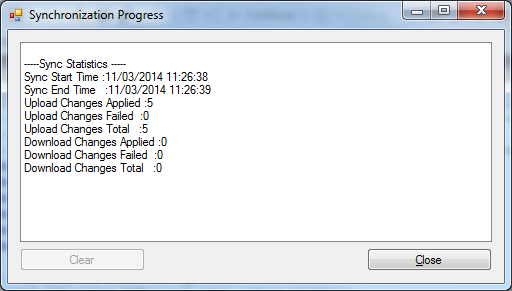
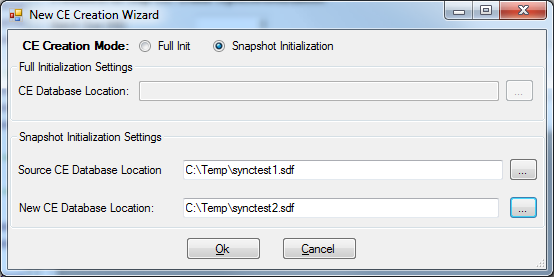
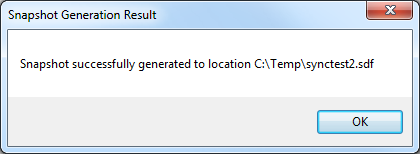
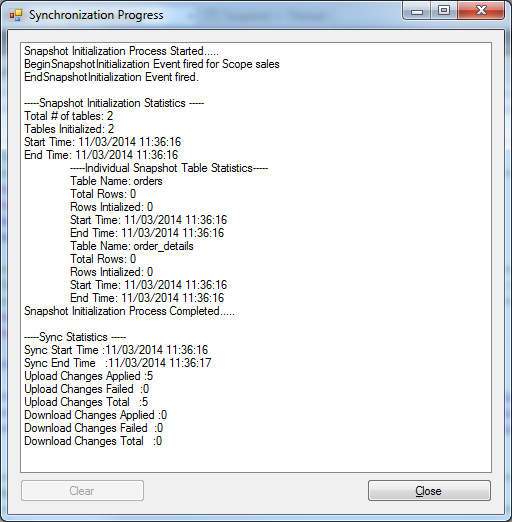
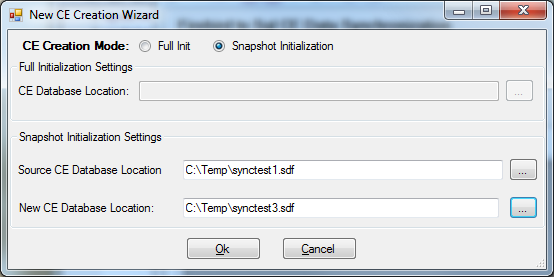
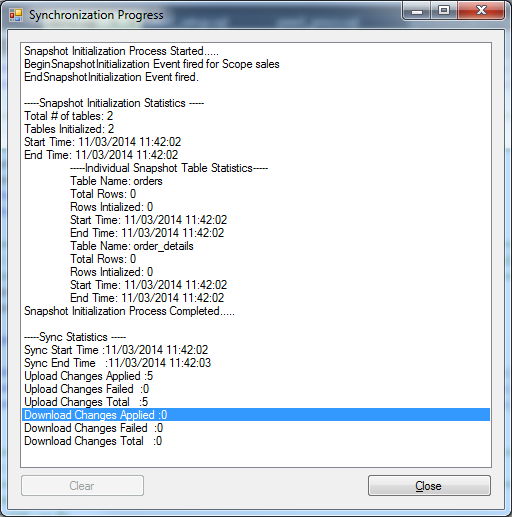
## Setup

* Create a new Firebird database  
    
  
* Run the script: **Firebird Setup\peer1\_setup.sql***(ensure that you commit the transaction)*
* Run the script: **Firebird Setup\peer1\_procs.sql***(ensure that you commit the transaction)*
* Setup the initial test scope and some test data which are in the first section of the script: **Firebird Setup\demo.sql***(only the sql statements up to test case 1)  
  (ensure that you commit the transaction)*

