Realtime publishers

The Definitive Guide™ To

Windows Application and Server Backup 2.0

sponsored by

Don Jones

Chapter 8: Other Concerns and Capabilities....131
Disaster Recovery and Bare-Metal Recovery....131
Recovery Times....134
Recovery Locations....135
Why Wait for a Disaster?....136
High Availability, Instant Recovery, and Replication....136
Hardware....137
Clustering....137
Replication for Redundancy....138
Replication for Offsite Recovery....139
Data Retention—Here Come the Lawyers....141
Compliance and Security—Here Come the Auditors....142
Building Compliance Islands....142
Integrating Backup 2.0 with Backup 1.0....145
Backups as a Form of Migration....146
Coming Up Next...147

Copyright Statement

Chapter 8: Other Concerns and Capabilities

There's more to a backup strategy than just grabbing the right files and making sure you can restore them in a pinch—although that's obviously a big part of it. A solid backup strategy also concerns itself with disaster recovery in a variety of scenarios. You need to make sure your backup system itself has some redundancy—nothing's worse than being without a backup system! Because backups inherently involve data retention, in this day and age, you also have to concern yourself with the safety and security of that data as well as any legal concerns about its retention. That's what this chapter is all about: Dealing with the "extras" that surround a backup strategy. I'll look at how traditional Backup 1.0 techniques addressed these extras, and suggest ways in which we might rethink them for a Backup 2.0 world.

Disaster Recovery and Bare-Metal Recovery

I've already written quite a bit about disaster recovery, or bare-metal recovery, which is what you do when an entire server dies and you need to restore it. In the bad, bad, bad old days, disaster recovery always started with re-installing the server's operating system (OS) from scratch, then installing some kind of backup solution, then restoring everything else from tape—a time-consuming process because spinning data off of tape isn't exactly the fastest activity in the world.

Even the Backup 1.0 mentality got sick of that process, though. Today, *most* third-party backup solutions provide some kind of "restore CD," which can be used to boot a failed server. The stripped-down OS on the CD, often based on DOS, WinPE, or a proprietary OS, is smart enough to find the backup server and receive data being streamed across the Internet; depending on the solution, it might also be smart enough to read data directly from an attached tape drive. Figure 8.1 shows an example of one of these recovery disks in action.

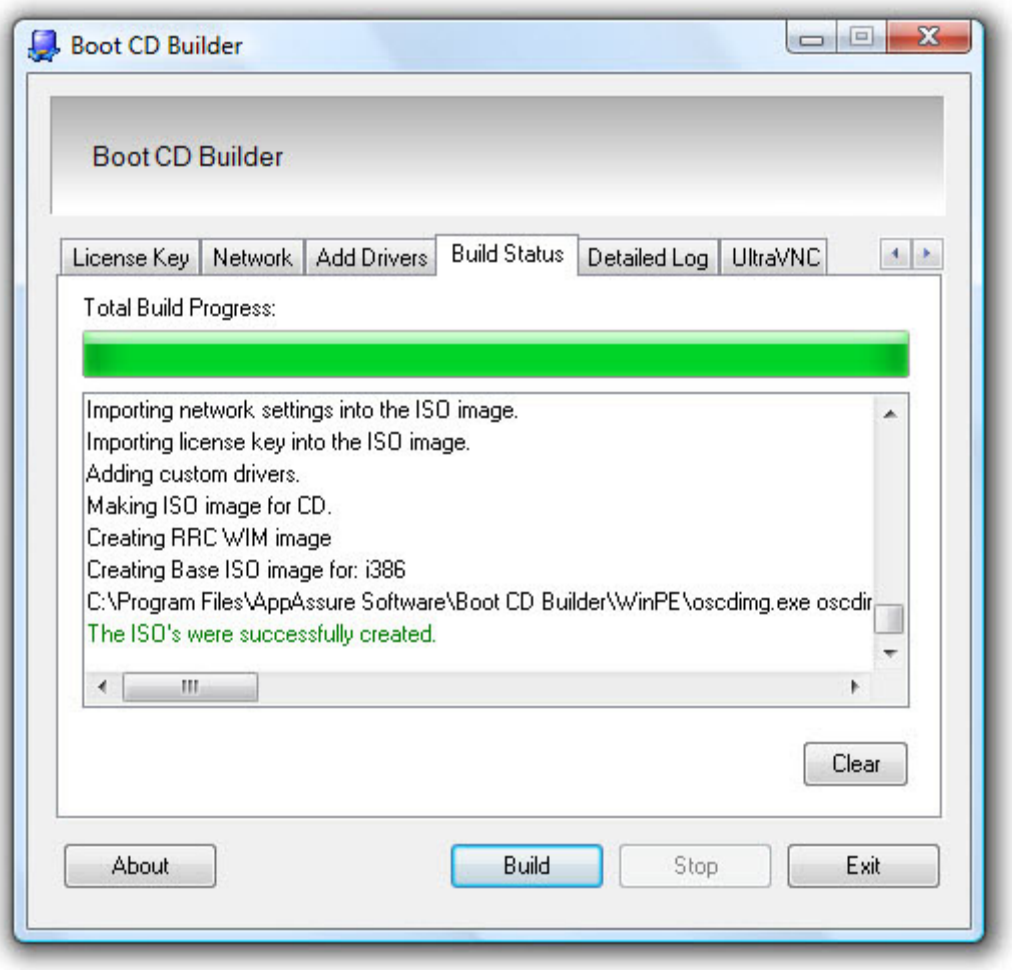

Figure 8.1: Recovering a server by using a recovery disk.

