Module 4: Tips, Tuning and Troubleshooting
Learning Objectives

At the end of this module, you will understand:

- Minimizing initial synchronization time
- DataKeeper™ registry settings
- The DataKeeper™ command line interface
- User-reported issues and resolutions
- Known restrictions
Local Initial Synchronization
In some situations, site-to-site bandwidth will be sufficient to address peak and sustained average rates of change, but insufficient to support initial synchronization in a reasonable amount of time. To address this initial synchronization process, we recommend relocating the target system to the primary site to perform the initial synchronization over the LAN. It should occur much more quickly.

This solution is not a panacea that will address all issues. Performing initial mirror synchronization over a LAN addresses the challenge once. If the mirror is subsequently broken or deleted and recreated, a full synchronization of the mirror must be performed.
To set up initial synchronization over a LAN, configure both systems with their final IP v4 configuration, including IP address, netmask, default route and DNS server(s).
Create routes on both the source and target systems for the remote and local subnet respectively.

Add a route to each system that uses its partner as the gateway for the remote subnet. The syntax is:

```
route add <remote subnet> mask <local subnet mask> <remote system IP address> metric 1
```

For example:

On the source system:
```
route add 192.168.252.0 mask 255.255.255.0 192.168.252.98 metric 1
```

and
```
route add 192.168.252.98 mask 255.255.255.0 192.168.252.0 metric 1
```
On the target system:

route add 192.168.251.0 mask 255.255.255.0 192.168.251.98 metric 1

Confirm that each system can ping the other.
Once the routes have been created and tested, start the DataKeeper GUI and connect to all servers. Use the DataKeeper GUI to create the job and mirror(s) with the appropriate compression settings. Wait until the initial synchronization is complete. Shut the target system down, without pausing the mirror. Delete the route to the target system from the source system.
Relocate the target system to the remote site. Start the target system on the remote site subnet. There is no need to delete the route since it was not persistent across a reboot. Once the source and target can communicate again, all mirror(s) should automatically perform partial resynchronization from the source to the target.
Performance Optimization
DataKeeper uses “zlib” for compression. Zlib is the defacto standard compression mechanism. It supports ten levels of compression, from none or 0 to 9. As the level is raised, the compression algorithm becomes more aggressive. Network packet compression can have a major, positive impact on wide area network throughput, particularly when bandwidth is constrained. Compression provides little, if any value when bandwidth exceeds 45Mbps. At bandwidths at and below 45Mbps, the greatest benefit may be achieved by setting compression level 1. Depending upon the compressibility of the data, bandwidth utilization may decline by a factor of 2 to 4. SteelEye has found that, on average, a 30% to 40% improvement may be achieved.

At low levels of compression, there is little processor overhead (<3%). At the highest level
of compression, processor overhead can exceed 10%. Clearly, there is a tradeoff to be made between the cost of bandwidth and processor utilization. SteelEye has found that at different rates of change and available bandwidths there is a “knee” in the replication performance curve where higher compression settings do not yield additional throughput. Ideally, the user should test varying levels of compression against throughput and processor utilization to obtain the optimal compression level for the replication workload at hand.

It should be noted that compression is not a panacea for low bandwidth. It cannot and should not be expected to resolve situations where the rate of change continuously exceeds the bandwidth between source and target servers.
Each mirror may have its bandwidth utilization throttled to balance individual mirror bandwidth consumption against the requirements of all mirrors and/or network consumers.
DataKeeper’s driver parameters are located in the registry under:

```
HKEY_LOCAL_MACHINE\SYSTEM\Current
ControlSet\Services\ExtMirr\Par
ameters
```

When replicating a volume, parameter settings may need to be altered to improve performance.
The first registry parameter is “DontFlushAsynchQueue”.

SteelEye has found that some Windows applications flush the NTFS buffer cache regularly to ensure on-disk data consistency. The DataKeeper filter driver sees these requests and flushes its own asynchronous write queue. This effectively causes the mirror to become synchronous when each flush command occurs. If the amount of data in each write operation is small, a negative impact on the performance of the source system application will occur.

The “DontFlushAsynchQueue” registry parameter allows the user to specify a volume or volumes that should not flush their asynchronous write queues when the DataKeeper driver receives a flush request. Acceptable values are the drive letter(s) of the
volume(s) being replicated. Drive letters may be adjacent to each other (i.e. XY), or space separated (i.e. X Y). Do not place colons after each drive letter.

The next two parameter settings, “WriteQueueHighWater” and “WriteQueueLowWater” determine how the DataKeeper driver manages the asynchronous write queue.

