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 4: Exchange Server Backups

Ask anyone in the organization what their most mission-critical piece of infrastructure is, and you'll probably hear "email" as a common answer. Or you might not: Many folks take email for granted, although they expect it to be as available and reliable as a telephone dial tone. Users who have never suffered an email outage almost can't imagine doing so; once they do experience an outage, they make sure *everyone* knows how much they're suffering. As one of the most popular solutions for corporate email, Exchange Server occupies a special place in your infrastructure. It's expected to be "always on," always available, and always reliable. Disasters simply can't be tolerated. What's more, users' own mistakes and negligence become very much *your* problem, meaning you have to offer recovery services that are quick and effective, even when you're recovering something that a user mistakenly deleted on their own.

Native Solutions

Exchange Server's native backup and restore capabilities are tied in part to the underlying Windows operating system's (OS's) capabilities—which isn't always a good thing. Part of Exchange's recovery capabilities come from the fact that deleted messages aren't actually deleted from the system. Email clients such as Microsoft Outlook automatically move deleted messages into a Recycle Bin, where they stay for a configurable period of time or until the user manually empties the Recycle Bin. When using other email clients, such as a generic IMAP client, deleted messages are retained on the Exchange Server computer even if they're not actually moved to the Recycle Bin; deleted messages are simply left in their original folder and hidden from the user's view until a configurable amount of time has passed, or until the user specifically purges deleted messages as part of a "cleanup" operation. As Figure 4.1 shows, Outlook actually displays these deleted-in-place messages in a special font rather than hiding them completely, illustrating how IMAP messages are left in-place when deleted.

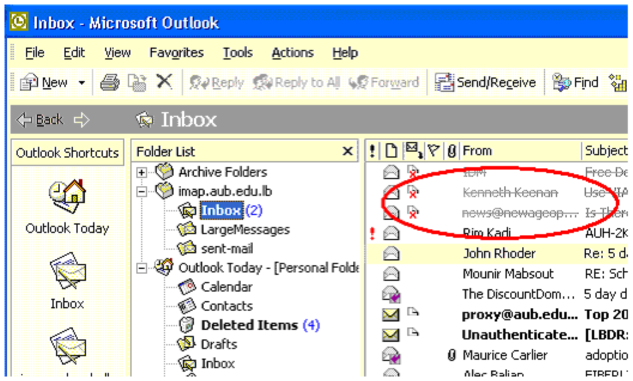


Figure 4.1: Deleted messages in Microsoft Outlook.

All of this functionality is designed to provide users with a self-service recovery option: If they accidentally delete a message, they can either undelete it or retrieve it from the Recycle Bin.

Once a message has passed beyond the realm of the Recycle Bin or other undelete options, recovering a message becomes *your* problem. Unfortunately, Exchange Server doesn't include any built-in backup and recovery mechanisms; it does include an application programming interface (API) that allows other applications to interact with Exchange Server to make backups and perform restores. Exchange provides support for Windows' Volume Shadow Service (which I discussed in the previous chapter) for backups, too—the Windows-native Backup utility, included with Windows server 2003, can use the Volume Shadow Service interface to make backups of Exchange.

Note

In what must have been a miscommunication between product teams, the new Windows Server Backup utility included in Windows Server 2008 does *not* offer support for Exchange Server backups, meaning you have no native option for producing backups of your Exchange data. Microsoft does offer an additional-cost backup solution that supports Exchange backups, and obviously numerous third parties provide varying levels of support for Exchange backups. As of Exchange Server Service Pack 2, Exchange includes a new plug-in that does enable Windows Server Backup to natively back up and restore Exchange databases.

As of Exchange Server 2007, Exchange includes a feature called Clustered Continuous Replication. CCR is designed to replicate Exchange database transactions—the individual changes that are made to the database—to a separate Exchange Server computer. There are specific hardware, software, and environmental requirements to make CCR work, and it does require that you have additional Exchange Server computers in the environment. CCR is Microsoft's preferred solution for whole-server recovery because it essentially keeps a spare copy of the Exchange database on a separate machine. The costs involved in CCR can be quite high, however, because you're basically maintaining a complete, spare Exchange Server machine—hardware and all, unless your spare is virtualized—just sitting around waiting for the first server to fail.

Figure 4.2 shows what CCR looks like; notice that a third *witness* exists here. The witness' job is to make sure that the primary active node is working; if it isn't, the witness is key in making an automatic failover to the passive node happen.