Note

Some vendors might make a bootable USB flash drive or some other form of removable media bootable. That's fine, too, and actually can be faster than a disk, provided your servers can boot from the selected type of media.

Not every backup solution relies on a disk, though—after all, some folks do configure their servers without optical drives. Other backup solutions can instead install their bare-metal recovery software to a separate partition of the server's boot disk, making the recovery software available as a boot option. I don't favor this approach because if you lose a whole server, it's entirely possible that you lost the disks, too, in which case your recovery software is gone as well. Stick with solutions that provide a bootable CD or DVD, and simply make sure your servers have an optical drive installed. Even today's low-profile, 2U rackmount servers can almost always be configured with a slimline optical drive—usually the same slimline drives that the manufacturer uses in their laptop computers.

A Third Option: PXE

A third option is to have servers whose BIOS and network cards support network boot, using Intel's PXE protocol. In this scenario, your backup server software would need to support network boot, meaning that the server would be able to "find" it on the network, request a boot OS image, and download that image from the server. That image is usually the same thing that would be on a recovery disk.

Microsoft makes great use of PXE for their automated deployment tools; even the older Remote Installation Services (RIS) supported network-boot scenarios. It's less common to see PXE used in conjunction with a backup and restore solution, though, simply because you're—hopefully—not using the feature *that* often, so using a recovery disk is simpler, requires less setup and infrastructure, and is easier to use under a wider variety of situations.

Backup 2.0 isn't much different in terms of process: You'll usually use a boot disk of some kind, and that disk is smart enough to look for the backup server on the network. You may be able to browse for a specific server image to restore, then the restoration begins. Other times, you might boot using the disk, then go to the backup server's console to select an image to "push" to the server you're recovering.

The difference with Backup 2.0 is *what* gets restored back to the server.

Because a Backup 2.0 solution was streaming disk block changes in near-real time up until the minute the server failed, you're going to lose a *lot* less data than in a Backup 1.0-style scenario where your most recent full server backup may have been from a week ago—or longer. So Backup 2.0 gets you back online with less data loss—if there's even any loss at all.

Also, remember that Backup 2.0 relies *primarily* on a disk-based backup store. That offers a couple of advantages: First, you won't be sitting around waiting for data to stream off tape, which even with today's improved magnetic tape and drives, is still time-consuming. A Backup 2.0 solution can start restoring your server *immediately.*

Finally, Backup 2.0 doesn't mess around with full, incremental, and differential backups. Or technically, I guess, it *does*: It grabs a full backup of your server in the beginning, then continually grabs changed disk blocks, which is technically like an incremental backup. But you don't have to keep track of any of that, or find the right tapes, or remember the correct order to restore those tapes. The Backup 2.0 solution simply asks you what point in time you want to restore the server to, and it gets the correct disk blocks automatically.

Recovery Times

Now, you might think that streaming an entire server's worth of disk blocks across the network would be pretty time consuming in and of itself. I mean, just trying to copy a 2GB file across the network can take time, and servers might have hundreds of gigabytes worth of disk blocks. That's where you want to dig into some of the details of a Backup 2.0-style solution before actually buying one.

Later in this guide, I'll be discussing compression and de-duplication, two techniques that help a backup solution store server data using less disk data than that data originally took on the server it came from. "Less space on disk" also means "less time on the network," so exactly *how* a backup solution implements its recovery can matter a lot. Figure 8.2 shows what you *don't* want: A solution that expands compressed and de-duplicated disk blocks on the backup server, streaming each disk block to the recovering server exactly as they'll be written to disk. In other words, a server with 200GB of storage will have 200GB of data transmitted across the network; even with a dedicated 1000Mbps network connection, you're probably looking at an hour or so.

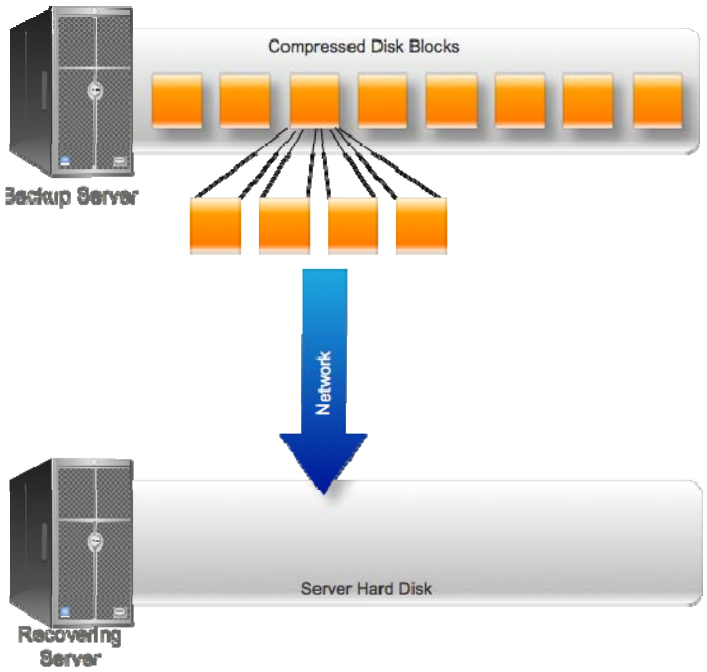

Figure 8.2: Uncompressing and re-duplicating data at the source.

Figure 8.3 shows a slightly smarter solution: Leave everything compressed and de-duplicated until it *gets to* the recovery server. There, the backup solution's recovery disk software can de-compress and re-duplicate the data. Less data is transmitted across the network this way.