• “WriteQueueHighWater” specifies the high water mark of the asynchronous write queue. If the queue length reaches this value during heavy write traffic, the DataKeeper driver momentarily pauses the mirror, drains the queue down to the “WriteQueueLowWater” mark, then automatically starts a partial resynchronization. The maximum setting for this value depends on the available non-paged memory in the system. This should only be a factor in 32-bit Windows Server environments. Here the maximum non-paged memory is 256MB unless the /3GB switch is used, in which case it is 128MB.

• “WriteQueueLowWater” specifies the low water mark of the asynchronous write queue, and the number of writes that can be outstanding on the network at the same time. In some ways, this parameter is not properly named. When the mirror is in the resynchronizing state, it governs the number of blocks that are queued for transmission. If WriteQueueLowWater is set to 50, the default, when a resynchronization begins, the driver reads 100, 64KB blocks of data and places them in the queue. As blocks are removed from the queue, more are added. When DataKeeper is deployed on a fast network of 1Gbps or above, the queue can drain entirely before blocks are added, making resynchronization less efficient and slower than necessary. In these environments, it is recommended that the WriteQueueLowWater parameter be set at 200.

In general, these parameter settings do not need to be changed. However, if the peak sustained rate of change only occasionally causes the asynchronous write queue to exceed the high water mark, it may be possible to prevent the pause/partial resynch from occurring by increasing this value. It is important to understand that the asynchronous write queue contains data that has been written locally, but not written or acknowledged by the target. Increasing “WriteQueueHighWater” increases the amount of data that could be lost if the source system fails and cannot be recovered.

In high-performance networks, the “WriteQueueLowWater” parameter setting may be increased to allow more packets to be sent over the network at once, thus improving throughput. SteelEye is performing internal testing to determine optimal values for this parameter.
The last parameter in this table, “MaxResyncPasses” specifies the maximum number of bitmap file scans that will be performed during resynchronization. If more than the default number of passes occur and the bitmap file is not empty, DataKeeper will pause the mirror. When might this situation occur? When there is sustained, heavy write activity that first exceeds the “WriteQueueHighWater” setting, and then continues for a long time, while the mirror is being resynchronized. This condition indicates that the network connection between systems is severely undersized.
Other registry parameters that may be tuned generally involve WAN-based mirrors. It should not be necessary to modify these settings when doing LAN-based replication.

This table provides recommendations for a typical configuration in a WAN environment. Optimal values will vary depending on specific network conditions.

- **PingInterval** specifies the interval in milliseconds between pings. Use a higher value for Wide Area Networks or unreliable networks.
- **MaxPingMisses** specifies the maximum number of pings that can fail before the source drops the existing connection to the target and attempts to re-establish a new one. Use a higher value for Wide Area Networks.
or unreliable networks.

PingInterval and MaxPingMisses may be altered to adjust mirroring to network performance.

- **NetworkRetries** specifies the number of times DataKeeper will retry a network command.
- **NetworkRetryDelay** specifies the interval (in milliseconds) between each retry attempt.
- **NetWriteTimeout** specifies the interval (in milliseconds) before a pending network command will timeout. When a timeout occurs, the DataKeeper driver will perform a pause/partial resynchronization of the mirror(s).
- **ValidateTargetRetries** specifies the number of additional attempts a source volume will make during system boot to connect with its target volume. There is always one attempt. If the target system is available the response will normally be immediate. This value identifies the number of additional attempts made, should the first one fail or the target system be down. The interval between each additional attempt is approximately 30 seconds. Target validation is performed using the DataKeeper driver’s ControlPipe and must be done serially by volume. The default of 9 retries will consume 5 minutes per volume. Therefore, if a system is booting with two source volumes and the target for both of them is down, the last volume will not complete its initialization process until the driver has been running for about 10 minutes. The source volume is available for writing during this phase, and writes are tracked in the bitmap to be sent to the target when communication with the target is established.
In addition to tuning the DataKeeper driver parameters, it may be necessary to tune the system’s global TCP Window Size parameters by creating the following registry entries:

\begin{verbatim}
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters
  GlobalMaxTcpWindowSize (REG_DWORD) = Window Size
  Tcp1323Opts (REG_DWORD) = 1
\end{verbatim}

For NIC being used for replication:

\begin{verbatim}
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{GUID}
  TcpWindowSize (REG_DWORD) = Window Size
\end{verbatim}

Where: \textit{Window Size} = \textit{Bandwidth} \times \textit{Latency}
Where: \( \text{Window Size} = \text{Bandwidth} \times \text{Latency} \)

