Realtime publishers

The Definitive Guide™ To

Windows Application and Server Backup 2.0

sponsored by

AppAssure HOME OF BACKUP 2.0

Don Jones

How can this guy
be so calm
when a mission-critical
server just failed?

LIVE RECOVERY
with Replay 4

Find out how you can be up, even when you are down.

FREE TRIAL!

Replay4

www.appassure.com
703-547-8686

Copyright Statement

Chapter 3: Whole-Server Backups

Recovered from the horror stories of the previous chapter? Ready to start *ensuring* solid backups in your environment, the Backup 2.0 way? That's what this chapter is all about, and what I call "whole server backups" is definitely the right place to begin. This is where I'll address the most common kinds of servers: file servers, print servers, directory servers, and even Web servers—the workhorses of the enterprise. I'll show you what some of the native solutions look like, discuss some of the related Backup 1.0-style techniques and scenarios, and detail why they just don't cut it for today's businesses. Then I'll assemble a sort of Backup 2.0 wish list: All the things you *want* in your environment for backup and recovery. I'll outline which of those things are available today, and wrap up by applying those things to some real-world server roles to show how those new techniques and technologies impact real-world scenarios.

The Technical Details

What's so technical about whole-server backup? Windows stores critical data in a number of places, and some of them are files and databases that are continually open and under modification by the operating system(OS): the Windows registry, Active Directory's (AD's) database, and certain critical OS files are just a few examples of these. These files are difficult to back up and restore simply because the OS itself can't "lock" the files to provide a "clean" image of the file. In other words, because the files are constantly open and constantly changing, traditional backup solutions can't easily "see" the complete file to back it up.

Whole-server backup also includes user files, such as those stored on a file server, along with numerous configuration databases associated with Windows itself. Although all of these might not be open at all times, they can still be tricky to back up because they *may* be open during the brief window when the backup software is running.

So the goal with whole-server backup is to get a good, usable backup of the *entire server.* And by "usable," I mean that the backup can serve our main business goals related to backup and recovery, which I stated in Chapter 1:

> Backups should prevent us from losing any data or losing any work, and ensure that we always have access to our data with as little downtime as possible.

As little downtime as possible suggests that we need to do more than just back up the entire server in a way that facilitates restoring the *entire* server; we may also need to recover a single file, or a single configuration database, or a single AD object. Being able to recover *just* the data we want from a backup will help reduce downtime, as recovering a single file is obviously—or should be—much faster than recovering an entire server.

We also have to recognize that *downtime* doesn't just apply to the server that we're recovering; it also applies to the people who are waiting for the recovery to be complete. We can get a user back to work faster by recovering the one file that the user needs rather than having to recover the entire server. That said, we'll certainly want the *ability* to recover the entire server, in the event that a complete disaster occurs and we lose the entire server. When that happens, the ideal is to lose as little work as possible, meaning whatever we're using for backups should be *continuous.*

Traditional Techniques

The technical details here present one significant technical challenge: open file access. That is, the ability of a backup solution to get a complete, consistent backup of a file that's currently open and in-use. Over the years, a number of techniques have been developed to address these issues.

One technique is called a *locked file backup (LFB)*, and it's a fairly generic technique that's in use on many OSs. Basically, when an application requests a backup of a file, the LFB logic—which may be embedded in the OS or provided by an agent of some kind—checks to see whether the OS has the file open for any other applications. If it does, the LFB logic waits for a pause in write activity to the file—a pause lasting for some predetermined amount of time. When a pause occurs, the LFB makes a copy of the file as it exists at the time, and offers that copy to the backup application. In some instances, LFB will operate on an entire disk volume rather than on a per-file basis. Working across an entire volume helps ensure that the files are consistent with one another but also makes it more difficult to find a pause during which all the open files can be copied and cached.

Windows introduced a new technique with its Volume Shadow Copy (VSC) service. I'll discuss VSC's operation in more detail later in this chapter; for now, suffice to say that the VSC keeps a local, on-volume backup of changed files. It can then offer those files to the backup application when a file's *current* version is open at the time of the backup. That means the backup is typically getting the *previous* version of a file rather than the current version; you might say that because the file is open, the "current" version hasn't yet been created.

Note

Microsoft refers to this feature as VSS, or Volume Shadow copy Service.

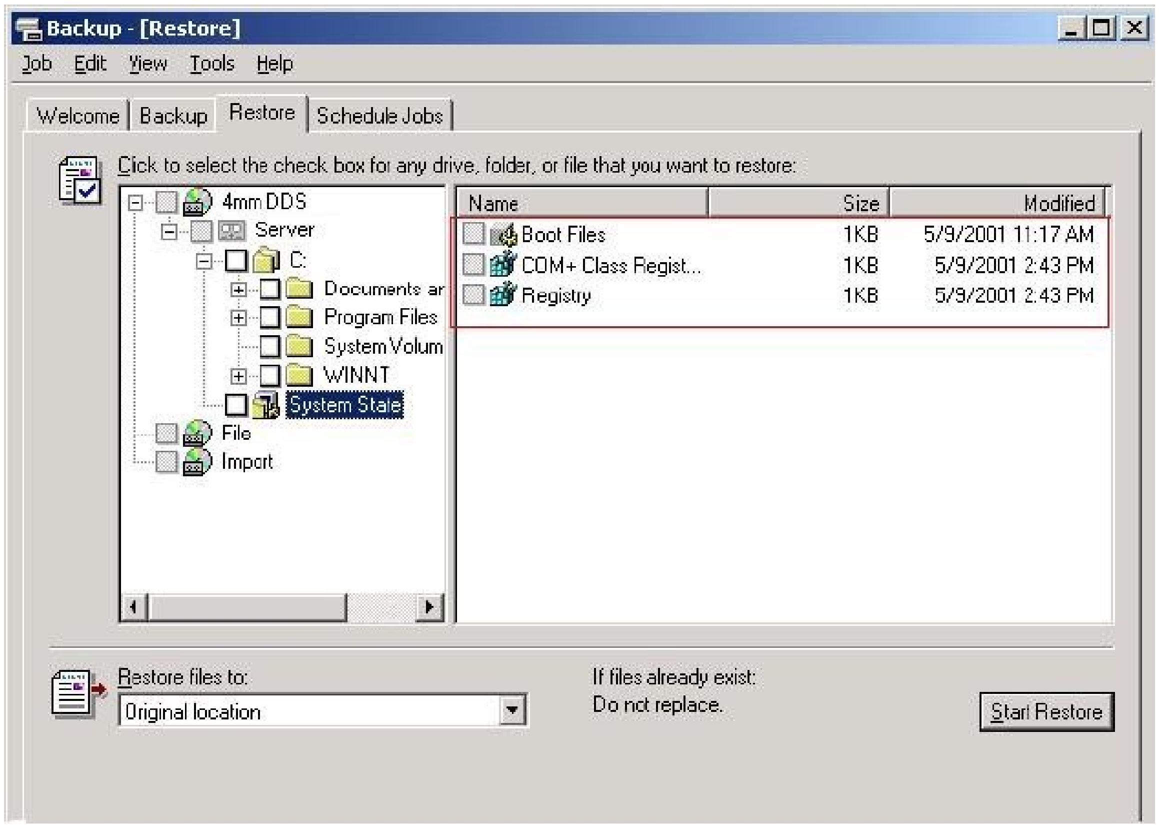
Both of these techniques are, as you can see, pretty complicated, and they don't work every time. Files that are *constantly* open will defeat techniques such as VSC, for example. LFB can be defeated by files that never have a long enough pause in write activity, or in the case of whole-volume LFB logic by large, busy volumes that never have a long enough volume-wide pause in write activity.

