

[**Editor's Note:** The following excerpt is from Chapter 2 of the free eBook *The Definitive Guide to Exchange Disaster Recovery and Availability* (Realtimerepublishers) written by Paul Robichaux and available at <http://cc.realtimerepublishers.com/portal.aspx?pubid=338>.]

## Chapter 2: Availability Building Blocks: Disaster Recovery

Chapter 1, which focused on the causes and potential solutions for downtime, gave you a high-level overview of what to look for and think about as you begin to consider how to improve the availability of your Exchange infrastructure and servers. There are several technologies that can be used to provide disaster recovery and business continuance capability. This chapter will explore the fundamental principles behind disaster recovery and look at technical solutions that purport to improve disaster recovery. It will then examine some of the design choices and tradeoffs you face in trying to design an effective disaster recovery plan.

### What Is Disaster Recovery?

Depending on whom you ask, the definition for *disaster recovery* can range from restoring data from a backup to restarting operations at an alternative business continuance site. This book will use the Wikipedia definition from [wikipedia.org/wiki/Disaster\\_recovery](http://wikipedia.org/wiki/Disaster_recovery), which defines disaster recovery as “The ability of an infrastructure to restart operations after a disaster.” Let’s add to that definition by clarifying that *disaster* means an event that causes loss of data or interruption in services. The combination of these definitions gives us a good starting point for a discussion of disaster recovery, although there are some additional nuances that need to be explored before talking about technical means of implementing disaster recovery.

The first is how you know when you’ve successfully recovered from the disaster. There are two common metrics for recovery:


- A recovery point objective (RPO) specifies the point in time at which your capabilities will return when recovery is complete. Let’s say you do a daily full backup at 2am. If you have a failure Tuesday at 8am, your RPO will probably be 2am Tuesday; in other words, your recovery will succeed if you can recover the state of your Exchange data to that particular point in time.
- A recovery time objective (RTO) specifies the maximum amount of time allowed for a recovery. For example, if your service level agreement (SLA) promises that you’ll restore operations within 6 hours of a disaster, you have a 6-hour RTO.

Although these metrics are clearly related, there are significant differences between them that become apparent as you start to consider how to reduce the RTO or move the RPO closer to the actual beginning of the outage. To shorten the interval between a failure and your RPO, you must make more frequent copies of your data with whatever protection mechanism you’ve chosen. To shorten the RTO, you need to take measures to increase the speed of your recovery. The remainder of this chapter will discuss both types of measures.


## How Exchange Backs Up Data

Conventional backup tools copy files from a source location to the backup media—*xcopy* is actually a backup tool in disguise. Leaving aside the question of how you select which files should be backed up, file-copy-based backups are an acceptable strategy for file-based data. However, Exchange (and other database applications such as Oracle and SQL Server) keeps its data files open all the time. There are commercial solutions that purport to allow you to safely and consistently back up files that are open by another application, but this method is somewhat unreliable—there is no guarantee that the backup copy of the file will have *exactly* the same data as the original version because a write operation to the source file might change its contents after that portion of the file has been copied.

Exchange attacks this problem in two ways: the Extensible Storage Engine (ESE) provides an API for backup tools to use to copy Exchange databases and transaction log data, and the Volume Shadow Copy Service (VSS) provides a way to cleanly copy volumes that contain data.

 This chapter will explore VSS in more detail later.

In both cases, you must have software that uses the appropriate API. Both of these alternatives allow you to make an *online backup*; that is, one that is made while the databases being backed up are mounted and available to users. In the case of the ESE API, Microsoft ships a modified version of the *ntbackup* utility as part of Exchange, and every major backup software vendor—including Computer Associates, Hewlett-Packard, VERITAS, and Yosemite—offers an Exchange-aware agent or version of their software. You can always ignore these products and make your own *offline backups*, which are made while the database is dismounted; however, these backups require you to interrupt service, and they don't check the integrity of the file as it is being backed up.

 If you're not familiar with how Exchange transaction logging works, a quick review might help. Microsoft's guide to using Exchange Server 2003 recovery storage groups (available at <http://www.microsoft.com/technet/prodtechnol/exchange/guides/UseE2k3RecStorGrps/6923d327-f5a1-48a6-bfdb-1ef9ef9a928c.mspx>) has an appendix that covers Exchange transaction logging in detail.

### Backup Types

Exchange supports four types of backup:

- Full
- Incremental
- Differential
- Copy

Each has associated benefits and drawbacks, most of which revolve around whether the transaction logs are purged as part of the backup.

## Full Backups

*Full backups* are straightforward. When you make a full online backup of an Exchange storage group, all of its databases are copied, and the log files for that storage group are removed after their data are incorporated into the databases (a process known as *truncation*). Full backups are self-sufficient because you can restore a complete copy of the database without anything else. For example, if you have a full backup taken at 0600 Monday, you can roll back to that point in time using only that backup.

## Incremental Backups

*Incremental backups* are pretty straightforward, too; they record only those database pages that changed since the most recent full backup. The Exchange online backup mechanism actually does this by copying the log files, not the entire database. As a result, to restore a database that has incremental backups, you need to have the full backup plus *all* of the incremental backups between the full backup and the desired RPO. It's common for Exchange shops to do weekly full backups combined with daily incrementals.

Suppose that your weekly backup is taken on Saturday at 0400, with incrementals Sunday through Friday. If you want to restore the database to Tuesday's data, you need the Saturday full backup, plus the incrementals for Sunday, Monday, and Tuesday. This requirement puts extra emphasis on backup media tracking and control procedures—if you lose an incremental backup set you can't restore any data *past* that backup. In this example, if you lose Monday's tape, you can only restore back to Sunday's system state.

