# logo-sql08.gif

Failure Is Not an Option: Zero Data Loss and High Availability

SQL Server Technical Article

**Writer:** Thomas H. Grohser, SQL Server MVP

**Contributors:** Lindsey Allen, Sanjay Mishra

**Published:** November 2010

**Applies to:** SQL Server 2008, SQL Server 2008 R2

**Summary:** This paper describes how to combine SQL Server high availability technologies to provide a zero data loss, highly available solution for database applications involving financial transactions.

***Note****: This document is current as of its most recent publication date, November 1, 2010. Be sure to check www.sqlcat.com to obtain the most recent version of this paper.*

Copyright

The information contained in this document represents the current view of the author and/or Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in, or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

© 2010 Microsoft Corporation. All rights reserved.

Microsoft and SQL Server are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

Contents

[Introduction 4](#_Toc276211842)

[How to Use This Document 4](#_Toc276211843)

[Document Focus 4](#_Toc276211844)

[Document Prerequisites 4](#_Toc276211845)

[Understanding Zero Data Loss 4](#_Toc276211846)

[Understanding High Availability 4](#_Toc276211847)

[Service-Level Agreement 5](#_Toc276211848)

[Step 1 – Use Reliable Hardware 5](#_Toc276211849)

[Step 2 – Perform Regular Full Backups 7](#_Toc276211850)

[Step 3 – Take Transaction Log Backups 7](#_Toc276211851)

[Step 4 – Use Log Shipping to Restore the Log 8](#_Toc276211852)

[Step 5 – Implement a Database Mirror 9](#_Toc276211853)

[Step 6 – Implement a Mirror Witness 10](#_Toc276211854)

[Step 7 – Make Log Shipping and Backup Jobs Work with Mirroring 11](#_Toc276211855)

[Conclusion 12](#_Toc276211856)

# Introduction

When it comes to financial transactions of customers and their money, it is absolutely imperative that none of these transactions is ever lost due to an IT system failure (or any other type of failure). This white paper shows you how to create a Microsoft SQL Server infrastructure that enables operation for zero data loss while having high availability and performance in mind.

# How to Use This Document

After establishing the required service -level agreement (SLA), the document walks you from a single server installation that may lose 100 percent of its data and has absolutely no redundancy to an infrastructure across multiple data centers that provides zero data loss and high availability.

## Document Focus

This document is meant to provide general guidance on how to build zero-data-loss and high-availability solutions with SQL Server 2008 and SQL Server 2008 R2 using technologies like backup, log shipping, and database mirroring to create an appropriate infrastructure.

This document does not provide you with instructions or details on the technologies and features used; it focuses instead on demonstrating how to combine them.

## Document Prerequisites

Readers of this document should have a general understanding of database transactions, SQL Server backup (both database and log), log shipping, and database mirroring in order to better appreciate the concepts presented here.

# Understanding Zero Data Loss

*Zero data loss* means that all measures are taken so that if the worst assumed or any simpler failure occurs, no transaction that was committed to the customer is lost.

The key elements for a successful zero data loss implementation are:

* Reliable server, storage, and network infrastructure
* A reliable backup implementation[[1]](#footnote-1)
* Continuous and reliable transaction log backups
* Distribution of copies of the data across multiple data centers (preferably far away from each other)

## Understanding High Availability

*Highly Available* means that the required data in the database can be accessed whenever it is needed, not just the normal business hours. In other words, a database used only in an office is highly available if it is running every weekday from 9:00 A.M. to 5:00 P.M. It can be down all other days and during the night, and if you ask people that use it, they will tell you that it’s always there when they need it. It’s a completely different story with a database used in a 24x7 worldwide active online shopping site; even a failure at 3:00 A.M. can keep someone on the other side of the planet from working with the website.

This document shows you how to build an infrastructure that enables you to run a highly available solution, but it’s not a magic silver bullet. The environment and applications must also support high availability so that the solution at a whole will run as expected.

## Service-Level Agreement

Before you can start to implement any kind of high availability or reliability solution, you must establish the service-level agreement (SLA) with your customer.