Native Solutions

I want to take a look at the native backup solutions that are built-in to the Windows OS. There are really two: the bundled backup application and the aforementioned VSC.

Windows Backup / Windows Server Backup

Prior to Windows Server 2008, Windows Server came with a fairly primitive backup and restore application that was essentially unchanged since its introduction in Windows NT 3.1 in the early 1990s. Figure 3.1 shows Ntbackup.exe, which offered basic backup and recovery of files on disk and, as shown, of the critical *System State*.



Name	Size	Modified
Boot Files	1KB	5/9/2001 11:17 AM
COM+ Class Regist...	1KB	5/9/2001 2:43 PM
Registry	1KB	5/9/2001 2:43 PM

Restore files to: Original location

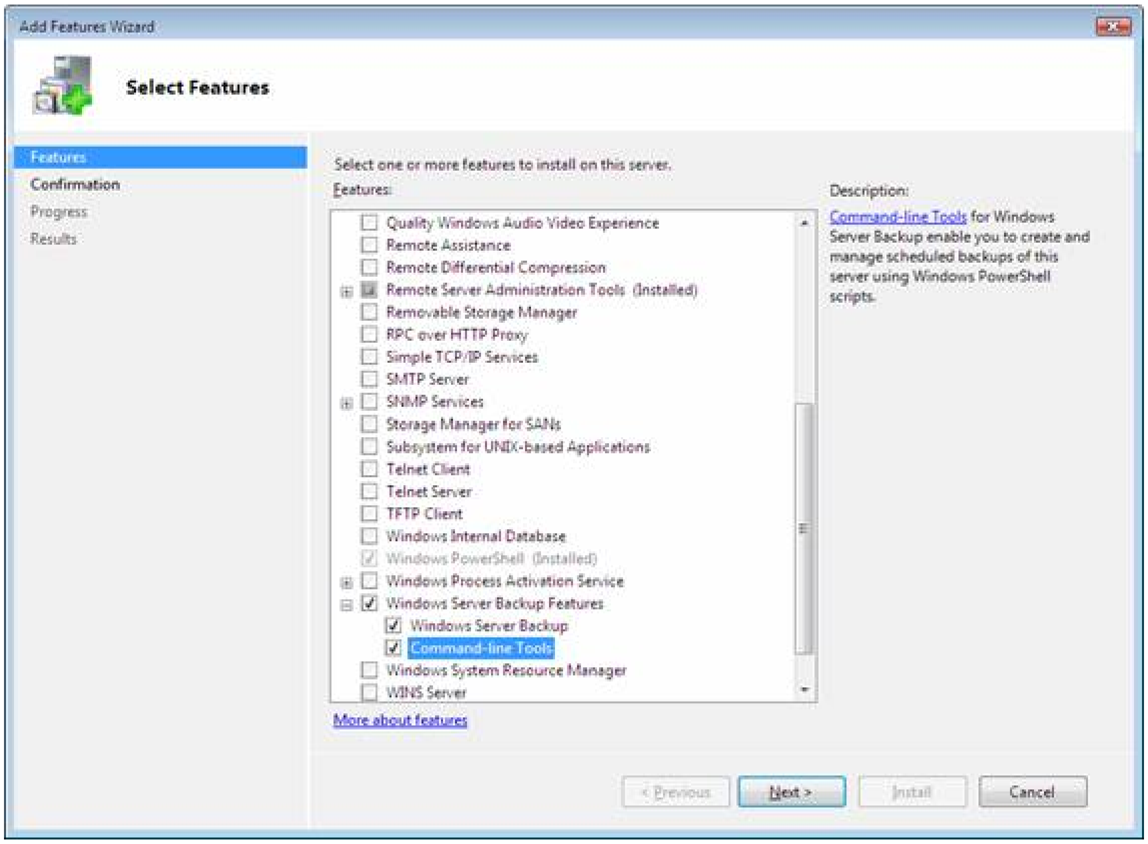
If files already exist: Do not replace.

Figure 3.1: Windows Backup, or Ntbackup.

The System State consists of Windows' boot files, the Windows registry, COM class registration (primarily the names and locations of DLL files), and other configuration data related to the OS itself. The application could back up to locally-attached tape drives or to a file, and it could recover the entire server or individual files. A shortcoming in the whole-server recovery method is that you first had to install Windows, then use the application to recover the server. In other words, the application didn't support any kind of *bare-metal* recovery, where the backup could be applied to a brand-new server that had no OS installed. The need to manually install Windows prior to beginning the recovery added a few hours, at best, to the recovery process, and made the tool unsuitable for all but the very smallest and risk-tolerant environments.

The application also lacked internal scheduling of backups. Instead, it used the Windows Scheduled Tasks functionality to schedule the command-line version of the tool. This provided the basic ability for scheduling backups, but from a business perspective, it was suitable only to the very smallest environments.

Although valuing the many third-party applications that sprang up to fill the backup and recovery gap left by the native application, Microsoft eventually recognized the need to provide somewhat more modern and sophisticated backup capabilities in the OS's native toolset. So, in Windows Server 2008—after more than a decade of Ntbackup—Microsoft introduced Windows Server Backup. As Figure 3.2 shows, this feature must be explicitly added to the server by an administrator.



Select one or more features to install on this server.

