*An **InterSystems** Technology Guide*

One Memorial Drive, Cambridge, MA 02142, USA
Tel: +1.617.621.0600 | Fax: +1.617.494.1631
http://www.intersystems.com

HIGH AVAILABILITY STRATEGIES

HA Strategies for InterSystems Caché, Ensemble, and HealthShare Foundation

Ray Fucillo, Product Manager (**ray.fucillo@intersystems.com**)
Mark Bolinsky, Technology Architect (**mark.bolinsky@intersystems.com**)

April 6, 2015

INTRODUCTION

This document is intended to provide a survey of various High Availability (HA) strategies that can be used in conjunction with InterSystems Caché, Ensemble, and HealthShare Foundation. This document also provides an overview of the various types of system outages that can occur, as well as how each strategy would handle a given outage, with the goal of helping you choose the right strategy for your specific deployment.

The strategies surveyed in this document are based on three different HA technologies: Operating System Failover Clusters, Virtualization-Based HA, and Caché Database Mirroring. Table 1 below highlights some key differences between these technologies.

	Caché Database Mirroring	Operating System Failover Clustering	Virtualization High Availability
Failover after Machine Power Loss or Crash	Handles machine failure seamlessly in version 2015.1 or later. Prior versions did not fail over automatically in this scenario; alternatives required careful planning.	Handles machine failure seamlessly	Handles physical and virtual machine failures seamlessly
Protection from Storage Failure and Corruption	Built-in replication protects against storage failure; logical replication avoids carrying forward many types of corruption	Relies on shared storage device, so failure is disastrous; storage-level redundancy optional, but can carry forward some types of corruption	Relies on shared storage device, so failure is disastrous; storage-level redundancy optional, but can carry forward some types of corruption
Failover after Caché Shutdown, Hang, or Crash	Rapid detection and failover is built in	Can be configured to fail over after Caché outage	Can be configured to fail over after Caché outage
Caché Upgrades	Allows for minimum-downtime Caché upgrades*	Caché upgrades require downtime	Caché upgrades require downtime
Application Mean Time to Recovery	Failover time is typically seconds	Failover time can be minutes	Failover time can be minutes
External File Synchronization	Only databases are replicated; external files need external solution	All files are available to both nodes	All files available after failover

Table 1: General Feature Comparison

* Requires a configuration in which application code, routines, and classes are in databases separate from those that contain application data

OPERATING SYSTEM FAILOVER CLUSTERING

A very common approach to achieving HA is to use failover solutions that are provided at the operating system level. Examples of such solutions exist on all platforms and include Microsoft Windows Clusters, HP Serviceguard, Veritas Cluster Server, and IBM SystemMirror (PowerHA), as well as the respective clustering packages from Red Hat and SUSE Linux. While the specifics of the configuration may differ slightly among the various platforms, the model is generally the same: two identical servers with a shared storage device (often a SAN or iSCSI targets) and a shared IP address, one actively serving production workload, and one standing by in case of failure. When an outage occurs on the active system, the failover technology transfers control of the shared disk and the shared IP address to the standby node, and then starts application services, including Caché.

Caché is designed to integrate easily with these failover solutions. The production instance of Caché is installed on the shared storage device so that both members of the failover cluster recognize the instance, then added to the failover cluster configuration so that it will be started automatically as part of failover. When Caché starts on the newly active node during failover, it automatically performs the normal startup recovery from WIJ and journal files (again, located on the shared storage device); data integrity is preserved just as though Caché had simply been restarted on the original failed node.

Pros:

- ↑ Handles machine failure seamlessly
- ↑ Most common HA choice
- ↑ Available on all supported platforms through OS or 3rd party vendors
- ↑ All files (database and external) available to both nodes

Cons:

- ↓ Storage failure is disastrous
- ↓ Upgrades require downtime
- ↓ Application Mean Time to Recovery can be minutes

The appendixes in the Caché High Availability Guide contain detailed information on how to correctly configure Caché with some of the more popular OS failover clusters.

VIRTUALIZATION-BASED HIGH AVAILABILITY

Virtualization technologies, such as VMware vSphere ESX/ESXi, provide High Availability capabilities, which typically monitor the overall health and viability of the physical hardware, as well as the guest operating systems running therein. On failure, the Virtualization HA software will automatically restart the failed virtual machine on an alternate surviving hardware. When Caché restarts, it automatically performs the normal startup recovery from WIJ and journal files; data integrity is preserved just as though Caché had simply been restarted on the original failed node.

In addition, guest operating systems can be relocated to other servers within the virtual environment, allowing for a virtual machine to be uplifted to alternate physical infrastructure, for maintenance purposes, without downtime. This feature is available as VMware vMotion, IBM Live Partition Mobility, HP Live VM Migration, and others.

Pros:

- ↑ Handles machine failure seamlessly
- ↑ Most common HA choice in virtual environments
- ↑ All files are available after failover
- ↑ Planned physical hardware maintenance requires little or no application downtime

Cons:

- ↓ Storage failure is disastrous
- ↓ Software upgrades require downtime
- ↓ Application Mean Time to Recovery can be minutes for unplanned hardware failures

Proper infrastructure is required to effectively support high availability in a virtual environment. This includes storage, networking, and processor capacity. Please refer to your virtualization supplier's documentation for best practices.

CACHÉ DATABASE MIRRORING

A mirror consists of two physically independent Caché systems, called *failover members*. The mirror automatically assigns the role of *primary* to one of the failover members, while the other member automatically becomes the *backup* system. Data is replicated from the primary to the backup failover member, thus providing built-in data redundancy. Caché Database Mirroring (Mirroring) is designed to provide an economical solution for rapid, reliable, robust, automatic failover between two Caché systems for planned and unplanned outages.