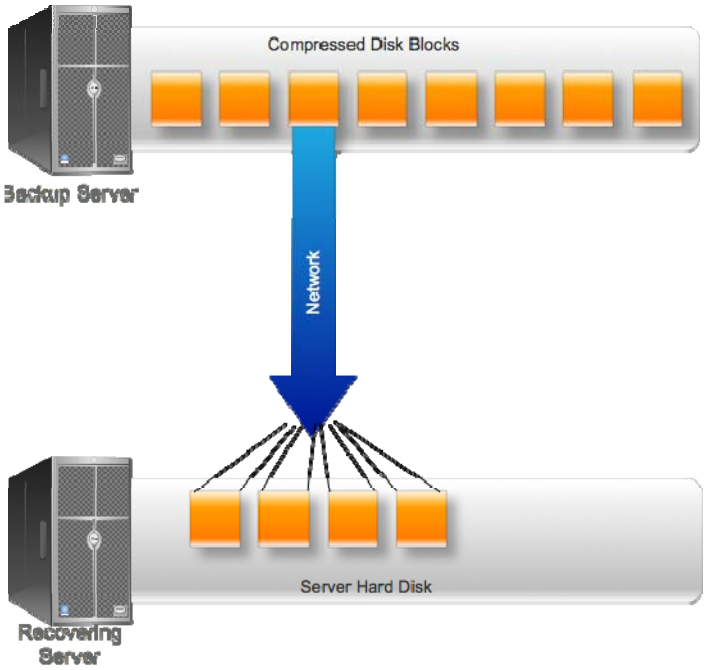

Figure 8.3: Uncompressing and re-duplicating data at the destination.

These sorts of fine details can help speed the recovery process tremendously. As you're evaluating solutions, ask vendors for benchmarks for bare-metal recovery times. They might be able to tell you, for example, how many minutes it takes to recovery *x* gigabytes of data across a network connection of *y* megabits per second. If they can't, it's certainly a benchmark you can figure out on your own in a lab using trial software.

Recovery Locations

A key capability for any bare-metal recovery solution is the ability to restore a server's image to a different server—either a different physical box or to a virtual machine. *Both* capabilities—physical server and virtual machine—are needed. In some cases, a server might fail entirely, leaving you with no choice but to get a new physical machine to replace it. In other cases, you might only need a few days to repair a physical server, meaning restoring to a virtual machine will provide a less-expensive interim option. Being able to restore to a virtual machine also provides vastly more flexibility for disasters that affect your entire data center: Rather than needing an off-site recovery location that has dozens of servers ready to go, you can just dump all your servers into virtual machines.

Expect *some* limitations around these capabilities. For example, you obviously shouldn't expect to restore a 64-bit OS onto older, 32-bit hardware. Server hardware with vastly different configurations may present problems, too, although the Windows OS has become more adept at dealing with those types of changes.

> **Tip**
> Have your Windows serial numbers or license keys handy, and if you're using internal Volume Activation licenses, make sure that an activation server is part of your recovery plan. When Windows detects massive hardware changes, it may demand immediate re-activation.

The feature to look for as you're evaluating solutions is "dissimilar hardware," which often means the vendor has provided technologies for making the recovery process onto different hardware—or onto a virtual machine—a bit smoother.

Why Wait for a Disaster?

Here's an idea worthy of our Backup 2.0 "let's rethink everything" motto: Why wait for a disaster to start your disaster recovery? Why not just have a spare server ready to run—for *every* one of your servers?

Hear me out. In the recent, bad old days, you simply couldn't afford to keep a spare server in the closet for every production server you owned. Not practical. However, with the advent of high-performance, reliable virtual machines, you *can* keep a spare. Here's what a really slick Backup 2.0 solution could offer: Once a night, say, the solution could construct a ready-to-run virtual machine from your current backups. You'd just have some big hard disk somewhere, filled with virtual machine files that were constantly being re-created or updated. When disaster struck, *you wouldn't need to recover anything*—you'd just boot one of the virtual machines. We're talking a "recovery time" of *a few minutes*!

Depending on how you've built out your virtualization infrastructure, the virtual servers might not perform as well as the physical ones. But "slower" is far better than "completely gone," right? And with today's live migration capabilities—offered in most major hypervisors, provided you've bought the right management tools to enable the feature—you can always shift virtual machines to different virtualization hosts, balancing the workload to get the best performance you can from the most critical resources. And those virtual servers might well just be an interim strategy while you repair a physical machine or get a new physical machine in place.

High Availability, Instant Recovery, and Replication

A server dies. A user is missing a file. Someone's email is corrupted. You confidently turn to your Backup 2.0 solution, having read this book and fully embraced the Backup 2.0 lifestyle.

And your backup server is dead.

Whoops.

High availability is an appropriate feature for *any* mission-critical function within your business, and your backup solution should certainly qualify as mission critical. There are a few different ways in which a backup solution can be made more redundant and more resistant to single points of failure.

Hardware

Obviously, running your backup solution on redundant hardware can help. Redundant power supplies, redundant drive controllers, redundant network cards, RAID arrays for disks, redundant fans, and so on—those features are all widely available from most server manufacturers, and are usually worth the extra price.

Don't forget about the infrastructure that your backup solution relies on, though. Having redundant network cards is nice, but it's less nice if they both plug into the same switch, and that switch fails. Some redundancy in your infrastructure can help, too. Of course, you have to decide where the tradeoff is for your company: Redundancy costs money, and at some point you'll have to decide how much redundancy—and expense—is practical.