Features:

- ☐ Quality Windows Audio Video Experience
- ☐ Remote Assistance
- ☐ Remote Differential Compression
- ▨ Remote Server Administration Tools (Installed)
- ☐ Removable Storage Manager
- ☐ RPC over HTTP Proxy
- ☐ Simple TCP/IP Services
- ☐ SMTP Server
- ☐ SNMP Services
- ☐ Storage Manager for SANs
- ☐ Subsystem for UNIX-based Applications
- ☐ Telnet Client
- ☐ Telnet Server
- ☐ TFTP Client
- ☐ Windows Internal Database
- ✓ Windows PowerShell (Installed)
- ☐ Windows Process Activation Service
- ✓ Windows Server Backup Features
  - ✓ Windows Server Backup
  - ✓ Command-line Tools
- ☐ Windows System Resource Manager
- ☐ WINS Server

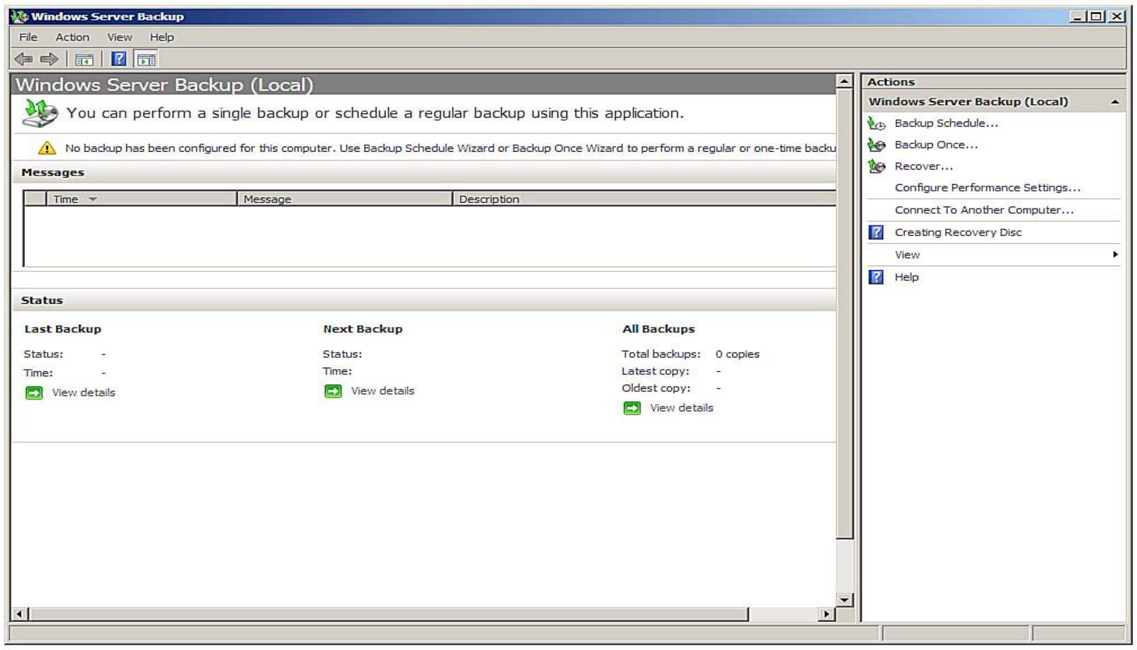
More about features

Description:

Command-line Tools for Windows Server Backup enable you to create and manage scheduled backups of this server using Windows PowerShell scripts.

Figure 3.2: Adding Windows Server Backup to a server.

Once added, the application, see Figure 3.3, offers a number of improvements over its predecessor, including the ability to natively schedule backups and the ability to connect to remote servers and manage their backups. The application can create a *recovery disc,* which makes it easier and faster to do bare-metal recovery.



Windows Server Backup (Local)

You can perform a single backup or schedule a regular backup using this application.

No backup has been configured for this computer. Use Backup Schedule Wizard or Backup Once Wizard to perform a regular or one-time backu

Messages

Time	Message	Description

Status

Last Backup
Status: -
Time: -
View details

Next Backup
Status:
Time:
View details

All Backups
Total backups: 0 copies
Latest copy: -
Oldest copy: -
View details

Actions

Windows Server Backup (Local)
- Backup Schedule...
- Backup Once...
- Recover...
- Configure Performance Settings...
- Connect To Another Computer...
- Creating Recovery Disc
- View
- Help

Figure 3.3: Windows Server Backup in Windows Server 2008.

Aside from improvements in its user interface (UI) and the addition of native scheduling, Windows Server Backup isn't significantly different from its predecessor. It still works on a file-by-file basis (although it backs up entire volumes), still has the ability to back up the Windows System State, and so on. Some of the UI improvements are great—such as the ability to restore from a backup file that is located on the network—but in many ways, Windows Server Backup provides features that third parties were providing a decade earlier.

Note

Microsoft doesn't oversell Windows Server Backup; it specifically positions the tool as useful for "smaller organizations or individuals who are not IT professionals."

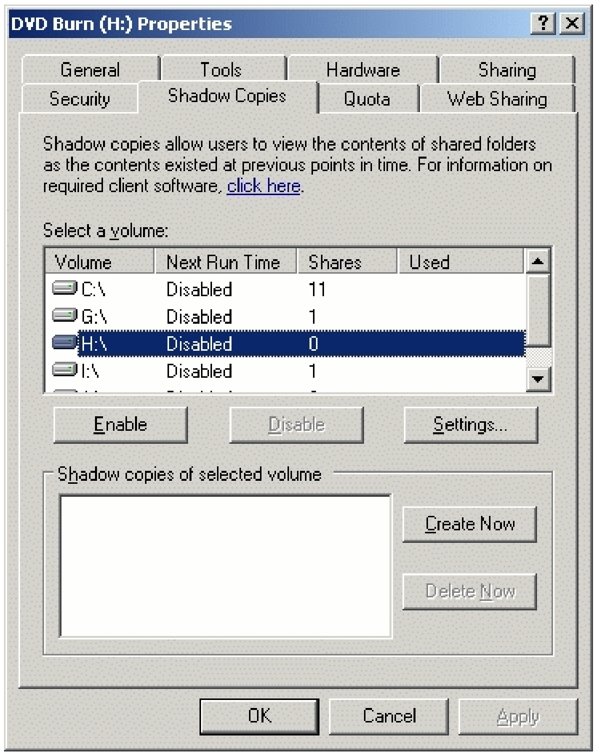
You can read more about Windows Server Backup at http://technet.microsoft.com/en-us/library/cc754572(WS.10).aspx.

Both of these native applications have a major shortcoming in that they are *snapshot* based, meaning they grab backups only during a designated backup window. Any files that changed after the backup was made, and before the next one was completed, would be lost in the event of a disaster or other problem.