Mirroring additionally allows asynchronous replication to other members called *async members*. Async members can be used to meet a variety of demands including disaster recovery, reporting, data warehousing, and business intelligence. Async members are not available for automatic failover, but async members that are designated for disaster recovery can be quickly promoted to take over as part of your disaster recovery procedures. For more information on the disaster recovery features of mirroring (specific to versions 2013.1 and later), see the Caché documentation section on Promoting a DR Async Member to Failover Member and Mirror Outage Procedures. The remainder of the discussion of mirroring in this document pertains to failover members and the high availability features of mirroring.

Pros:

- ↑ Rapid, automatic and safe failover for almost any type of hardware failure, operating system failure, or Caché failure.
- ↑ Allows for minimum-downtime Caché upgrades
- ↑ Data replication protects against storage failure on the primary
- ↑ Failover time is typically seconds, providing fast application mean time to recovery
- ↑ Can be less expensive than clustering solutions
- ↑ Logical data replication can protect against physical corruption being carried forward to the other system
- ↑ Failover members may be in separate data centers, possibly allowing for HA and DR goals to be met with only two servers (allowable latency is dependent on the application)
- ↑ Async members for disaster recovery and reporting allow you to meet multiple needs with one technology.

Cons:

- ↓ Only databases are automatically replicated; external files needed by the application (i.e., file streams, images, etc.) need a third party replication solution
- ↓ Security and configuration management is currently decentralized

MIRRORING FAILOVER STRATEGIES

Mirroring can be used to meet a variety of high availability needs. The strategy for meeting these demands will encompass the mirroring settings, hardware configuration, data center configuration, and sometimes manual procedures.

In all cases, in order to take over as primary, it must be definitively determined, automatically through software or through manual intervention, that the primary failover member is down, and that the backup failover member has all of the journal data that the primary failover member has durably committed. The mechanism for making that determination differs for each of the mirroring failover strategies described. For more details, see the Caché documentation section on Automatic Failover Mechanics.

The remainder of this section describes the various mirroring failover strategies. The general mirroring pros and cons listed above apply to each of the failover strategies; specific pros and cons for each strategy are separately listed below.

FAILOVER WITH MIRROR ARBITER

Starting in version 2015.1, mirroring employs a separate system called the arbiter to provide safe, built-in, automatic failover under scenarios in which communication between the failover members themselves is not possible: when the primary's host has either failed or become network-isolated. If the arbiter is not configured, the arbiter is down, or the backup system was not up to date at the time of the failure, mirroring automatically falls back to the mode of operation described in Failover with ISCAgent Only until the failover members are connected to the arbiter and caught up.

Pros:

- ↑ Provides rapid failover in almost any failure scenario.
- ↑ Completely safe failover; no risk of split-brain (that is, two servers both acting as primary)
- ↑ No specialized hardware or software needed
- ↑ Failover members may be in separate data centers, possibly allowing for HA and DR goals to be met with only two servers (allowable latency is dependent on the application)
- ↑ Mirror continues to operate normally if arbiter fails. (ISCAgent-based failover can still occur until the arbiter becomes available again.)

Cons:

- ↓ If failover members are in separate data centers, a third location should be used for the arbiter in order to allow automatic failover after complete data center failure.

To implement this strategy, identify and configure a host to act as arbiter as described in the Caché documentation section on Locating the Arbiter to Optimize Mirror Availability.

FAILOVER WITH ISCAGENT ONLY

When the backup mirror member detects a failure of the primary, it attempts to contact the ISCAgent on the primary machine. If the backup successfully contacts the ISCAgent, it can then confirm that the primary is down or force it down if it is unresponsive, download any journal information required for it to be fully caught up, and safely take over as primary.

If the ISCAgent cannot be contacted (for example, if the primary server is down), failover does *not* occur. Of course, the administrator can take manual steps to confirm that the primary is down and that the backup has the necessary journal data, then initiate failover. See Appendix D for instructions on Manual Failover After Unplanned Outage of Primary.

Pros:

- ↑ Completely safe failover; no risk of split-brain (that is, two servers both acting as primary)
- ↑ No specialized hardware or software needed
- ↑ Allows rapid failover after Caché shutdown, a hung Caché instance, and many hardware or software failures that prevent Caché from working, so long as the ISCAgent remains reachable from the backup member
- ↑ Failover members may be in separate data centers, possibly allowing for HA and DR goals to be met with only two servers (allowable latency is dependent on the application)

Cons:

- ↓ No automatic failover occurs after failure that renders the ISCAgent unreachable, such as host failure
- ↓ If the primary host is unavailable, it can be difficult to determine whether the backup has all the required journal information in order to verify that it is safe to initiate manual failover

In 2015.1 and later this is the default strategy until you configure an arbiter. To implement this strategy in versions prior to 2015.1, leave the *Agent Contact Required for Takeover* configuration setting at YES (the default).

FAILOVER WITH CUSTOM SOLUTION: RELIABLE NETWORK PING

Important: Starting in version 2015.1, this strategy is no longer be available, and sites using this strategy will, upon upgrading, need to switch to use Failover with Mirror Arbiter, a simpler and safer way to achieve the same goals.

When the backup mirror member detects a failure of the primary, it attempts to contact the ISCAgent on the primary machine. If the backup successfully contacts the ISCAgent, it can then confirm that the primary is down or force it down if it is unresponsive, download any journal information required for it to be fully caught up, and safely take over as primary.

If the ISCAgent cannot be contacted (for example, if the primary server is down), network pings over the public and private network are utilized to determine the status of the primary server (this requires custom programming which is implemented in `$$IsOtherNodeDown^ZMIRROR()`). If the primary does not respond to the pings on either the public or the private network, the backup assumes that the primary is down, and takes over as primary. Because the lack of ping response from the primary server does not strictly guarantee that the server is down, there is a risk of split-brain (two servers simultaneously acting as primary) that cannot be completely eliminated with this strategy. Other mirroring failover strategies discussed in this document carry no risk of split-brain. To minimize the risk, the following is required:

- The networking between the failover members must be redundant, reliable, and highly available.
- The failover members should be hosted directly on physical machines, not on a virtualization platform; on virtualized platforms, activity at the host/hypervisor level may cause a member to become temporarily unresponsive to ping while it is still running. See the Hybrid HA Strategy for information on how to safely extend mirroring in its default configuration to provide higher availability in a virtualized environment.

