Microsoft

SQL Server Replication: Providing High Availability using Database Mirroring

Writer: Gopal Ashok (Microsoft Corporation), Paul S. Randal (SQLskills.com)

Technical Reviewers: Hilary Cotter (Relevant Noise), Prem Mehra, Lindsey Allen, Sanjay Mishra, Glenn Berry (SQL Server MVP), Jimmy May, Mike Ruthruff, Michael Redman, Joseph Sack, Jean-Yves Devant, Mike Weiner, and Mike Habben

Technical Writer: Glenn Minch

Project Editor: Diana Steinmetz

Originally Published: August 2008

Last Updated: March 2011

Applies to: SQL Server 2008, SQL Server 2008 R2

Summary: This white paper describes how to use database mirroring to increase the availability of the replication stream in a transactional environment. The document covers setting up replication in a mirrored environment, the effect of mirroring partnership state changes, and the effect of mirroring failovers on replication. In addition, it describes how to use LSN-based initialization to recover from the failover of a mirrored Subscriber database.

Although brief overviews are given of both replication and database mirroring, it is easier to understand this white paper if you have some experience with one or both of these technologies, and have at least a rudimentary knowledge of database concepts such as transactions.

The information about architecture and Publisher failover are valid for SQL Server 2005; however, the section on Subscriber failover applies only to SQL Server 2008 and later.

Copyright

Contents

Introduction

Transactional replication is the mechanism that Microsoft® SQL Server® provides to publish incremental data and schema changes to subscribers. The changes are published (the replication stream) in the order in which they occur, and typically there is low latency between the time the change is made on the Publisher and the time the change takes effect on the Subscriber. This enables a number of scenarios, such as scaling out a query workload or propagating data from a central office to remote offices and vice-versa. This form of replication always uses a hierarchical hub and spoke topology.

The addition of peer-to-peer transactional replication in SQL Server 2005 simplifies the implementation of a bi-directional transactional replication topology, where the replication stream flows both ways. In this topology, any participating node may read or update the data. Properly partitioned modifications are propagated between all nodes in a full mesh topology (as shown in Figure 1), allowing the data to be highly available in the event that one server is unavailable. This feature has been further improved in SQL Server 2008 with conflict detection and online changes for peer-to-peer topologies.

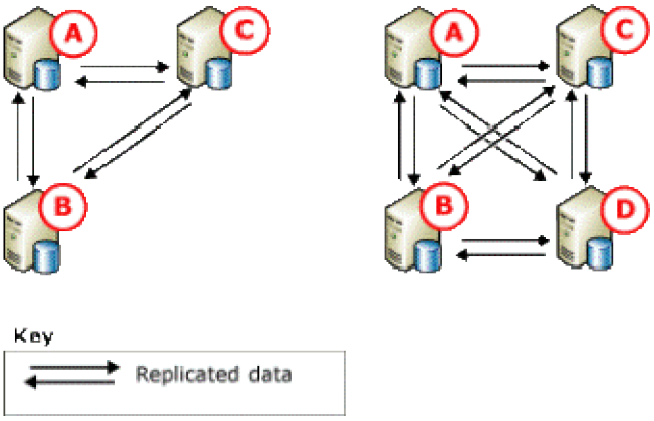

Figure 1: Full mesh topology for peer-to-peer replication with three and four nodes

Transactional replication topologies can be made more resilient to server failures, and hence more highly available, by adding redundant copies of the various databases involved. This is especially important for hub and spoke topologies. Care must be taken, however, because replication is reliant on the server names of the servers in the topology, so any failover to another server can result in the replication stream being broken.

Various mechanisms in SQL Server provide database-level redundancy, such as backup/restore, log shipping, and database mirroring (in SQL Server 2005 and later). Database

mirroring is the only mechanism that provides a real-time, exact copy of the protected database with the guarantee of zero data loss (when the mirror is synchronized).

This white paper describes how to use database mirroring to increase the availability of the replication stream in a transactional environment. It covers setting up replication in a mirrored environment, the effect of mirroring partnership state changes, and the effect that mirroring failovers have on replication. In addition, it describes how to use LSN-based initialization to recover from the failover of a mirrored subscriber database.

Technologies

Transactional Replication Architecture

Transactional replication and peer-to-peer replication use the same architecture to move changes between the servers in a replication topology. The following illustration is an overview of the components involved in transactional replication.