Clustering

Redundant software simply means that your backup solution is running on more than one server, either in some kind of cluster configuration or using replication. A cluster, for me, isn't the most efficient use of resources. Here's why: Look at the cluster arrangement in Figure 8.4.

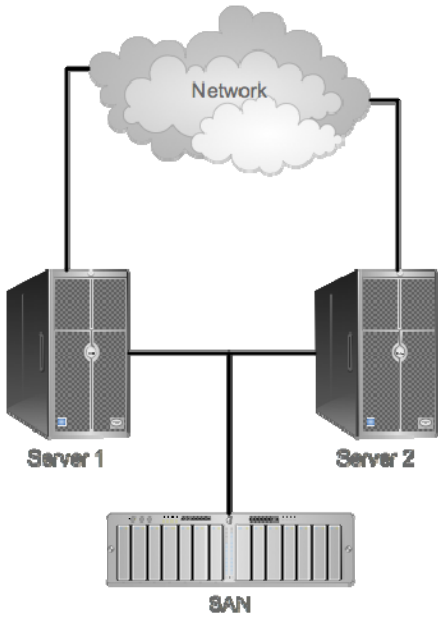

Figure 8.4: Typical two-node cluster.

In this type of cluster, you have two complete physical servers (they could be virtual, too, I suppose), connected to some kind of shared storage—like a Storage Area Network (SAN). Your backup software would run on each server, although only one server at a time would be active. In the event of a failure, the other server would take over, and would have access to all the same storage, so it could pick up where the other one left off.

For a backup solution, this is kind of inefficient—and you still have your storage network as a single point of failure (although those are often redundant in and of themselves, using RAID arrays and the like). For a backup solution, *replication* often offers a better form of high availability—and brings with it additional flexibility for solving other business problems.

Replication for Redundancy

With replication, your backup software is continuously streaming copies of its backed-up data to another copy of the same software. That means your backup data lives in two places, so if one of those places gets attacked by dinosaurs or something, the other place is still able to provide restore and recovery services. This can be a great way to get data off-site entirely, especially for smaller branch offices that might not be able to mess around with tape backups. Figure 8.5 shows what it might look like.

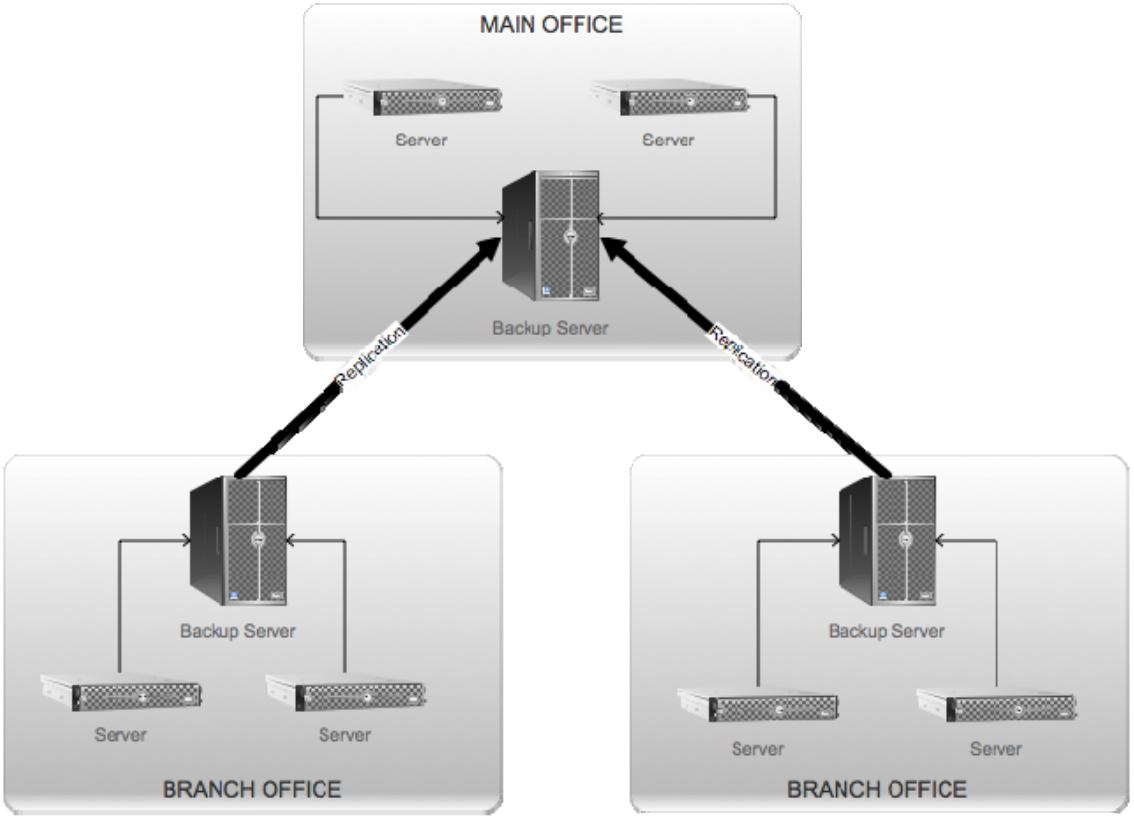

Figure 8.5: Replicating backup data.

