

LEGATO RepliStor®

LEGATO solutions



Maintaining Data Integrity through Email and Database Replication

The benefits gained from replicating business critical applications like Microsoft Exchange, Microsoft SQL Server and Oracle are numerous. Not only can businesses deliver the critical disaster protection these environments require, they can also realize significant cost savings, administration efficiencies, and application efficiencies through data consolidation for tape backup. However, these gains are lost if the replication solution does not preserve database integrity through the replication process.

LEGATO understands this and has designed RepliStor® to preserve write-order as part of the replication process so that database integrity is always maintained.

RepliStor's Write-Order Preservation Capability Maintains Database Consistency
When you replicate databases like SQL Server and Oracle, and even for some types of complex applications like Exchange, you need to ensure that the data remains consistent throughout the replication process. In other words, the replication process must write changes in the same order on the target as was written by the applications on the source.

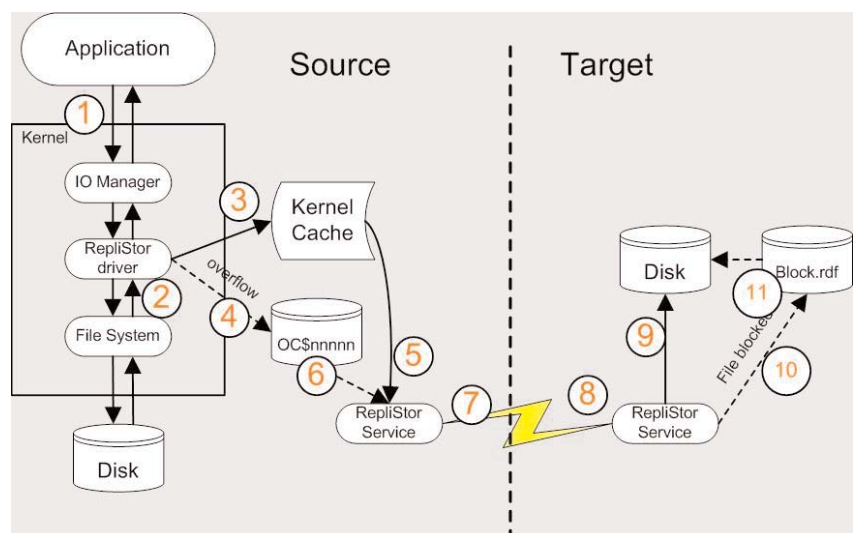
To ensure write-order consistency from the source to the target, RepliStor was designed to immediately replicate the I/O stream on the source system as it is written by the Windows kernel. RepliStor preserves it in a local cache, and then sends it in the same order to the target system where it is replayed – in the same order as was done on the source system.

The “secret sauce” in RepliStor's process, is in capturing the I/O data at the time that it is written instead of going back to capture it at the time that it will be sent over the network

like other products do. Capturing the data at the time that it is written ensures that RepliStor is following the exact order of how data is written by the application.

For example, if RepliStor is being used to replicate SQL Server data, then as SQL Server makes updates to its transaction log and index files along with its database file, RepliStor will capture those changes in the exact order that they are written and replay them in the same write-order on the target system.

In this way, RepliStor preserves disk operation ordering from the source system to the replicated target system within each file, across multiple files, and across RepliStor specifications. RepliStor's capability even maintains ordering when databases are stored “striped” across multiple LUNs, so that your options for configuring systems are as flexible as ever.



The above figure illustrates how RepliStor works with the Windows Kernel to capture file operations and data in the order that they are written on the source so that write-order is preserved in the replication process to the target system.

RepliStor Works with SQL Server and Oracle to Ensure Transactional Integrity in the Event of Server Failure

Databases like SQL Server and Oracle are designed to preserve transactional integrity when system failures occur. These applications do their best to preserve data integrity by making updates to the database in a consistent, specific sequence. When the database is opened after a crash, the application service may find that the database is corrupt since the crash may have occurred mid-transaction. In this event, the application service automatically rolls the database back to the last completed transaction in order to repair the database and make it usable.

Since RepliStor preserves write-ordering from the source to the target system, the same restart sequence applies when using RepliStor's failover capabilities. If the source can fail mid-transaction, then the data replicated to the target might also be mid-transaction at the time of the failure. In this case, when RepliStor restarts SQL Server or Oracle services on the target system, then those services open and repair the database by rolling it back to the last completed transaction just as would be done in a manual restart on the source system.

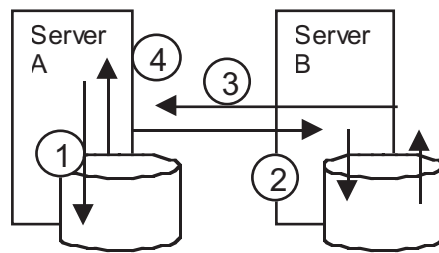
Asynchronous Replication Does Not Compromise Data Integrity

A common misconception is that asynchronous data replication will somehow compromise data integrity. This is untrue; whether the data movement is asynchronous or synchronous does not affect whether or not the data is preserved with integrity. The key is that the data must be written in the same order, from the source to the target, as it is replicated.