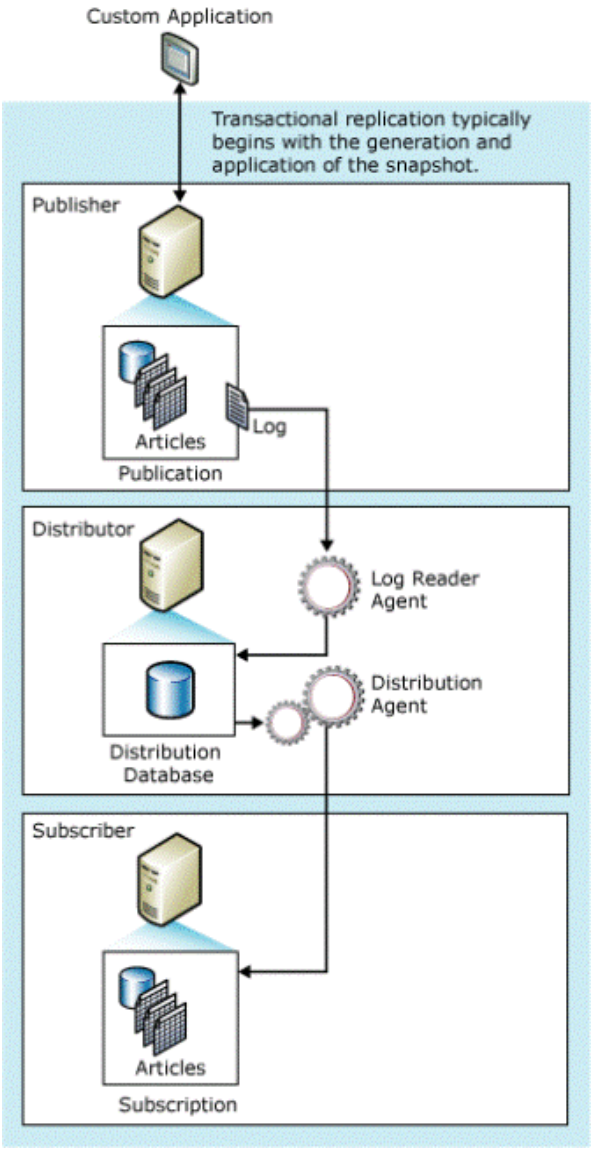

Figure 2: Transactional replication architecture overview

A minimum of three server roles are required for transactional replication:

- **Publisher,** hosting the publication database
- **Distributor,** hosting the distribution database
- **Subscriber,** hosting the subscription database

Depending on the complexity of the replication topology, there may be multiple Subscriber servers or, in the case of peer-to-peer replication, multiple peer servers with the replication stream flowing in both directions between the peers. Furthermore, the roles of the various replication servers can be played by one server or by individual servers (the more common case), and it is possible for a server to play any combination of roles. Regardless, the various

servers and databases must be protected to ensure that the replication stream is highly available.

Transactional replication relies on various agents to perform the tasks associated with tracking changes and distributing data. These agents are:

- **Snapshot Agent,** which runs at the Distributor. This agent prepares schema and initial data files of published tables and other objects, stores the snapshot files, and records information about synchronization in the distribution database.
- **Log Reader Agent,** which runs at the Distributor. This agent connects to the Publisher and moves transactions marked for replication from the transaction log of the publication database to the distribution database.
- **Distribution Agent,** which runs at the Distributor for push subscriptions, and at the Subscriber for pull subscriptions. This agent applies the (optional) initial snapshot to the Subscribers and moves transactions held in the distribution database to Subscribers
- **Queue Reader Agent,** which runs at the Distributor. This agent is only used for transactional replication with updateable subscriptions and moves changes made on the Subscribers back to the Publisher.

It should be mentioned that there is also a Merge Agent, but it is used only for merge replication, which is not covered in this paper.

This white paper focuses mainly on the Log Reader Agent and the Distribution Agent.

For more detailed information on SQL Server Replication, see the following "SQL Server Replication" topics in SQL Server Books Online:

- http://msdn.microsoft.com/en-us/library/ms151198(SQL.100).aspx for SQL Server 2008
- http://msdn.microsoft.com/en-us/library/ms151198(SQL.90).aspx for SQL Server 2005

Database Mirroring Architecture

Database mirroring works at the database level and provides a single copy of the mirrored database that must reside on a different server instance, usually on a separate physical server in a different location. One server instance serves the database to clients (the *principal server*). The other instance acts as a hot or warm standby server (the *mirror server*), depending on the configuration of the *database mirroring session* and the *mirroring state* of the mirrored databases. The two servers are said to be partners in the mirroring session.

When the mirrored database is synchronized, database mirroring provides a hot standby server that supports rapid failover without loss of data from committed transactions. When the database is not synchronized, the mirror server is typically available as a warm standby server (with possible data loss).

A database mirroring session runs with either synchronous or asynchronous operation. Under asynchronous operation (also called *high-performance* mode), transactions commit without waiting for the mirror server to write the log to disk, which maximizes performance. Under synchronous operation (also called *high-safety* mode), a transaction is committed on both

partners, but at the cost of increased transaction latency. In high-safety mode, it is possible to allow automatic failovers by adding a third *witness* server. In all other configurations, failovers must be performed manually.

The transaction safety level of a mirroring session is controlled by the SAFETY property of the ALTER DATABASE statement. Synchronous mirroring is when SAFETY is FULL; asynchronous is when SAFETY is OFF.

The following illustration shows an example database mirroring configuration.