## Differential Backups

*Differential backups* combine the best aspects of full and incremental backups. A differential backup must be used in combination with a full backup because it copies *all* of the transactions that have occurred since that full backup. A Saturday full backup coupled with daily differentials can be restored to any point in time by restoring the full backup first, then restoring the differential backup taken closest to the point of failure. This method makes media management somewhat easier, although the differentials increase in size as more time passes since the original full backup was created. In addition, later differentials take longer, which is often a bigger concern than the amount of space they consume.



Each of these backup types updates the header information in the database file that indicates when the database was last backed up. Depending on the backup type, the backup operation may change the checkpoint, which keeps track of which logged transactions Exchange has already committed to the database. This is important because the checkpoint plays a central role in restore operations and log playback.

## Copy Backups

*Copy backups* do nothing except copy the database data. They don't purge the transaction logs, and they don't update the database header to indicate that the database was actually backed up. Copy backups are the most transparent type of backup operation, which makes them a good choice for times when you want a known-good copy of the database but don't want to interfere with your existing backup processes.


## Choosing a Backup Type

Choosing a backup type may seem complicated, but it's not. The bottom line is that the amount of time required for a restore is roughly double the amount of time required to make the backup in the first place. Factor in your RTO to quickly determine how much time you can afford to do a restore, which in turn tells you how long your backup *can* take if you're going to hit your SLAs and RTO. You can always tweak your backup solution's hardware (for example, by adding more tape drives and striping data across them, or switching to a higher-capacity, faster solution), but the time required for the backup window will ultimately be the number one factor in determining your backup pattern.

Let's say that your RTO is 8 hours and that you have a total of 120GB of mail data evenly distributed over four servers. Thus, within that 8-hour window, you need to notice that a failure has occurred, locate any needed backup media, start the backup, wait for it to finish, and wait for any pending transaction logs to be replayed. You must also include a fudge factor to cover you in case something unexpected happens. Suppose that you actually have only 6 hours worth of restore window to work with. (In fact, during most restores, IT staffers waste time trying various procedures before they decide that a restore is *necessary*—be sure to factor this time into your planning!) Thus, your backup time should be at or below 3 hours. What kind of backups should you use?

- Full backups take the longest—assuming that your backup solution can handle 10 to 20GB per hour, you can restore one server's worth of data in 1.5 to 3 hours—assuming that nothing goes wrong.
- Incremental backups are smaller, so they take less time to capture and restore. However, they trade time for space; in addition, if the same database page changes more than once over the time span of an incremental set, you'll end up having to play back transactions to change that page over and over, adding to your restore time.
- Differential backups give you easier management and less overhead than incrementals, at the cost of storage growth over time. It's easy to grab a full backup, plus the differential for a given RPO, but you must factor in the time it takes to restore two backups instead of just one.

Without lab testing, it's difficult to pinpoint which combination of full, incremental, and differential backups will best allow you to meet your 8-hour RTO. However, if you have sufficient hardware to support it, daily full backups offer relatively easy restoration, little additional media management overhead, and integrity checking.

 In fact, full and copy backups are the only backup types in which the contents of every database page is checked for integrity, so you should run them periodically even if the other backup types suit your needs better.

Most organizations use a combination of weekly or intra-week daily backups with daily differentials, although your combination may vary. As disk space continues to drop in purchase cost, an increasing number of organizations are doing full backups to disk and intra-day differentials—doing so gives great coverage at the expense of storage space.



The online backup process checks the physical integrity of each database page, not the logical structure of the database itself. In other words, each individual page of the database is checked to make sure it's OK, but the links, tables, and views that collect groups of pages into folders, messages, attachments, and the like aren't checked; for that, you have to use Microsoft's *isinteg* tool.

### **Exchange Online Backups**

An Exchange backup using the ESE API follows a predictable set of steps:

1. The backup utility asks for a list of all the Exchange Server systems.
2. The backup utility connects to the specified Exchange Server system and makes a request to back up a particular storage group or database. The ESE API allows simultaneous backup or restore of as many as four storage groups, but you can only back up or restore one database within each storage group at a time.
3. If the ESE online maintenance task is performing maintenance on any databases in the storage group, that maintenance stops.
4. ESE flushes any dirty database pages to disk. These pages are those that have been changed but haven't yet been written to the on-disk copy of the database. At this point, the checkpoint is frozen.
5. The backup utility opens the first database file to be backed up. On Exchange 2000 and Exchange Server 2003, each individual EDB and STM file is backed up separately. For a full or differential backup, the database header is updated to point to the low anchor log file.
6. The backup utility issues repeated calls to read data from the file. It can then write that data using any backup mechanism.
7. When the backup tool is finished reading, it closes the database file.
8. Steps 3 through 5 are repeated with each additional file in the selected storage group.
9. The backup utility opens the first transaction log file for the selected storage group and copies its data, closing the file when done.
10. Step 7 is repeated for each additional transaction log in the selected storage group.
11. Once all the log files have been backed up, any log files marked for truncation are removed.
12. The backup program calls the ESE API to indicate that it's done with the backup.

Sharp-eyed readers will wonder what happens to transactions created while the database is being backed up. The answer might surprise you: they're logged to the transaction logs just as they would be during normal operation. Once the checkpoint is frozen in step 3, additional logs can be generated, but their transactions will not be committed until sometime after the backup completes. This method works because the log files generated while steps 4 through 7 are taking place will themselves be backed up in steps 8 through 10.