Understanding Synchronous Data Movement

Synchronous data movement, often called mirroring, is a method of replicating data in which the file system changes must be known to be committed on the target system before they will be fully committed on the source system.

The figure below illustrates how synchronous data movement operates:



1. Application, such as SQL Server, performs a write operation.
2. Synchronous mirroring, such as LEGATO Co-StandbyServer AAdvanced, captures the change and writes it immediately to a second disk.
3. An acknowledgement of the write from the second disk is received on the first server.
4. The application is allowed to continue processing.

As in the LEGATO Co-StandbyServer AAdvanced solution, this process ensures that the target system and the source system are always writing data from the same moment in time and with the source system never getting ahead of the target system.

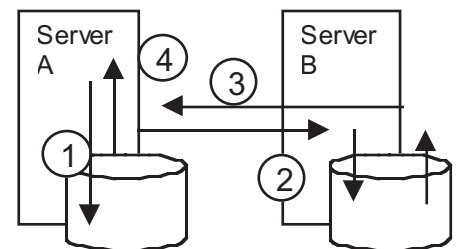
Synchronous movement of data is always preferable to asynchronous because it is always up-to-date. This makes synchronous mirroring the method of choice for local high availability clustering solutions, including Co-StandbyServer AAdvanced. However, synchronous movement of data is not always possible to use.

When data needs to be moved over long distances such as for disaster recovery or data consolidation, then waiting for the acknowledgement to be sent over the network puts too much latency into the process for synchronous data movement to be practical. This latency, which for the most part is attributable to normal TCP/IP "chatter", slows applications too much for end-users to work normally. For wide-area data movement, therefore, asynchronous data movement is the method of choice.

Understanding Asynchronous Data Movement

Asynchronous data movement, often called replication, is a method of replicating data in which the file system changes do not need to be committed on the target system before the source system is allowed to continue. Instead, the source system is allowed to operate normally and as a parallel operation the data is replicated as quickly to the target system as the network and target system will allow.

The figure below illustrates how asynchronous data movement operates:



1. Application, such as SQL Server, performs a write operation and the local application is allowed to continue without any delay.
2. Asynchronous replication, such as LEGATO RepliStor, captures the change immediately, caches it locally, and then forwards it to the second server.
3. An acknowledgement of the write from the second disk is received on the first server.

Because RepliStor is an asynchronous replication product which allows replication over great network distances between the source and the target system, the data that has reached the target at the time of the failure may be slightly behind the source system. How far behind the data is depends on a number of factors, including available network bandwidth and amount of data being sent.

However, even in cases in which the data may have fallen behind the source, because RepliStor preserves write-order it can always restart database services smoothly, quickly, and with usable data.

For example, in the case of a SQL Server database, let's assume that Server A performs a series of SQL Server operations. If the server fails, then RepliStor will restart SQL Server on Server B. SQL Server opens the replicated copy of the database on Server B and, if the failure occurred when Server B was mid-transaction, then SQL Server repairs its database by rolling back to the previous commit.

Whether you are looking to replicate your critical email or database application for consolidated backup, disaster recovery or content publication across your enterprise, you can be confident that RepliStor will get your data where you need it with the utmost integrity.

System Requirements

Each RepliStor protected server should satisfy the following software requirements:

- Microsoft Windows NT 4.0 Server or Enterprise Edition
- Microsoft Windows 2000 Standard, Advanced or Datacenter Edition
- Microsoft Windows 2003 Server, Enterprise or Datacenter Edition
- NAS Devices with Microsoft 2000 SAK or Windows Storage Server 2003



LEGATO Software, a division of EMC

2350 West El Camino Real, Mountain View, CA 94040 USA

Tel (650) 210.7000 • (888) 853.4286 | Fax (650) 210.7032 | www.legato.com

For a complete listing of LEGATO Software offices worldwide, please visit <http://www.legato.com/offices/>

LEGATO and the LEGATO logo are registered trademarks, and LEGATO NetWorker, NetWorker, Co-StandbyServer, RepliStor, SnapShotServer, QuikStartz, AlphaStor, ClientPak, Xtender, XtenderSolutions, DiskXtender, ApplicationXtender, ArchiveXtender, EmailXtender, and EmailXaminer are trademarks or registered trademarks of LEGATO Software. This is a nonexhaustive list of LEGATO trademarks, and other trademarks may be the property of their respective owners.

Information regarding products, services and offerings may be superseded by subsequent documents. For the latest information and specifications regarding LEGATO Software and any of its offerings or services, please contact your local sales office or the Division Headquarters. ©2003 LEGATO Software. Printed in the U.S.A.