Pros:

↑ Allows rapid failover following server/host failure

Cons:

↓ Requires implementation of a custom ^ZMIRROR routine (InterSystems can provide a sample).

↓ Requires specialized networking hardware configuration to provide very robust networking.

↓ Recommended that the failover machines are located in the same data center to avoid network isolation.

↓ The risk of split-brain (two servers both acting as primary) cannot be completely eliminated.

To implement this strategy:

1. Create a hardware configuration that provides an extremely reliable network between the primary and backup failover members. Please reference Appendix A for an example of a reliable network configuration between two failover members.
2. Customize the sample implementation of `$$IsOtherNodeDown^ZMIRROR()` from the routine provided in Appendix C. In any scenario under which the ping mechanism cannot adequately determine that the primary is down, your implementation must assume that it is up so that automatic failover will not occur. Of course, the administrator can take manual steps to determine that the primary is down and that the backup has the necessary journal data, and then initiate failover. See Appendix D for instructions on Manual Failover After Unplanned Outage of Primary
3. Set *Agent Contact Required for Takeover* to NO.
4. Adjust *Trouble Timeout Limit* to allow sufficient time for the `$$IsOtherNodeDown^ZMIRROR` mechanism to operate. For example, while testing failover you may notice a message similar to the following in the cconsole.log file on the backup failover member: `Mirror recovery time of 7.101 seconds exceeded trouble timeout of 6 seconds. Restarting`. In this example, you might consider increasing the Trouble Timeout Limit to 8 seconds.

HYBRID HA STRATEGY

Database Mirroring can be used in conjunction with Virtualization HA to provide extremely robust high availability strategies for planned and unplanned outages.

Database Mirroring provides the first line of defense with rapid automatic failover for planned and unplanned outages. Virtualization HA automatic restarts the virtual machine hosting a mirror member following unplanned machine or OS outages, making the failed member available again to act as backup (or to take over as primary if necessary). The data replication built in to Mirroring provides protection against storage failures and Mirroring allows for additional async members for disaster recovery and reporting.

Prior to Caché version 2015.1, Virtualization HA plays a larger role in failover scenarios where the virtual machine hosting the primary crashes. In that case, Virtualization HA is key in making the host available for the backup member to contact as described in Failover with ISCAgent Only. In 2015.1 and later, Failover with Mirror Arbiter occurs immediately in this scenario, even before Virtualization HA restarts the failed machine.

Pros:

- ↑ Provides higher availability than Virtualization HA alone. A natural choice for virtual deployments that wish to take advantage of mirroring features.

Cons:

- ↓ Higher complexity
- ↓ Multiple technologies to manage and learn

To implement this strategy, see the Hybrid HA Solution section in Appendix A.

GENERAL SYSTEM OUTAGES

There are two broad categories of system outages: planned and unplanned. Which HA strategy you choose will largely determine the likelihood of automatic failover during a given type of outage. Table 1 summarizes the various types of outages and indicates whether each HA strategy would likely fail over automatically in response to that particular outage.

Automatic Failover by Outage Type and HA Strategy		OS Failover Cluster	Virtualization HA	Mirroring: Failover w/ ISCAgent Only	Mirroring: Failover w/ Mirror Arbiter	Mirroring: Failover w/ Custom Solution - Reliable Network Ping	Hybrid Strategy
Planned Outage	Caché Upgrade	NO	NO	YES	YES	YES	YES
	OS Upgrade	YES	NO	YES	YES	YES	YES
	Application Upgrade	NO	NO	MAYBE	MAYBE	MAYBE	MAYBE
	Server Upgrade/Maintenance	YES	YES	YES	YES	YES	YES
	Storage Upgrade/Maintenance	MAYBE	MAYBE	YES	YES	YES	YES
Unplanned Outage	Caché hang, crash, or shutdown	MAYBE	MAYBE	YES	YES	YES	YES
	All networking fails on primary	YES	YES	NO	YES	N/A†	YES
	Public network interface fails on primary	YES	YES	MAYBE	MAYBE	MAYBE	YES
	Server (host) failure, crash, or power loss	YES	YES	NO	YES	YES	YES
	Storage failure	MAYBE	MAYBE	MAYBE	YES	MAYBE	MAYBE

Table 2: Automatic Failover by Outage Type and HA Strategy

The following sections describe the reasoning for the outcomes in Table 2.

† This is a special case that can result in split-brain (two servers both acting as primary). See additional details below.

PLANNED OUTAGE TYPES

- **Caché Upgrade**: OS Failover Clusters and Virtualization HA require downtime for Caché upgrades because there is only one instance of Caché in this configuration. Mirroring, on the other hand, allows you to perform a rolling upgrade between compatible versions by upgrading Caché on the backup failover member first, then failing over, and upgrading the other node. Since the Hybrid Strategy includes Mirroring, the same benefit applies to this HA strategy.
- **OS Upgrade**: All outlined HA strategies, except Virtualization HA, typically provide rolling upgrade options; Virtualization HA requires downtime because there is only one instance of the guest OS.
- **Application Upgrade**: OS Failover Clusters and Virtualization HA require downtime for application upgrades because there is only one production instance of Caché in this configuration. Mirroring may provide a way to minimize the downtime for application upgrades, assuming no data conversion is necessary (before, after, or during the upgrade), and assuming that the application code and data are stored in separate databases (a best practice). Since the Hybrid Strategy includes Mirroring, the same benefit applies to this HA strategy.
- **Server Upgrade/Maintenance**: All outlined HA strategies typically provide rolling maintenance and upgrade options.
- **Storage Upgrade/Maintenance**: OS Failover Clusters and Virtualization HA may‡ require downtime for storage upgrades or maintenance because there is one shared storage device in this configuration. Mirroring, on the other hand, typically uses independent storage devices, allowing you to perform rolling storage upgrades or maintenance. Since the Hybrid Strategy includes Mirroring, the same benefit applies to this HA strategy.