### **Exchange Offline Backups**

Not every Exchange backup is performed using the ESE APIs; it's possible to copy Exchange databases under a variety of other circumstances. By convention, any backup that doesn't use the online backup APIs is called an *offline backup*. This categorization includes making copies of dismounted databases using xcopy and using various tricky methods to make copies of open database files without closing and dismounting them. The Microsoft article "Offline Backup and Restoration Procedures for Exchange," which is available at <http://support.microsoft.com/kb/296788/EN-US/>, describes the process that Microsoft recommends for taking offline backups that include the log files necessary to do a complete restoration.

The big downside to offline backups is that they require you to do more manually, which is a concern during disaster recovery operations; more steps mean more possible ways to make mistakes, as well as more time spent performing the steps. For example, most savvy administrators will run esutil with the /K switch to check the restored database's integrity; doing so can add significantly to the restore time required. Microsoft's official position is pretty much that anything other than an online backup is an offline backup; this includes point-in-time and replicated copies.


## **Disaster Recovery Technologies**

The basic technologies used for disaster recovery will be familiar to most Exchange administrators. However, there are some relatively new technical wrinkles brought on by the increasing availability of low-cost storage and high-speed networks.

### **Basic Backup and Restore**

Backup and restore processes remain the fundamental disaster recovery tool for a simple reason: they offer a time-tested, proven way to get data back onto a server after it fails. Having solid, well-tested backup processes and being able to successfully perform restores on demand, is the first key indicator of an organization that has good disaster recovery capability. Accordingly, developing those processes and skills is a critical part of any plan to increase your uptime, because being able to restore from backup is a necessary part of meeting an RTO.

The basic idea behind backups is simple: take data from a server and store it, usually in compressed format, on another server or on removable media that can be stored in a safe location. Of course, the mechanics of how these steps are performed has a huge impact on how long the backup takes, how safe the data are, and how likely it is that a restore will be successful.

 The preceding sections in this chapter describe how Exchange backs up data.

## Backup to Tape: Pros and Cons

Tape backups have been the gold standard in disaster recovery for more than 40 years—but are they still? Tape-based systems have some important pros and cons that you should consider as part of your overall disaster recovery planning.

The biggest factor driving the use of tape is the total ownership cost. If you look back at the storage pyramid that Figure 1.2 shows, you'll see that tape systems occupy the bottom tier. The reason is simple economics: tape offers relatively large storage capability coupled with relatively low media costs. For example, a DLT-III tape currently sells for between \$25 and \$40 in single quantities. For that, you get between 35 and 70GB of storage space, depending on compression. These figures don't seem too impressive in the current environment in which disk drives offer storage costs of well under \$1/GB, but in quantity, the price of tapes looks better, especially when you factor in the cost of arrays, controllers, and the other paraphernalia that disk-based systems require. For true offline systems—where backup media are taken to a separate physical site and stored for long terms—tape is difficult to beat.

That's not the only reason tape technology is ubiquitous, though. It's a familiar and well-understood technology, and it scales relatively well on individual servers. If your backups are too slow, you can add more tape drives to back up more data concurrently, or you can move to a more expensive tape drive type to increase throughput. Library vendors such as STK and Exabyte offer large-scale tape libraries that can hold hundreds or thousands of individual tapes and switch between them very quickly, which provides near-line access to extremely large volumes of data. For larger numbers of servers, vendors such as CommVault and VERITAS sell backup solutions that allow automated backup of dozens, hundreds, or even thousands of servers to a central set of backup servers.

What are the downsides of tape backup technology? First, and most important for our purposes, tape restores are generally slower than disk-based restores. Microsoft's standard rule of thumb is that you should multiply the time it takes to capture an Exchange backup and double it to estimate the time required for a restore, and tape's relative slowness just exacerbates the problem. Next, tape-based restore processes are error-prone—one analyst firm estimates that more than 40 percent of tape-based restores initially fail. When you get ready to restore from tape, you're betting that the tape isn't damaged or suffering from media errors brought on by improper handling or inappropriate storage or environmental conditions. In fact, you're making a more fundamental bet: that you can *find* the tapes in the first place, and that once found, you can get them back to your recovery site in a timely manner. Of course, you can work around these potential problems by building redundancy into your backup processes, but that comes at an extra cost.

## Disk-Based Multi-Stage Backup and Restore

Tape's advantages as a long-term storage mechanism are clear, but so are the disadvantages of using tape as the linchpin of a backup system. This conundrum has led to the common deployment of multi-stage backups: protected data is initially backed up to disk and kept there for a limited period; the disk-based backups are then archived to tape. This approach has several advantages:

- It's fast—Backing up data to disk means that the backup runs at the speed of your storage subsystem, which can exceed the speed of tape systems by a factor of ten or more. This speed shortens the required backup window, which means you can take backups more often.
- It offers more frequent RPOs—Because the backup time window is smaller, you can easily decrease the intervals between backups, which gives you a way to quickly recover to a point in time.
- It puts less load on the Exchange server—In fact, depending on how you implement the disk-based portion of the backup process, there may be essentially *no* load on the server because the work is all done by the SAN controller when it makes a copy of the volume being backed up.

These advantages come at a cost, though. Per-gigabyte storage costs for tape are still significantly lower than for fixed disks, so if you have a large volume of data to back up, you'll need to maintain enough spare storage capacity to hold the backups *and* keep them around for the backup retention period. In addition, adding disk-based stages to your backup procedures makes them more complicated, so you'll need to spend some extra time and attention to ensure that backed-up data moves from stage to stage appropriately and that you have adequate storage monitoring and control technology so that you don't run out of storage space.