In this example, each branch office has its own backup server, which backs up its local servers. The backup data is replicated to the main office, whose backup server also backs up the production servers in that office. The main office would likely have access to tape backup drives and an IT staff, so the main office's data is redundant through tape backups. The branch offices wouldn't need tape backups or IT staffers because their data is being made redundant through replication. In a worst-case scenario, a branch office server could be restored from the main office's backup server. Might not be ideal, and wouldn't likely be as fast, but *slow* is better than *out of business.*

The downside of this approach is that you might well be replicating data—across WAN links, mind you—that you don't care *that much* about. In other words, maybe a branch office has two servers, and you're backing up them both, but you're only really super-worried about one of them. So I'll suggest a modification: Your backup solution should let you *specify* which servers' backup data will be replicated upstream.

Replication for Offsite Recovery

There's another big, big, big advantage to this kind of replication. Take a look at Figure 8.6.

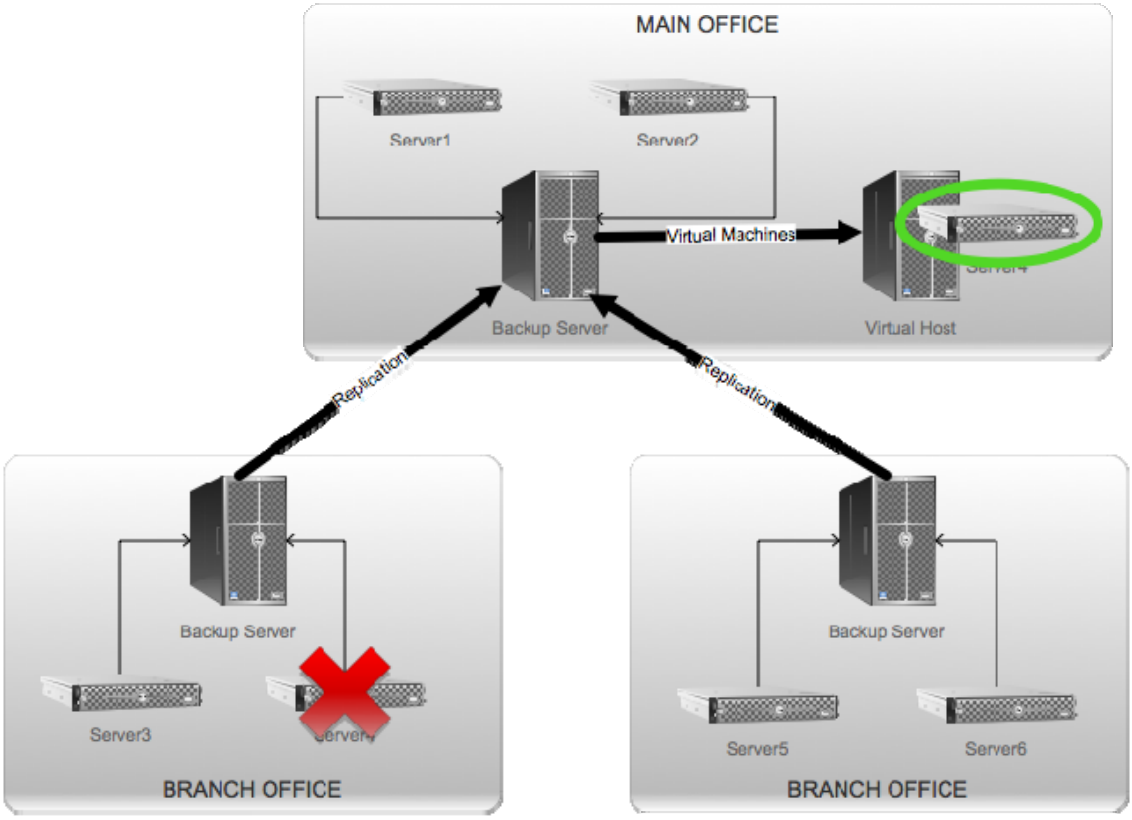

Figure 8.6: Replication + virtualization.

Server4, one of the servers in the first branch office, has died. Rest in peace. Fortunately, Server4 was being backed up, and its backup data was being replicated to the main office. And, by strange coincidence, the main office has a virtualization host. Each night, the main office backup server creates a fresh set of virtual machines on that virtualization host. So when Server4 dies, someone at the main office pushes a button, and Virtual Server4 starts up. Total down time? However long it takes to make a phone call and push a button.

Virtual Server4 is still being backed up, so when the real Server4 is fixed, someone pushes another button and the *current* Server4 image is restored to the Server4 hardware. Virtual Server4 is shut down, and Server4 resumes its existence. That "fail back" can be done during a maintenance window, of course, to avoid further inconveniencing the branch office users.

In this instance, having the backup data replicated wasn't covering for a failure in the backup system, it was offering a faster and more convenient recovery scenario for a failed server. That's what's called *instant recovery,* and it's a very useful capability.

Changing the Nature of Off-Site Recovery

I can see this type of technology really changing the way companies think about off-site recovery. Today, many companies invest a lot of money in disaster recovery facilities. Typically, companies that offer off-site recovery also offer off-site tape storage, and they keep a varied inventory of hardware on hand. If your data center is toast, you pack up your staff and head to the off-site facility. You get your latest tapes and sit back for a nice long day (and night) of restores, getting your servers up and running on the loaner hardware. Typical Backup 1.0 mentality, right?