At a minimum, you should establish answers to the following topics:

* Size of the database (current size, expected growth per month or year)
* Workload volume (how much transaction log is created on average and during peak times)
* RPO – Recovery Point Objective, or how much data can be lost
* RTO – Recovery Time Objective, or how much time is allowed to restore the database in case of a disastrous failure (this is different from high availability, because it specifies the time you have to recover from a backup if all high availability measurements fail)
* Desired time to fail over if a component fails
* Assumed worst-case scenario: for example, complete data center failure with loss of all data in the data center, or just a single server failure

For the scenario for which this paper builds an infrastructure, the RPO is as close to zero as possible, the failover time is seconds, and the worst-case scenario is the loss of a data center, as described earlier. The size of the database, the workload, and the RTO have no influence on the design of the solution. They do have an influence on the sizing of the servers, but we can assume the database is rather large and the workload is high. The design goals of this scenario are, in order of priority:

1. Zero data loss
2. High availability
3. Performance

# Step 1 – Use Reliable Hardware

Both zero data loss and high availability are implemented by failover technologies that can cope with the failure of a server, but failures cause downtime, and the risk of something going wrong increases with every incident. Therefore the more reliable each server is, the more robust the solution.

#### Each server should have the following:

**Redundant power supplies** that are connected to different electrical circuits. At least one of the circuits should be connected to a UPS (and it is best to connect all of them, each one to a different UPS).

**Redundant network cards** (adapter teaming) that are connected to different switches in the network, which on the other end is configured to transport the network traffic over redundant routes to the required destinations.

**A redundant disk subsystem** that uses either a redundant SAN (minimum of two HBAs, two fabrics, two SAN switches, and a storage array that is high available in itself) solution as shown in the following picture



or a RAID 1, RAID 10, or RAID 101 solution with your direct-attached SCSI disks and RAID controllers.



The resulting server is now highly reliable in the first data center (called Datacenter A for this example).

This is a great starting point, but based on the example SLA, the worst case scenario is complete data center loss, causing the customer to lose all the data.



Implementing the server as a cluster is an option; but a cluster does not protect the data any more than a single server -- there is only one copy of data. So it is best to design a solution that does not use clustering.

# Step 2 – Perform Regular Full Backups

The starting point for any data recovery is a reliable backup. Datacenter A will need a backup file server where a full database backup is performed every 24 hours.



For more information about implementing a reliable backup, see the white paper “A Technical Case Study: Fast and Reliable Backup and Restore of a Multi-Terabyte Database over the Network” at <http://sqlcat.com/whitepapers/archive/2009/08/13/a-technical-case-study-fast-and-reliable-backup-and-restore-of-a-vldb-over-the-network.aspx>.” This white paper recommends alternating the backups to two file servers where at least one of them is in a remote data center, as shown in the following figure.



If you implement all of the steps described in the white paper, you can reduce the possible data loss (worst-case scenario) from everything to about a maximum of 60 to 72 hours, depending on how long the verify process takes.

The availability is not even worth mentioning. If Datacenter A fails, the data is in Datacenter B, but there is no hardware to restore to. It could take two days to get the hardware and start recovering.

It’s not where you want to be, but it’s a starting point.

# Step 3 – Take Transaction Log Backups

The next step is to start backing up the transaction log. The transaction log backups are the most important backups after at least one successful full backup, because they enable you to close the gap between the last full backup and the time of the failure.

As with full backups, transaction log backups must be implemented in both data centers. However, alternating backups does not work in this case. The backup must occur in both locations at the same time. The BACKUP command has a built-in feature to write to multiple destinations at the same time, with the following syntax.

BACKUP LOG MyVLDB

TO DISK = '\\LOGBACKUP01\backup\MyVLDB00000001.trn'

MIRROR

TO DISK = '\\LOGBACKUP02\backup\MyVLDB00000001.trn'



You can also back up the logs to the database file servers themselves, but I normally place them on different servers for more predictable performance: the database backup file server is optimized to handle a few large files, and the transaction log backup file server is optimized to handle thousands of small files, so with two servers the two jobs don’t stand in each others’ way.

The frequency with which you back up the logs really depends on your business needs. Everything from once per hour to every minute might be sufficient. Check your SLA to see how much data you can afford to lose. If you cut this time in half, you have a starting point for the log backup frequency.

To minimize the chance of any data loss you can implement a technique I call continuous log backup. The easiest way to describe it is this: As soon as you finish a log backup, just start the next one.

# Step 4 – Use Log Shipping to Restore the Log

As with the full backup, just backing up the data is not enough. You need to make sure the transaction log backups are valid. The way to verify that the logs are valid is to restore them. You can restore each backed-up transaction log to a separate server.



