

**Microsoft IT: A case study on Business Continuity and Disaster Recovery – SQL Server 2012**

**Writer:** Srinivas Venbakam Vengadam, Microsoft | Srikiran Tadikonda, Microsoft | Kishore B V Bondada, Microsoft

**Reviewed By:** Manoj Badgujar, Microsoft | Mandi Ohlinger, Microsoft

**Contributors:** Supriya Naik, Microsoft | Manoj Badgujar, Microsoft | Kishore B V Bondada, Microsoft |

Vijay Shankar, Microsoft

**Published**: August 2013

**Applies to**: SQL Server 2012, BCDR

**Summary**: This paper shares the approach used by Microsoft IT to do the following:

* Upgrade a database server from SQL Server 2008 R2 to SQL Server 2012.
* Set up BCDR for all the dependent applications and services with minimal downtime.

When the database sizes are huge, are transactional, and have replications from/to another SQL Server (that does not support available groups), you can use some of the alternatives discussed in this white paper to manage your downtimes and avoid replication re-initializations. This white paper walks through disaster scenarios related to a database server; it does not cover disaster scenarios of web or application servers.

This content is suitable for developers or architects and database administrators. It is assumed that readers of this white paper have basic knowledge of SQL Server 2012 AlwaysOn, SQL transactional replications, and SQL Server administration.

Copyright

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2013 Microsoft. All rights reserved.

Contents

[Introduction 4](#_Toc365580033)

[Business Continuity and Disaster Recovery (BC/DR) 4](#_Toc365580034)

[Legacy Architecture: Database Mirroring for High Availability, Log Shipping for Disaster Recovery 4](#_Toc365580035)

[Limitations 5](#_Toc365580036)

[Current Architecture: AlwaysOn Availability Groups for High Availability and Disaster Recovery 5](#_Toc365580037)

[Benefits of AlwaysOn Availability Groups in SQL Server 2012 6](#_Toc365580038)

[SQL Replication Techniques 7](#_Toc365580039)

[HomeDepot Case Study 7](#_Toc365580040)

[Context 7](#_Toc365580041)

[Objective 8](#_Toc365580042)

[BCDR on HomeDepot 8](#_Toc365580043)

[HomeDepot Architecture 9](#_Toc365580044)

[BCDR Testing 12](#_Toc365580045)

[Failover 12](#_Toc365580046)

[Failback 14](#_Toc365580047)

[Challenges 16](#_Toc365580048)

[Conclusion 16](#_Toc365580049)

[Appendix 16](#_Toc365580050)

[Acknowledgements 17](#_Toc365580051)

# Introduction

SQL Server 2012 added a set of features for High Availability (HA) and Disaster Recovery (DR). The AlwaysOn feature is now available in a cluster environment and can be used to implement Business Continuity and Disaster Recovery (BCDR) for critical applications and services. In addition to having the databases highly available, there are different strategies to support BCDR so that an enterprise application or service is available/recoverable with minimal downtimes and data loss.

This paper shares the approach used by Microsoft IT to do the following:

* Upgrade a database server from SQL Server 2008 R2 to SQL Server 2012.
* Set up BCDR for all the dependent applications and services with minimal downtime.

When the database sizes are huge, are transactional, and have replications from/to another SQL Server (that does not support available groups), you can use some of the alternatives discussed in this white paper to manage your downtimes and avoid replication reinitializations. This white paper walks through disaster scenarios related to a database server; it does not cover disaster scenarios of web or application servers.

This content is suitable for developers or architects and database administrators. It is assumed that readers of this white paper have basic knowledge of SQL Server 2012 AlwaysOn, SQL transactional replications, and SQL Server administration.

# Business Continuity and Disaster Recovery (BC/DR)

BCDR are closely related practices that enable organizations to prepare for any unexpected events such as earthquakes, terror attacks, national emergencies, and so on. In general, Disaster Recovery (DR) refers to the specific steps taken to resume operations in the aftermath of disaster. Business Continuity (BC) describes the processes and procedures an organization must put in place to ensure that mission-critical applications can continue during and after a disaster.

# Legacy Architecture: Database Mirroring for High Availability, Log Shipping for Disaster Recovery

Before SQL Server 2012, SQL Server deployment architecture uses Database Mirroring for high availability and uses Log Shipping for disaster recovery. Database Mirroring is configured within the primary data center and Log Shipping is configured within the cross-data center. To achieve automatic failover, synchronous database mirroring with a witness (a third SQL Server instance) is configured. When zero data loss is required, the database mirroring high-safety mode (synchronous) setting is enabled to help ensure zero data loss between the two servers located in the primary data center. To improve database availability within the primary data center, a third SQL Server instance is configured to act as a witness to enable automatic failover between the database mirroring partners.

