Follow the Data

Abstract

Follow the Data

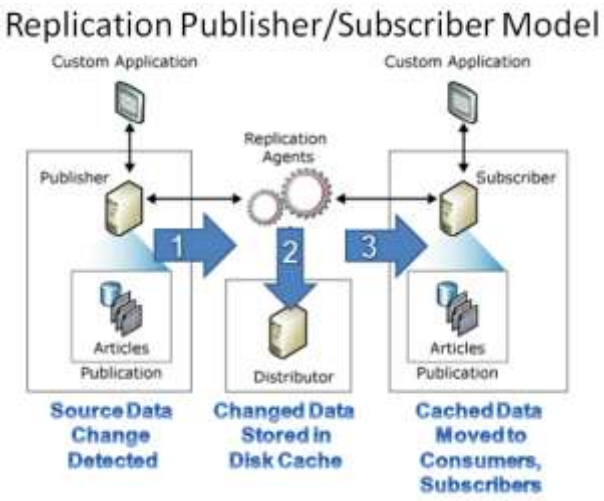
Many technology solutions use a STORE and FORWARD model to move data though out the business. It begins when source data change is detected, collected, and STORED in a cache. This cache data, often on a schedule, is processed and FORWARD to one or more destinations. Microsoft technologies using "store and forward" include Log Shipping, Database Mirroring, Transactional Replication, Merge Replication, Change Data Capture, Change Tracking, Sync Services, SSIS ETL packages, and SQL Azure. As we "Follow the Data" for Transactional Replication, look at the players, their roles, auditing options, supportability features, and ability to handle changing business needs.

Store and Forward

In Microsoft SQL Server Replication technologies "Store and Forward" model is called Publish-Subscribe. In this paper I'll cover techniques used by Microsoft SQL Server Transactional Replication for moving data throughout a topology. Topics covered include Roles, Jump Start, What's Normal, Auditing, "NOT" Replicated, Really Big Batches, Open Transactions, Do Over, Divide-n-Conquer, Cleanup, and Supportability, can be applied to many technologies using similar "Store and Forward" methodology to move data throughout an organization.

Big Picture

Microsoft SQL Server Transactional Replication designed goal is to move data from one Publisher to one or more Subscriber in near real-time. What's "near", about 5-10 seconds for simple topology, up to a few minutes for complex environment. Using a built-in Create Publication Wizard or via stored procedures, you can select which objects, like tables, views, and stored procedures, to replicated to subscribers.



Replication topology consists of 3 major components. First, is the Log Reader Agent, a standalone executable LogRead.exe, watching for changes committed on the Publisher. Those changes are picked up and stored in the 2nd component, the Distribution Database. This database is a storage, or cache, of the changes to be distributed. Depending on individual needs, the Distribution database can be stored at the Publisher, a standalone SQL Server, or even on the Subscriber. The 3rd component, the Distribution Agent, is another standalone executable, Distrib.exe,

Written by Chris Skorlinski, Edited by Gaurav Mathur
Microsoft SQL Server Escalation Services

periodically scans the Distribution database cache for updates and sends copies of new data to downstream consumers, called Subscribers. One Distribution Agent is created for each subscriber requesting data. When the Distrib.exe is executing on the Distributor, we use term Push replication. When executing on the Subscriber, Pull replication is running.

Unique business needs can be met by selecting which database objects get distributed. For example, downstream Subscriber may be aggregating selected columns from just a few tables from multiple sources into one business reporting or data warehouse solution. The target subscriber may include tables or indexes not found on the Publisher to support the subscribers reporting activity. The Subscriber may also be republishing data to other Subscribers thereby acting as both Subscriber and Publisher.

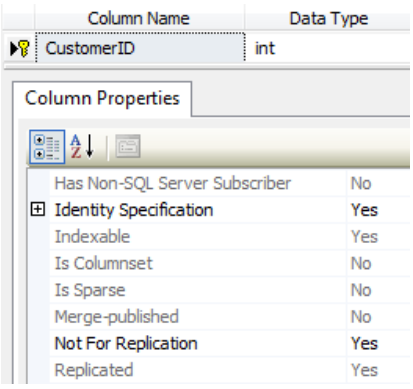
Jump Start

Transactional Replication provides 3 options to preload subscribers as the initial step in the store and forward process. Advantages and disadvantages of these options is discussed in SQL Server Books Online.