UNPLANNED OUTAGE TYPES

- **Caché hang, crash, or shutdown**: OS Failover Clusters and Virtualization HA provide monitoring scripts that can be configured to induce failover in the event of Caché failure. Mirroring immediately detects this condition and triggers an automatic failover. Since the Hybrid Strategy includes Mirroring, the same benefit applies to this HA strategy.
- **All networking fails on primary**: OS Failover Clusters and Virtualization HA typically detect network loss and automatically trigger a failover. Mirroring behaves differently based on the strategy that has been deployed:
    - *Failover with ISCAgent Only*: Since contact with the ISCAgent on the failed node is not possible, mirroring does not automatically fail over.
    - *Failover with Mirror Arbiter*: Automatic failover occurs as long as the networking between the backup and arbiter remains intact because both the arbiter and the backup see that the primary has disconnected. The primary remains in a trouble state where it stops function as primary. When the members can communicate again, the original primary in trouble state will be automatically forced down.
    - *Failover with Reliable Network Ping*: This strategy is predicated on having a 100% reliable/redundant network, so failures of this type should be precluded. If a failure of this type were to occur, the `$$IsOtherNodeDown^ZMIRROR()` procedure would allow mirroring to automatically fail over, resulting in split-brain (two servers both acting as primary).
    - *Hybrid Strategy*: Since the Hybrid Strategy includes Virtualization-based HA, automatic failover occurs with this sort of failure.
- **Public network interface fails on primary**: OS Failover Clusters and Virtualization HA typically detect network loss and automatically trigger a failover. If the mirror systems are configured with only one network for both external traffic and internal mirror communication, then see *All networking fails on primary* above; the discussion of public network failure that follows assumes that an additional private network is configured between the mirror members. Mirroring does not explicitly detect failures of the public network interface. However, it is possible for you to create a custom monitoring script to induce failover in the event of public network interface failure on the primary server and this would apply to any of the mirroring strategies. Since the Hybrid Strategy includes Virtualization-based HA, automatic failover occurs with this sort of failure.

‡ Some SAN storage systems allow for non-disruptive updates (NDU), which may allow storage upgrade/maintenance without downtime.

- **Server (host) failure, crash, or power loss**: OS Failover Clusters and Virtualization HA detect these types of failures and automatically fail over. Mirroring behaves differently based on the strategy that has been deployed:
    - *Failover with ISCAgent Only*: Since contact with the ISCAgent on the failed node is not possible, mirroring does not automatically fail over.
    - *Failover with Mirror Arbiter*: Automatic failover occurs because both the arbiter and the backup see that the primary has disconnected.
    - *Failover with Reliable Network Ping*: The `$$IsOtherNodeDown^ZMIRROR()` procedure will detect that the primary is down, allowing mirroring to automatically fail over.
    - *Hybrid Strategy*: Since the Hybrid Strategy includes Virtualization-based HA, automatic failover occurs with this sort of failure.
- **Storage failure**: OS Failover Clusters may be able to tolerate a storage failure if disk-based replication is utilized; otherwise, a storage failure is catastrophic, as both nodes would be operating on a shared storage device. Similarly, Virtualization HA may be deployed on replicated storage. Mirroring, assuming the failover members are configured to use independent storage devices, may be able to automatically fail over in the event of storage failure. The ability to automatically fail over depends on the nature of the storage failure and the mirroring failover strategy; if the storage failure prevents the ISCAgent from operating properly, then only *Failover with Mirror Arbiter* can fail over automatically. The data replication provided by Mirroring protects production data from storage failures, allowing you to manually induce failover if needed. Since the Hybrid Strategy includes Mirroring, the same benefit applies to this HA strategy.

APPENDIX A: SAMPLE RELIABLE NETWORK CONFIGURATIONS

This appendix describes various possible reliable network configurations used to deploy Mirroring.

The *Mirroring Failover with Custom Solution: Reliable Network Ping* strategy requires that the networking between the two failover members is redundant, reliable, and highly available in order to prevent split-brain (i.e., two servers both acting as primary). The following sample configurations are examples.

Figure 1 illustrates a very simple configuration whereby the two failover members are directly connected to each other with crossover cables for mirror communications over a private network, entirely avoiding an external switch or hub; this reduces the likelihood of a network-related failure between the two systems. The systems are also connected to an external bank of switches for public communication. This configuration is highly economical and simple in design.