Volume Shadow Copy / Previous Versions

Introduced in Windows Server 2003, VSC is a native, OS-level feature that creates *shadow copies* of files on any volume for which the feature is enabled. Administrators specify a maximum amount of storage to be used for the *shadow cache,* and the service automatically makes copies of shares files that are changed as well as files that are open when a backup is requested.

As Figure 3.4 shows, VSC is configured on a per-volume basis on the server. Once enabled, it will automatically begin creating shadow copies of files that are contained within shared folders—one reason the configuration UI displays the number of active shares. VSC will maintain several shadow copies for a given file, providing multiple old versions of a file. It will continue making shadow copies until its administrator-allotted space is full, and will then discard older copies to make way for new ones.



Shadow copies allow users to view the contents of shared folders as the contents existed at previous points in time. For information on required client software, click here.

Select a volume:

Volume	Next Run Time	Shares	Used
C:\	Disabled	11	
G:\	Disabled	1	
H:\	Disabled	0	
I:\	Disabled	1	

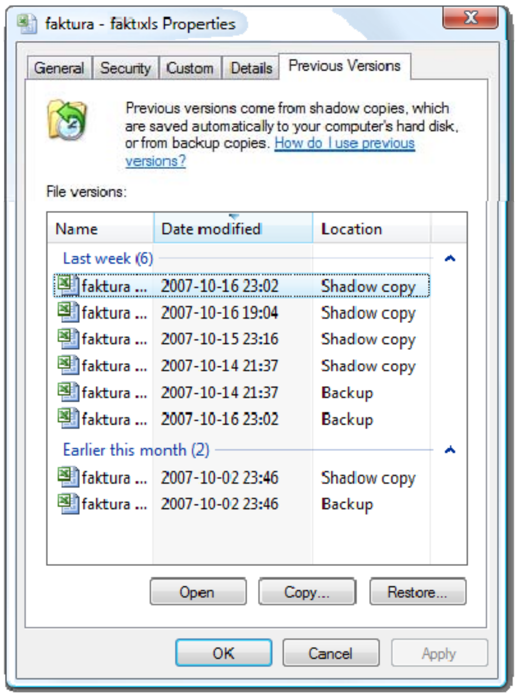
Figure 3.4: Configuring VSC.

Note

Windows Server Backup and later versions of Ntbackup include support for VSC, meaning they can natively make use of VSC stores to back up applications that expose their data through VSC. Such applications include Microsoft SQL Server.

VSC has two distinct functions:

- It will make shadow copies of open files automatically when those files are accessed for backup. This requires a backup application that is VSC-aware, meaning the backup application has to *ask* for a shadow copy. Some applications, such as Microsoft SQL Server, are designed to expose data through VSC, providing a common interface that backup applications can use to access application-specific data.
- It will make shadow copies of shared files, and make those shadow copies accessible to end users through the Previous Versions feature in Windows client OSs, such as Windows Vista. This provides end users with a self-service method for recovering individual files and folders that have previous versions available as shadow copies. Figure 3.5 shows an example.



Previous versions come from shadow copies, which are saved automatically to your computer's hard disk, or from backup copies. How do I use previous versions?

File versions:

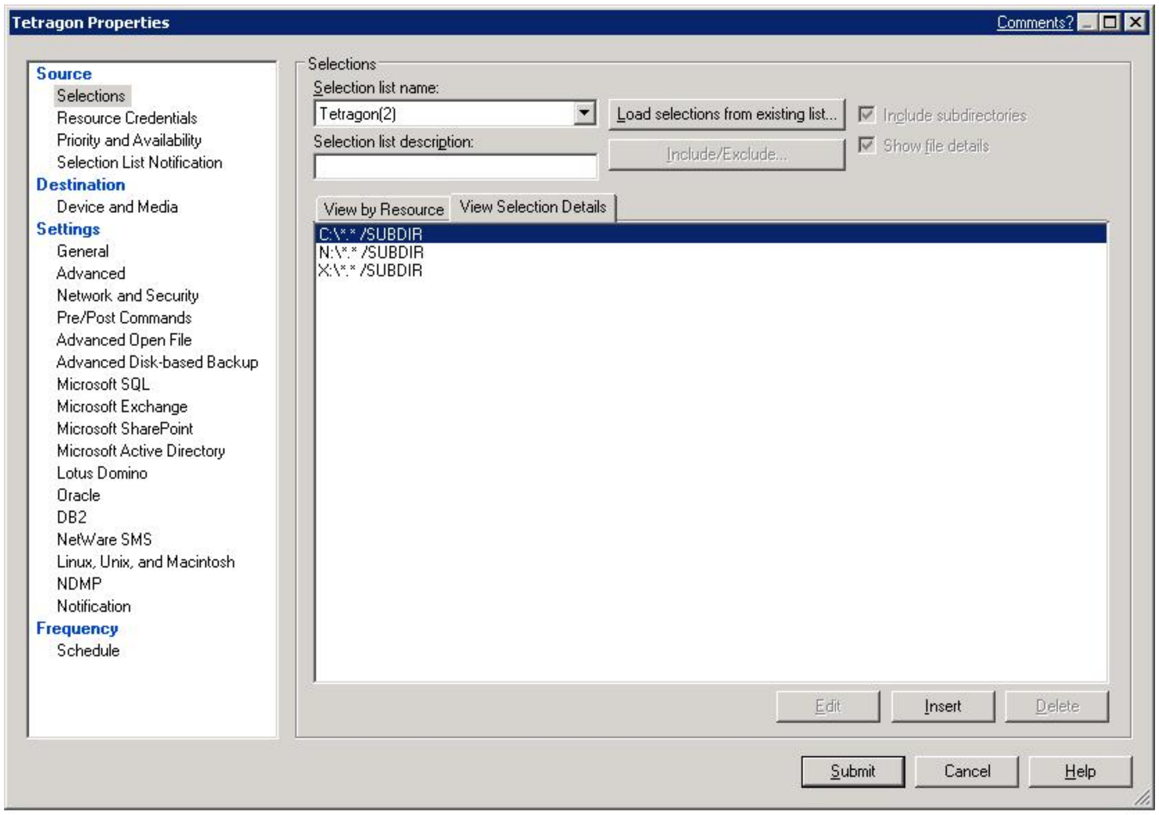
Name	Date modified	Location
Last week (6)		
faktura ...	2007-10-16 23:02	Shadow copy
faktura ...	2007-10-16 19:04	Shadow copy
faktura ...	2007-10-15 23:16	Shadow copy
faktura ...	2007-10-14 21:37	Shadow copy
faktura ...	2007-10-14 21:37	Backup
faktura ...	2007-10-16 23:02	Backup
Earlier this month (2)		
faktura ...	2007-10-02 23:46	Shadow copy
faktura ...	2007-10-02 23:46	Backup

Figure 3.5: The Previous Versions feature in Windows Vista.