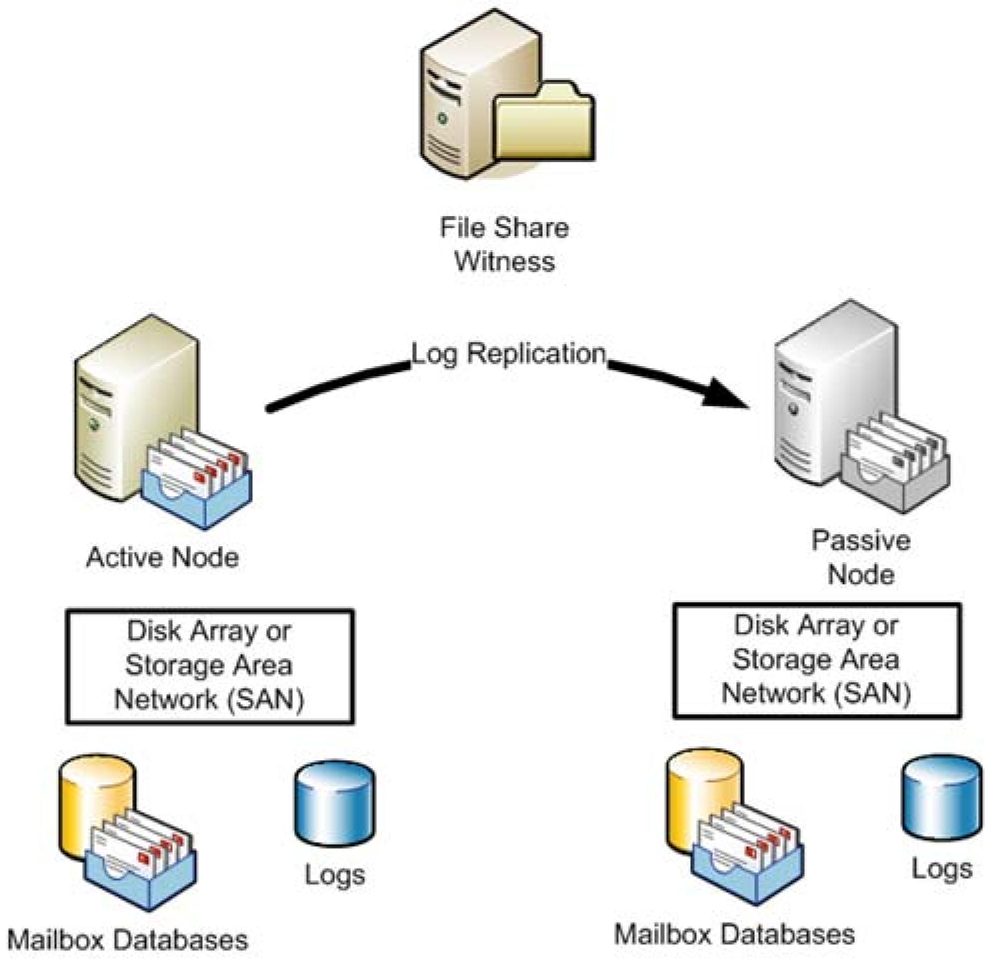


Figure 4.2: CCR.

A variation of CCR is Local Continuous Replication; LCR differs in that it uses transaction replication to create a copy of the Exchange database *on the local server*, on a separate set of disks. This gives you a copy of the database without the need for a separate server, although your Exchange Server hardware obviously remains a single point of failure in that scenario. LCR is less expensive than CCR but does require extra locally-attached storage on the Exchange Server computer.

CCR is primarily useful for whole-server failures—a full-on disaster, in other words (LCR is only useful at protecting against a database failure—if the entire server fails, you lose the LCR replica, too). Neither of these replication techniques is especially useful for recovering single messages. In fact, neither Windows nor Exchange offers a particularly effective solution for single-message recovery. Instead, they rely entirely on the Recycle Bin functionality implemented in Outlook, the most commonly-used Exchange client application.

Problems and Challenges

Exchange offers unique problems and challenges in the backup arena. I'll discuss many of these in more detail later in this chapter, but for right now, I want to briefly introduce them from a business, rather than a technology, perspective.

A Bit About How Exchange Server Works

Most of these derive from Exchange's architecture, so it's worth talking a bit about how that architecture works. Exchange is built around a transactional database. In this regard, Exchange is similar to SQL Server, although the underlying structure of Exchange databases is very different from the relational ones found in SQL Server. *Transactional* means some important things:

- When changes are made to the database, a note of those changes is first made in the transaction log, a special file managed by Exchange Server. *Changes* might include new messages, incoming messages, or even changes to messages—such as marking a message as having been read or replied to. The transaction log is basically a "to do" list—"mark message #164783 as read" or "insert new message #7847829 with the following contents."
- After noting the change in the transaction log, Exchange searches its database for the bits of data that actually need changed. Those bits are loaded into the server's memory, and the changes are applied to the data in-memory.
- After a certain amount of time, the changed data is written back to the database on disk. This happens quickly, but often not immediately; Exchange may choose to cache data in-memory so that multiple changes can be applied in succession.
- Once changed data is successfully written back to the database on disk, Exchange goes to the transaction log and "checks off" the change as having been completed and committed.