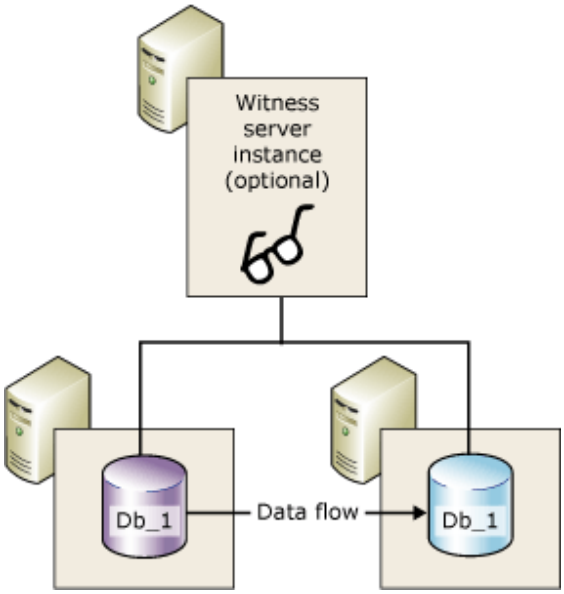

Figure 3: Example database mirroring configuration

In this example, the mirrored database is Db_1, with data flowing from the principal to the mirror server. Database mirroring maintains an exact copy of the database on the mirror server by *redoing* every insert, update, and delete operation that occurs on the principal database onto the mirror database as quickly as possible. Redoing is accomplished by sending every active transaction log record to the mirror server, which applies log records to the mirror database, in sequence, as quickly as possible. Unlike replication, which works at the transaction level (by harvesting and forwarding INSERT, UPDATE, and DELETE operations from the transaction log of the publication database), database mirroring works at the level of the physical log record (by sending the actual log records to the mirror server).

This white paper discusses mirroring the publication and subscription databases, and details the behavior of the various replication agents in each case.

For more detailed information on database mirroring, see the "Database Mirroring" topic in SQL Server Books Online:

- http://msdn.microsoft.com/en-us/library/bb934127(SQL.100).aspx for SQL Server 2008
- http://msdn.microsoft.com/en-us/library/ms177412.aspx for SQL Server 2005

Deploying Database Mirroring and Replication Together

The degree to which database mirroring can be combined with transactional replication depends on which replication database is being considered, as the level of support varies by database. Peer-to-peer replication with database mirroring is not supported. The following table lists the replication databases and corresponding level of integration with database mirroring.

Replication database	Integration with database mirroring
Distribution	No. Not a supported combination of technologies.
Publication	Yes, with automatic failover of replication agents.
Subscription	Yes, with manual failover and configuration of the replication agent.

Table 1: Mirroring support for the various replication databases.

The degree to which database mirroring is supported depends on whether the replication agents that connect to it are designed to cope with a mirroring failover.

The distribution database is the simplest to consider. The distribution database is also where the replication configuration is stored, and it is tightly coupled with the server name where replication is configured. Hence, any failover of the distribution database cannot be tolerated and so it cannot be mirrored. In the interests of high availability, this means that it is advisable to configure the Distributor to be on a different physical server from the Publisher (called a *remote Distributor*) rather than on the same server (called a *local Distributor*). If a remote Distributor is configured, it should be clustered for maximum availability.

All replication agents that connect to the publication database (that is, Snapshot Agent, Log Reader Agent, Queue Reader Agent, and for completeness, the Merge Agent) are mirroring-aware. They can be configured such that if a mirroring failover occurs, they automatically reconnect to the new principal server, and then replication continues. Therefore, mirroring the publication database is fully supported, but the state of the mirroring partnership and the specific failover scenario can affect the behavior of the Log Reader Agent. This is discussed more fully in the next section, Mirroring the Publication Database.

The subscription database is more complicated. None of the agents involved in replication are designed to cope with a failover of the subscription database. Prior to SQL Server 2008, the only way you could reliably restart the replication stream using supported methods was to drop and recreate the subscription to utilize the new principal Subscriber database, or you needed to bring back the original principal within the replication retention period. SQL Server 2008 added the capability to create a subscription to a Subscriber mirror using an LSN as the starting point from which to start synchronization. This allows you to use the data stored in the mirror as of the last synchronization before the principal failed. Initializing a subscription from an LSN can significantly reduce the time needed to return the Subscriber to service compared to full initialization of the Subscriber from a snapshot or backup. To use the method described in Mirroring the Subscription Database, the Publisher, Distributor, and Subscriber must all use SQL Server 2008 or later.

This white paper looks at some of the concepts, best practices, and steps that help redirect the Distribution Agent's subscription database connection to the mirror. These steps allow a mirrored subscription database to exist in the topology and a failover of the subscription database to be achieved with minimal downtime.

> **Important:** Care should be taken to ensure that database mirroring will perform as expected. For more information, see Database Mirroring Best Practices and Performance Considerations on Microsoft TechNet.

Mirroring the Publication Database

Configuring Replication with a Mirrored Publication Database

This section provides an overview of the steps to configure replication when the publication database is mirrored.

> **Note:** This mechanism does not work for peer-to-peer topologies, as the Publishers are also Subscribers.

For the purposes of this section, assume the following environment, with database mirroring already configured:

- SERVERPP – the database mirroring principal server and initial replication Publisher
- Database TicketOrdersPub – the publication database that is mirrored
- Publication name TicketOrders
- SERVERPM – the database mirroring mirror server
- SERVERD – the replication Distributor
- SERVERS – the replication Subscriber
- Database TicketOrdersSub – the subscription database

Replication can be configured by using either SQL Server Management Studio or Transact-SQL.

To configure replication with a mirrored publication database

1. Configure SERVERD as the Distributor for SERVERPP and SERVERPM

   It is highly recommended that you use a remote Distributor to increase the availability of the replication stream. If a local Distributor is used and the Publisher becomes unavailable, the replication stream is also unavailable.

   a. On SERVERD:
      - Configure SERVERD to be a Distributor, with a named distribution database and working directory.