Imagine instead that you're paying an off-site facility simply to host a copy of your backup solution, with enough disk space to replicate your backup data. All you need them to provide, in the event of a disaster, is a bunch of Hyper-V servers. When something bad happens, you might not even have to pack up the whole team—just get onto a Web interface, push a few buttons, and your pre-created virtual machines (which your backup solution has dutifully been making from your replicated backup data) move over to a Hyper-V host and start up. You establish Virtual Private Network (VPN) connections so that your employees can get to the recovered servers, mangle some IP addresses and DNS entries perhaps, and you're back online. Heck, I imagine someday there will be a backup solution that even takes care of little details like DNS entries.

Off-site disaster recovery facilities suddenly become a *lot* less expensive, because all they need is some good WAN bandwidth and a bunch of Hyper-V servers in a data center. They can get by with a lot less *variety* of hardware, and they can enable faster off-site recovery without actually requiring many (or maybe even any) of your team members to actually *go* off-site. Call it "cloud recovery," perhaps. It's not here yet, although you can certainly build your own private version of it; it'll happen someday, though.

Data Retention—Here Come the Lawyers

Backups contain data. Kind of the whole point, really. But a lot of companies these days are under some pretty strict regulations on how long they *must* keep certain types of data, and how long they *may* keep certain types of data. Working those data retention policies in with your backup solution can be awfully tricky. In a tape-based, Backup 1.0 world, it usually means cycling tapes off-site, and keeping them for however long you're required to keep the data. In a Backup 2.0 world, you can get just a bit more flexibility.

Remember that each disk block backed up in a Backup 2.0-style solution is time-stamped, and that the backup data lives primarily on disk, not tape. It's therefore very easy to specify minimum or maximum retention times for different types of data—the solution simply makes sure that each disk block stays around for that long, and removes anything that's no longer needed.

That idea gets a bit more exciting when you remember that a Backup 2.0 solution might use disk blocks as its lowest form of granularity, but that it also relates each disk block to the real-world data it represents—like a file, or a message, or a database table. Figure 8.7 shows how a Backup 2.0 solution might re-assemble disk blocks into an Exchange message store, mount that store, and then display the individual mailboxes and their messages. By combining this capability with a retention policy, a solution might allow you to specify that "all messages be kept for 1 year," rather than worrying about what that means at a disk block level.

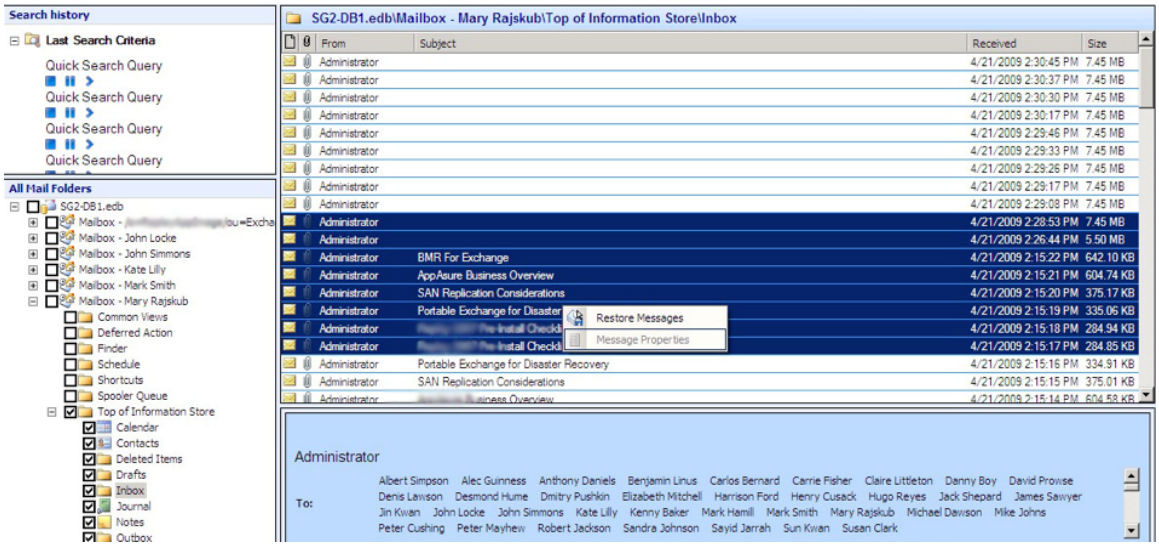

Figure 8.7: Managing individual messages from a backup store.

Obviously, data retention can increase the amount of disk space that your backup solution requires—but if it's a business requirement, it's simply something you have to plan for.

Compliance and Security—Here Come the Auditors

Now the tricky, tricky part: Making sure your backups are compliant with your security policies, and with any externally-imposed security requirements. Even many midsize companies today face tough industry and legislative security requirements:

- If you're in the medical industry, you're probably dealing with the Health Insurance Portability and Accountability Act, commonly known as HIPAA.
- Anyone dealing with financial services has to comply with the Gramm-Leach-Bliley Act, or GLBA.
- Any publicly-traded companies have to meet the requirements of the Sarbanes-Oxley Act, or SOX.
- Any company who accepts credit cards has to deal with the Payment Card Industry (PCI) Data Security Standard (DSS), which is enforced by companies like Visa, MasterCard, and American Express.
- Federal contractors of any size are often subject to Chapter 21 of the Consolidated Federal Rules (21 CFR) and various FISMA requirements.