## Common Backup Pitfalls


You've probably heard television sports commentators say that a team did well or poorly based on the amount of emphasis the team gave to the fundamentals. Such is certainly true for backup and restore operations—it's the simple things that you do, or don't do, that can spell the difference between successful and failed restores when the chips are down.

First, be sure that your backups are actually *working*. It might shock you to know how many otherwise competent administrators have been undone over the years by the sudden discovery that their backup tapes were blank. This disaster is 100 percent preventable. Every Windows backup utility includes logging and reporting features that can tell you whether the backup completed, and the Exchange information store service logs a number of informational events that tell you when backups started and completed. More important, the information store also logs events that tell you whether errors were encountered.

For full backups, the information store calculates a checksum for each 4KB database page as it's read; the calculated checksum is compared against the checksum stored with the page. If they don't match, the information store logs a -1018 error to indicate the mismatch, and the backup terminates. (In addition, the backup process also checks that each page's "next page" pointer is pointing to a valid page.)

The one exception to this circumstance is that, as of Exchange 2003 SP1, the information store can fix some types of single-bit errors that would otherwise cause a -1018 error. However, if you see a -1018, -1019, or -1022 error generated by the information store, the error indicates a serious problem that warrants your immediate attention. The Microsoft article “Understanding and Analyzing -1018, -1019, and -1022 Exchange Database Errors,” which is available at <http://support.microsoft.com/kb/314917/>, describes more about these errors, what causes them, and how to troubleshoot them.

The gold standard method for verifying backup integrity is actually to restore the backup and see whether it contains the expected data. You can't rely solely on logging because all that tells you is that the data was successfully *read* (and verified, if you haven't turned verification off to save time). You don't need to check every backup, but you should do so often enough to maintain confidence that your backup procedures are working. As a happy side effect, if you regularly restore backups to test them, you'll vastly improve your disaster recovery skills, which will pay off in the event of an actual failure that requires restoration.

 Chapter 4 will talk more about the specifics of validating your backup design and processes as part of your overall HA design validation.

Third, remember the old adage “out of sight, out of mind.” If you use any type of offsite storage, be sure that you include it in your test plans. Can you get the media you need within the allotted recovery period? It's a wise idea to find out before you actually *have* to. In the same vein, be sure to test retrieved media to make sure that it hasn't been damaged in storage.

### **Vendor Snapshots and Point-in-Time Copies**

The Holy Grail of Exchange disaster recovery is to be able to instantly jump back to a particular RPO with as low an elapsed time as possible. Taken to the extreme, this ideal would mean being able to roll back to any point in time with zero elapsed time. We're not quite there yet, but vendors—including Microsoft—ship solutions that can drastically cut the recovery time for Exchange. These solutions depend on specific hardware and software combinations from particular vendors; examples include EMC's TimeFinder and VERITAS' Storage Foundation.

### **How Vendor Solutions Work**

These solutions typically work by exploiting disk mirroring, or RAID-1. In a conventional mirror implementation, there are two disks, one of which is the primary. (VERITAS calls the individual volumes *plexes*, a term which will be adopted here.) Software or hardware copies data from the primary plex to the mirror plex as it is written. Windows 2000 (Win2K) and Windows Server 2003 (WS2K3) provide software mirroring support, as do most Linux distributions. Many motherboards offer hardware mirroring, and mirroring is a standard feature on RAID disk controllers. The relationship between the plexes can be broken at any time, at which point the mirrored plex becomes a point-in-time copy of the data on it. In a conventional two-disk mirror setup, though, breaking off the mirrored plex leaves you without a spare copy of your data, and the whole point of mirroring is to provide data protection.

Point-in-time copy solutions that depend on mirroring keep *two* plexes of the master. Data written to the master volume is mirrored to both plexes simultaneously. To create a point-in-time copy, the mirror relationship between the master volume and the third plex is broken; at that point, the third mirror can be used as a point-in-time copy by moving its data to another computer, or logically moving the third mirror to another host on a storage area network (SAN).

One problem with this approach: when the third mirror is broken off, what guarantee is there that it contains complete and consistent data? The answer is simple—there is no such guarantee. Suppose that your Exchange information store is in the midst of writing a 4MB mail message into a user’s mailbox. That’s about 1000 4KB database pages. If the mirror is split after, say, only 920 of those pages have been written, the mirrored plex will be missing 80 database pages. The seriousness of this shortcoming depends on *which* 80 pages are missing, but this situation is not good no matter how it works out. Microsoft offers a solution that solves this problem—, as described in the next section on VSS.

Different vendors work around this problem in different ways. The safest way is to dismount the databases and storage groups that you want to back up before breaking off the mirror. This process is simple and quick and guarantees a complete and consistent copy. Of course, it has the undesirable side effect of kicking all users off that mailbox database, which means you probably shouldn’t do it during business hours.

## Vendor Solutions: Pros and Cons

Snapshot solutions offer essentially instant backups and very fast restores, which makes them a preferred solution in environments in which restore time is important. However, they cost more than traditional backup solutions, and there are some supportability issues that mean you must be very careful when choosing a product.