If a primary data center outage renders both database mirroring partner instances unavailable, log shipping is used for disaster recovery. Log shipping involves ongoing transaction log backups of the principal databases. These transaction log backups are then copied to a SQL Server instance in the disaster recovery data center. Incoming transaction log backups are restored in sequence on an ongoing basis. You could also choose to configure log shipping for read-only workloads, but read-only connections must be disconnected before incoming transaction log backups can be applied. Figure 1 shows a representation of this solution architecture:

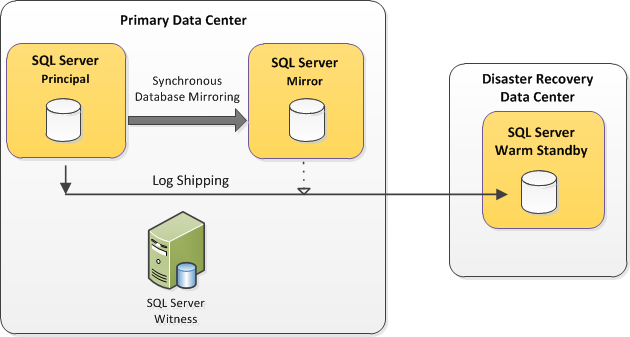


Figure 1: Database mirroring for high availability and log shipping for disaster recovery

More information on Database Mirroring can be found [here](http://msdn.microsoft.com/library/ms189852(v=sql.105).aspx) (<http://msdn.microsoft.com/library/ms189852(v=sql.105).aspx)> and Disaster Recovery can be found [here](http://msdn.microsoft.com/library/bb895393(v=sql.105).aspx) (http://msdn.microsoft.com/library/bb895393(v=sql.105).aspx).

## Limitations

1. Database Mirroring does not allow reads on secondary server.
2. Log Shipping is used across data centers. As a result, there can be data loss that is equal to the frequency at which log shipping job is scheduled.
3. Maintainability is low with log shipping because two jobs are created for every database on the server.
4. Log shipping does not support automatic failover.

# Current Architecture: AlwaysOn Availability Groups for High Availability and Disaster Recovery

BCDR in SQL Server 2012 supports AlwaysOn Availability groups. The AlwaysOn Availability Groups feature is a high-availability and disaster recovery solution that provides an enterprise-level alternative to database mirroring. AlwaysOn Availability Groups maximizes the availability of a set of user databases for an enterprise. An availability group supports a failover environment for a discrete set of user databases, known as availability databases that fail over together. An availability group supports a set of read-write primary databases and one-to-four sets of corresponding secondary databases. Optionally, secondary databases can be made available for read-only access and/or some backup operations. More information on AlwaysOn Availability Groups can be found [here](http://msdn.microsoft.com/library/ff877884.aspx) (http://msdn.microsoft.com/library/ff877884.aspx).

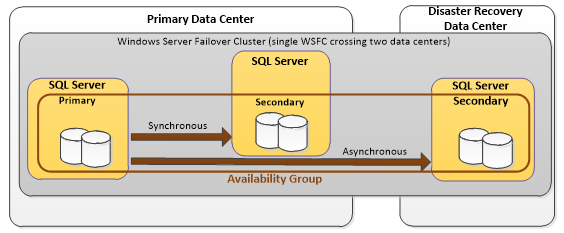


Figure 2: Availability groups and Disaster recovery in SQL Server 2012

# Benefits of AlwaysOn Availability Groups in SQL Server 2012

The following are some significant benefits of AlwaysOn Availability Groups in SQL Server 2012:

* More than one user database can be grouped into a single unit of failover. In contrast, database mirroring permits only one user database as a unit of failover.
* Availability groups multiple secondary replicas enables users to unify the HA and DR solution into one technology, instead of using multiple technologies used in SQL Server 2008 R2.
* Secondary replicas can also be configured to permit read-only workloads to obtain close to real-time data. Unlike with log shipping, ongoing read-only connections to the secondary replicas can be active to see ongoing data modifications against the primary replica. Secondary replicas can also be used to offload full database and transaction log backup operations.
* Availability groups and the associated availability group listener support automatic client redirection to the primary replica or redirection to available readable secondary replicas. Availability group listeners remove the need to designate a failover partner in the client connection string.
* Up to five availability replicas are supported. An availability replica is an instantiation of an availability group that host a specific instance of SQL Server and maintains a local copy of each availability database that belongs to the availability group. Each availability group supports one primary replica and up to four secondary replicas.
* Encryption and compression are supported; which provide a secure, high performing transport.

More information on the benefits is available [here](http://msdn.microsoft.com/library/hh510230.aspx#Benefits) (http://msdn.microsoft.com/library/hh510230.aspx#Benefits).