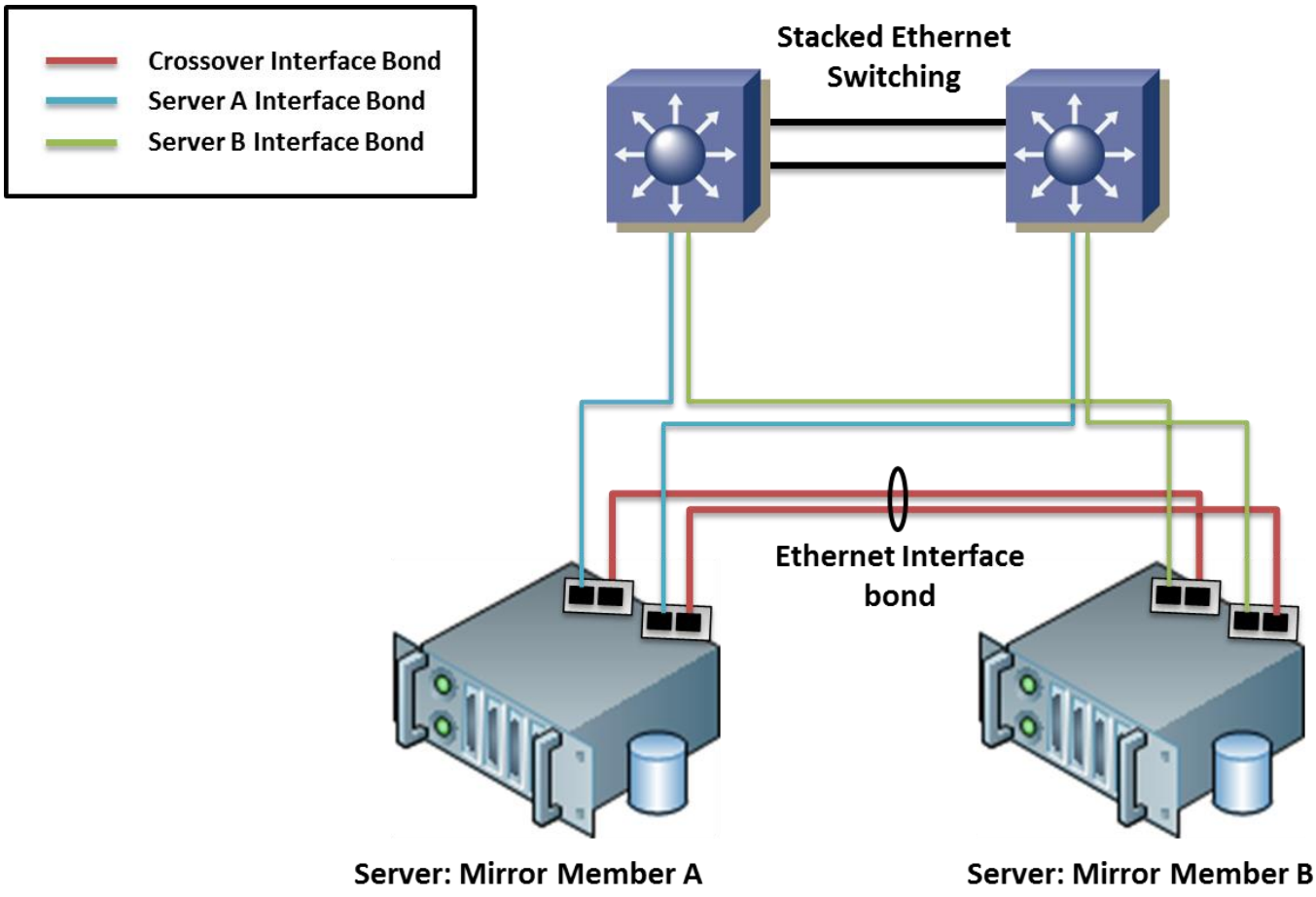

Figure 1: Direct-Connect Sample Configuration

A few additional notes about the sample configuration presented in Figure 1 follow:

- Each system is shown with two dual-ported NICs. This is for reliability and redundancy purposes, and is a recommended configuration.
- Support for using bonded or teamed interfaces ports with crossover cables is OS and NIC driver dependent. Not all operating systems and NIC drivers support this type of bonding. Please refer to system documentation for compatibility.

Note that the two failover members could either be configured with internal storage (SAS), or have external storage (SAN, NAS, and so on); in the case of a shared SAN, however, it is recommended that the two systems use independent physical spindles.

This example could be extended to include highly available network switching, which will mitigate the physical co-location requirement; however, it is imperative that the networking be reliable. Figure 2 illustrates a sample configuration using dual computer rooms (same campus) each with highly available stacked switching using EtherChannel bonds.