1) BCP out data using Snapshot Replication
2) Manually backup the Publisher database and restore to Subscriber
3) Use any data copy solution like SSIS packages to ensure data matches

The BCP out option is default when configuring using the built-in Replication Wizard. This Wizard can be launched from SQL Server Management Studio. Data in tables selected to be Replicated are exported using Bulk Copy Program, BCP.exe, utility included with SQL Server. Data is stored in a shared folder and made available to the Distribution Agent to preload the Subscriber using the 1st run.

If you are using the "Manual Backup and Restore" method to load the Subscriber, manually cleanup database objects not needed on the Subscriber. This is also an opportunity to add objects needed on the Subscriber but not on the Publisher, such as indexes used to support online reporting query activity.



Before setting up Replication you may want to FLAG objects created on the publisher as "NOT for Replication", this instructs Distribution Agent to ignore those objects following the Subscriber restore. In this example the CustomerID is an identity column. Data inserted on the Publisher will automatically be assigned a new, incremented ID. The "Not For Replication" setting will ensure when this record is distributed to the Subscriber the identity value will not be recomputed and overwrite the original value. This option is also available for objects such as TRIGGERS.

Written by Chris Skorlinski, Edited by Gaurav Mathur
Microsoft SQL Server Escalation Services

Once source and destination are in synch, Transactional Replication starts a SQL Agent Job to call LogRead.exe. This "Store" component makes a connection to Publisher and requests list of transactions. These transactions are extracted from the Publisher's transaction log, and stored in the Distribution database. The Distribution DB, will "cache" this data in table until "forward" phase moves the data to Subscribers/Consumers.

Prior to "forwarding" data the Distribution Agent will create Replication support stored procedures and tables need to be installed on the subscriber. This happens automatically as the first step in the Distribution Agent "forward" phase.

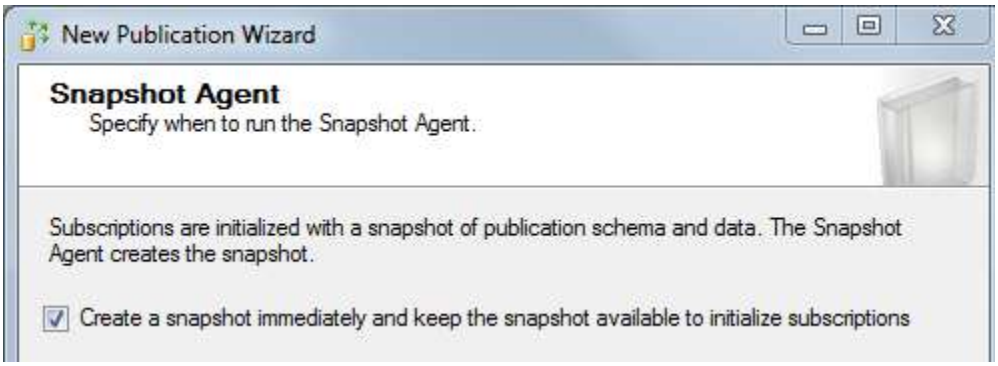
In Flight

How does the system handle the 24x7 business needs during initial setup? What happens to the in-flight data changes?

When Transactional Replication is created using the Replication Wizard, the Snapshot Agent is contracted to begin BCP-ing out data already stored in the Publisher. This data is cached in the "Snapshot Folder" and later used by the Distribution Agent to preload the Destination\Subscriber with the initial data. The Snapshot Agent will take a quick Schema-Lock to obtain the tables definition. It releases the lock and begins BCPing out the data. For large tables it will bulk export the data in multiple parallel BCP sessions creating multiple BCP files.