For example, if bandwidth is equal to 45Mbps (a DS3) and latency is equal to 100ms:

\[
\text{Windows Size} = \left( \frac{45 \text{Mbps} \times 100 \text{ms}}{10^6 \text{ ms/s}} \right)
= 4.5 \text{ Mbits}
= 0.56 \text{ Mbytes}
\approx 587202 \text{ bytes}
\]

When unsure of the exact values for bandwidth and latency, it is best to overestimate \( \text{Window Size} \), rather than underestimate.
Command Line Interface (EMCMD)
The DataKeeper command line interface, EMCMD, includes six groups of subcommands. Here are the Service, Job and the first group of Volume related subcommands.
The Volume related subcommands are continued here, along with Mirror related subcommands.
Finally, here are the Bitmap and Rewind related subcommands. LifeKeeper uses the Rewind related subcommands to perform assisted target data rewind functions.
The DataKeeper Command Line Interface, EMCMD, may be used to obtain basic information about a volume participating in a mirror on a specific system. In this example, the command is being used to query the status of the E: volume. EMCMD reports that the system at IP address, 192.168.251.98, is the source, the system at IP address, 192.168.252.98, is the target, and the mirror is in Mirroring state.
Here are a couple of common use cases for EMCMD.

The first case is used when manual switchover to the target system is required, and the source system is down, but repairable. The target role will be switched and the mirror will be in resynch pending state.

The second case is used when manual switchover to the target system is required, and the source system has been destroyed entirely or its system disk has crashed. Here the local mirror or target is deleted and the switchover flag associated with the mirror is cleared. The effect here is to entirely delete this leg of the mirror. If the source system was not destroyed and it comes back in-service, it will re-establish its mirror to the target system and perform a full resynchronization. Any data changed on the target system will be lost.

The mirror target should never be brought on-line for write, while the mirror source is also on-line for write. This is a different situation than when a mirror source is paused and a target is unlocked for access. In this environment, DataKeeper believes that both volumes have the source role. Two volumes, with different contents that cannot be automatically reconciled can result. The

Use Case: Manual Switchover, Source Out of Service

EMCMD <target ip> SWITCHOVERVOLUME <letter>

Use Case: Manual Switchover, Source Destroyed

EMCMD <target ip> DELETELLOCALMIRRORONLY <letter>
EMCMD <target ip> CLEARSWITCHOVER <letter>

Source must be off-line, otherwise a mirror split-brain condition may be created.
DataKeeper GUI will indicate that the mirror is in an inconsistent state that must be manually reconciled by the user.
This use case has been employed by several customers. In it the user wishes to use the target as a nightly tape backup source, typically at a disaster recovery site. An automated script pauses the mirror on the source system, and unlocks the volume on the target system. Then the script invokes the appropriate backup utility.
When the backup is complete, the script locks the volume on the target and continues the mirror on the source. When the mirror is paused and unlocked, the bitmap on the source system is used to track source volume changes. If data is written to the target volume while it is unlocked, this data will be overwritten by the corresponding blocks from the source system when the target is locked and the mirror continued. Any changes that occurred on the source volume are also sent to the target during the partial resynchronization of the mirror.
DataKeeper includes the ability to perform disk-to-disk copy on a local system. A template batch file and associated VisualBasic script file are provided under C:\Program Files (x86)\SteelEye\DataKeeper\D2dbackup. The use of these scripts, with customization for the local configuration, allow the user to set up an automated task to create a mirror image copy of a local disk for the purposes of nightly backup.
Troubleshooting, Known Issues and Restrictions
If communication between source and target systems is lost and the target volume is MANUALLY made a source, when communications between the source and target systems are re-established, DataKeeper will recognize a “split brain” condition. Both mirrors will be paused waiting for manual intervention.
The DataKeeper GUI on the target system will display the message shown here. Clicking on the link will bring up a resolution dialog box.
To resolve the issue, the user MUST select which system should be the mirror source. It is strongly recommended that the user review the system event logs from both systems to determine when this condition was created and confirm which volume should be the source. Further research and resolution will be required if writes occurred to both volumes. If writes only occurred on the source system, it is safe to make it the mirror source by selecting its radio button and clicking the “OK” button. The mirror will partially resynchronize and return to a mirroring state.
Many reported issues can be attributed to DataKeeper installation pre-requisites that have not been met or lack of understanding. These include:

- Failure to configure firewalls to allow ports on which DataKeeper depends to pass. On Windows Server 2008, the DataKeeper installer will automatically configure the firewall appropriately.
- Mistakenly believing that DataKeeper can replicate the System volume, typically C:, or volumes containing page files.
### Issues and Resolutions