VSC has a few shortcomings: First, it isn't exactly *continuous.* The feature doesn't make backups of a file each time the file is saved, although it will periodically make a new shadow copy of changed files. VSC doesn't provide fine-grained control, either. In other words, an administrator can't decide which files are more important and should be retained longer; VSC discards shadow copies in a first-in, first-out cycle; administrators can only determine the total amount of space available for all shadow copies. VSC isn't useful in full-server recovery, only in single-file backup and recovery. VSC doesn't create shadow copies of files that aren't shared, and it doesn't protect the entire OS—it ignores System State, for example. Generally speaking, most administrators regard VSC as a good *supplement* to a proper backup application; VSC can help reduce Help desk calls for single-file recovery by making that functionality more self-service, but VSC itself is not a backup solution.

Traditional Non-Native Solutions

Since the introduction of Windows server OSs, their fairly weak native backup and restore capabilities have spawned an enormous ecosystem of third-party solutions. For years, however, these solutions essentially replicated the basic features of the native backup application. To be sure, they added a great deal of administrative convenience and flexibility, but they did basically the same job. Figure 3.6 shows one such application.



Selection list name: Tetragon(2)

Selection list description:

☑ Include subdirectories

☑ Show file details

```
C:\*.* /SUBDIR
N:\*.* /SUBDIR
X:\*.* /SUBDIR
```

Figure 3.6: An example traditional third-party backup application.

With Windows Backup, administrators could select the files they wanted to back up; with third-party solutions—as shown—they did basically the same thing. The third-party solution could compress the backed-up data for transmission across the network to a central backup server with an attached tape library, of course, which is a major improvement. Many third-party solutions provide powerful scheduling capabilities, designed to maximize the amount of data that could be copied in a limited backup window. Most provided backup media management, and most provided application-specific agents to include applications such as Exchange Server, SQL Server, and so on in their backups. Most provided single-item recovery as well as whole-server recovery, and most provided the means to perform bare-metal recovery in the event of a complete disaster. Ultimately, though, their reliance on the same basic principles and techniques as Ntbackup always led to the same basic problems and challenges. It's what makes this entire category of backups feel like Backup 1.0.

Problems and Challenges

Both the native Windows backup capabilities and the similarly-designed traditional third-party solutions have the same challenges and problems:

- They're schedule-based, meaning they're snapshot-based. Without *continuous data protection,* you're always at risk for losing data. In most cases, backup windows are in the evening, so you're always at risk for losing an entire day's worth of work.
- Media management is a significant challenge, and many administrators using these solutions spend a few hours each week just shuffling backup tapes.
- Restoration is a time-consuming process, as file indexes and data must be loaded from tape. Even restoring a single file can take a long time: the right tape must be located and loaded and read, the file must be selected from a list, the tape must be wound to the correct location, and finally the file can be read.

What is it we want from our backups, again?

> Backups should prevent us from losing any data or losing any work, and ensure that we always have access to our data with as little downtime as possible.

These Backup 1.0 techniques miss out on preventing the loss of *any* data or *any* work; there's always data at risk because of these solutions' snapshot basis. They don't ensure we *always* have access to our data, and the downtime they offer certainly isn't minimal. Even solutions that eschew tape in favor of more expensive disk-based backup still have to perform an incredible amount of file operations to locate the right data and bring it back into production.

There's a deeper problem that these old-school techniques also tend to miss: data de-duplication. Today's enterprises have a *lot* of duplicated data—duplicated files in users' home folders, for example. Traditional backup solutions tend to blindly copy everything they see, meaning you're wasting not only space *storing* that duplicate data but also time and capacity backing it up. Data de-duplication is starting to become a watchword in storage management, and a Backup 2.0-style solution would certainly include de-duplication capabilities.

In the Old Days

I want to take a brief section to summarize some of the key Backup 1.0 principles and elements so that we can pull out their specific problems and think of ways to solve them or improve upon them.

Backup Techniques

The solutions I've discussed so far rely on a single primary technique with a few supporting ones. Mainly, these solutions are *snapshot-based,* meaning they seek to back up a file as it exists at the moment. They typically back up groups of files during a single backup window, which is often in the evening or during other periods of low or no utilization. Supporting techniques primarily revolve around backing up open files, and may include open file management utilities, LFB agents or features, or special features such as VSC.

Recognizing that some servers may contain too much data to be backed up in a single backup window, some organizations may choose to use a partitioning scheme. For example, half the server's data might be backed up in full one evening, while the other half receives an incremental or differential backup that evening. These management techniques help make the most of limited backup windows but also complicate restore procedures.

The main problem is that backup is snapshot-based, meaning there is always some data at risk of loss. A secondary problem is that of duplicated data, which wastes backup storage space.

Restore Scenarios

Restoration typically involves restoring a single file or other resource, often at the request of a user who accidentally changed or deleted a file. Self-service solutions such as Windows' Previous Versions feature (in conjunction with VSC) work well when the desired data is available for restoration; when such features aren't available or the needed data isn't available, administrators must turn to their backup solution. This typically involves identifying the correct version of the resource to be restored, locating the storage media containing that backup, and then reading the data from that media—which, in the case of tape, can require some time, as the tape doesn't support random access and must be wound to the correct read location.

The main problem is in identifying the needed data and locating the media on which it is stored. With complicated backup schemes, this may involve identifying a full backup, one or more incremental backups, and/or a differential backup. Another problem is in the time it takes to perform the restore, particularly with tape-based backups. A final problem relates to the snapshot-based nature of the backup, meaning there will be times when the desired data simply isn't available on a backup.

Disaster Recovery

Whole-server disaster recovery usually comes in one of two forms:

- The base OS must be installed first, possibly along with any recent service packs, using standard installation procedures. Then, one or more software applications must be installed to support the recovery. Finally, the recovery can begin in earnest, reading data from the backup media and restoring it to the server.
- A bare-metal recovery usually involves a special boot disc that includes a stripped-down OS, such as WinPE or Windows Recovery Environment, that contains enough smarts to access backed-up data, format the server's disks, and copy the backup data to those disks. This form usually involves fewer steps and is much faster than the previous form.