# SQL Replication Techniques

SQL Replication is a process of copying and distributing data and data objects from one database to another; and then synchronizing both the databases for consistency. Through SQL Replication, you can achieve high-availability, application performance, and so on.

There are three types of SQL Replications:

* Snapshot Replication
* Transactional Replication
* Merge Replication

Refer to the Appendix at the end of this white paper for details on Replication.

# HomeDepot Case Study

## Context

HomeDepot is a critical data warehouse server that distributes product data across all the Microsoft IT systems in the enterprise. Currently, it has around 1 TB of data, 7000 SQL objects, and acts as an upstream for various external systems, including enterprise data warehouse, SAP, and systems that exchange data that uses flat files in Excel. The HomeDepot server stores different types of databases, including static and Transactional (which include small and large databases). BCDR setup and testing has never been conducted for the HomeDepot server due to these multiple dependent systems. Loss of the HomeDepot server due to a disaster negatively impacts customers and can result in loss of key data. To maintain business continuity and better IT manageability, a replica server outside datacenter and a technical upgrade on all the databases are required.

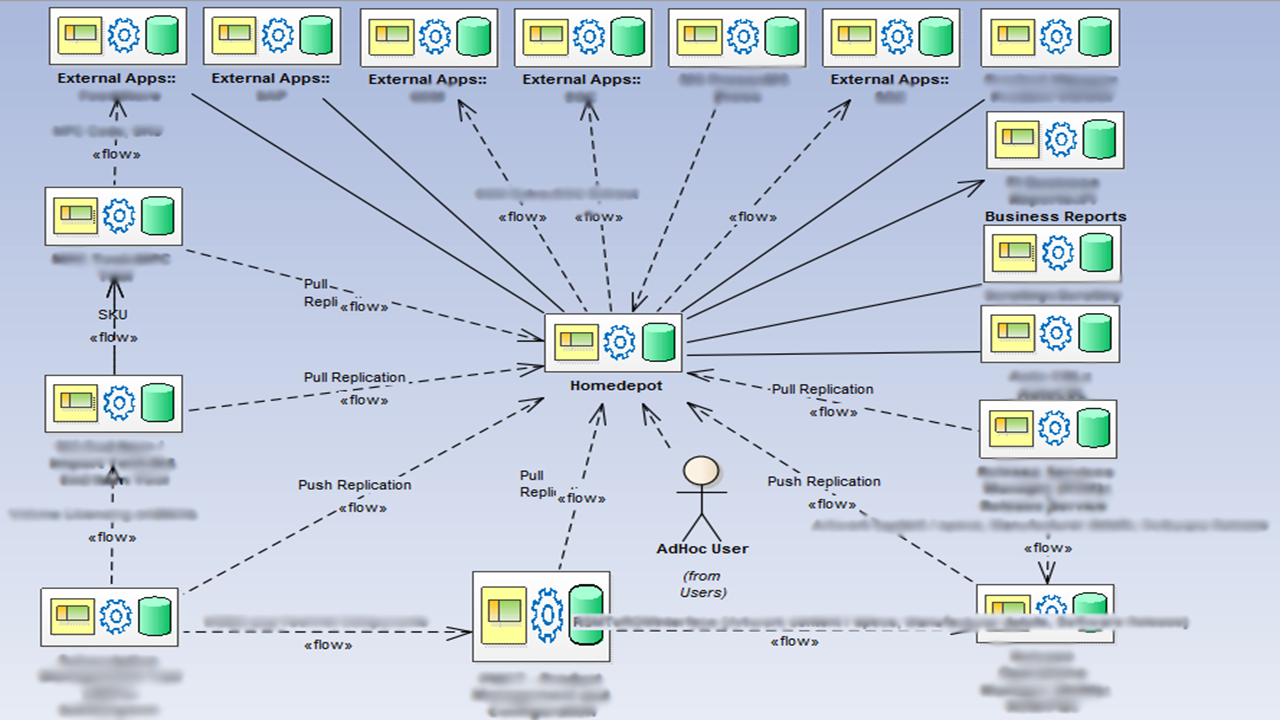


Figure 3: HomeDepot Server

## Objective

* Migrate the HomeDepot database server from SQL Server 2008 R2 to SQL Server 2012.
* Using Geographic Redundancy, enable High Availability (HA) and Disaster Recovery (DR) with minimal loss of data.
* Minimize downtime by reducing Recovery Time Objective (RTO) and Recovery Point Objective (RPO).

## BCDR on HomeDepot

Industry best practices, including failover clustering, database mirroring, log shipping, peer-to-peer replication, and AlwaysOn are considered:

**Failover Clustering** - A failover cluster is a group of independent computers that work together to increase the availability and scalability of clustered roles (formerly called clustered applications and services). The clustered servers (called nodes) connect to physical cables and by software. If one or more of the cluster nodes fail, other nodes begin to provide service (a process known as failover). The drawback of failover clustering is using shared-storage devices, like a storage area network (SAN); which could be a single point of failure.