<table>
<thead>
<tr>
<th>Component</th>
<th>Message/Issue</th>
<th>Cause</th>
<th>Suggested Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface</td>
<td>Failed to create mirror. Target System event log entry contains error code: 0xC0000055</td>
<td>Distributed Link Tracking Client service enabled and active</td>
<td>Stop and disable service on both source and target</td>
</tr>
<tr>
<td>User Interface</td>
<td>Mirror created, but not stored in job. Event log entry contains “File: ;GuiThread.cpp Line 3099 ...”</td>
<td>Local security policy for &quot;Network Access: Let Everyone permissions apply to anonymous users&quot; is disabled</td>
<td>Enable policy on all servers.</td>
</tr>
<tr>
<td>System Event Log</td>
<td>Create mirror failed.</td>
<td>vmms.exe program has volume open preventing DataKeeper from obtaining volume lock</td>
<td>Stop vmms.exe program</td>
</tr>
</tbody>
</table>

- Failure to stop and disable the Distributed Link Tracking Client service. In DataKeeper version 7.1 the installer automatically takes these actions.
- Failure to enable the local security policy Network Access rule to that allows Everyone permissions to apply to anonymous uses.
- Having programs active that lock a volume, preventing DataKeeper from acquiring an exclusive lock.
### Issues and Resolutions

<table>
<thead>
<tr>
<th>Component</th>
<th>Message/Issue</th>
<th>Cause</th>
<th>Suggested Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Clustering</td>
<td>Newly created resource appears off-line but is unlocked</td>
<td>Expected behavior</td>
<td>Bring resource on-line</td>
</tr>
<tr>
<td>Windows Clustering</td>
<td>DataKeeper Volume resource not available for use</td>
<td>DataKeeper installed before cluster name resource created</td>
<td>Perform repair install of DataKeeper</td>
</tr>
<tr>
<td>Windows Clustering</td>
<td>DataKeeper Volume resource not available for use. Event log contains “DataKeeperVolume.dll, Error: 70”</td>
<td>All cluster nodes were not on-line</td>
<td>Bring all cluster nodes on-line</td>
</tr>
</tbody>
</table>

- It is expected behavior that a newly created DataKeeper Volume resource does not take the mirror off-line, even though the resource itself is off-line.

- Installing DataKeeper Cluster Edition prior to forming the Windows cluster by creating a cluster name and management IP address.

- Failure to have all Windows cluster nodes on-line when attempting to create a DataKeeper Volume resource.
The last two items are also known issues. DataKeeper-specific performance counters are not compatible with the 64-bit version of Performance Monitor on Windows Server 2008. The recommendation is to use the included 32-bit version of Performance Monitor.

DataKeeper-specific performance counters do not work on Windows Server 2003. This is a known issue that will be resolved in a future update to DataKeeper.
Finally, a list of known restrictions and recommendations.

Use caution with the length of directory names if you override the default installation directory for DataKeeper. Directory names of more than 32 character will cause performance counters to fail and may cause other issues with functions that need to read the environment variable EXTMI RRBASE.

Any volume maintenance activity that generates a large number of writes, including CHKDSK and defragmentation, should be done with the mirror paused. This will prevent heavy write traffic from pushing a mirror into a pause/resynchronization cycle. It will also de-dup changes made to the volume by multiple write operations to the same block(s). The subsequent partial resynchronization will
complete much more quickly than if the mirror had not been paused.

As mentioned in earlier modules, heavy write activity when using a synchronous mirror may have a noticeable negative impact on application performance. Even when deploying DataKeeper in a local area network, it can be advisable to use asynchronous mirrors.

Resizing of mirrors that use Dynamic Disks is not supported. The mirror must be deleted before resizing both source and target volumes. When the mirror is recreated, DataKeeper will create a new, appropriately sized bitmap file.

If bitmap files will be relocated to another directory for performance reasons, the new directory must be created prior to attempting to move existing bitmap files.

Bitmap files on Dynamic Disks are not supported unless the disk is also the System disk.
Summary
In this module, we have discussed the following topics.

- How to use routes to help perform initial synchronization of WAN-based mirrors.
- DataKeeper registry settings, what they mean and how to tune them to improve network performance.
- The DataKeeper command line interface, EMCMD. Its subcommands, the purpose of these commands and several use cases involving the command line interface.
- How to recover from a mirror split-brain situation.
- A list of common user configuration oversights and errors.
• Finally, a list of known restrictions and explanations for those restrictions.
End Module 4

Lab:  Tips, Tuning and Troubleshooting

Replicate Any Data. Protect Any Application