- Add SERVERPP and SERVERPM as Publishers.

b. On both SERVERPP and SERVERPM, configure SERVERD to be the Distributor, using the distribution database and working directory previously specified.

Note: The principal and mirror must be configured to use the same Distributor, with identical settings.

For more information on configuring a Distributor, see the following SQL Server 2008 Books Online topics:

- How to: Configure Publishing and Distribution (Replication Transact-SQL Programming) for Transact-SQL
- How to: Configure Distribution (SQL Server Management Studio) for SQL Server Management Studio

2. Create a publication and configure replication:
   a. On SERVERPP, create a publication using snapshot, transactional, or merge replication.
   b. On SERVERS, create a push or pull subscription.

3. To configure the replication agents for failover, set the -**PublisherFailoverPartner** parameter for any of the following replication agents that exist:
   - Snapshot Agent
   - Log Reader Agent
   - Queue Reader Agent
   - Merge Agent
   - SQL Server Replication Listener (replisapi.dll)
   - SQL Merge ActiveX Control

   The recommended method for specifying this parameter is to include it in an agent profile. Create a custom profile by using a system-defined profile as a template, and then add this parameter to the profile. The custom profile can then be used when running the replication agents.

For more information, see Replication and Database Mirroring in SQL Server 2008 Books Online.

Effect of the Mirroring State on the Replication Log Reader

When a publication database is mirrored, Log Reader Agent behavior is governed by the mirroring state of the database. By default, the Log Reader Agent can only replicate log records that have been hardened in the log on the mirror server (the process in which the mirror server writes a transaction log record to the transaction log file is termed *hardening* the log). This means it cannot replicate log records with a Log Sequence Number (LSN) that is higher than the LSN of the last log record that was hardened on the mirror.

It is possible for the principal to become *exposed* (the mirror is accessible but there are log records that have not yet been hardened on the mirror) or *isolated* (the mirror is inaccessible). In both cases, if the principal is still able to serve the database, any changes made to the database are not replicated until the corresponding log records are hardened on the mirror.

This behavior introduces latency in the replication stream, and is done so that if a mirroring failover occurs, it is guaranteed that replication Subscribers cannot be "ahead" of the new principal database.

When the Log Reader Agent is waiting for transaction log records to be hardened on the mirror, the following message is entered in the Log Reader Agent history:

```
Replicated transactions are waiting for next Log backup or for
mirroring partner to catch up.
```

The following table lists the default Log Reader Agent behavior for various mirroring states. The information assumes that the principal is able to serve the database (either because there is no witness server, or there is a witness and there is a quorum).

Safety Level	Initial Mirroring State	Event	Resulting State	Principal Exposed?	Log Reader Agent Behavior
FULL or OFF	-	Mirroring partnership established	SYNCHRONIZING	N	Waiting (for synchronization to complete)
FULL or OFF	SYNCHRONIZING	Synchronization completes	SYNCHRONIZED	N	Running
FULL or OFF	SYNCHRONIZED	Session is paused	SUSPENDED	Y	Waiting
FULL or OFF	SYNCHRONIZED	Redo errors on the mirror	SUSPENDED	Y	Waiting
FULL or OFF	SYNCHRONIZED	Mirror unavailable	DISCONNECTED	Y	Waiting
FULL or OFF	SUSPENDED or DISCONNECTED	Session is resumed	SYNCHRONIZING	Y	Waiting (for synchronization to complete)
OFF	SYNCHRONIZED	SEND queue grows	SYNCHRONIZING	Y	Switching between Waiting and Running

Table 2: Default Log Reader Agent behavior in various mirroring states.

Changing the Replication Log Reader Behavior by Using Trace Flag 1448

In some situations, the effect of mirroring on the latency of the replication stream is not acceptable. A new trace flag, 1448, allows replication to continue even when the principal database is running exposed or is isolated. In such situations, the Log Reader Agent usually waits for log records to harden on the mirror before replicating them to the Distributor. When the

server is started with trace flag 1448 enabled, this restriction is removed so the Log Reader Agent can continue replicating changes regardless of the mirroring state.

The trace flag is available in SQL Server 2008, and as a hot-fix in SQL Server 2005. For more information on obtaining the fix that implements this trace flag in SQL Server 2005, and how to enable the trace flag in either version, see Knowledge Base article 937041.

> **Note:** Although replication latency may be of paramount importance, and so it may seem worthwhile to always have this trace flag enabled in configurations involving database mirroring and replication, be aware that there are some situations where using this trace flag could cause issues after a mirroring failover. These are explained in the next section.

Effect of a Mirroring Failover on the Replication Log Reader

When a mirroring failover occurs, whether automatically or manually, the Log Reader Agent should automatically connect to the new principal server and continue replicating transactions (as long as the –**PublisherFailoverPartner** parameter was set correctly, as described above).

There are two situations in which this may not happen. The first is where the failover does not succeed because the mirror server is unable to become the new principal for some reason. This is described in Log Reader Agent Behavior if the Mirroring Partnership is Broken later in this paper. The second case can only occur when trace flag 1448 is enabled, as described in this section.