And that's just (mainly) in the US; many other countries have similar rules and laws. Typically, these all boil down to a fairly common set of technical requirements:

- Your data has to be protected against unauthorized disclosure. That breaks down into a few pieces:
  - You usually need to encrypt your data—both on-disk and in-transit across your network
  - You need to be able to place access controls on your data, so that only authorized individuals can get to it
- You have to prove that the correct access controls remain in place on your data. That normally means you have to capture any changes to access controls, such as file permissions, in a tamper-evident audit log
- You have to track who actually touches your data, which also means a tamper-evident audit log

Auditors, the folks who come along to check and make sure you're doing all these things, are fun people who completely understand if your *backed-up data* doesn't comply with these rules. You wish. In reality, data is data; auditors don't care where it is or why it's there, but it has to follow these rules.

Building Compliance Islands

One popular technique for making it easier to comply is called *islanding.* Figure 8.8 illustrates; basically, you designate one or more servers that will contain all the data that you *have* to follow the rules on, and everyplace else you *don't* follow the rules. See, these various laws only cover certain types of data: HIPAA covers healthcare and patient information; PCI DSS concerns itself with credit card holder data.

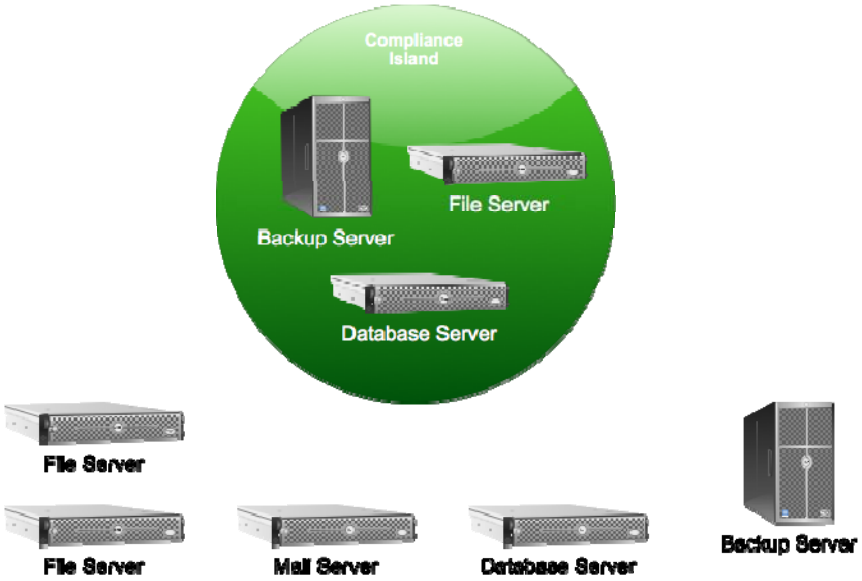

Figure 8.8: Creating an island of compliance.

The idea is that the laws or other requirements only care about some kinds of data, so you sequester all that data and just manage compliance on a few servers. Of course, if you're caught with data-of-interest *outside* the island, you're probably in trouble and might pay some fines—so it pays to be very clear with everyone in the company about what is supposed to live where.

With this islanding technique, you can have a separate backup server—because your backed-up data will have to be compliant just like "live" data is. To make sure your backup solution *can* be made compliant, take a look and see if it can do the following:

- **Encrypt data on disk.** If your solution can't encrypt data on-disk, then at least see if it is compatible with Windows' Encrypting File System (EFS) or BitLocker technologies, which can handle the encryption for you.
- **Encrypt data on the network.** This refers to backed-up disk blocks that are flying from servers to the backup solution, or from the backup solution to a server under recovery. Again, if the backup solution can't provide this, you can always use Windows' native IP Security (IPSec) features to create encrypted network channels between servers—although IPSec imposes additional processing overhead.

Note

It's possible to obtain network cards that handle IPSec encryption in hardware, placing little or no overhead on the server itself. These can be useful to have within your "island of compliance."

- **Audit all access to data.** A Backup 2.0 solution makes it easy to get to data, so it's going to need to keep an audit log of every access to that data. Unfortunately, simply logging to the Windows event logs is usually *not* sufficient, simply because that log is pretty far from "tamperproof" or "tamper-evident;" administrators can clear the log very easily, erasing any evidence of their own wrongdoing. Typically, acceptable logs are in a database like SQL Server, which can be independently secured, encrypted, and so on.

  Note

  Log consolidation solutions, such as Microsoft's Audit Collection Service (MACS; part of System Center Operations Manager), may provide an acceptable way to turn Windows' Security event log into a tamperproof or tamper-evident audit log.

- **Secure the data.** You can't simply allow all users, or even all administrators, access to backed-up data. You need a way to secure the data so that only authorized individuals can get to it. In some cases, this may mean configuring a backup solution so that only members of a "Security Administration" team can access backed-up data.

  Note

  One way to approach this is to create dedicated user accounts that have permission to access backed-up data. These accounts would have long, complicated passwords, which would be broken into two or more pieces and often written down and stored in a safe. That way, two or more people are needed to access the account, and thus to access the data, ensuring that no one person can get to the data unobserved.

- **Audit configuration changes.** Because the configuration of your backup system is critical to maintaining compliance, you also need to audit any changes to that configuration and store those changes in a tamperproof or tamper-evident log.

If your company *isn't* subject to external security requirement or similarly-strict internal security policies, then you're in luck. You won't have to worry about any of these concerns.

Integrating Backup 2.0 with Backup 1.0

With all the glories of Backup 2.0, why in the world would you ever want to *integrate* Backup 1.0? We live in a world that requires flexibility. You may, for example, simply want an old-school tape backup of your servers on an occasional basis as an extra layer of redundancy—the "belt and suspenders" approach, if you will. That may seem paranoid to some, but I say go for it—the more backups, the merrier. *No piece of software is 100% perfect,* meaning whatever backup solution you're using—even Backup 2.0-style software—may have some imperfection that causes a problem for you. Having an old-fashioned tape backup, made using different software, provides a "Plan B" and prevents you from being completely in the lurch if a problem *does* come up.