The bottom line from an Exchange perspective is that Microsoft expects you to treat vendor-produced snapshot backups as offline backups, as described in the Microsoft article “Hot Split Snapshot Backups of Exchange,” which is available at <http://support.microsoft.com/default.aspx?scid=kb;en-us;311898&sd=tech>. Thus, when you restore a snapshot backup, the associated transaction logs have to be replayed. This requirement increases recovery time; to cut the recovery time, you can do periodic online backups to purge the logs, but that takes time too. Most vendors who sell these solutions offer clever scripts and tools to hide some of the complexity of recovery operations, and these solutions have many satisfied customers. However, Microsoft is very clear that the primary support responsibility for these tools rests with the vendor, and that Microsoft’s involvement and support don’t include troubleshooting problems with, or caused by, a third-party snapshot solution. (This is not to say they won’t *try* to help, merely that they don’t see themselves as the primary support provider.)

The following quote is a joint statement from Microsoft and VERITAS from July 2004 about VERITAS' Storage Foundation product:

VERITAS Storage Foundation replaces core Microsoft Windows disk management services with a proprietary VERITAS solution. The result is a solution that is in part supported by Microsoft and in part supported by VERITAS.

To be very clear: Microsoft will provide support for Microsoft Exchange issues if you run Exchange on a VERITAS Storage Foundation platform. However, Microsoft will only troubleshoot and attempt to resolve Exchange-specific issues up to the point that the source of the problem can be reasonably attributed to an issue or incompatibility with VERITAS software. This same principle also applies to other third party products.

 For more information about this support issue, see <http://support.veritas.com/docs/269626>.

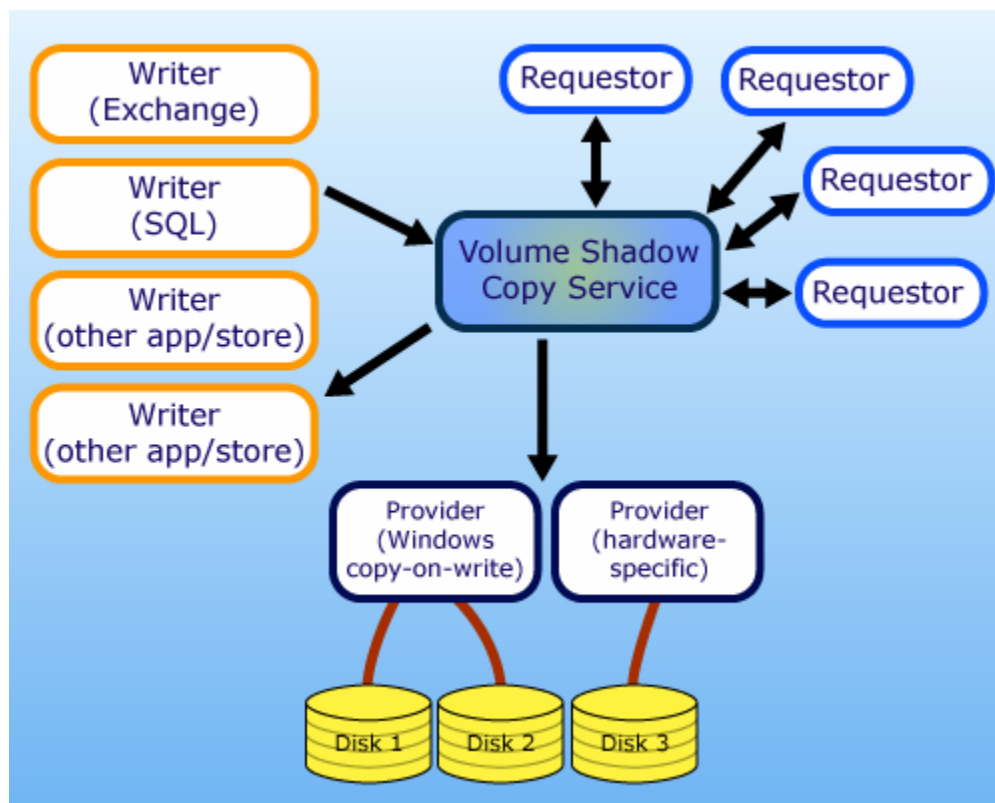
You could easily substitute Hewlett-Packard or EMC in the previous statement still have a true statement. This policy doesn't intend to paint these solutions as bad or troublesome; it merely tells you that you must understand who is responsible for providing support when things go wrong.

## VSS

The Exchange product team has long known that Exchange administrators want the benefits of point-in-time copies. However, building support for these backups into Exchange was problematic because Exchange doesn't handle its own disk I/O; that is delegated to the Windows I/O manager component of the kernel. Without a way to freeze I/O requests to a set of Exchange volumes, there wasn't a good way to enable Exchange to take safe point-in-time copies. Enter VSS, introduced in WS2K3. Exchange Server 2003 supports VSS; together with a VSS-compatible backup program, these two products give you a safe and Microsoft-supported way to make point-in-time copies of your Exchange databases and transaction logs.

## How VSS Works

VSS is conceptually pretty complex. There are several interlocking components, some provided by the OS, some by Exchange, and some by the vendor of your backup hardware and software. Figure 2.1 shows a static representation of how these parts work together. However, a better way to understand how VSS pieces interact is to step through an actual VSS backup. The Microsoft article "Exchange Server 2003 Data Backup and Volume Shadow Copy Services" (which is available at <http://support.microsoft.com/?id=822896>) describes the process in fairly dry detail.