If the Exchange Server computer crashes or loses power, no work is lost provided the transaction log is intact. Even though changes existing only in memory may not have been safely written to disk, Exchange can go back to the transaction log and simply re-do, or *replay*, any transactions not "checked off" as committed. This transaction log also provides the basis for LCR and CCR: Individual transactions are replicated from the Exchange Server log and replayed, creating an exact duplicate database.

Why Exchange Server Backups Can be Tricky

So the Exchange database files are actually the result of millions of transactions being applied in-memory and then saved to disk. The database is obviously indexed so that individual messages can be found easily, and a great deal of structured data—such as which messages belong to which users—is stored in the database along with actual message data. Normally, while Exchange Server is running, only its processes have access to the database. All of this conspires to make certain backup and recovery tasks a bit more complicated:

- Individual Item Recovery. Because individual messages are stored in a monolithic database rather than as individual files on disk (as is the case with most Unix-based mail applications), recovering an individual message can be tough. You might, for example, have to retrieve the right whole-database backup, then dive into it as Exchange would to find the message or messages you're after.
- Data Corruption. The slightest data corruption on a backup can render the entire database useless—meaning whatever you're using to make backups has to be 100% reliable and capable of detecting and repairing data corruption.
- Data De-Duplication. Exchange actually has built-in data de-duplication of a sort. Individual messages (and, if configured, attachments) addressed to multiple users are stored only once in the database, provided all the users' mailboxes are in the same message store. Thus, backup software gets the advantage of this de-duplication, creating smaller backups than if each message was extracted individually.

  However, in order to facilitate single-message recovery, many backup solutions not only back up the entire database but also extract individual messages and back up those independently. This lets them index, search, and recover individual messages—but with some solutions loses the built-in de-duplication and increases the size of the backup data.
- Search and e-Discovery. Exchange doesn't have good built-in capabilities for searching across the entire database, which is often needed when you need to recover a message or perform e-Discovery for legal purposes. Because many backup solutions grab the entire Exchange database, they're often incapable of searching through that database either.

Obviously, some of these special concerns are ones we'll have to address in a Backup 2.0 solution.

In the Old Days

I want to look at how old-style Backup 1.0 solutions address both the basic and special needs of Exchange Server recovery. It's important to recognize what works and what doesn't in these traditional solutions so that we can identify areas for improvement as well as areas that should be retained in a new-style, Backup 2.0 solution.

Backup Techniques

Exchange Server backups can be *complicated* from a process viewpoint. Consider Figure 4.3, which is excerpted from a blog entry at http://blog.thejpages.com/2008/03/18/why-are-we-still-backing-up-exchange.aspx. The author proposes that Exchange Server *not* be backed up. Instead, he suggests enabling *circular logging*—meaning Exchange's transaction log will automatically overwrite older entries as needed to write new ones. Using CCR, the Exchange database is replicated—in this proposal—to *two* passive nodes, making two complete copies of the database. One passive node sits in the data center, ready to take over in the event of a failure in the production Exchange computer. The other passive node is in an offsite location and delays replaying incoming transactions by 7 days—meaning its copy of the database is always 7 days old.