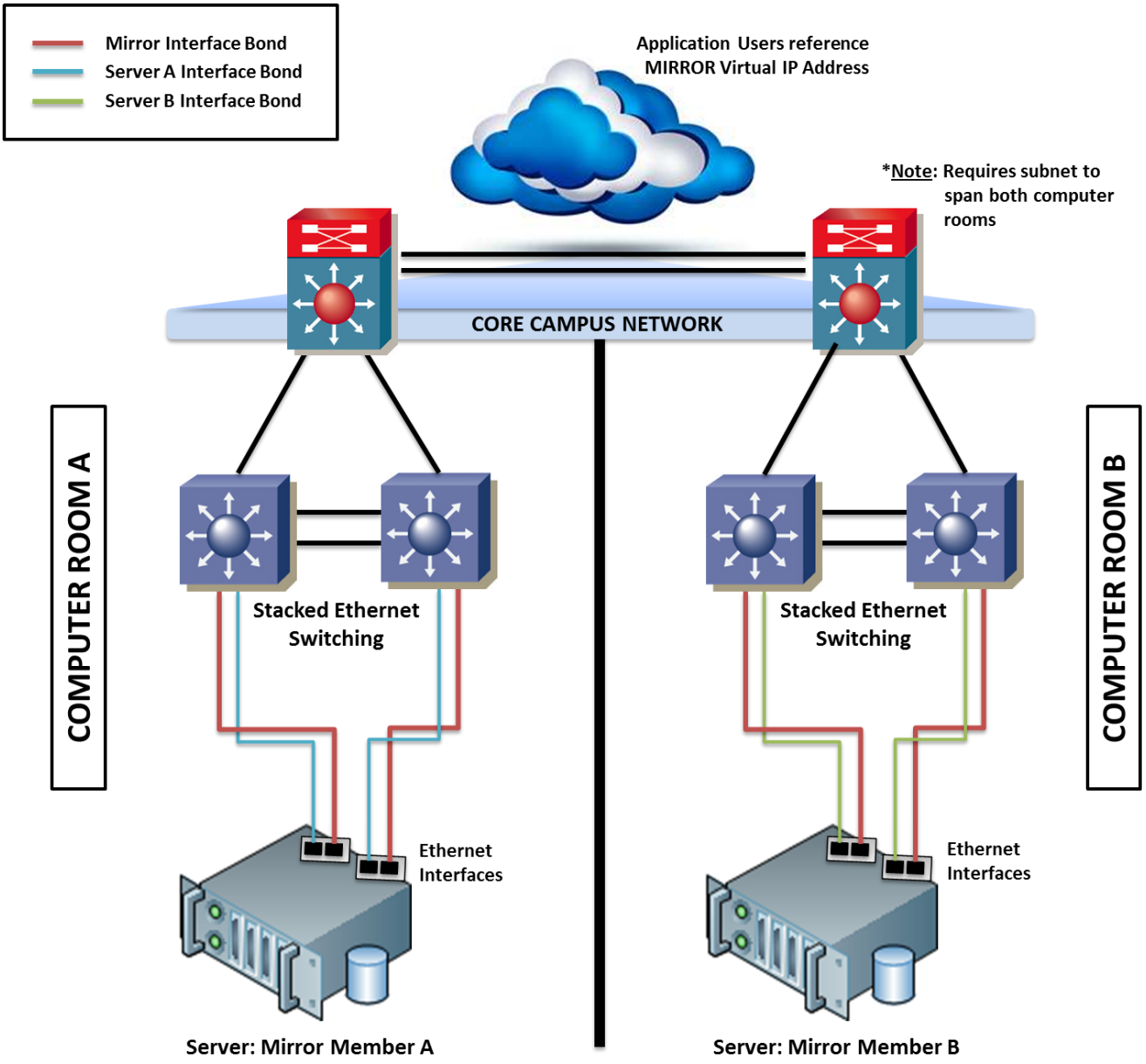

Figure 2: Redundant Switching Sample Configuration

APPENDIX B: HYBRID HA SOLUTION

A Hybrid HA solution is one that leverages the strengths of both types of HA technologies – Virtualization-based HA plus Caché Database Mirroring – to provide very high levels of availability. Database Mirroring provides the first line of defense with rapid automatic failover for planned and unplanned outages. Virtualization HA automatic restarts the virtual machine hosting a mirror member following unplanned machine or OS outages, making the failed member available again to act as backup (or to take over as primary if necessary).

See Mirroring in a Virtualized Environment in the Caché documentation for additional guidelines on Caché configuration and operation in this environment. The remainder of this appendix will detail the configuration of the virtualization environment. If using a Caché version prior to 2015.1 configure mirroring using *Agent Contact Required for Takeover* = YES. On version 2015.1 or later, configure an arbiter as described in the 2015.1 Mirror Architecture and Planning Guide.

The use of virtualization carries with it additional requirements to provide acceptably high availability. When configuring the guest virtual machines, each of the mirror members should have anti-affinity rules defined. This will ensure that the primary and backup failover members will avoid running on the same physical server.

Installing a minimum of six (6) Ethernet interfaces spread across at least two (2) or three (3) physical adapters is highly recommended. In blade server technologies, the use of VICs (*Virtual Interface Cards*) or CNAs (*Converged Network Adapters*) is common, and support both Ethernet and Fiber Channel communications, so consult your respective hardware supplier's best practices for virtual interface definitions.

Figure 3 illustrates a sample 3-node VMware ESX[§] cluster with redundant networking.

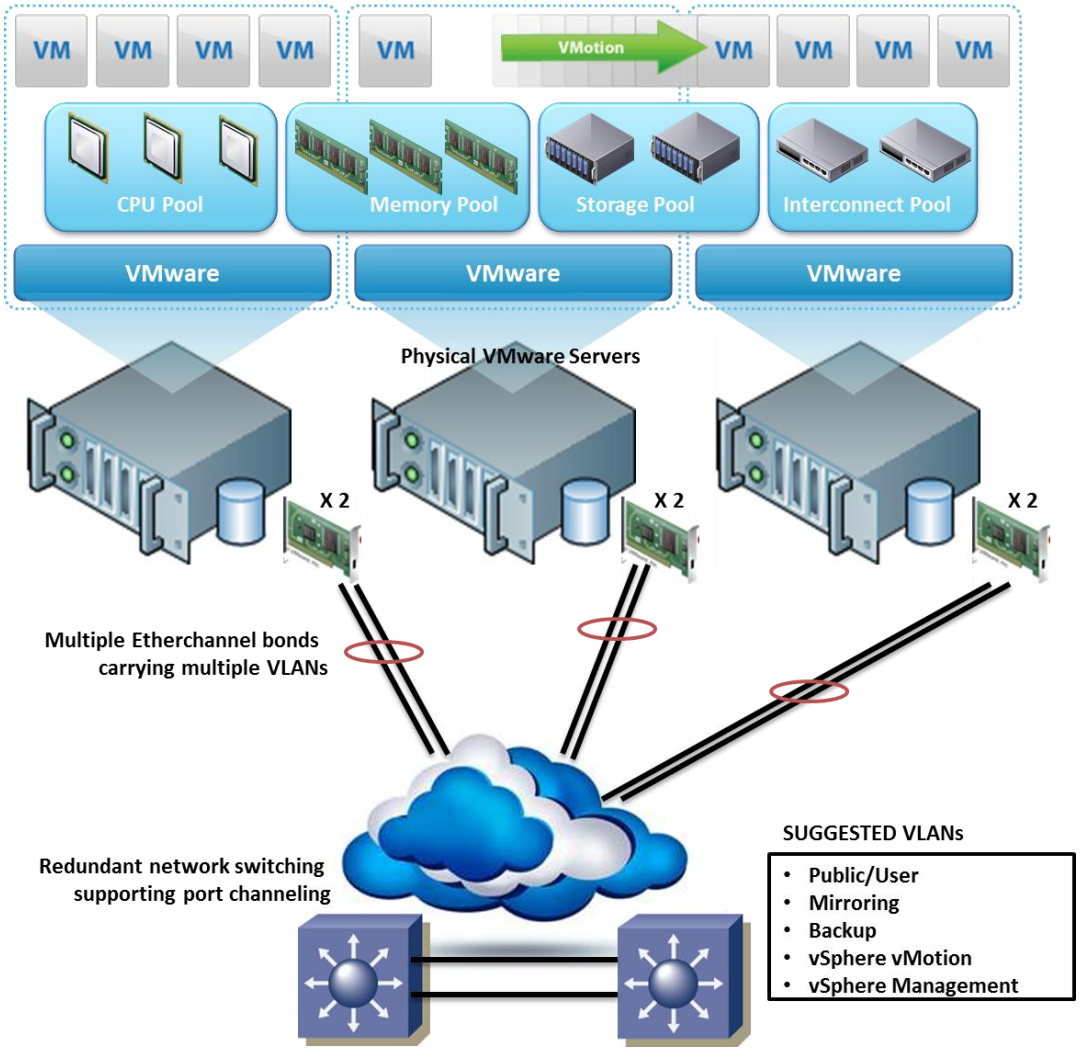

Figure 3: Redundant Networking with VMware vSphere

[§] Please note VMware is used only as an example. Other virtualization technologies from IBM, HP, Microsoft, and Linux KVM variants such as Red Hat RHEV-M that supports automatic partition or virtual machine mobility/restart can be used as well.