Transactional Replication option "immediate_sync" begins queuing changes immediately following the start of the Snapshot Replication BCP-out of the published data. This ensures Replication accounts for all "in flight" transactions. Distribution Agent is responsible for loading new subscribers with the bulk data BCP'ed out by the Snapshot Agent followed by applying of the queued data stored in the Distribution database by the Log Reader Agent.

Picking up these "in flight" changes occurs automatically when you select option to "create a snapshot immediately" in the New Publication Wizard dialog. The data changes will be queued in the Distribution Database for 24 hours while new subscribers are being established.



Written by Chris Skorlinski, Edited by Gaurav Mathur
Microsoft SQL Server Escalation Services

If creating publications via stored procedures you can enable “in flight” capture by setting the `@immediate_sync = N'true'` in the `sp_addpublication` stored procedure call.

If this feature is not selected, you must ensure no “in-flight” data changes occur until the publication is created, new subscribers are configured, and Log Reader is picking up new changes.

What’s Normal?

As with any data transformation, data movement system, understanding what’s normal provides baseline to help isolate when problems occur. Ask yourself how long it is taking to move data end-to-end through your topology? How much data are you moving hourly, daily, and monthly. Is the flow consistent throughout the day or peaks and valley caused by end-user activity, or scheduled job processing.

In the Transactional Replication flow it all begins when a transaction is committed at the source. This new data is detected by the Log Reader agent, extracted from the database’s transaction log and stored in the Distribution database **msrepl_commands** table. The transaction header information is “stored” in the Distribution database **msrepl_transactions** table and queued until the Distribution Agent polls for newly cached data. When found, the new data is “forward” to 1 or more consumers or Subscribers.

Sending individual transactions wouldn’t be very efficient, therefor both Replication Agents collect transactions into batches and distributes the entire batch. “Batch Size” is one of the adjustment you can change to tune Replication flow for your environment in the Agent Profile settings. For overview see: How to: Work with Replication Agent Profiles.

The Store and Forward methodology for Transactional Replication has some latency as data is moved across a network from server to server. Normal in some environments may be 5-10 seconds, while others 1-2 minutes. When higher latency is observed a bottleneck is occurring. Further investigation is required to determine if either data volume has increased or if a system component is restricting the normal flow of data. The auditing features of any Store and Forward system like Transactional Replication provide insight into normal v. abnormal data flow.

Auditing Features

Auditing feature of each step is critical in understanding the flow of data. Transactional Replication stores audit activity in the Distribution database. The Log Reader Agent stored audit records in the **MSlogreader_history** table while the Distribution Agent stored in the **MSdistribution_history** table. You can query these tables directly from SQL Server Management Studio, or view them using the Replication Monitor utility.

Written by Chris Skorlinski, Edited by Gaurav Mathur
Microsoft SQL Server Escalation Services

You can also control the "level" of auditing using Agent Profiles by adjusting the History Verbose Level. While investigating abnormal flow, increasing the Audit level may provide clues as to volume change or bottleneck. However, by default this audit data is only maintained for 2 days. It is controlled by the Distribution History Cleanup job. Temporarily stopping the cleanup job or increasing the history retention is a common practice to capture longer historical data.

"NOT" Replicated

When configuring Transactional Replication you select which tables to replicate. Data changes stored in the Transaction log for non-replicated data must still be read, then skipped. Active database with large amount of non-replicated data to be skipped can slow the detection of data to be replicated.

Other logged database operations such as index maintenance can also generate non-replicated data. These transactional log records are also read and skipped. While this activity can't be avoided, it should be investigated as possible root cause when flow of data is slowed.

Is the database transactional log (*.ldf) file >20gb? If so, you may notice Log Reader Agent taking a long time upon startup to scan the log for replicated transactions. Transactions logs approaching 100gb may take 1-2 hours to scan depending on disk subsystem performance. Frequent log backups or using "bulk logged" operation will reduce size of the transaction log and improve the Log Reader latency.