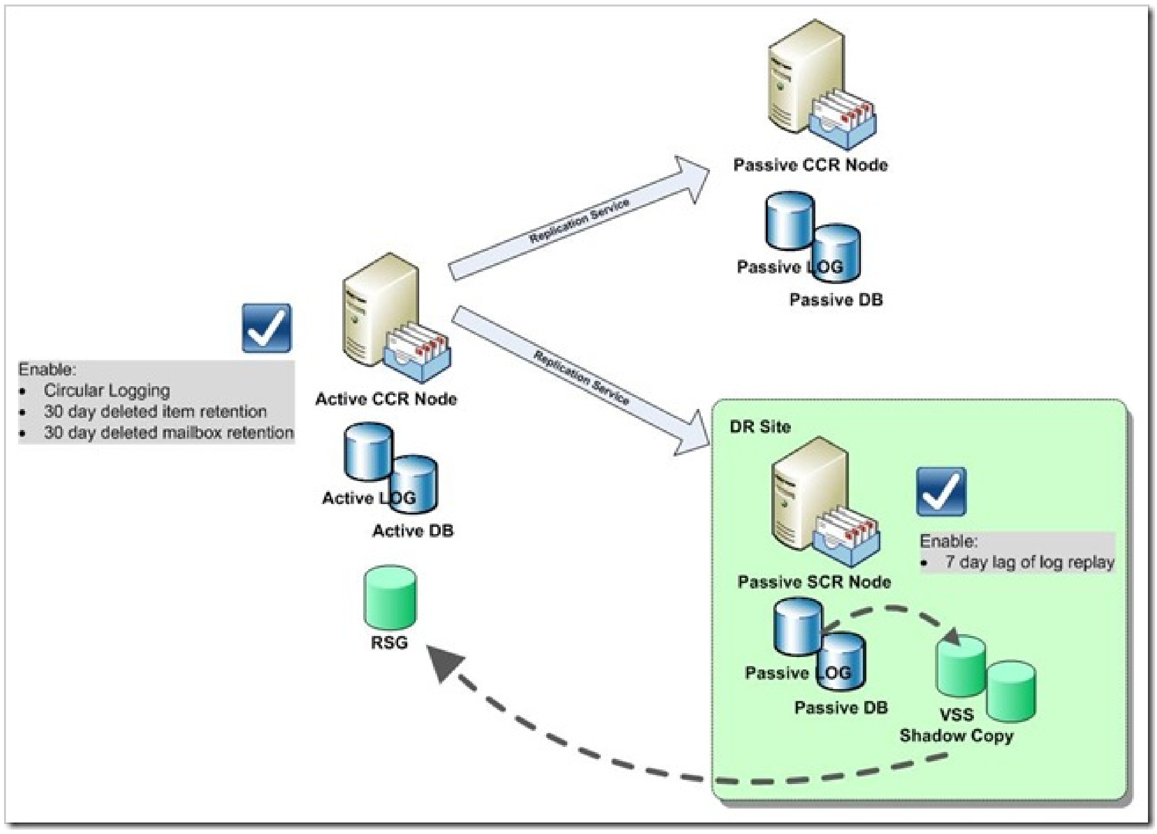


Figure 4.3: Proposal to *not* back up Exchange.

There are some downsides to this approach. Although it provides great almost *instant* recovery capabilities with very little lost data, it would be *very* expensive to implement. You're looking at two additional Exchange Server licenses, even if they're on virtual machines. If they're not virtualized, that's two additional Exchange Server machines, too, and a great deal of network bandwidth. You'd still have to have *some* kind of backup running to support users who delete messages and want them back—in the proposal, Exchange retains deleted items for just 30 days and many organizations must—due to legal or industry regulations—retain messages for a far longer period of time.

Traditional Exchange backups typically seek to grab the entire database, usually by connecting to Exchange Server's Volume Shadow Service API. As mentioned earlier, these solutions may also extract individual messages through other APIs, giving them not only a complete copy of the database—used for disaster recovery—but also access to individual messages.

Restore Scenarios

The most common restore scenarios in Exchange are single-message recovery or single-mailbox (including all of its messages) recovery. An article at http://www.msexchange.org/tutorials/ExMerge-Recover-Mailbox.html details a fairly common way of doing this in Exchange Server 2003: Start by installing the free ExMerge utility (available from http://microsoft.com/download). Restore your database backup from tape or whatever—you may wind up restoring it to a different Exchange Server so that you're not affecting your production server. As Figure 4.4 shows, you'll be able to use ExMerge to export the desired mailbox to a PST file, which can be opened with Microsoft Outlook. If you want to recover a single message, you attach that PST to an Outlook client and go hunting for the message you want. Messages can be "dragged" out of the PST file, via Outlook, and "dropped" into an active Exchange mailbox to get the message back onto the server.