**Database Mirroring –** Mirroring is a High availability solution used in SQL 2008 R2 but depreciated in SQL 2012.

**Log Shipping** - Log shipping allows you to automatically send transaction log backups from a primary database on a primary server instance to one or more secondary databases on separate secondary server instances. The transaction log backups are applied to each of the secondary databases individually. Log Shipping is used for Large and replicated databases in HomeDepot.

**Peer-to-peer replication** – Peer-to-peer replication provides a scale-out and high-availability solution by maintaining copies of data across multiple server instances, also referred to as *nodes*. Built on the foundation of transactional replication, peer-to-peer replication propagates transactionally consistent changes in near real-time. This enables applications that require scale-out of read operations to distribute the reads from clients across multiple nodes. Because data is maintained across the nodes in near real-time, peer-to-peer replication provides data redundancy, which increases the availability of data.

**AlwaysOn -** The AlwaysOn Availability Groups feature is a high-availability and disaster-recovery solution that provides an enterprise-level alternative to database mirroring. AlwaysOn Availability Groups maximizes the availability of a set of user databases for an enterprise. An availability group supports a failover environment for a discrete set of user databases, known as availability databases that fail over together. AlwaysOn is used for High Availability in HomeDepot.

To achieve High Availability, AlwaysOn is chosen. For Disaster Recovery, the replications on the secondary server are reinitialized using a Log Sequence Numbering (LSN)-based approach.

## HomeDepot Architecture

In HomeDepot, the databases are separated into three categories:

* **Static Databases**: Static Databases contain configuration tables that are used by other databases. They are not used by any external applications. HomeDepot has three small static databases.
* **Transactional Non-Replicated Databases**: Transactional Non-Replicated databases contain tables where external applications perform read-write operations. HomeDepot has such databases whose size ranges from 2 GB to 70 GB.
* **Transactional Replicated Databases**: Transactional Replicated Databases contain tables that get data from another database server (to which upstream applications feed data) through pull or push subscriptions. HomeDepot has seven such databases whose size ranges from 2 GB to 70 GB.

HomeDepot gets data from eight upstream applications and provides data to four downstream applications. HomeDepot has 18 SQL Jobs that perform data operations.

The architecture of HomeDepot server on SQL Server 2008 R2:

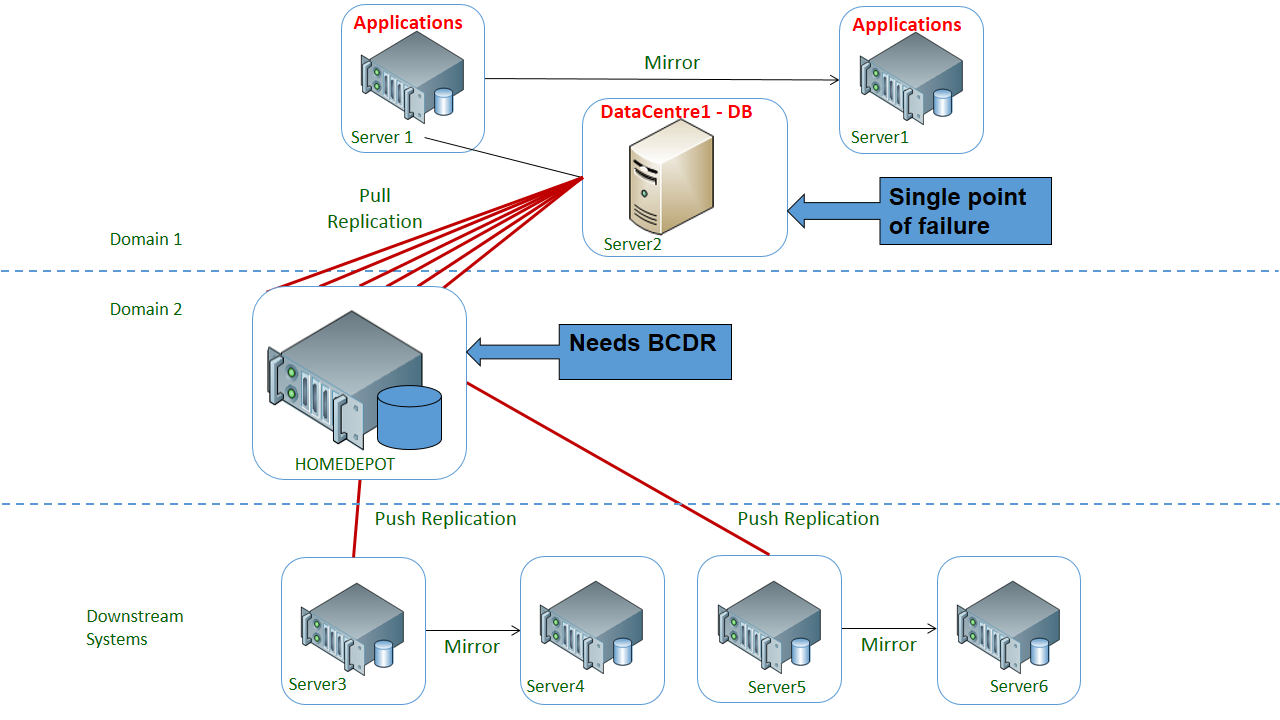


Figure 4: HomeDepot Server architecture before BCDR.

After migrating to SQL Server 2012 and enabling BCDR, the architecture consists of the following:

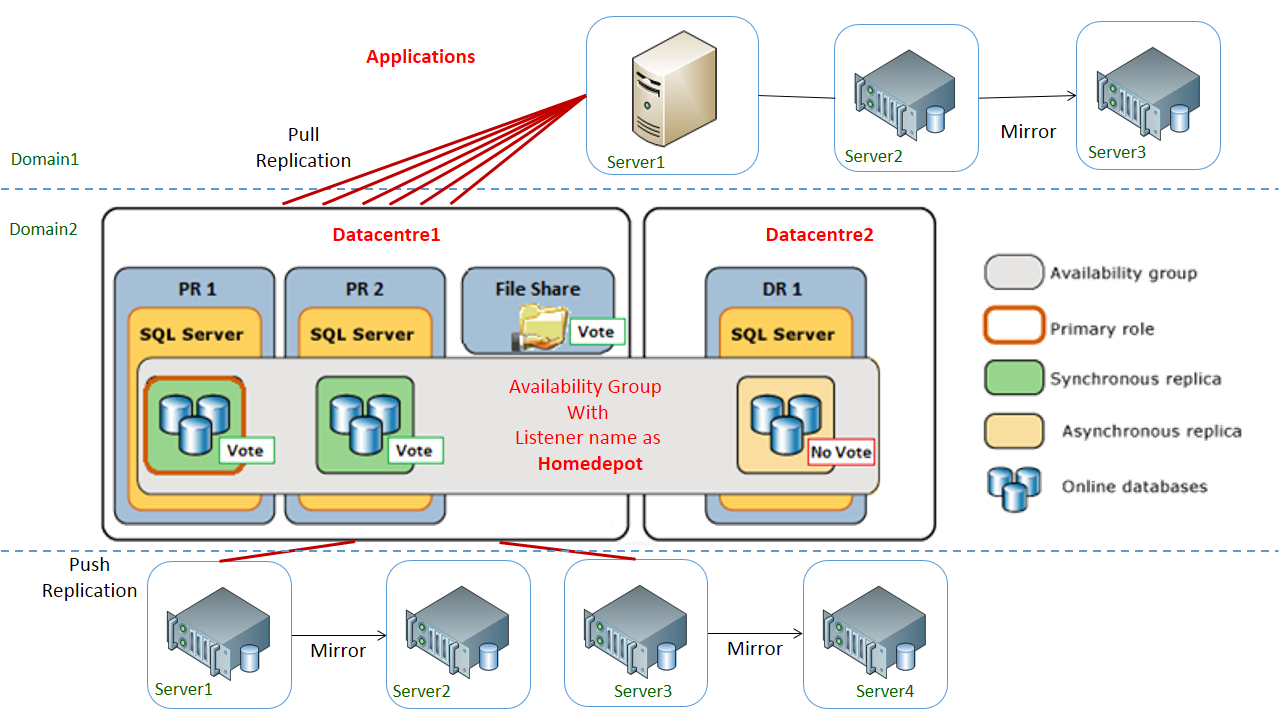
**

Figure 5: HomeDepot Server architecture after BCDR.

*Primary Server:* A primary HomeDepot server that contains all the databases and SQL objects is described in the Architecture.

*Primary Replica:* An exact copy of the primary HomeDepot server. The primary server and primary replica communicate with each other synchronously.

The primary server and primary replica exist in the same datacenter.

*Secondary Replica:* An exact copy of the primary server. It communicates with primary server in asynchronous mode. The secondary replica is kept in different datacenter to achieve Geo-redundancy.

HomeDepot uses a publisher-subscriber model for transactional replication.