Both techniques can be time-consuming and suffer from the inherent snapshot-based nature of the backup, meaning there will always be some data missing even in a perfectly-executed recovery.

Backup Management

Backup management can be complicated using these Backup 1.0-style techniques. Backup schedules and types (full, incremental, or differential) can be complex, and require careful management not only of schedules but also of storage resources, network capacity, and so forth. Physically managing tapes—rotating them off-site, verifying their accuracy, and so forth—can be time-consuming and error-prone.

These days, backup management is complicated by many companies' data retention requirements. Do the backups include regulated information? In that case, they may need to be retained for a specific period of time or discarded after a specific period of time. They may need special security precautions, as well, to protect the data contained in the backup.

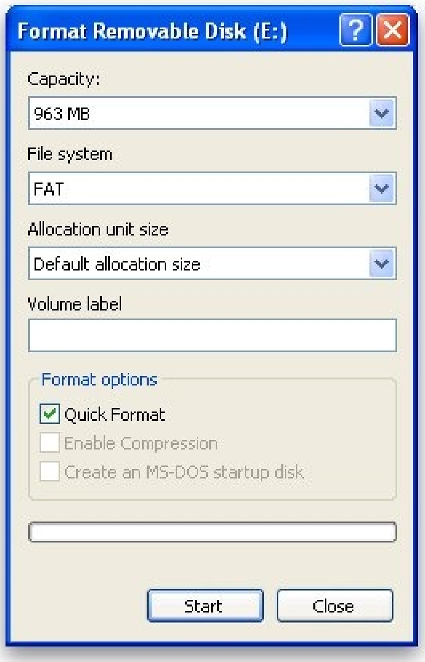
Rethinking Server Backups: A Wish List

Let's start thinking Backup 2.0: What are our old whole-server backup techniques missing that we'd like to add? The sky's the limit; at this point, we don't need to worry about what's possible or available—just what, in a perfect world, we'd be able to improve.

New and Better Techniques

I think the most important improvement that Backup 2.0 can offer is a change from point-in-time snapshot backups to *continuous* data protection. Here's how it might work: Everything starts when an application—of any kind—makes a change to disk. Applications do so by passing data to the Windows OS's file system application programming interfaces (APIs), basically asking Windows to "save this data to this file." When that happens, Windows' file system takes the data and begins breaking it into blocks.

Blocks are the basic unit of management for data on a disk. When you format a disk, you select the *allocation unit*—or block—size. That determines how much data is contained within a given block of disk space. Figure 3.7 shows that Windows usually picks its own default value, which is what most people go with.


Format Removable Disk (E:)

Capacity: 963 MB

File system: FAT

Allocation unit size: Default allocation size

Volume label:

Format options

[✓] Quick Format

[ ] Enable Compression

[ ] Create an MS-DOS startup disk

Figure 3.7: Formatting a disk and selecting block size.

A single block can hold the contents *only* for a *single* file. If your block size is 2 kilobytes, for example, and you save a 512-byte file, then three-quarters of the block will be empty and wasted. Larger files are split across multiple blocks, and NTFS keeps track of which blocks go where: "file XYZ.txt consists of block 324, then block 325, then block 789," and so forth.

Microsoft recognizes that third-party utilities might need to be notified of file operations, and might in fact need to modify or cancel those operations. That's how most third-party disk quota systems and file-filtering solutions work: they get the file system to tell them when files are changed on disk, and they update their quota database or block files from being saved, or whatever. The technique Microsoft provides for accomplishing this is called a file system *filter* or *shim.* Essentially, the shim registers itself with the OS, is notified of file operations, and is given the chance to block or allow each operation.

In the case of a Backup 2.0-style solution, the shim might just pay attention to which disk blocks were being modified. As blocks are modified on disk, the shim could copy the blocks' data, compress it, and transmit it across the network to a backup server. Figure 3.8 illustrates the process I'm proposing:

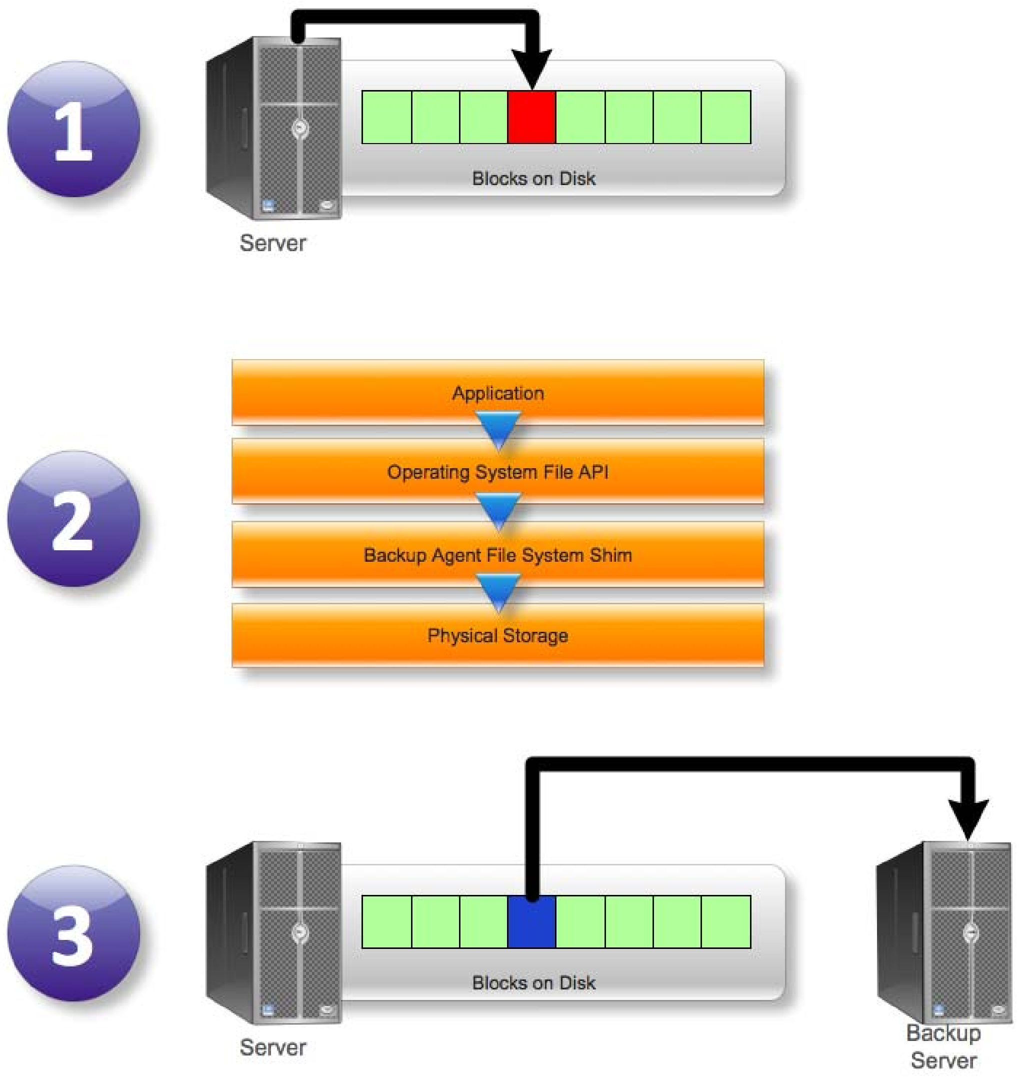


Figure 3.8: Capturing changed disk blocks.