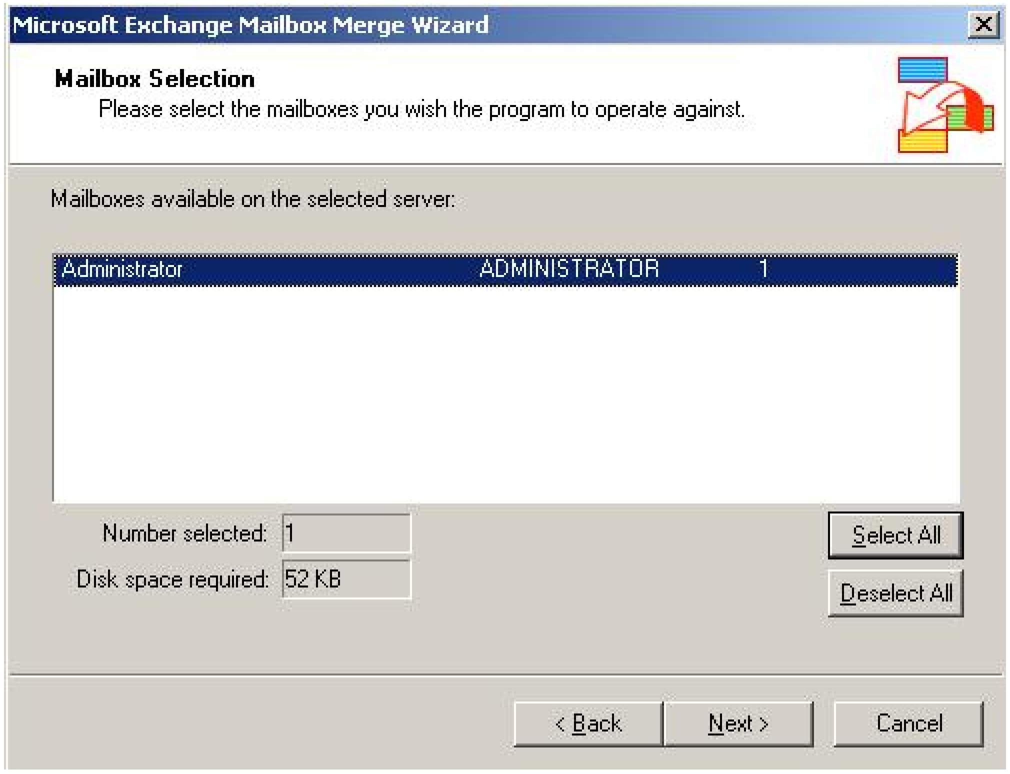

Microsoft Exchange Mailbox Merge Wizard
Mailbox Selection
Please select the mailboxes you wish the program to operate against.
Mailboxes available on the selected server:
Administrator   ADMINISTRATOR   1
Number selected: 1
Disk space required: 52 KB
Select All
Deselect All
< Back
Next >
Cancel

Figure 4.4: Recovering a single mailbox in ExMerge.

Newer versions of Exchange offer a Mailbox Recovery Center that performs essentially the same task without the need for ExMerge. You still have to add a storage group to the Recovery Center, restore an Exchange database, mount the database into the Recovery Center storage group, then go browsing for the mailbox you want to recover. Figure 4.5 shows what the Recovery Center looks like. It's still time-consuming, but perhaps prettier than ExMerge.