**Figure 2.1: How the VSS components work together.**

The following list highlights the VSS process:

1. The administrator launches a VSS-aware backup tool and starts a backup. In this case, “VSS-aware” means that the backup tool incorporates a VSS requestor, the component that is responsible for telling VSS which data should be backed up and when the backup is starting.
2. VSS accepts the request and finds the requested application, returning a list of data sources (in Exchange’s case, storage groups) that can be backed up. Only those applications that have registered their writer components with VSS will have data available to the backup tool. The writer is responsible for performing application-specific actions to prepare the application data for copying by VSS.
3. The backup application selects the data it wants to back up, then tells VSS to start the backup. VSS in turn notifies the Exchange writer, which Microsoft ships as part of Exchange Server 2003, to prepare its data to be backed up. The writer turns around and asks VSS to freeze write requests to the volumes that hold the Exchange data to be copied. Read operations, and any writes that are already partially complete, can continue; new write operations are queued.
4. After VSS finishes freezing I/O for all target volumes (the logs and databases are probably on separate volumes but need to be backed up together), it notifies the writer that it’s clear to proceed. In Exchange’s case, the writer will close the currently active log file, rename it, then create a new temporary log file. The writer also calculates the range of log files that need to be included in the storage group backup and passes that information to the requestor.

5. When the writer is finished preparing the application data, it signals VSS, which notifies the requestor that it can proceed with the backup. The requestor then asks a VSS provider to actually copy the data. VSS includes a basic provider for copying data between disk volumes; SAN hardware vendors such as EMC and XioTech often ship their own provider to enable additional kinds of backup mobility.
6. When the provider finishes making its copy, it notifies VSS, which notifies the backup application. Once the backup tool decides that it's finished, it tells VSS to release the application data, at which point I/O to the relevant volumes is unfrozen and normal operation resumes.

### VSS: Pros and Cons

VSS was designed to provide the fast backup and recovery of third-party snapshot solutions in a way that would be supported by the Windows kernel and Microsoft applications such as Exchange and SQL Server. The biggest pro of VSS is that it delivers this capability in a way that is guaranteed to always produce consistent backups; because Exchange is integrated with VSS, you can directly restore a VSS backup to an Exchange server in the same manner (although with a different API) as the online backup mechanism.

Getting this degree of support required some fairly extensive changes both to Windows and to Exchange, which is why you must be running Exchange Server 2003 on WS2K3 to use VSS. This requirement can be a pro or a con, depending on the combination of versions you're using and what your upgrade plans look like.

In the same vein, to use VSS, you will need to have compatible hardware and backup software. This requirement might mean upgrading what you've already got. VERITAS, CommVault, Yosemite, and other major vendors support VSS in their latest versions. Depending on what type of SAN you have, you might be able to take VSS shadow copies and logically move them between hosts on the SAN, which enables off-host backup. When considering VSS deployment, factor in the cost of upgrades required to your SAN and backup infrastructure.

#### Off-Host Backup

Point-in-time copies can be quickly made and quickly restored, but they have a significant drawback: they take up lots and lots of disk space, because they contain exact copies of the Exchange data you back up. For longer-term storage, you still need to provide a way to back up this data to a more permanent archive medium, such as tape. However, if you need to do tape backups on your production servers, you're essentially doing away with many of the advantages of point-in-time copies, particularly the speed of making the backup and the low system load imposed during the actual backup.

One way to solve this problem is to let another computer perform the backup by moving the point-in-time copy to another system and then backing it up to tape. This method offloads the backup process from the production Exchange server, neatly solving the timing and system load problems (or at least moving them to another location). Off-host backup is normally implemented on SANs using a feature known as volume transport. The idea is that you can take a point-in-time copy onto a volume (using either a third-mirror system or VSS), then tell the SAN to logically reassign the volume from its existing host to another host on the SAN, then put it back. This process requires a SAN that supports logical volume transport, but if you have support for this feature, it can be a valuable addition to your backup processes.

If you're interested in learning more about off-host backup, The Microsoft white paper "Backup Process Used with Clustered Exchange Server 2003 Servers at Microsoft" (available at <http://www.microsoft.com/technet/itsolutions/msit/operations/exchbkup.mspx>) is a pretty interesting read that explains how Microsoft uses multi-node clusters with dedicated SANs to do exactly this.

## Replication

Keeping backups of your data is necessary, but it's not necessarily sufficient to give you complete protection. After all, what happens if your computer room or data center catches on fire, gets flooded, or is sealed off by the FBI? The terrorist attacks of September 11th spurred a surge in interest in solutions that allow for replication of critical data between physically separate locations. There are several different ways in which these solutions can be implemented, each with its own pros and cons.

### How Replication Solutions Work

Fundamentally, the basic idea behind replication solutions is simple: when a change is made to protected data on a source machine, that change should be captured and replicated to a target machine somewhere else. The details of how this basic idea is implemented are where things get interesting. You can generally categorize replication solutions in two ways: hardware versus software and synchronous versus asynchronous. All of these solutions require a certain amount of network bandwidth between the source and target, but the type and amount required vary.

#### *Hardware vs. Software Replication*

Hardware-based replication systems are pretty much relegated to the world of high-end SANs because they perform replication at the disk-block level, and the replication itself is performed by the SAN controllers. When a changed block is written to a protected volume, the SAN controller captures the data and ships it to the remote SAN controller, which writes the changed block on the target replica. This approach has some advantages—notably that it happens well below the level where Windows and Exchange live. Assuming that the replication setup works properly, Exchange will never know that its underlying data are being replicated, which is a huge bonus. Of course, these solutions tend to be expensive because they require identical SANs on either end, and most vendors sell the replication capability as a separate option. On top of the initial hardware purchase cost, you must provide enough low-latency bandwidth to handle disk-speed replication, which might involve such exotica as long-distance runs of optical fiber and all the assorted paraphernalia that come with it.