Along with redundancy in the network, added attention to storage is highly recommended to maintain isolation for each mirror member. To provide isolation, each mirror member should use separate data-stores which should be at a minimum on separate disk groups in a single SAN storage array or ideally on two separate storage arrays altogether. Figure 4 illustrates the storage layer isolation in a virtualized environment.

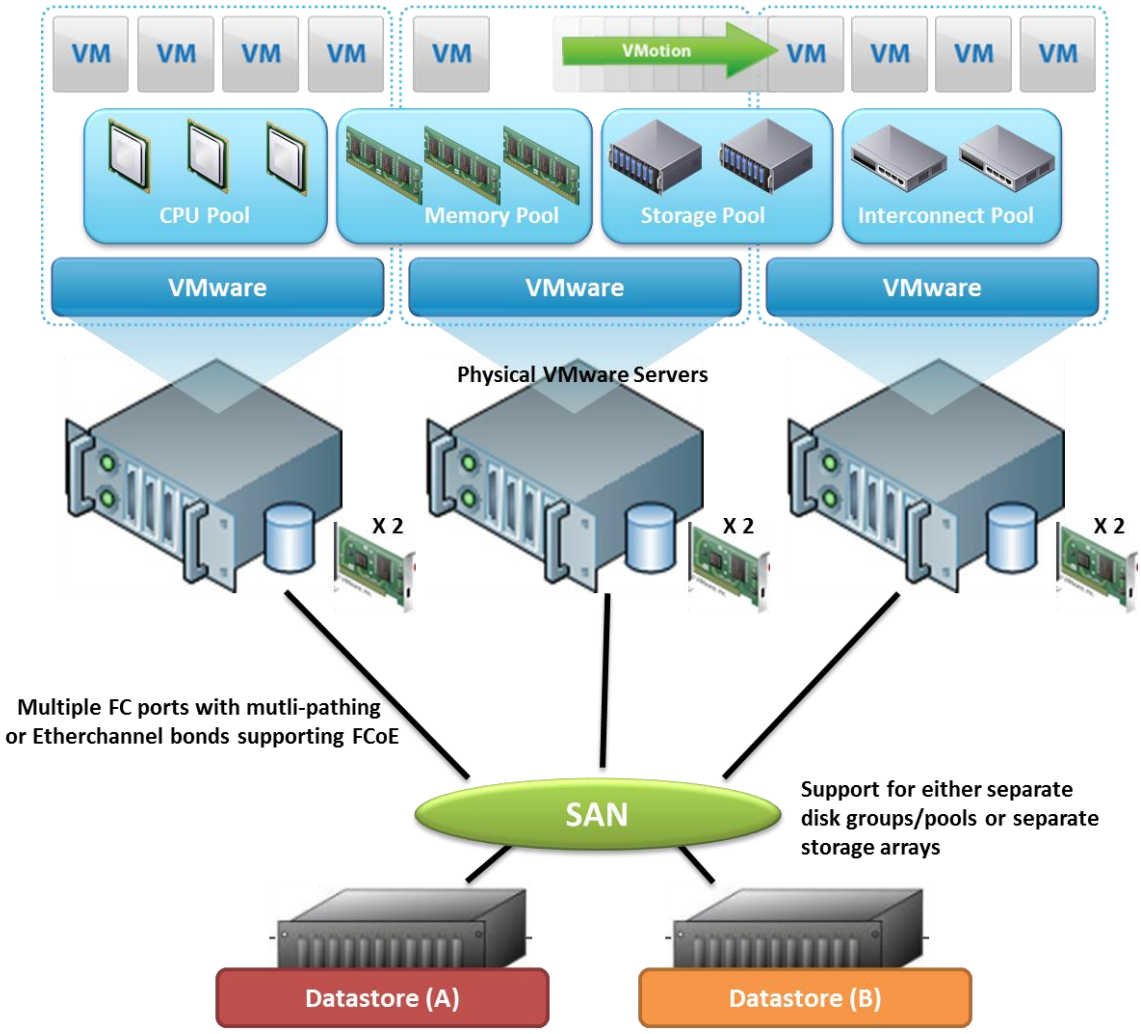

Figure 4: Redundant Storage with VMware vSphere

Figure 5 illustrates VMware HA** recovering a failed virtual machine from other physical resources. Here, a minimum of three (3) servers is required.