In Step 1, something on the server modifies a block of disk space. In Step 2, this change is passed to the backup software's file system shim, and the modified block is copied and transmitted to a backup server.

This technique would allow for *continuous* data protection. Of course, in addition to just copying blocks, the software would also make a note of which file the block went with. On the backup server, software would keep track of files and their associated blocks. This is a powerful technique: Rather than messing around with cumbersome and complicated open file techniques, as Backup 1.0 solutions would do, this system is grabbing the low-level data changes as they are physically inscribed on the hard disk by the OS. The data is being grabbed *below* the level of an entire file, so even partial changes to a file—such as a Microsoft Access database—can be grabbed immediately, almost in real-time, and the backup solution doesn't care if the file happens to be open or not at the time. When someone needs to restore a file, the backup server's software simply looks up the most recently-saved blocks for the file, and uses them to reconstruct the file in its most recent condition. Best yet, the central backup server could *also* save *past* copies of a file's blocks—meaning it could reconstruct the file as it existed in any particular point in time.

If the saved blocks were stored on a high-speed storage system, such as a RAID array, files could be reconstructed almost instantly and saved to any location on the network. Considering our prime directive for backups:

> Backups should prevent us from losing any data or losing any work, and ensure that we always have access to our data with as little downtime as possible.

This Backup 2.0 technique would do the trick. We really wouldn't lose any data or work, because low-level changes would be captured continuously. We wouldn't need to worry about backup windows because the entire day would be our backup window. We'd always be able to re-construct data as needed, with minimal downtime.

> **Note**
> In fact, solutions exist that do just this: Although they may use different under-the-hood techniques than the file system shim I've described, the Backup 2.0 idea of copying blocks nearly instantly is very much a real thing, not just a wish.

This technique also has the advantage of being able to capture *everything* that goes to disk, including file permissions, alternate NTFS streams, registry changes, AD changes, and more, all without necessarily needing any special knowledge of file types or structures.

In fact, this technique is logically (although not physically) similar to RAID 1 disk mirroring, which also operates at a block level—albeit normally via a RAID controller card and not via software, although software RAID 1 is also possible. Rather than mirroring blocks in real time to a separate disk, this backup technique "mirrors" the blocks across the network to a backup server. And, rather than only keeping the *current* block data, as in a disk mirror, the backup solution can keep *past* copies, too, enabling recovery to any earlier point in time.

Better Restore Scenarios

The trick to all of this, of course, is having great management software on that central backup server. It needs to be able to track which changed blocks go with which files, for example, and for convenience may want to keep track of special files like the Windows registry or AD databases.

The backup server software could easily expose a self-service UI, if desired, to provide functionality similar to—but more controllable than—Windows' Previous Version / VSC feature. The software could more easily recover data in applications such as AD because the software could re-construct a portion of the database file or even the entire database file, down to a specific point in time (in all probability, you would want the software to be able to attach "points in time" to specific AD operations, such as the deletion of a user or the creation of a group or whatever—just so you could more easily identify the point in time you want to roll back to).

Better Disaster Recovery

With a copy of *every block of disk space,* restoring an entire server would be straightforward: You'd need some sort of recovery disk, such as a bootable DVD, to get the server up and running and to talk to the backup server software. You'd then simply stream the most recent version of every disk block back to the server, write those blocks to disk, and then restart the server normally. With good compression to speed up the transfer of data across the network, bare-metal recovery could be done quickly—and you'd lose very little, if any, data.

In fact, the opportunity exists to recover the server to a *virtual* server, if need be—an excellent disaster recovery scenario as well as a powerful physical-to-virtual migration technique. You might well be able to recover to dissimilar hardware, too, since in many cases Windows can re-adjust itself when it finds itself suddenly running on different hardware.

Tip

As you begin looking at block-based recovery solutions, ask for details on how they deal with dissimilar bare-metal recovery scenarios. Would Windows require re-activation when it finds itself running on different hardware? *How* dissimilar can the hardware be? The more flexibility the solution offers, the greater the number of scenarios when it will be able to save the day when it's needed.

Here's a killer scenario: Imagine grabbing real-time disk blocks from a production server, and then immediately applying ("restoring") those blocks to a virtual machine. Instant virtual standby! If the main production server fails, the virtual server can step in and take over with little downtime and little or no data loss. Depending on how you architect the virtual infrastructure, a single virtual host might support several virtual standbys. Those standbys might operate with less performance than the non-virtual production machines (again, depending on how you set things up), but you'd have a "hot spare" any time you needed on. Figure 3.9 illustrates this idea.