In a mirroring session where transaction safety is set to OFF, or the principal is running exposed or isolated, it is possible for there to be committed transactions on the principal that have not yet been hardened on the mirror. If the principal database goes offline and the mirror is brought online by manually forcing service with failover, it is recovered with data loss (transactions that were committed on the principal but not yet hardened on the mirror are essentially lost to the application when the mirror becomes the new principal).

Furthermore, if trace flag 1448 is enabled and this situation arises, it is possible that some of the committed transactions from the old principal may have been replicated but not hardened on the mirror. This means that the Distributor is effectively “ahead” of the new principal when it comes online after the manual failover. This causes the Log Reader Agent to fail with the following message:

```
The process could not execute 'sp_repldone/sp_replcounters' on
'SERVERB'.
```

There will also be a more detailed message showing the log sequence number that the Log Reader Agent tried to read. This indicates that the Log Reader Agent has already retrieved log records further ahead in time than the new principal has in its transaction log.

The Log Reader Agent can be recovered by using the **sp_replrestart** stored procedure, which re-synchronizes the metadata between the Publisher and the Distributor. This can result in Subscribers potentially having more data than the Publisher. In the event that these changes

are made to the publication database again and subsequently replicated, the Distribution Agent will fail with data consistency errors. To override these errors use the Distribution Agent profile "Continue on data consistency errors."

> **Note:** Care should be taken when overriding these errors. For more information, see Skipping Errors in Transactional Replication in SQL Server 2008 Books Online.

Log Reader Agent Behavior if the Mirroring Partnership is Broken

As previously described, the Log Reader Agent will failover gracefully to the new mirroring principal server, providing the –**PublisherFailoverPartner** parameter is configured correctly. As long as the mirroring partnership remains in place, the Log Reader Agent can even be restarted after a mirroring failover and it will still connect to the correct server (the new principal server).

If the original principal server is likely to be unavailable for a long period of time, it is common practice to remove mirroring so that problems with transaction log growth on the mirror database do not occur. If the mirroring partnership is broken after a failover occurs, the Log Reader Agent continues to function as long as it is running at the time that mirroring is removed. If it is subsequently restarted, the Log Reader Agent tries to connect to the original principal server, which is no longer serving as the Publisher, and fails. The Log Reader Agent history will then contain a message such as:

```
User-specified agent parameter values: -Publisher SERVERA
```

There will also be more detailed messages about the connection failure.

To force this situation for testing purposes:

1. Configure database mirroring and replication.
2. Manually fail over from Server A to Server B.
3. On Server B, remove mirroring.
4. On Server D (the Distributor), stop and restart the Log Reader Agent job.
5. Observe the failure by looking at the job's history.

In this case, it is possible to easily make the Log Reader Agent continue to work by creating an alias for the original Publisher on the Distributor. The steps for doing this are as follows (continuing with the same test situation previously described), all on Server D.

1. Start SQL Server Configuration Manager, and then:
   a. Expand the SQL Native Client Configuration section.
   b. Right-click Aliases and add a new alias, from the original principal/Publisher to the new principal/Publisher. (In this example, the Alias Name is SERVERA and the Server is SERVERB.)
2. Restart the Log Reader Agent job.
3. Observe that replication continues to work as expected.

When it comes time to reestablish the mirroring partnership, the sequence of steps should be the following (continuing with the example situation):

1. Reestablish mirroring between Server B and Server A.
2. Delete the alias on the Distributor.
3. Restart the Log Reader Agent.
4. Optionally failover from Server B back to Server A.

Note: If the Distributor is not a dedicated server, the addition of an alias for the original mirroring principal server may cause problems for other applications running on the Distributor. For maximum flexibility, we recommend that you have a dedicated, remote Distributor.

Mirroring the Subscription Database

Even though you can mirror the subscription database, there is no support for automatic failover of the Distribution Agent if a mirroring failover occurs. After a mirroring failover of the subscription database, the Distribution Agent fails because it can no longer connect to the original subscription database on the original Subscriber.

Prior to the release of SQL Server 2008, mirroring the subscriber was not a practicable solution because the potential advantages of database mirroring were mostly negated by the requirement to perform a full initialization of the Subscriber after a failover.

The subscriber failover procedures described in this document take advantage of the **initialize from lsn** option introduced in SQL Server 2008. Creating the new subscription with **initialize from lsn** allows you to quickly synchronize the new subscription database with the publication database without resorting to initializing from a snapshot or backup, and without taking the publication or distribution databases offline. This technique enables you to integrate replication with database mirroring to achieve rapid failover with zero data loss.

Despite the advantages of mirroring the subscription database, a few words of caution are in order. All of the steps needed to perform a successful failover of a mirrored subscriber database are supported; however, the success or failure of an actual attempt to failover with **initialize from lsn** will depend on careful preparation and execution of the subscriber failover procedure described in this document.

Failover to the mirrored subscription database is a manual procedure. This is a relatively complex procedure, and to achieve reliability and zero data loss, you must perform all of the steps correctly and in the proper order.

To successfully initialize from an LSN, the following considerations apply:

- The Publisher, Distributor, and Subscriber must all be running SQL Server 2008 or higher.
- The distribution retention period must be set to an appropriate, non-zero value. For more information, see Configuring the Distribution Retention Period.