One of the great things about the speed and power of modern CPUs is that we can use them to perform many tasks that would formerly have required specialized hardware. Data replication is no exception; software replication solutions work in much the same way as their hardware ancestors did, just without the hardware. These solutions can be implemented in multiple ways, depending on the software vendor's design goals:

- File system filter drivers use the Windows kernel mechanisms for hooking drivers into the file system; the driver's job is to watch any changes to a specified file (or folder) and copy those changes to the target
- Block-level drivers monitor changes to volumes at the disk-block level rather than at the file level; this setup has the advantage of letting the driver combine operations in much the same way that hardware replication solutions do—the tradeoff is that these solutions normally require you to monitor an entire volume, which may not match up well with the disk topology you're using for your databases and transaction logs

Vendors of both types of solutions have implemented optimizations to avoid replicating data needlessly. For example, most products are smart enough to only transmit changes to files instead of retransmitting the entire file any time it changes; in addition, many products will aggregate changes whenever possible to reduce the amount of bandwidth required.

Speaking of bandwidth—it's the limiting factor in how well software-based solutions work. Most of them are designed to work well over LAN or WAN bandwidth, which means that they normally implement both queuing and throttling. Updates are added to a queue on the source as they happen; when bandwidth usage permits, the update at the head of the queue is transmitted. This method helps to smooth the flow of updates between source and target. However, if the link between source and target is overloaded or goes down for a long period, the queue will fill up, preventing further updates until the link comes back up, at which point you'll probably have to resynchronize source and target.

Throttling is the other significant component of software-based replication; depending on the vendor, you can either specify a percentage of bandwidth usage or an absolute value. Either way, the replication controller is responsible for controlling how fast updates are sent. A useful feature to look for is the ability to vary the throttling limits by time of day so that you can change the amount of bandwidth at different times throughout the work day.

One of the functions of resynchronization is to allow you to bring a source and target that have diverged back into synchronization. For example, let's say that you wanted to do an offline defragmentation on the source machine. Because doing so will touch every database page, it would be pretty pointless to let replication run during the defrag. However, once it completes, you would need to resynchronize the source and target data.

### ***Synchronous vs. Asynchronous Replication***

The other key difference between replication solutions is how replication happens. Suppose that a particular data block (let's call it A) on the source server is written, followed by a write request for block B. In synchronous systems, the write request for block B on the source machine will be blocked until the source replication driver gets an acknowledgment that block A has been written to the target. This process has the advantage of guaranteeing that the source and target are in lockstep, as no further writes can be applied to the target until block A is written and acknowledged. The problem, though, is that this method can impose an unacceptable degree of latency on Exchange, which generally expects its write requests to take no more than 500 milliseconds. Thus, synchronous replication is normally the exclusive province of hardware systems, which essentially hide the increased latency by using their large caches to soak up the extra time required for synchronous acknowledgement.

Asynchronous systems decouple I/O on the source and I/O on the target. To revisit the earlier example, after block A is written on the source, it's queued for transmission to the target, and block B can immediately be written. The replication controller becomes responsible for making sure that blocks A, B, and so on are transmitted to the target, and the target's replication controller must ensure that the received updates are applied in the correct order. Asynchronous replication tends to be well-suited to WAN-based replication because it doesn't force the source system to suffer from transient increases in latency. Most software replication products default to asynchronous replication.

## Replication: Pros and Cons

Is replication a sensible solution for your requirements? It depends. On the positive side, replication can provide a means to duplicate your most critical data in an alternative location. This benefit can add significant redundancy. Software replication solutions tend to be pretty flexible; you can easily use them to replicate file server data, Exchange databases, Exchange transaction logs, or any other data that you want to protect.

On the opposing side, replication technology is not officially supported by Microsoft, which means you have to lean on your replication vendor for first-line support. In addition, both software and hardware replication require a significant amount of bandwidth for adequate Exchange performance. Hardware replication solutions also tend to be very expensive; software solutions are more affordable.

## Design Choices and Issues

As you prepare to start assessing your Exchange environment to see how to improve its availability, you face some design choices that will strongly influence what your disaster recovery capabilities will look like.

### ***Onsite vs. Offsite***

Where you keep your backups will influence how quickly you can perform restores, but there are other implications too. Let's start with the obvious tradeoff—it's convenient to store your backups near your servers so that you can get to them quickly when needed. In contrast, if something bad happens to your servers, it's quite possible that your backups will be affected too. Offsite storage offers the possibility that your backup data will remain available to you even if your building burns to the ground or all your servers are blown away by a tornado.


The best strategy for most organizations is probably a mix of the two: keep some backups onsite and others at a secure offsite location. For example, if you keep backups older than 1 week offsite, you can still do intra-week restores without having to wait for your backup tapes to be retrieved from the offsite location. Depending on your backup strategy, and the difficulty of moving media to and from your offsite location, you might even choose to take an extra intra-week full backup just for offsite storage.

Whichever storage method you choose, there is another risk to be aware of. Two large companies (Time Warner and Wells Fargo) have recently been forced to deal with a firestorm of negative press (and potential liability issues) because backup tapes containing confidential customer information were stolen in transit between their data centers and their offsite storage location. Remember, anyone who can get access to your backup tapes can use them to restore your Exchange data and then read it (unless, of course, you encrypt it), so whether you use onsite or offsite storage, you should protect your backup media to the same degree you protect the data on your servers.

### **Recovery vs. Redundancy and Resiliency**

Let's assume you have a finite budget (I know, it's a stretch, but play along!). You can spend money on improving your ability to recover from failures or on adding redundancy and resiliency to reduce the odds that you'll *need* to recover from a failure. What should you do?