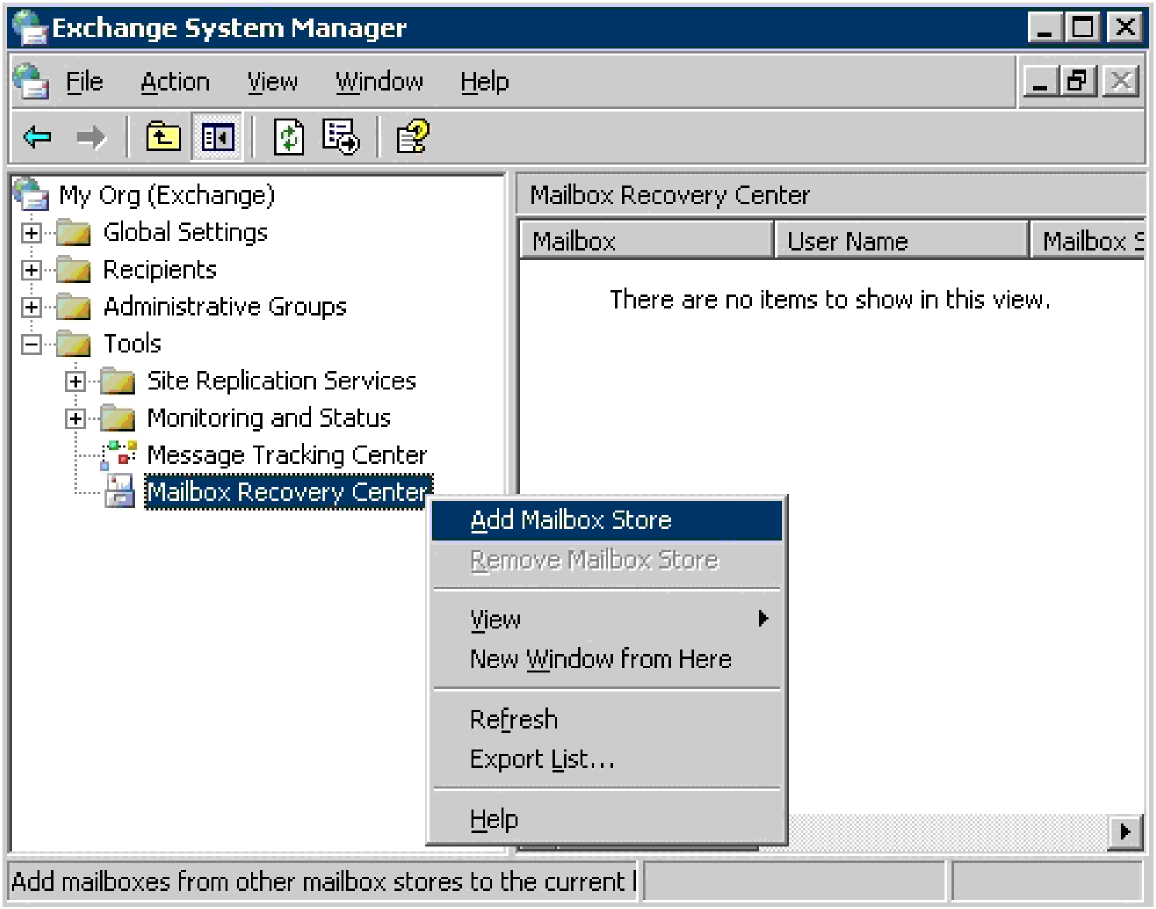


Figure 4.5: Mailbox Recovery Center.

Frankly, this whole process is nuts. It's slow, cumbersome, and incredibly labor-intensive for administrators. This is why I've never been in a single environment that doesn't have some kind of third-party backup solution—often *just* to provide more efficient single-message and single-mailbox recovery.

Third-party backup solutions work more quickly but involve substantially the same process. You go get your backup data off of tape—which will take some time because you might have to restore a full backup and multiple incremental or differential backups to re-create the point in time you want to restore from. The backup software usually maintains its own indexes of available messages and mailboxes, so you browse or search through that until you find what you want. What the solution typically automates is the pain of extracting the mailbox or message from the database and putting it back into Exchange Server—so the process is less labor-intensive for the administrator but still pretty awkward and not necessarily very fast.

Disaster Recovery

Disaster recovery in Exchange is straightforward—and usually time consuming. You restore the most recent full backup. You restore any incremental or differential backups made since then. You restore any transaction log backups made since *then.* Finally, you stand back and let Exchange sort it all out—and be prepared to wait because the process can take hours. The development of CCR and LCR technologies was driven in large part by the time-consuming nature of more traditional backups; with CCR or LCR, you've got a spare database sitting right there, ready to be used—provided what you're after is the *most recent* database. In true disaster recovery situations—a total hardware failure or even a data center disaster—you usually *do* want to get the latest version of the database back up and running quickly.

Where CCR and LCR fail is if something goes wrong that *gets replicated*—in those cases, the copy of the database will also contain whatever went wrong, and you'll be back to time-consuming tape restores to rebuild your databases.

Backup Management

Depending on the backup solution you're using, managing traditional backups can be quite a science. Because Exchange databases can grow quite large, some organizations don't have the time to grab a full database backup as often as they'd like. That means you're stuck managing full backups, incremental backups, differential backups, and in many cases, *log* backups—backups of the Exchange transaction log.

In fact, managing transaction log backups is crucial to minimizing data loss in the event of a failure. The transaction log literally contains every single piece of work that Exchange needs to do; Exchange's ability to replay that log to re-create work is an effective recovery technique. Some organizations will grab transaction log backups throughout the day; many third-party Exchange backup solutions will bundle transactions from the log into 15-minute "recovery points." Of course, losing 15 minutes of email traffic is still a pretty big disaster.

The downside to all this is simply management overhead and storage space. Exchange backups can occupy a lot of disk and tape space, and keeping track of all the files can be complex. In fact, most third-party Exchange backup solutions are *primarily* solutions for managing backup files—after all, the actual backup functionality comes from Exchange's APIs.

Rethinking Server Backups: A Wish List

Let's revisit our Backup 2.0 "mission statement" from 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.

This is a tricky statement to evaluate when it comes to Exchange. Certainly, with CCR or LCR, we can achieve backups that offer very little downtime; in the case of CCR, downtime might amount to a few seconds. We would certainly lose *very little* data, although some data loss is possible because both CCR and LCR utilize asynchronous replication, meaning it's possible for a few minutes' worth of transactions to occur on the source, yet not replicate to the mirror when a failure occurs.

But CCR and LCR have two distinct problems: First, they only maintain a current, working copy of the database; they don't provide a long-term archive and they don't allow you to restore to a particular point in time. If you accidentally delete a mailbox, that deletion is replicated; after the deleted mailbox retention period elapses, the mailbox is gone forever no matter how many CCR or LCR replicas you have. Second, both CCR and LCR are expensive—in terms of storage resources for both, and in terms of additional hardware and software licenses for CCR. Neither LCR nor CCR are designed to provide single-message recovery beyond the deleted item retention period in Exchange.

> **Note**
>
> It's been suggested to me by some administrators that they could simply set the deleted item retention period very high—as high as 5 years in one case I saw. I don't recommend it; the Exchange database will get *huge*, and it simply isn't intended to be a permanent archive. Performance will suffer, and it'll become harder and harder to get real backups as the database bloats.