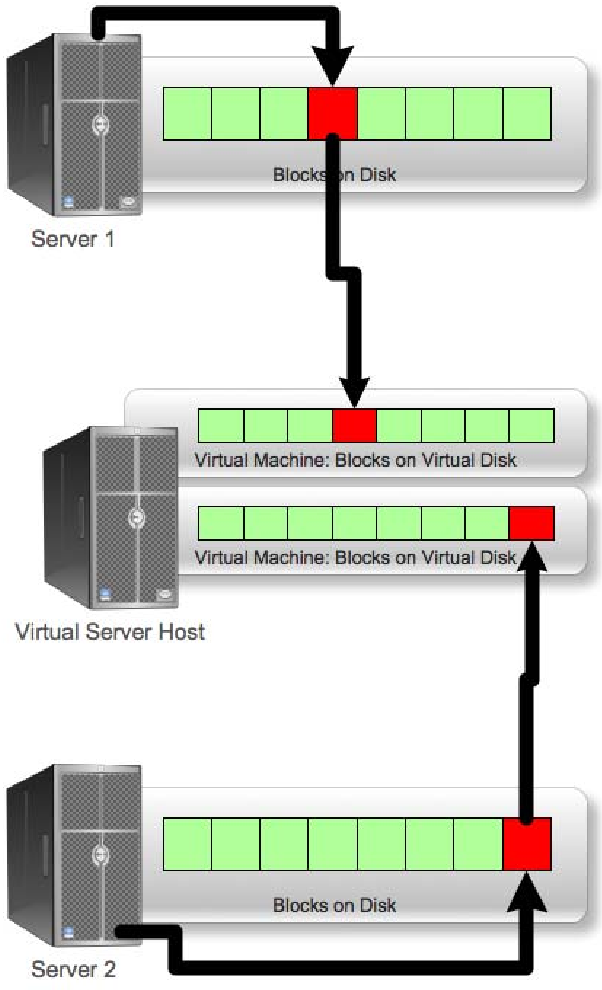

Blocks on Disk
Server 1
Virtual Machine: Blocks on Virtual Disk
Virtual Machine: Blocks on Virtual Disk
Virtual Server Host
Blocks on Disk
Server 2

Figure 3.9: Block-based virtual hot spares.

Easier Management

Managing block-level backups can be much easier because they'll typically be stored first on disk. You can then make tape-based backups from the disk-based backups—meaning your production servers don't participate in the tape-based backup, and your "backup window" can be as long as you like. Tape backups would truly represent time spans, and could be complete and internally consistent—unlike today's mix of full, incremental, and differential backups, which must be treated as a set in order to retain their usefulness.

Block-based backups would also be an effective way to implement data de-duplication, as sets of blocks could easily be indexed and compared to check for duplication. That would help cut down both on disk storage and tape storage as well as all the management overhead that comes with any kind of storage resource.

Note
Some data de-duplication vendors claim that you can reduce the size of your backups by up to 70%. Even if that's an optimistic number, you could conservatively expect to save a significant amount of space!

The backup software could also, in theory, allow you to mount backed-up data as a real disk volume. It would simply need to provide some kind of disk driver for Windows that could talk to the backup software and reassemble, in real time, the most recent backed-up blocks into a single disk volume. You could then mount and explore backed-up data as easily as live data, allowing you to compare files, drag and drop files to different locations, and so forth. If the backup solution was storing *past* versions of block data, you could mount a disk that resembled *any given point in time*, making it easy to compare files or data from different points in time.

Great for…

Again, most of the capabilities I've wished for *are available today* from a variety of third-party vendors. Backup 2.0 for entire servers isn't something you have to wish for; it's something you can choose to do now, helping you achieve the capabilities that backups have *always* alleged to provide:

> Backups should prevent us from losing any data or losing any work, and ensure that we always have access to our data with as little downtime as possible.

The following sections highlight a few specific scenarios where these Backup 2.0 techniques could save the day.

Domain Controllers

Today, companies spend thousands on AD backup solutions that create point-in-time backups of AD and allow for single-object recovery as well as more complete whole-directory disaster recovery. A Backup 2.0-style solution, however, would be able to natively back up AD, because in the end, AD's complex database is just blocks on disk. Certainly, a backup solution would need to offer AD-specific functionality for restores, or might even offer an AD management console extension that made AD recovery capabilities available right from that console. But when you start using block-level backups, AD suddenly isn't so difficult to back up. You can get real-time backups of *every change*, automatically, with no extra effort.

Infrastructure Servers

Infrastructure servers are perfect for Backup 2.0: You'll capture every DNS change, every DHCP change, and more—on servers that you might only back up once a week in the Backup 1.0 world. Why bother with real-time backups on an infrastructure server?

- If you have to do a bare-metal recovery, things like DHCP leases will still be intact—your network won't have to go through a period of recovery and readjustment.
- Both static and dynamic DNS records will be maintained—again eliminating the period of confusion that normally occurs when a DNS server crashes and is brought back online.
- Still using WINS? Even that database can be backed up in real time and recovered to a specific point in time—helping eliminate the need for your network to start massive broadcasts and re-registrations, as would happen when an empty or out-of-date WINS database was recovered.
- Infrastructure servers can be easily restored to a virtual machine in the event of a complete disaster—even an off-site virtual machine, making it easier to re-create your exact production infrastructure, virtually, in a disaster recovery mode.

Web Servers

Web servers are just files and folders, right? Why not capture *every* change, and make it easier to roll back an entire Web site to a known-good point in time, including Web server configuration files and metadata? It doesn't matter if you're using IIS or Apache or something else as your Web server, it's all blocks on disk, and a Backup 2.0 solution can grab it all.

Backup 2.0 can even help make Web farm management easier. Designate one Web server as your "master"—perhaps it's even a protected server that doesn't accept live traffic. Back that up in real time using a Backup 2.0-style solution, capturing newly-uploaded files from your Web developers as well as configuration changes from Webmasters. Restore those changes to multiple "hot spares" in your Web farm—with no effort on your part, every member of your Web farm now has identical Web content and server configuration files! Your master content would remain protected, so even if one of your Web servers was compromised, you can easily—and quickly—restore it to the most recent, correct version, bringing it back into production quickly and effortlessly.

Want to rebuild your Web farm as virtualized servers? No problem: Just use your Backup 2.0 solution to "restore" your master Web server to one or more virtual machines. Reconfigure a few settings such as computer names, reconfigure your load balancer to point to the new servers, and your Web farm is moved. With the right management tools on top of the basic disk block-based backup system, it's all possible—and it brings value to your backup solution that extends beyond mere backup and recovery.

Public Key Infrastructure

If you operate a Public Key Infrastructure (PKI), you know how critical those Certification Authority (CA) servers are. With disk block-based backups, you'll always have a reliable backup of every CA—including every certificate, every public key, every revocation, and every outstanding certificate request. In the worst-case scenario of a complete CA failure, you can still bring the PKI back up quickly, either by restoring the most recent disk blocks to the original hardware, to dissimilar hardware, or even to a virtual server.

Coming Up Next…

In the next chapter, I'll be taking the same approach as this one but focusing exclusively on Microsoft Exchange Server. There's no question that Exchange forms a big part of most organizations' "mission critical" list, and having solid, reliable Exchange backups is equally critical. But Exchange certainly makes backup and recovery a bit more challenging, as you need to support a high level of granularity from single-message recovery to bare-metal server restores. You've got to do all that despite the fact that Exchange's database is essentially a big black box, not a bunch of more easily-manipulated little files. It's a real test of the Backup 2.0 methodology.