Really Big Batch

What happens when the data volume changes, for example, a large batch of data changes are detected and moved through the topology. By default, a batch change of 10 million rows is detected and passed to the Log Reader as single transaction. This is then inserted into the Distribution databases as a single transaction and finally to the Subscriber(s) as a single transaction. If the topology takes 15 minutes to read 10 million changes, 15 minutes to write to Distribution database, 15 minutes to read from Distribution database, and 15 minutes to write to the Subscriber, the Subscriber could be waiting 1 hour after the transaction has committed on the Publisher until it is observed on the Subscriber.

During this time you may observe the Replication Agents reporting "no activity in the last 10 minutes". This message doesn't always indicate an error. In this example it is just an informational message that the Agents are still running and still processing the current batch. Once the batch has moved through each step in Replication, the Agents update their respective audit history tables.

First option is to recognize the subscribers may observe high latency while batch is being moved through the Replication Topology. If no failure is occurring and latency can be tolerated, waiting for Replication to catch up may be an acceptable solution.

Written by Chris Skorlinski, Edited by Gaurav Mathur
Microsoft SQL Server Escalation Services

If changes impact majority of the data it may be faster to drop Replication environment, make required changes, then rebuild the Replication topology. This solution is often used in combination of Backup/Restore method for setting up a subscriber. For example, if business needs require updating every row in an 845 million row table and that table is 90% of the entire database. Dropping the Replication, changing the data, then setting up Replication using Backup/Restore method is faster than moving individual record changes through the Distribution databases to the subscriber. See also: SQL 2005/2008 Books Online top: How to: Initialize a Transactional Subscriber from a Backup

A third option is to publish the "execution" of a stored procedure instead of the results from a stored procedure. This solution is best used when data changes can be applied via stored procedure. Some creativity may be needed to implement this solution. For example if updating and inserting, a temp table of new data may need to be imported into both the Publisher and Subscriber before the stored procedure executes.

Another option is to instruct the Log Reader to break the single transaction into batches using the MaxCmdsInTrans option. The same workload is transferred through the Replication Topology, however because the Replication Agents are updating their audit table following each commit and agents are committing in smaller batches, audit tables are updated more frequently providing "warm fuzzy" about Replication agent status. Breaking up 10 million row update into batches of 100k changes also allows the Replication agents to restart at 100k batch size should network or system problem occur.

These options are discussed further on ReplTalk blog

Who left the door open?

Another cause for transaction log growth is the presence of open transactions which start when client issues a BEGIN TRAN statement, makes data changes, but no corresponding COMMIT TRAN, or ROLLBACK TRAN statement was issued. During log backup, the transaction log can't be purged because "open transaction" still exists. The transaction log will continue to grow until the open transaction is committed or closed.

To look for open transactions execute the DBCC OPENTRAN command as show below.

```
USE AdventureWorks2008
GO
DBCC OPENTRAN
GO

Transaction information for database 'AdventureWorks2008'.

Oldest active transaction:
    SPID (server process ID): 61
    UID (user ID) : -1
    Name          : user_transaction
    LSN           : (61:375:2)
    Start time    : Oct 17 2011  7:37:42:737PM
```


Written by Chris Skorlinski, Edited by Gaurav Mathur
Microsoft SQL Server Escalation Services

Do Over

If problems occur, what features are available to "do over". How does the system know where it left off and what work was last completed?

Transactional Replication Log Reader Agent tracks the last successful transaction retrieved from the database's transaction long in the actual transaction log in the form of checkpoint entries. Should SQL Server stop and restart, such as in an automatic cluster failover, the Log Reader retrieve the last successful transaction Log Sequence Number (LSN) from the transaction log, verifies that record made it to the Distribution Database, then retrieves new records from that point forward. You can track the last distributed and next to distribute LSN values for the Log Reader by executing the "DBCC OPENTRAN" command.

Sample output

```
--Are there any "pending" transactions?
USE [AdventureWorksLT]
DBCC OPENTRAN

Transaction information for database 'AdventureWorksLT'.

Replicated Transaction Information:
        Oldest distributed LSN     : (31:1696:6)
        Oldest non-distributed LSN : (31:1741:1)
```

The Distribution Agent for each unique Subscriber tracks status of last committed transaction delivered to that subscriber in **MSreplication_subscriptions** table stored in each subscriber. It uses the Subscriber's stored LSN as the starting point to begin retrieving rows from the Distribution database. This allows the Subscriber database to be restored to an earlier time as long as the pending data is still cached in the Distribution database. More advanced features such as "sync with backup" are discussed in the SQL Server online books.