Traditional third-party backup solutions, which rely either on the Exchange transaction log or the Volume Shadow Service API, have all the failings of any Backup 1.0 solution—which I explained at length in the previous chapter. Backup file management, backup windows, time-consuming restores, and other disadvantages have been frustrating Exchange administrators for more than a decade.

Let me clue you in to a little secret: The reason traditional Exchange backups can be frustrating or incomplete—even in the case of CCR and LCR—is that they rely on catching transactions as they occur or reading information from Exchange's own APIs. In other words, everything depends on *how Exchange works*. For an up-to-date backup, like what CCR and LCR offer, you have to replicate transactions. For a full point-in-time backup, you have to talk to Exchange and deal with the data the way Exchange gives it to you. When it comes to Exchange Backups 1.0, that's the ultimate problem. The solution? Stop dealing with Exchange for your backups.

New and Better Techniques

In the previous chapter, I posited a new type of backup solution that focused on disk blocks. Figure 4.6 illustrates my proposal: Forget about talking to APIs and just grab the information as it is written to disk.

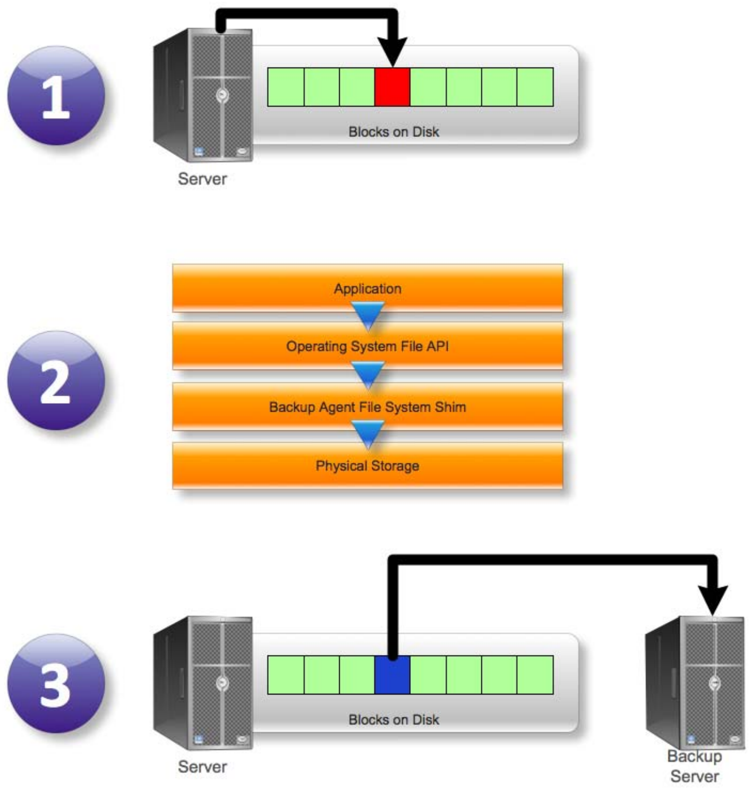


Figure 4.6: A proposal for Backup 2.0.

Think about it: Software developers—like the ones who wrote Exchange Server—know that server memory isn't reliable. A loss of power, a software crash, whatever, and memory is lost. Disk, however, is much more reliable and is persistent. Exchange's transaction log is designed to help provide a cover for unreliable memory: In the event of a memory failure, the transaction log allows work to be replayed. For that reason, the *log itself* sits on persistent disk storage.

That means a Backup 2.0 solution can simply grab blocks of disk space as they are changed on disk. That will grab any changes to the transaction log as well as changes to the main Exchange database files (there are a few files for every mail store). By immediately shipping copies of those disk blocks to a separate server, you have a *continuous* backup that *doesn't* rely on talking to Exchange APIs, replicating transactions, or any other complexity.

Now, that's all well and good for simple files on disk: If you want to restore something, just track down the disk blocks that comprise the file and put those blocks back on the server. Poof, file restored. For Exchange disaster recovery, as I'll explain in a moment, it's a fast way to get an entire server back online. However, how does it help with the more common restore scenarios like single mailbox recovery or single message recovery?

Better Restore Scenarios

This is where we put Backup 2.0 to the test: Does it meet the mission statement when it comes to Exchange?

> 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.

I think it does. We've got a nearly real-time backup of everything that gets written to Exchange's disks—including databases and transaction logs. That means we don't lose any data, and restoring data *doesn't* need to include taking Exchange offline (unless we're talking about a complete disaster recovery scenario—which I'll cover next).

With the right tools and interfaces, Backup 2.0 enables single-item recovery—something I'll outline a bit later in this chapter. That's really the key restore scenario, although you might also, from time to time, need to recover an entire Exchange database and mount it elsewhere for testing purposes.