The following steps are used to migrate from SQL Server 2008 R2 to SQL Server 2012:

1. Installed Windows Service Failover Cluster (WSFC) on the new HomeDepot server. This [blog](http://blogs.msdn.com/b/srinivas-v-v/archive/2013/06/25/setting-up-always-on-availability-group-on-sql-server-2012.aspx) (http://blogs.msdn.com/b/srinivas-v-v/archive/2013/06/25/setting-up-always-on-availability-group-on-sql-server-2012.aspx) lists the step-by-step procedures.
2. Configured the databases to use the Full recovery model with differential backups.
3. Copied the static and transactional nonreplicated databases backups and restored them on new servers using powershell scripts.
4. After restoring the databases on the new servers, restored SQL server-level logins and roles on the new SQL Server 2012 server using [sp\_help\_revlogin](http://support.microsoft.com/kb/918992) (http://support.microsoft.com/kb/918992).
5. SQL Agent jobs are scripted out and recreated in the new server, SSIS packages are redeployed with appropriate connection settings, and new file shares (DFS and non-DFS) are created on the new server.

The migration is complete and the new HomeDepot server is ready.

To set up the new environment for BCDR, did the following:

1. AlwaysOn feature is enabled on the primary server. The step-by-step procedures are listed at this [blog](http://blogs.msdn.com/b/srinivas-v-v/archive/2013/06/25/setting-up-always-on-availability-group-on-sql-server-2012.aspx) (http://blogs.msdn.com/b/srinivas-v-v/archive/2013/06/25/setting-up-always-on-availability-group-on-sql-server-2012.aspx).
2. For static databases, took a backup from the primary server and restored it on the primary replica and secondary replica using SQL server backup and restore feature. As static databases are of a small size, backup and restore took less time.
3. After databases are restored, they are in recovery mode. To bring them online, the “Restore database [$db] with RECOVERY” SQL Command is used.
4. Added these static databases to the Availability Group (AG) using PowerShell script; which can be done either with a PowerShell script or using SSMS.
5. For copying transactional replicated and nonreplicated databases to the primary replica and secondary replica, log shipping is used. For the step-by-step procedure used for log shipping, refer this [blog](http://blogs.msdn.com/b/srinivas-v-v/archive/2013/06/17/log-shipping-for-sql-server.aspx) (http://blogs.msdn.com/b/srinivas-v-v/archive/2013/06/17/log-shipping-for-sql-server.aspx) .
6. Break the log shipping. Bring all transactional replicated and nonreplicated databases on primary replica and secondary replica online.
7. Add these transactional replicated and nonreplicated databases to the AG using PowerShell script; which can be done either with a PowerShell script or using SSMS.
8. Set up replication on the transactional replicated databases. For the step-by-step procedure, refer to this [blog](http://blogs.msdn.com/b/srinivas-v-v/archive/2013/06/25/sql-server-replication.aspx) (http://blogs.msdn.com/b/srinivas-v-v/archive/2013/06/25/sql-server-replication.aspx).

## BCDR Testing

BCDR testing is performed to ensure that the system meets the Service Level Agreements (SLA), including a disaster scenario or when system maintenance requires the system to stop. BCDR testing is performed through Failover and Failback.

### Failover

When the primary server is down (due to a disaster a patching issue), failover to the secondary server or the disaster recovery server is completed. Then, drop the old subscriptions:

1. Check if the secondary server on the primary datacenter is up or the entire primary datacenter is down. If the primary server is down, the replications can break. During planned downtime windows (for example, patching operations), you can choose to wait until the primary server is up and running or reinitialize replication on the new primary server.
2. If the secondary server on the primary datacenter is still up and running, then execute the following command:

ALTER AVAILABILITY GROUP [$(ag\_name)] FAILOVER;

1. If the entire primary datacenter is down, force quorum on one of the DR nodes and ensure that the nodes in the primary data center do not form their own quorum.

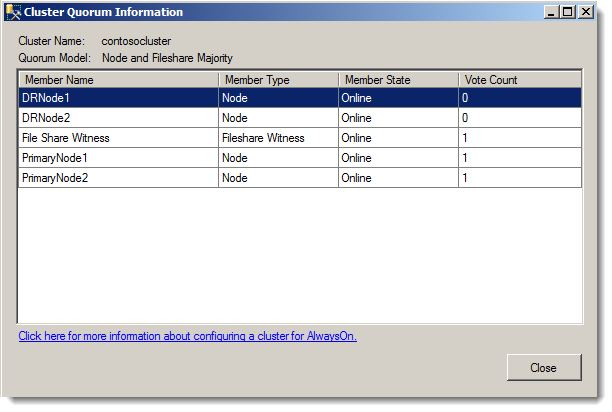
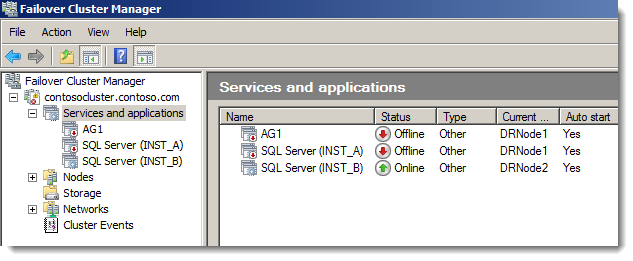


Figure 6: Vote count indicating primary server not forming its own quorum.