There's nothing about Backup 2.0's disk block-based technique that intrinsically prevents you from making a parallel Backup 1.0-style, file-and-folder-based backup to tape; in many instances, the two can run simultaneously. But not always. Keep in mind that most backup solutions rely on an agent of some kind, which bundles up data and ships it off to the backup server. Many agents rely on Windows' Volume Shadow Copy Service (VSS or VSC, depending on who you talk to) to copy in-use files like SQL Server databases. VSS works by creating a temporary snapshot on disk, and it's that snapshot that the agent grabs for its backup. Sometimes, even when not using VSS, agents have to make local copies of data so that they can perform compression or encryption or whatever.

Local copies. Snapshots on disk. Yes, your Backup 1.0 backup agents are *writing files to disk,* and you're simultaneously-running Backup 2.0 agent is going to see those changed disk blocks and try to back them up. So you're backing up the temporary files used to make the backup of the backup with…of… I'm lost. Figure 8.9 shows how chaotic this can be.

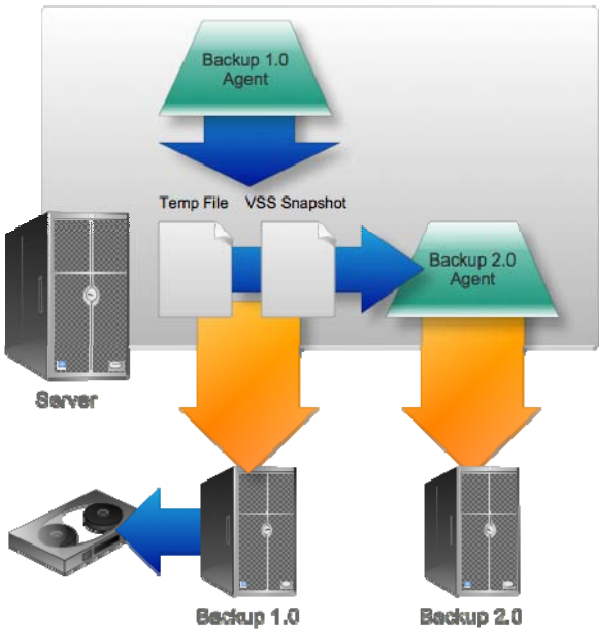

Figure 8.9: Double and triple backups.

There's no reason for the Backup 2.0 solution to be grabbing temp files and VSS snapshots; it's *already backed up that data* when the original disk blocks hit the disk drive; by backing up these temp files, it's just doing unnecessary work.

That's why, if you're going to take a "normal" backup of a server to tape, it's often best to suspend disk block imaging on your Backup 2.0 server. Ideally, you can have the Backup 1.0 agent writing to an area of disk that you can tell the Backup 2.0 agent to simply ignore; that way there's no overlap.

> **Note**
>
> As I imagine them working, it might not be enough to simply "shut off" a Backup 2.0 agent; it would likely scan for changes it missed as soon as you turned it back on. It depends a bit on how your Backup 1.0 agent works: If it cleans up its temp files, they won't be there to be "seen" when your 2.0 agent comes back online; if your 1.0 agent leaves stuff laying around in disk, though, your 2.0 agent will see it and back it up—unless you can exclude a certain directory structure or file type or something.

Backups as a Form of Migration

With virtualization becoming increasingly popular, we're also seeing an increase in the availability of tools to handle Physical-to-Virtual migrations, or P2V moves. There's also V2V, P2P, and V2P, if you want to be precise—in other words, moving a server's image to wherever it needs to be moved.

Traditional x2x migration tools work pretty much like an image-based Backup 1.0 solution: They take a snapshot of the server's hard drive, copy it elsewhere, and convert it into whatever virtual format or physical image layout is needed. The downside is that they often require the source server to be completely offline, or at very least not under very much use, so that they can get all the server's files in a "quiescent" state.

A Backup 2.0 solution, in contrast, would make the *perfect* x2x migration tool, provided it had some provisions for restoring server images to dissimilar hardware (something I identified as useful in the disaster recovery section of this chapter, also). A Backup 2.0 solution could easily back up either a physical or virtual server, and can do so *while it's running,* grabbing changes as they happen almost in real time. The "migration" part comes when you *restore* the server, again either to a physical or virtual machine. Because many of the major virtualization vendors have made their virtual disk image formats public, a Backup 2.0 solution could even produce a complete virtual machine image that's ready to be loaded onto a hypervisor and started up—a capability I've mentioned several times in this chapter.

Having a backup solution that can do double-duty as a migration solution—particularly a migration solution that can do on-demand production of physical or virtual images—is a very useful bit of flexibility.

Coming Up Next…

We've explored Backup 2.0 in almost every detail—except for some operational realities. Assuming you obtain a solution that implements the techniques and technologies I've discussed in this and previous chapters—how do you go about actually managing that solution on a daily basis?

In the next chapter, I'll look at one of the most important aspects of any backup infrastructure: storage architecture. As you might imagine, we're going to discard the notion that magnetic tapes are the be-all, end-all of backup storage, and look at Backup Storage 2.0, which will involve disks, SANs, and yes, tapes as well. I'll also explain how some new technologies—like de-duplication—combine with some older techniques—like compression—to change the *reasons* behind many of our storage choices.