- To avoid lost transactions and inconsistencies between the publication and subscription, re-initialization of the subscription using *initialize from lsn* must occur before the distribution retention period expires.
- You must identify and record the LSN of the last update that was successfully applied to the mirrored (secondary) subscription database before the principal mirror failed. For more information, see How Does Initializing from an LSN Work.
- Ensure that no event occurs to cause the **min_autonosync_lsn** value in the publication database to be updated after the principal subscription mirror fails.

It is necessary to immediately halt activities that could cause **min_autonosync_lsn** to be updated before the subscription can fail over to the new database. These activities can be safely resumed after you have pointed the successfully initialized the new Subscriber. Activities that will cause **min_autonosync_lsn** to be updated include the following:

- Do not add a new article to the publication.
- Do not make schema changes to existing tables in the publication database.
- Do not enable the **allow_initialize_from_backup** option on the publisher. For more information, see How to: Initialize a Transactional Subscription from a Backup (Replication Transact-SQL Programming) in SQL Server Books Online.

If any of the conditions are not met, it may be impossible to initialize the subscription from an LSN without loss of data, and a full re-initialization of the Subscriber from a snapshot or backup will be required to restore the subscription database to a state consistent with the publication database.

The rest of this section describes how to recover a Distribution Agent that failed due to a subscription database failover, without having to perform a full re-initialization of the Subscriber.

For the purposes of this section, assume the following example environment:

- SERVERSP – the database mirroring principal server and initial replication Subscriber
- SERVERSM – the database mirroring mirror server
- SERVERD – the replication Distributor
- SERVERP – the replication Publisher
- Database TicketOrdersPub – the publication database
- Database TicketOrdersSub – the subscription database that is mirrored
- Publication name TicketOrders

The steps for configuring replication are similar to those previously described in Mirroring the Publication Database.

To configure replication

1. Configure SERVERD to be a Distributor and allow SERVERP to publish through it.
2. Configure SERVERP to use SERVERD as the remote distributor.
3. Configure the transaction retention period on the SERVERD (see the next section, Configuring the Distribution Retention Period).

4. Configure the publication on SERVERP.
   So that it will be easy to redirect the Distribution Agent to the new subscription database after a mirroring failover, the publication must be set to allow initialization from a backup. For instructions on setting this option when creating or modifying a publication, see the following:
   - How to: Initialize a Transactional Subscriber from a Backup for Transact-SQL
   - How to: Enable Initialization with a Backup for Transactional Publications for SQL Server Management Studio
5. Configure the subscription on SERVERSP, which can be either a push subscription or a pull subscription.

Database mirroring between SERVERSP and SERVERSM can be enabled before or after replication is configured, but if the Subscriber will be initialized with a snapshot, it is more efficient to configure mirroring *after* the subscription database has been initialized. This is to avoid all the logging activity as a result of the subscription initialization process being mirrored.

Configuring the Distribution Retention Period

The default minimum retention period for transactions on the Distributor is zero, which means that transactions can be cleaned up as soon as they have been replicated to all Subscribers. The steps required to redirect the Distribution Agent to the new subscription database after a mirroring failover rely on the availability of replicated transactions in the distribution database even after they have been sent to the Subscriber. There are two main reasons why this is required:

- If the subscription database is mirrored using a high-performance configuration, and if the mirror is far behind the principal, in the event of a failover with data loss, the recovered mirror (that is, the new subscription database) will be behind the original principal. To bring the new subscription database to a consistent state for replication, the Distribution Agent must re-apply all transactions that the new subscription database has not seen. This includes the set of transactions that were already applied to the original subscription database on the old mirroring principal server, but had not yet been mirrored. This avoids having to completely re-initialize the new subscription database.
- In the scenario above, the Publisher server may also be lost, or the data from the publication database is no longer available to re-initialize the new subscription database. In that case, data loss will occur if the missing transactions are not available on the Distributor.

To avoid data loss or extended downtime when mirroring the subscription database, we recommend that you set the minimum transaction retention period to a reasonable time period. Using the above example, a reasonable time period would be the maximum amount of time that the mirror server could be behind the principal server, plus additional time to complete the Subscriber fail over procedure, thus guaranteeing that no replicated transactions are lost.

For more information on configuring Distributor properties, see the following:

- How to: View and Modify Publisher and Distributor Properties for Transact-SQL

- How to: Set the Distribution Retention Period for Transactional Publications for SQL Server Management Studio

Manual Synchronization Types for Subscriptions

SQL Server 2005 introduced two new synchronization types that allow easier ways to create subscriptions by using manual synchronization (sometimes called *no-sync* subscriptions). These synchronization types were introduced to address two key aspects of manual synchronization:

- Automate the tasks of preparing the subscription database for replication. This involves automatically creating the objects required by replication when setting up a no-sync subscription.
- Reduce the down time required to set up a no-sync subscription using a backup. In SQL Server 2000, the Publisher had to be taken offline until all the steps related to setting up a subscription using a backup were completed.

The two synchronization types introduced in SQL Server 2005 were replication support only and initialize with backup. These are both widely known and used.