Divide-n-Conquer

During any store and forward processing of data an occasional problem will occur in the end to end flow of data. Any system should provide tools to help isolate problem and allow you to focus on troubleshooting.

The Transactional Replication points below have largest impact on the data flow.

- Reading Transaction Log (Common)
- Writing Distribution, Blocking, Disk Speed, (Rare)
- Reading Distribution, Blocking Agent, Batch Cleanup, (Rare)
- Writing Subscriber, Blocking, Triggers, Indexes to Maintain (Common)

Follow the Data

To isolate the problem, Transactional Replication allows a tracer token to be inserted on the Publisher. The time to retrieve this token and write it to the Distribution Database and time to write token to the Subscriber is tracked by the token. Tracer Tokens can be inserted and tracked using the built-in Replication Monitor.

A common problem is the bottleneck at the Subscriber writes. As multiple transactions flow into the Publisher these are funneled into a single stream of data into the Distribution database by the Log Reader. The Distribution database containing just a few tables, a few indexes and only Replication Agent connecting can usually handle the Log Reader flow. The Distribution Agent, by default, makes single connection to the Subscriber database to push down these changes. It must contend with 100s of end-user connections accessing multiple tables, some of which have many indexes and/or triggers.

To increase Distribution Agent flow, SQL Server 2005 and above include a SubscriptionStreams option to write in parallel to the Subscriber. A range of values from 1 to 64 streams can be selected. Each stream will make a connection to the Subscriber, transfer data across the streams, and then together commit the entire batch. If contention occurs, the Distribution Agent will automatically switch to single stream, apply the changes, and switch back to multi-streams. This parameter is not supported for non-SQL Server Subscribers, Oracle Publishers or peer-to-peer subscriptions. For more insight into SubscriptionStreamscheck can be found on the MSDN Repltalk blog posting.

"Christopher, Clean up this Mess"

If store-n-forward solution is keeping data around for "restarts" or supporting multiple targets, somewhere along the way it will need to purge data no longer needed.

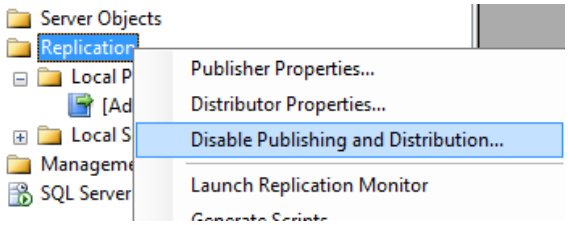
As discussed above, using Replication Wizard default settings, it begins by BCPing out table data into a Snapshot folder then BCPing that data into new Subscriber. New transactions are also cached in the Distribution Database. The Agents are also logging audit information in the Distribution Database. So when do we cleanup all this old data?

Replication includes a couple of "cleanup" jobs visible under SQL Server Agent. The frequency and/or amount of data to cleanup is controlled by the job parameters for each job. The "Agent history clean up: distribution" cleans up Agent audit data older than 2 days.

The "Distribution clean up: distribution" is responsible for removing delivered data from the Distribution Database. This Agent provides option to remove transactions from the Distribution Database cache as soon as the transactions are replicated to Subscribers, or keep data in the cache for 24 hours. How much historical data to keep is controlled by the cleanup job parameter. If data is kept in cache, a new subscribers can retrieve the last Snapshot BCP files then pick up all pending transactions. The option is controlled by the "Immediate_sync" option for Transactional Replication.