Generally, I advocate spending your money to improve your recovery capability *first*. Why? Because you'll always need the ability to do complete recoveries, no matter what kind of resiliency and redundancy you add with technologies such as clustering. No single technology gives you 100 percent protection, so you're always going to need to be prepared to do a full restore. Being able to do it right is a key requirement that should be met before you spend money on anything else.

 The next chapter will talk more about the specific things you can do to add redundancy and resiliency.

The one case in which this recommendation may not apply is when you can sharply reduce the odds of needing to perform a recovery. For example, let's say that your servers are using ordinary disks, with no RAID. Adding RAID will greatly reduce the risk of a disk failure that might necessitate recovery, so it might make sense to put your budget to work by adding redundancy to lower the odds that you will need to perform a recovery.

### **Service Providers vs. Do-It-Yourself**

A number of companies specialize in business disaster recovery, offering services ranging from canned disaster recovery plans to hosted offsite operations centers that provide equipment and bandwidth. The basic idea behind these services is that the service company will do a better job planning and executing a disaster recovery than you will because they have more experience and expertise. In many cases, this claim is probably true. However, outsourcing this type of business-critical operation is always dangerous because you are essentially betting your business that the service provider will deliver when the chips are down.

For most organizations, it's better to spend money on improving your own internal disaster recovery capability and competence than to pay someone else to do it for you. It might make sense to supplement your own planning and execution capabilities by using a service provider to review your plans or to provide services (like an offsite data center) that you can't afford to maintain on your own.

Besides the full-service companies, a growing number of outfits offer Exchange-specific disaster recovery services that claim to be able to recover data from damaged or corrupt databases. Many of them are using commercial tools such as Quest Recovery Manager for Exchange that you could just as easily buy and use yourself; others rely on their own tools, which work with varying degrees of success. If you've carefully designed and implemented your disaster recovery processes, you won't need these kinds of services; if you do, that's probably a sign that your disaster recovery processes aren't quite perfected yet.

### ***Planned vs. Unplanned***

The technologies described in this chapter can help you in two ways: they can reduce the risk of unplanned downtime, and they can make it easier to implement sensible planned downtime measures. As you might expect, though, in a chapter on disaster recovery, most of the benefits fall into the first category: replication, VSS, and even the humble conventional backup process can all be used to meet your RTO and RPO.

However, don't neglect the potential impact of these disaster recovery building blocks on your planned downtime. One key aspect of planning for maintenance downtime is that you must be prepared in case your planned downtime turns into unplanned downtime, and these technologies can be used to effect a quick recovery if that happens. From a broader perspective, disaster recovery planning (and the technology deployment that goes with it) should embrace the idea that you will sometimes need to make time for planned maintenance, and that such maintenance should be carried out without disrupting normal business operations.

## **Disaster Recovery Planning**

This guide focuses on Exchange disaster recovery and availability, so it's not the appropriate place to provide a complete guide to disaster recovery planning. However, disaster recovery planning is so important that it's worth mentioning, even if only briefly. There are three components to a successful recovery plan:

- Having a plan—Your plan must account for every possible contingency that might necessitate a recovery. At a minimum, this plan will include hardware failures, corruption or loss of your Exchange data, failure of the infrastructure components (such as Active Directory—AD—and electrical power) that Exchange requires, and interruption of physical access to your servers. For each of these contingencies, you need to have a response. This response might be simple (for example, if non-critical hardware breaks, you wait for the vendor's service technician) or complicated (if your Los Angeles data center is damaged by an earthquake, you fail over its operations in your Denver data center). The point is to accurately describe the potential problems you might run into, and to have solutions identified for them.
- Being able to follow the plan—Just having a plan is fairly useless if you don't also have the ability to put your plan into action. This action will probably require a combination of money, persuasion, education, management support, and acquisition. For every solution you identify in your disaster recovery plan, you must have the necessary mix of equipment, skills, and preparation to make it actually happen.

Every cliché you've ever heard about the value of prior planning applies here, in spades. The best way to make sure that your disaster recovery plan includes both of the necessary components is to write down the plan and then practice it. Writing down the plan is important because it sets out everything that you think should be included—and that makes it easier to identify what's *not* included but should be. Practicing the plan is important because prior testing will make it much easier for you to identify shortcomings in the plan, in your equipment or infrastructure, or in the people who have to implement it.

The third component of a successful disaster recovery plan is perhaps the most often overlooked—keeping the plan up to date as your IT operations, staffing, and business requirements change. For example, a disaster recovery plan originally written for Exchange 5.5 doesn't take advantage of some of the best new features in Exchange Server 2003, such as recovery storage groups (RSGs). A plan that assumes restore windows of 12 hours might not work well when the actual current SLA only allows for 6 hours of downtime. Performing regular and frequent tests of your disaster recovery plan will act as an antidote to this problem by highlighting areas of the plan that need to be brought up to date.

## Summary

The first chapter of this guide focused on downtime. This chapter takes a complementary approach by pointing out some of the basic technical building blocks you can use to implement better disaster recovery for your Exchange environment. Replication, VSS, and conventional backup and restore technologies all have their place, but you have to know which ones to use to get maximum benefit. The next chapter will move on to exploring some of the technologies that you can use to improve the availability of your Exchange servers.

**[Editor's Note:** This content was excerpted from the free eBook *The Definitive Guide to Exchange Disaster Recovery and Availability* (Realtimepublishers) written by Paul Robichaux and available at <http://cc.realtimepublishers.com/portal.aspx?pubid=338>.]