SQL Server 2008 introduces a new synchronization type, **initialize from lsn**, which is used internally to support online changes to a peer-to-peer topology. It is a powerful option that is similar to initializing from a backup except that initialization proceeds from a supplied LSN (a Log Sequence Number that identifies a point within a database's transaction log). This means the transactions that must be applied to the new subscription database are limited to those after the supplied LSN, providing the transactions still exist in the distribution database. This makes the option suitable for disaster recovery scenarios (such as in the example configuration used previously in this paper) because the initialization of the subscription database can be much faster than a full re-initialization, and the Distribution Agent starts from where it left off at the time of the mirroring failover.

After a mirroring failover of a subscription database to the mirror server, it is necessary to redirect the replication stream to the new subscription database on the new mirroring principal server (SERVERSM in the example configuration). Although this involves creating a new subscription, the subscription does not require a full re-initialization in this case, because the new synchronization option in SQL Server 2008 can be used.

The following section describes how initialization using an LSN (either user-specified or retrieved from a backup) works and how it enables the replication stream to be re-directed to the new subscription database after a mirroring failover without a full re-initialization.

How Does Initializing from an LSN Work?

In the simplest sense, all three manual synchronization methods rely on a consistent starting point for replicating transactions after the subscription has been set up. When a subscription is created using manual synchronization, the Distribution Agent must know from what point transactions should be replicated to guarantee that no changes are lost between the time the subscription is enabled and the changes are replicated.

This starting point is an LSN. The LSN is determined automatically in the case where the synchronization type is replication support only. With initializing from a backup, the LSN is retrieved from the header of the backup used to initialize the subscription. When initializing from an LSN, the LSN is user-specified.

When specifying the LSN, the important thing is to ensure that the LSN is valid so that the transactions corresponding to the LSN still exist in the distribution database. In a disaster recovery situation, this is why it is important that a non-zero transaction retention period is set. The following diagram and explanation show the various LSNs involved after the mirroring failover of a subscription database.

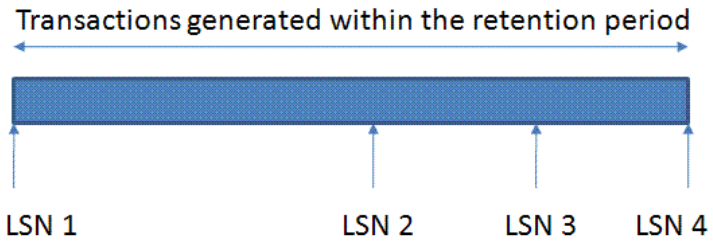

Figure 4: LSNs within the distribution database

Figure 4 shows the LSNs of interest within the distribution database after a mirroring failover of the subscription database. The LSNs are defined as follows:

- **LSN 1:** The LSN of the oldest transaction in the distribution database. This represents the transaction at the start of the retention period.
- **LSN 2:** The LSN of the last transaction that was applied to the subscription database on the mirroring principal server AND was mirrored to the mirror server.
- **LSN 3:** The LSN of the last transaction that was applied to the subscription database on the mirroring principal server.
- **LSN 4:** The most recent transaction that was read from the Publisher by the Log Reader Agent.

The difference between LSN 2 and LSN 3 represents how far the mirror server was “behind” the principal server before the mirroring failover. In the case where the principal and mirror were synchronized before the mirroring failover, LSN 2 and LSN 3 will be equal. When manually initializing the new subscription after the mirroring failover, LSN 2 should be used with the initialize from LSN method and can be found from the replication metadata in the subscription database.

There are two reasons why the LSN on the subscriber database from the current principal server (LSN 3) cannot be used to initialize the subscription:

- There are no LSNs greater than the specified LSN in the distribution database.
- The specified LSN must be less than the value of **min_autonosync_lsn** in the **syspublications** table in the publication database.

Note: The **min_autonosync_lsn** value is updated to the current database LSN whenever certain events occur; for more information, see Mirroring the Subscription Database. If the mirroring failover happens after the **min_autonosync_lsn** value was updated but before all changes after that point have been replicated, it is not possible to reinitialize by using the LSN value on the subscriber.

Care must be taken after a mirroring failover because transactions will continue to be cleaned from the distribution database, potentially allowing the transactions between LSN 2 and LSN 3 to be lost if the re-initialization of the new subscription database does not happen within the configured distribution retention period. You should allow extra time in the distribution retention period for the failure to be detected and the failover procedure to be completed.

If initializing from LSN is not possible, a full re-initialization of the subscription database must be performed by using a snapshot or a backup.

Recovering the Replication Stream Following a Mirroring Failover

This section presents the steps and code required to perform a manual initialization of the new subscription database after a mirroring failover, by initializing from an LSN. For clarity, these steps use the example environment previously described.

To recover the replication stream

1. To find the LSN of the last transaction that was applied to the new subscription database, run the following code on the new mirroring principal server (SERVERSM) after the mirroring failover, which will be the new Subscriber.