Written by Chris Skorlinski, Edited by Gaurav Mathur
Microsoft SQL Server Escalation Services

If you want to drop one Subscriber, or a Publication and all Subscribers, you can right-click an individual Subscriber or Publication in SSMS and "Delete". If you want to clean up EVERYTHING, I mean turn off all Replication you right-click "Replication" folder in SSMS and select "Disable Publishing and Distribution". However, be very careful selecting this option. If there are multiple publications originating from this Publisher, the replication configuration information about all publications will be purged. If you encounter problem purging Replication settings for a particular database you can disable all replication for that database by executing **sp_removedbreplication** from SSMS query. Warning, ensure you've selected the correct database on which replication is to be removed is selected before you run this stored procedure.




```
--SELECT database to remove all Replication settings
USE AdventureWorksLT2008
GO

sp_removedbreplication
GO
```

Advanced Supportability

In addition to the Agent history tables in the Distribution Database, the Replication Agents include supportability feature to further isolate performance problems. Each agent can OUPTPUT text file showing step-by-step agent activity. The OUTPUTVERBOSELEVEL setting controls the level of the details recorded. Both of these features can be enabled as SQL Agent Job command line parameters and are discussed on the ReplTalk blog "How to enable replication agents for logging to output files in SQL Server".

It's Just SQL

As you learn more about Transactional Replication you'll soon discover it boils down to 2 executables connecting to SQL Server, tables to store and forward changes, and stored procedure calls performing most of the work. These stored procedure calls appear in Profiler as RPC events. They also show in DMV execution stats. The same tools and troubleshooting techniques used for any SQL Server activity can be used when Replication is present.

What techniques would you use to investigate bottlenecks occurring in your SQL Server application? Would you execute "**sp_who2**" to check for blocking, or maybe run Profiler Trace or execute DMVs to determine long running queries.

Written by Chris Skorlinski, Edited by Gaurav Mathur
Microsoft SQL Server Escalation Services

Follow the Data

Because SQL Server Agent is much like user application, that is it connects to SQL Server, executes stored procedures, retrieves data, writes data, the same bottlenecks which occur for user applications also impact SQL Server Transactional Replication. Some of these include CPU, Disk, Network, Memory, Triggers, Constraints, and Indexes.

To simulate the behavior of the Replication Agents, most of the Replication created stored procedures can be called directly via SQL Server Management Studio. This provides further performance and troubleshooting diagnostic capabilities.

For example to simulate the Log Reader pulling pending transactions you can execute "**sp_replshowcmds**".

```
--The LogReader retrieves from the Transaction Log
sp_replshowcmds

xact_seqno                    command
---------------------- ---------------
0x0000001A000001060005 {CALL [dbo].[sp_MSupd_SalesLTProduct](,,1574.6500…
```

To simulate the Distribution Agent and retrieve pending transactions in the Distribution Database you can execute "**sp_browsereplcmds**" on the Distributor.

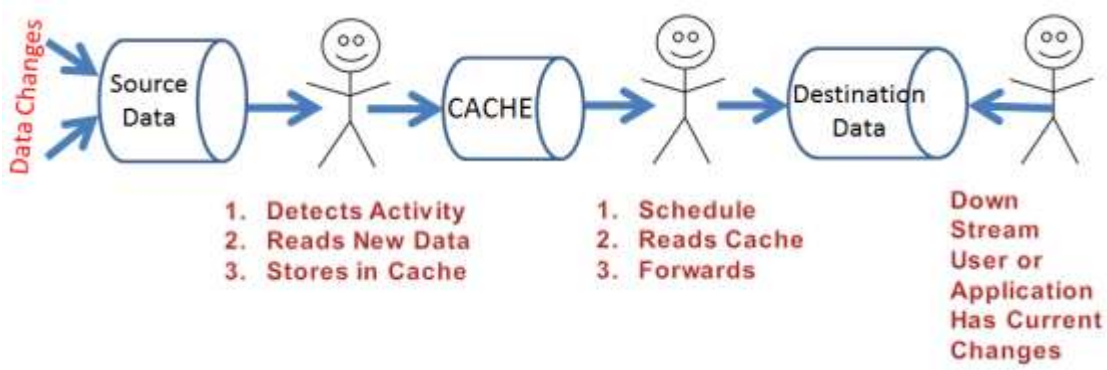
```
Use distribution
Go
sp_browsereplcmds

       0x00000039000000AC0005     {CALL [dbo].[sp_MSupd_SalesLTProduct]
(,,,1574.6500,,,,,,,,,,2010-01-22 15:31:13.170,680,0x200001)}
```

Replication Agents executing these commands are subject to the same design considerations of other SQL commands. For example, an UPDATE trigger on a subscriber table may cause delay in the Distribution Agent as trigger is fired for each update being applied. Additional indexes on the Subscriber to support ad-hoc reporting business need to be updated as Replication pushes down data changes.