The first one is performed with no delay, the second with a one hour[[2]](#footnote-2) delay. The first one is used for availability; the second is used if there is a logical problem like deleted data that needs to be recovered. If you encounter such a problem, you can easily roll forward the logs till the second before the data is damaged and then copy the data back to the main database. If you do not catch the problem in the delay time, you have to restore the last valid backup and start your recovery from there. However, if you don’t detect the problem within the delay time, the problem is not likely to be urgent.

Adding the log restore reduces the possible data loss to minutes (depending on the frequency of the log backup/restore cycle – you can assume the possible data loss to be about three times the duration of a log backup/restore cycle; meaning if you back up and restore your transaction logs every 5 minutes, you can lose about 15 minutes of data).[[3]](#footnote-3)

Adding the log restore also provides a tremendous boost: if the principal fails, the log shipping target can be brought online within minutes (with some possible data loss if a tail of the log backup on the principal was not performed after the system went offline).

# Step 5 – Implement a Database Mirror

Setting up a database mirror in the second data center in asynchronous mode reduces the possible data loss to seconds, and the use of synchronous database mirroring (FULL SAFETY mode) eliminates the possible loss of data completely. The solution discussed in this paper uses synchronous database mirroring.



Depending on how important zero data loss is compared to availability, if you really want zero data loss, the system must be taken offline as soon as the mirror is not available. In most cases, availability is more important. However, in the scenario this paper discusses, zero data loss is higher priority than high availability. Therefore, if either the principal or the mirror fails, the application is taken offline, and one of the log shipping secondaries is converted to the new synchronous mirror. Finally, the application is brought back online.

Here is a general guideline: if all four servers are running, everything is OK. If one server fails, you should try to come to a mirrored state that’s in sync as quickly as possible, and you should have one log shipping database with no delay. Try to restore the role of the delayed log shipping target as quickly as possible.

Note that the enhancements in Step 5 enable the system to meet the reliability goals established in the example SLA, but they do not improve availability. You still need to initiate the failover manually.

# Step 6 – Implement a Mirror Witness

The only way to increase availability is to implement a database mirror witness. The server hosting the witness must be located in a separate data center (Datacenter C in the example).



It is very important for the network connections from Datacenter A to Datacenter B, from Datacenter A to Datacenter C, and from Datacenter B to Datacenter C to be absolutely independent. If one data center fails, the connection between the other two data centers must continue to work.

# Step 7 – Make Log Shipping and Backup Jobs Work with Mirroring

Although log shipping and backup work fine, you still need to make one more adjustment: in the case of a failover to the mirror database, the full backup and the transaction log backups must be performed on the mirror server rather than on the principal server. (For more information, see the white paper Database Mirroring and Log Shipping Working Together at <http://sqlcat.com/whitepapers/archive/2008/01/21/database-mirroring-and-log-shipping-working-together.aspx>). Also, the former principal server should start trying to restore logs so that if the mirror is broken or out of sync for any reason, the logs are still applied.

So basically after a failover you must create the backup job on the new principal server and reconfigure log shipping. You should do this before you go into production.



To avoid hundreds of error messages while the infrastructure is operation in the normal configuration, place an additional job step in all agent jobs that have different behavior depending on the state of a mirrored database.

The job step basically checks to see whether the database is online or offline. Depending on the state, it continues with the next job step or fails gracefully (that is, without an error).

# Conclusion

This white paper provides an overview of a high availability and disaster recovery solution that provides a zero data loss solution for financial applications. Effective use of SQL Server technologies combined with good operational practices enables you to meet even the strictest service level agreements: failure is not an option.

**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](mailto:sqlfback@microsoft.com?subject=White%20Paper%20Feedback:%20Failure%20Is%20Not%20an%20Option:%20Zero%20Data%20Loss%20and%20High%20Availability).

1. For more information, see the white paper “A Technical Case Study: Fast and Reliable Backup and Restore of a Multi-Terabyte Database over the Network” at <http://sqlcat.com/whitepapers/archive/2009/08/13/a-technical-case-study-fast-and-reliable-backup-and-restore-of-a-vldb-over-the-network.aspx>. [↑](#footnote-ref-1)
2. This time depends on your business; it should be close to two times the time it would take to detect a logical error in the database caused by either human error or application error that results in data either being deleted or modified incorrectly. [↑](#footnote-ref-2)
3. If you see that a transaction log backup file is corrupt or can’t be applied, you should immediately start a differential backup of your database (except if a full backup is already running), so that you have a new starting point for your log restores. [↑](#footnote-ref-3)