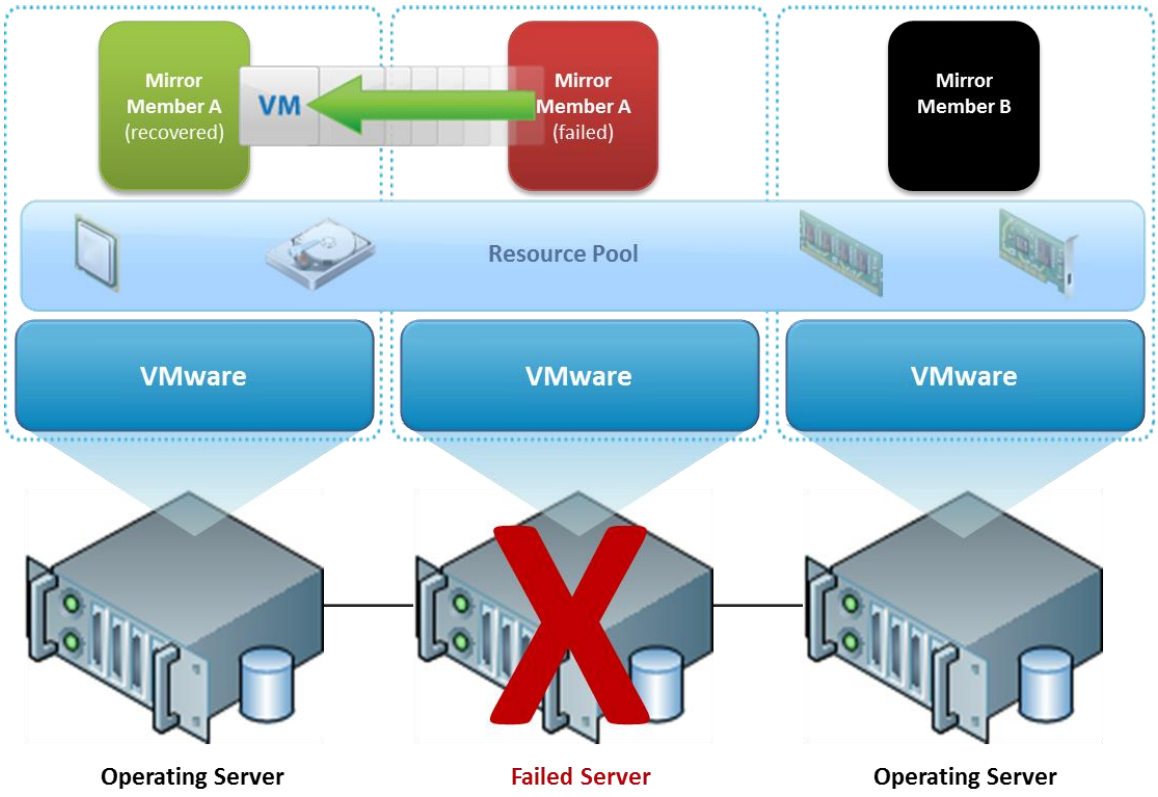

Figure 5: VMware HA with Database Mirroring

** Please note VMware is only used as an example. Other virtualization technologies from IBM, HP, Microsoft, and Linux KVM variants such as Red Hat RHEV-M that supports the automatic partition or virtual machine mobility/restart can be used as well.

APPENDIX C: SAMPLE ^ZMIRROR FOR RELIABLE NETWORK PING FAILOVER

A sample ^ZMIRROR routine is provided by InterSystems in order to assist you to implement the Mirroring Reliable Network Ping Failover strategy. This strategy is obsolete starting in versions 2015.1.

The latest sample file can be located at the following URL:
ftp://ftp.intersystems.com/pub/cache/ZMIRROR_RELIABLE_NETWORK_PING.xml

Download this file to your environment, and load it into the %SYS namespace as follows:

```
Do $System.OBJ.Load(''<path>/ZMIRROR_RELIABLE_NETWORK_PING.xml'',''ck'')
```

Once you have loaded this file, you will need to customize it to fit your environment. The routine itself contains self-explanatory steps and comments.

Please contact the InterSystems Worldwide Response Center for clarification, if needed.

APPENDIX D: MANUAL FAILOVER AFTER UNPLANNED OUTAGE OF PRIMARY

This appendix describes how to manually induce failover after an unplanned outage of the primary. It is used in conjunction with the strategies listed previously in this document when the software configuration alone cannot determine that it is safe to automatically fail over. This procedure also appears in the Mirror Outage Procedures section of the Caché documentation under Unplanned Outage of Primary Failover Member When Automatic Failover Does Not Occur .

Here, we assume that the primary system has completely failed. If the primary system is up, the backup system will automatically contact the ISCAgent on the primary to determine the status and possibly take over. In general, if simply restarting the failed primary is possible, it is preferred over any of the following manual failover steps.

If you have access to the journal files from the failed/down primary, you can use the following procedure without risking data inconsistency between the primary and backup:

1. Ensure that the primary machine is down and remains down during this procedure.
2. Shut Caché down on the backup machine.
3. Copy the necessary journal files to the backup. That is, re-copy the last mirror journal file that the backup has, and any newer mirror journal files that the primary may have created.
4. Delete the Mirror Journal Log file on the backup. This file is located in the *mgr* (manager) directory of the instance, and is named as follows: `mirrorjrn-{MirrorName}.log`. This will automatically be rebuilt in a later step.
5. Start up Caché on the backup machine.
6. Use the *Force Become Primary* function on the backup to become primary. This option is available from the ^MIRROR utility.

If you don't have access to the journal files from the failed/down primary, there may be a risk of data inconsistency between the primary and backup. If all of the following conditions are known with certainty, then a manual failover is safe to perform because the backup has all of the journal data from the primary as of the time of the failure:

- The primary server is down.
- The time when the backup disconnected from the primary matches the time that the primary server went down. The certainty with which you are able to make this determination is dependent on your environment and the type of failure. You can check the time that the backup disconnected by searching for a message similar to the following in the cconsole.log file on the backup: `MirrorClient: Primary AckDaemon failed to answer status request.`
- The backup was considered *active* at the time that it detected failure. An active backup is one that is caught up and is receiving data synchronously from the primary. You can confirm this by searching for a message similar to the following in the cconsole.log file on the backup: `Failed to contact agent on former primary, can't take over`. **Note**: presence of a message like `Non-active Backup is down` in the cconsole.log file on the backup indicates that it was not active at the time of detecting failure. In this case, it can be assumed that the backup does not have all of the necessary data from the primary, and it is not safe to perform manual failover, as it will result in data inconsistency.

To continue with the manual failover, use the *Force Become Primary* function on the backup to become primary. This option is available from the ^MIRROR utility.

If you decide to proceed with the manual failover, and there is data inconsistency between the two failover members, the following consequences can be expected:

- Your application may be missing data that was committed on the former primary but never transmitted to the backup.
- The former primary will refuse to re-join the mirror as backup when it is restarted.
- You will need to restore a backup of your databases from the current (new) primary to the other system in order for it to re-join the mirror as a backup.