Written by Chris Skorlinski, Edited by Gaurav Mathur
Microsoft SQL Server Escalation Services

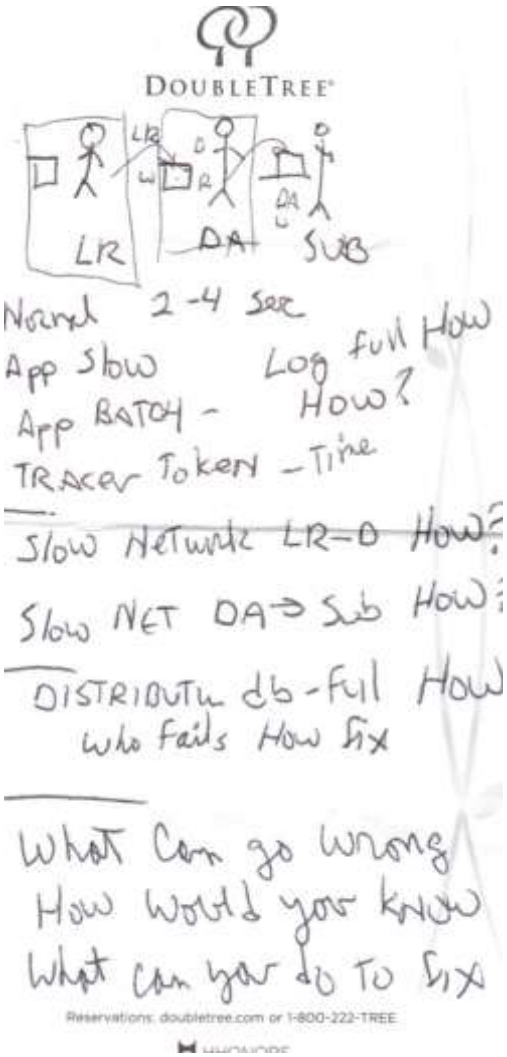
Wrap up

Many technology solutions use a STORE and FORWARD model to move data though out the business. It begins when source data change is detected, collected, and STORED in a cache. This cache, often on a schedule, is processed and FORWARD to one or more destinations. Microsoft technologies using “store and forward” include Log Shipping, Database Mirroring, Transactional Replication, Merge Replication, Change Data Capture, Change Tracking, Sync Services, SSIS ETL packages, and SQL Azure. As we “Follow the Data” look at the players, their roles, auditing options, supportability features, and ability to handle changing business needs.



Written by Chris Skorlinski, Edited by Gaurav Mathur
Microsoft SQL Server Escalation Services

How it all started



Ever wonder how ideas for technical papers get started? Do they really get started on a paper napkin in low lit restaurant. Okay, not on a napkin, sometimes it's on a hotel notepad. Here is the beginning of this technical paper, the Power Point presentations and blog postings which followed.

I was staying at the DoubleTree, little board, and started thinking about approaches for troubleshooting data latency when running Microsoft SQL Server Transactional Replication. I started to scribble down a few thoughts tracking the flow of data and breaking points. As I "followed the data", I noticed a similarity between Replication, Database Mirroring, Log Shipping, and various other technologies all of which have a similar "store and forward". I realized many of the Transactional Replication concepts and features discussed in this paper can also be applied to these technologies.

http://blogs.msdn.com/b/repltalk/

Transactional Replication – Follow the Data

http://www.sqlsaturday.com/viewsession.aspx?sat=87&sessionid=5248

Written by Chris Skorlinski, Edited by Gaurav Mathur
Microsoft SQL Server Escalation Services