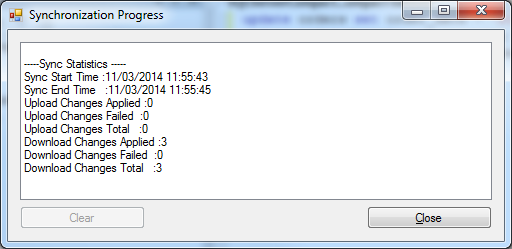
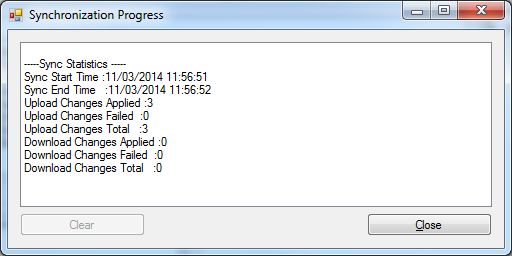
## Synchronization Demo

If the setup has been completed successfully you will be able to see the Firebird to SqlCe app look like the below screenshot



* Click on the **Add CE Client** button to open this dialog  
    
  
* Click on the … button to select the location to create a new Sql Ce Database.  
  *(for my demo I created c:\temp\synctest1.sdf)*
* Click **Ok** to create the new database  
  *(if you click the Client 1 tab, it will throw an exception stating that the tables don’t exist yet. This is correct as we have not sync’d any data or schema into it)*
* Click Synchronize to commence a sync from Firebird into the Sql Ce database  
  *(this will create the schema in Sql Ce also)*
* Now if you click between the Server and Client 1 tabs you can see that the details between the 2 databases are now in sync!
* Next we can create another SqlCe database using the synctest1 database as a snapshot.  
    
  
* Click Ok to create the next replica of our firebird database  
    
  
* Flicking between tabs we can see all 3 databases are in sync
* Now execute the Test Case 1 Sql Statements in the script: **Firebird Setup\demo.sql** against the Firebird database (and commit the transaction)
* Flicking through the tabs you should now see that the 3 databases are no longer in sync, there is only 1 order, and the order details for the floppy disk are now 13.
* Next ensure that the Source selected is the Server and the Destination is Client 2 and click Synchronize again.  
    
  
* As we now flip through the tabs we can see that Server and Client 2 are in sync, but Client 1 is still not behind. We could at this point sync that database to bring it up to date, however in this case we will create another replica based on Client 1 and sync that one.  
    
  
* Now as we flip through all the tabs server 1 and Client 2 are in sync, and Client 1 and Client 3 are in sync (but behind the server)
* Next ensure that the Source is the Server and Destination is Client 3 and click Synchronize  
    
  
* Now we can see that all are up to date, except for Client 2
* Now execute the Test Case 2 Sql Statements in the script: **Firebird Setup\demo.sql** against the Firebird database (and commit the transaction)
* Flicking through the tabs you should now see that the server has order 2, and the others do not.
* Synchronize the changes to Client 3
* Select the source provider as Client 2, and Destination as Client 1 and sync
* This brings 1 up to date with 2, but both are still missing the last set of changes
* Select the source provider as Client 3, and the destination as Client 1 and sync
* Now to bring client 2, select any of the databases as the source, and client 2 as destination and sync
* Now if you select any combination of source and destination (including server as the destination) there will be no changes propagates

This demonstration has shown that the firebird database can be used as a master of changes and sync to Sql Ce databases. The next tests will demonstrate how it can also receive changes made to the Ce database also. For this you will need SQL Server Management Studio (or some other tool to edit the Sql Ce database.

* Open the SqlCe database  
    
  
* Execute the below statements:  
  update orders set order\_date = '2014-8-8' where order\_id = 4;  
  update order\_details set quantity = 4 where order\_details\_id = 3;  
  insert into orders (order\_id, order\_date) values(9, getdate());
* Flick through the tabs to see that the new order has been created, and that the floppy disk quantity has been updated to 4.
* Select the Source as Server and Destination as Client 1 and sync  
    
  
* Select source as Server 1 and destination as Client 2 and Sync  
    
  
* Note that we have now sync’d the server and clients 1 and 2.
* Select Client 2 as the source and Client 3 as the destination and sync
* Now all databases are up to sync again, and we didn’t have to go through the Server database in all cases.

Using these test cases you can now go ahead and further experiment with making changes to any of the databases and syncing the changes between.