When you open Failover Cluster Manager on a disaster recovery node, it may not provide useful information (initially) on the state of the Failover Cluster because the cluster no longer has quorum:

To enable Quorum on the Failover Cluster node, execute the following command on DR node:

Start-ClusterNode –Name "DRNODE1" –FixQuorum

You should see output similar to the following:

Name State

------- --------

drnode1 Joining

**Note**: If the cluster service is still running on “DRNODE1”, you can use the following Windows PowerShell command to stop the service before you start the cluster service again with force quorum:

Stop-ClusterNode –Name "DRNODE1"

If a data loss risk is acceptable and the service must be restored at the data center, execute the following T-SQL syntax at the disaster recovery Failover Cluster Instance (FCI) to force failover:

ALTER AVAILABILITY GROUP [AG1] FORCE\_FAILOVER\_ALLOW\_DATA\_LOSS;

1. Modify the Windows Server Failover Cluster (WSFC) voting configuration on the DR Node.
2. Re-initialize the SQL replication using LSN on the new primary server. Follow this [blog](http://blogs.msdn.com/b/srinivas-v-v/archive/2013/06/25/re-initialization-with-log-sequence-numbering-lsn-for-sql-server-failover.aspx) (<http://blogs.msdn.com/b/srinivas-v-v/archive/2013/06/25/re-initialization-with-log-sequence-numbering-lsn-for-sql-server-failover.aspx>) for details.
3. Enable the SQL Agent jobs on the new primary server.

Note: If your server has dependent file shares, it is good to have them on Distributed File system (DFS). The advantage is that during failover and failback, the files shares are still available.

### Failback

When the primary data center issues are resolved and the nodes in the primary data center are enabled, the nodes attempt to connect to the Windows Server Failover Cluster (WSFC). After they are reconnected to the WSFC with the cluster services running, the node weights assigned at the disaster recovery node should be in effect. This scenario also assumes that the original SQL Server installations and associated databases are still intact.

Failback steps include:

1. Start the controlled migration back to the primary data center by adding back the quorum votes to the two primary data center nodes. After configuring this setting, verify that all nodes in the WSFC have a vote.
2. To resume each database that participates in the availability group, execute the T-SQL ALTER DATABASE commands on the primary data center Failover Cluster Instance (FCI):

ALTER DATABASE AppDB SET HADR RESUME;

GO

ALTER DATABASE ConfigDB SET HADR RESUME;

GO

ALTER DATABASE ReportDB SET HADR RESUME;

GO

1. To synchronize before failover, modify the availability group on the DR FCI to temporarily use the synchronous commit availability mode:

ALTER AVAILABILITY GROUP [AG1] MODIFY REPLICA ON N'SQLFCIPrimary\INST\_A' WITH (AVAILABILITY\_MODE = SYNCHRONOUS\_COMMIT);

**Tip**

The synchronous commit setting should be made during a period of low application activity to minimize the impact of transaction latency on users.

1. Confirm the synchronization status between the two locations:

SELECT role\_desc,synchronization\_health\_desc

FROM sys.dm\_hadr\_availability\_replica\_states;

Both replica states should say “healthy” before moving to the next step; meaning that both synchronous-commit replicas are synchronized.

1. To fail over from the disaster recovery data center FCI to the former primary data center FCI, connect and execute the following script on the primary data center FCI:

ALTER AVAILABILITY GROUP [AG1] FAILOVER;

The primary data center FCI becomes the new primary replica.

1. If your topology uses high-performance mode, as mentioned earlier, change the disaster recovery FCI replica node back to asynchronous commit. Execute the following Transact-SQL on the primary replica:

USE [master]

GO

ALTER AVAILABILITY GROUP [AG1]

MODIFY REPLICA ON N'SQLFCIDR\INST\_B' WITH

(AVAILABILITY\_MODE = ASYNCHRONOUS\_COMMIT);

GO

USE [master]

GO

ALTER AVAILABILITY GROUP [AG1]

MODIFY REPLICA ON N'SQLFCIPrimary\INST\_A' WITH

(AVAILABILITY\_MODE = ASYNCHRONOUS\_COMMIT);

GO

1. Remove quorum votes from the disaster recovery nodes.
2. Once the failback is complete, use the following steps for Log Sequence Numbering (LSN) to reinitialize the subscriber:
   1. Validate the LSN with the distributor.
   2. Clean up the new subscriber.
   3. Add subscription for the new subscriber on the publisher.
   4. Check the type of subscription and add appropriate subscription agents:
      1. Check the subscription agent at subscriber.
      2. Check the subscription agent at publisher.