   This code examines the Distribution Agent information for the old subscription in the subscription database. The “old subscription” is the one existing on the mirroring principal server before the failover occurred (SERVERSP). The LSN returned (`transaction_timestamp` in the following code) should be saved, as it will be used in step 3, and corresponds to LSN 2 in Figure 4.

```
USE TicketOrdersSub;

GO

SELECT transaction_timestamp, *

FROM dbo.MSreplication_subscriptions

WHERE publisher     = 'SERVERP'

   AND publisher_db = 'TicketOrdersPub'

   AND publication  = 'TicketOrders';

GO
```

2. To clean up the old subscription from the new subscription database, run the following code on the new Subscriber.

   This code cleans up the old subscription so that there is no entry for it in the replication metadata in the subscription database. If it still exists when the new subscription is created, the transaction timestamp is reset to 0x00, resulting in all transactions from the distribution database being re-applied to the subscription database, causing data consistency errors.

```
USE TicketOrdersSub;
GO
EXEC sp_subscription_cleanup
   @publisher    = 'SERVERP',
   @publisher_db = 'TicketOrdersPub',
   @publication  = 'TicketOrders';
GO
```

3. To create the new subscription, run the following code on the Publisher (SERVERP).

   The code defines the new subscription on the new Subscriber, using the initialize from LSN synchronization method with the LSN/timestamp saved in step 1. This must be done for both pull and push subscriptions. Ensure that the correct subscription type is set in the `@subscription_type` parameter. Substitute the saved LSN/timestamp from step 1 as the value for the `@subscriptionlsn` parameter.

```
USE TicketOrdersPub;
GO
EXEC sp_addsubscription
   @publication       = N'TicketOrders',
   @subscriber        = N'SERVERSM',
   @destination_db    = N'TicketOrdersSub',
   @subscription_type = N'pull',
   @sync_type         = N'initialize from LSN',
   @article           = N'all',
   @update_mode       = N'read only',
   @subscriber_type   = 0,
   @subscriptionlsn   = 0x00000013000013380004000000000000;
GO
```

4. If the subscription type is pull, go to step 5. Otherwise, go to step 6.
5. To finish configuring the pull subscription, if appropriate, run the following code on the new Subscriber.
   This code finalizes creating a continuously running pull subscription.

```
USE TicketOrdersSub;
GO
EXEC sp_addpullsubscription
   @publisher         = N'SERVERP',
   @publication       = N'TicketOrders',
   @publisher_db      = N'TicketOrdersPub',
   @independent_agent = N'True',
   @subscription_type = N'pull',
   @description       = N'',
   @update_mode       = N'read only',
   @immediate_sync    = 0;
GO
EXEC sp_addpullsubscription_agent
   @publisher      = N'SERVERP',
   @publisher_db   = N'TicketOrdersPub',
   @publication    = N'TicketOrders',
   @distributor    = N'SERVERD',
   @frequency_type = 64
GO
```

Go to step 7.

6. To finish configuring the push subscription, if appropriate, run the following code on the Publisher.
   This code finalizes creating a continuously-running push subscription.

```
USE TicketOrdersPub;
GO
EXEC sp_addpushsubscription_agent
   @publication               = N'TicketOrders',
   @subscriber                = N'SERVERSM',
   @subscriber_db             = N'TicketOrdersSub',
   @job_login                 = NULL,
   @job_password              = NULL,
   @subscriber_security_mode  = 1,
   @frequency_type            = 64,
   @dts_package_location      = N'Distributor';
GO
```

Continue to step 7.

7. To remove the old subscription, run the following code on the Publisher.
   This code drops the old subscription (from SERVERSP) to ensure that its agents do not run automatically and cause problems with the new subscription.

```
USE TicketOrdersPub;
GO
EXEC sp_dropsubscription
   @publication    = N'TicketOrders',
   @subscriber     = N'SERVERSP',
   @destination_db = N'TicketOrdersPub',
   @article        = N'all';
GO
```

Conclusion

This white paper shows how database mirroring can be combined with transactional replication to achieve higher availability of the replication stream.

It describes the mechanism for mirroring the publication database, along with in-depth explanations of setup steps and options. The behavior of the Log Reader Agent is analyzed, and trace flag 1448 discussed. By using this trace flag it is possible for replication to continue from a mirrored publication database if the mirroring partnership is not synchronized.

The most important part of this white paper is the description of the initialize from LSN synchronization method introduced in SQL Server 2008. By using this new feature, it is possible to effectively mirror the subscription database without performing a full re-initialization in the event of a mirroring failover, leading to less downtime in the event of a disaster.

For more information:

http://www.microsoft.com/sqlserver/: SQL Server Web site

http://technet.microsoft.com/en-us/sqlserver/: SQL Server TechCenter

http://msdn.microsoft.com/en-us/sqlserver/: SQL Server DevCenter

Did this paper help you? Please give us your feedback. Tell us on a scale of 1 (poor) to 5 (excellent), how would you rate this paper and why have you given it this rating? For example:

- Are you rating it high due to having good examples, excellent screen shots, clear writing, or another reason?
- Are you rating it low due to poor examples, fuzzy screen shots, or unclear writing?

This feedback will help us improve the quality of white papers we release.

Send feedback.

Change History

Date	Description
March 7, 2011	• Added considerations for a successful failover to Manual Synchronization Types for Subscriptions. • Inserted a note in How Does Initializing from an LSN Work? to emphasize the importance of **min_autonosync_lsn** in the **syspublications** table in the publication database. • Altered the definition of "a reasonable time period" in Configuring the Distribution Retention Period to include additional time for completion of the failover process. • Inserted additional references to product documentation throughout the Mirroring the Subscription Database section. Numerous and sundry small changes to improve readability of the text.
March 7 – March 19, 2011	Technical review: revise, review, & repeat as required.
March 21, 2001	Draft completed.