This [blog](http://blogs.msdn.com/b/srinivas-v-v/archive/2013/06/25/re-initialization-with-log-sequence-numbering-lsn-for-sql-server-failover.aspx) (<http://blogs.msdn.com/b/srinivas-v-v/archive/2013/06/25/re-initialization-with-log-sequence-numbering-lsn-for-sql-server-failover.aspx>) lists the details and scripts used in this step.

1. Modify the WFSC voting configuration.
2. Configure DFS to point to the file location on new primary server.
3. Set up the replication on new primary server.
4. Enable the SQL Agent jobs on new primary server.

# Challenges

A split brain scenario is one of the challenges in this HomeDepot BCDR.

Split brain is a typical scenario that occurs during a network outage at the primary datacenter. The underlying databases at the primary datacenter are still online and are not able to communicate with the DR site.

A sample split brain scenario includes:

1. Failover to DR node takes place. The Availability Group (AG) is up and running on the DR node.
2. Voting configuration for DR and PR node is changed.
3. If the network comes back at the primary datacenter, both sites have their own copy of the availability group and the group is accessible.

This scenario can cause serious data consistency problems.

The only clean way to come out of this scenario involves the physical presence of an on-site support team. The support team manually takes the nodes at the primary data center down or out of network. Then, do manual changes before bringing up the nodes in a specific order with a quorum vote of 0 and prevent the quorum setting. But, this solution may not be a viable solution during a real disaster.

# Conclusion

This case study describes how a critical Microsoft IT database server is moved to SQL Server 2012 and BCDR is enabled for all the dependent applications and services.

SQL Server 2012 has significant enhancements to support High availability scenarios. This white paper describes the enhancements to the High availability and Disaster Recovery features of SQL Server 2012 and explained approaches related to failover/failback for Business Continuity and Disaster Recovery. This white paper also enlists different dependencies on a typical enterprise database server, like databases (categorized as large, transactional, static, and small), Server objects (like SQL jobs, logins/roles, linked servers, file system dependencies, SSIS packages), replications from a prior version of SQL Server that move large data from/to SQL Server, applications connecting to the database server, and different strategies to allow fail-over and failback of applications/services during a disaster. You can refer to the blog references in this white paper to learn the step-by-step processes for different aspects of SQL Server 2012 AlwaysOn.

# Appendix

**Recovery Time Objective (RTO):** Recovery time objective is the time period in which a system has to be restored after disaster. Any further delay in restoring the system can result in loss of information.

**Recovery Point Objective (RPO):** Recovery point objective defines the maximum tolerable time limit after which data might be lost. Reduced value of RPO refers to the need for high availability.

**Snapshot Replication:** Snapshot replication distributes data exactly as it appears at a specific moment in time and does not monitor for updates to the data. Snapshot replication is best used to replicate data that changes infrequently or when the most current values (low latency) are not required. When synchronization occurs, the entire snapshot is generated and sent to Subscribers. For more details on snapshot replication, refer to [this](http://msdn.microsoft.com/library/ms151832.aspx) (http://msdn.microsoft.com/library/ms151832.aspx).

**Transactional Replication:** Transactional replication typically starts with a snapshot of the publication database objects and data. As soon as the initial snapshot is taken, subsequent data changes and schema modifications made at the Publisher are delivered to the Subscriber as they occur (in near real time). The data changes are applied to the Subscriber in the same order and within the same transaction boundaries as they occurred at the Publisher. Within a publication, transactional consistency is guaranteed. For more details on Transactional replication, refer to [this](http://msdn.microsoft.com/library/ms151176.aspx) (http://msdn.microsoft.com/library/ms151176.aspx).

**Merge Replication:** Merge replication, like transactional replication, typically starts with a snapshot of the publication database objects and data. Subsequent data changes and schema modifications made at the Publisher and Subscribers are tracked with triggers. When connected to the network, the Subscriber synchronizes with the Publisher and exchanges all rows that have changed between the Publisher and Subscriber since the last synchronization occurred. For more details on Transactional replication, refer to [this](http://msdn.microsoft.com/library/ms152746.aspx) (http://msdn.microsoft.com/library/ms152746.aspx).

# Acknowledgements

Many thanks for the technical information and input provided by Kishore B V Bondada, Manoj Badgujar, Supriya Naik and Vijay Shankar. We would like to acknowledge the leadership and support provided by Kishore B V Bondada. A special thanks to Mandi Ohlinger for the technical review and valuable insights on this white paper.

**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:%20[Title%20of%20White%20Paper]).