For example, I have one client who routinely restores Exchange databases to a disconnected Exchange Server, where they perform vulnerability scanning and anti-spam testing. Backup 2.0 excels at this, as it can quickly bring back an individual file—even one as large as an Exchange database—very quickly. But that's kind of a Backup 1.0 mentality: With Backup 2.0, you could make that testing operation even faster by simply exporting *the entire server's disk image* into a bootable virtual machine. That virtual machine can be easily segregated for testing so that it wouldn't interfere with the production network (that kind of testing is where virtualization saw its first widespread uses, in fact).

> **Note**
>
> Backup 2.0 is all based on disk blocks—raw, disk-level data. Where that data sits doesn't matter, meaning Backup 2.0 is also a great way to do physical-to-virtual, physical-to-physical, virtual-to-virtual, and virtual-to-physical moves and migrations. I've used Backup 2.0-style solutions in many cases to move physical Exchange Server computers into virtual machines as part of a larger enterprise virtualization project.

Better Disaster Recovery

Disaster recovery is what Backup 2.0 is all about, and Exchange Server is no exception. With a disk block-based backup image, you can quickly restore your entire Exchange Server to not just the most recent backup but also to any given point in time. You can even restore your Exchange Server to a virtual machine, which is great for huge disaster recovery scenarios where you might be hosting those virtual machines at a recovery facility or even in some online hosting provider.

So how does Backup 2.0 complement or interfere with CCR, Exchange's built-in recovery solution? Keep in mind that CCR requires a passive, standby server. That means the expense of additional Windows and Exchange licenses, and possibly the expense of dedicated hardware, all as a standby to a failure. That's an expense some organizations are happy to bear, but it's not for everyone. In some cases, your CCR passive node might be a less-capable machine (that might be the case if it was running in a virtual machine, for example) designed to get you through a tough time with less-than-normal performance. In those instances, Backup 2.0 can help by getting Exchange up and running more quickly on your *original,* full-powered hardware. For organizations that can't afford CCR, Backup 2.0 provides what is perhaps the next best thing: A fast way to quickly bring Exchange back online in a bare-metal recovery situation.

Backup 2.0 complements CCR in that Backup 2.0 provides the ability to roll back to a previous point in time; CCR does not. CCR's goal, remember, is to create an exact replica of your Exchange databases with as little latency as possible. That means if you do something *wrong,* that something is going to replicate via CCR very quickly, meaning your "backup" is also messed up. CCR can't undelete a mailbox or a message, and it can't help recover from accidental or malicious actions. Backup 2.0, however, can do so—and it can protect your passive CCR nodes at the same time it protects your active Exchange Server computers.

What I like best about Backup 2.0 is that the backups can be restored almost anywhere. Lose an Exchange Server computer and don't have a spare handy? No problem: Restore to a virtual machine (I tell clients to always have at least one virtualization host hanging around that has some spare capacity—for just these emergencies). No reinstalling Windows, reinstalling Exchange, reconfiguring Exchange, and waiting on tape backups to unspool your backed-up databases—just dump the entire server disk image into a virtual machine. It's a fast process and the result is a completely-configured computer that *is* the computer you lost. Clients just reconnect and start working.

Easier Management

My Backup 2.0 idea does need to have a few more Exchange-specific capabilities. For example, Exchange is designed so that transactions remain in its log until a backup of the database is made; that way, you're assured of the log serving as a backup for transactions until the related data is safely on tape. In a Backup 2.0 world, traditional backups don't occur—so the backup agent running on the Exchange Server computer needs to be smart enough to truncate the Exchange transaction log after transmitting the related disk blocks off to the backup server.

From there, you're left without much to manage. No log backups, no full backups, no differential backups—just the ability to restore, from an image, any bit of Exchange you need to, at any time. As you'll see in the next few sections, though, Backup 2.0 can enable some pretty impressive new management scenarios.

What About Performance?

Doesn't all this Backup 2.0 magic place some serious burden on your Exchange Server computers? In my experience, no. The majority of the overhead kicks in when you start de-duplicating, compressing, indexing, and saving data that is usually offloaded to a centralized "backup server." All the Exchange Server-based agent has to do is transmit disk-block deltas over the network—you might see low single-digit increases in things like processor utilization, but that's it. Backup 1.0 solutions tend to hit Exchange Server harder because they're not grabbing data in small chunks all day; they're trying to cram all their backup activity into a small evening window.

Exchange-Specific Concerns

So how will Backup 2.0 work with Exchange? I've already pointed out how specialized Exchange is in the way it works; will Backup 2.0 be able to work with it and still provide a better backup solution than Backup 1.0 does? Much of Exchange's functionality and architecture are specifically designed to accommodate and work within a Backup 1.0 world—will turning that world into Backup 2.0 break everything?

CCR

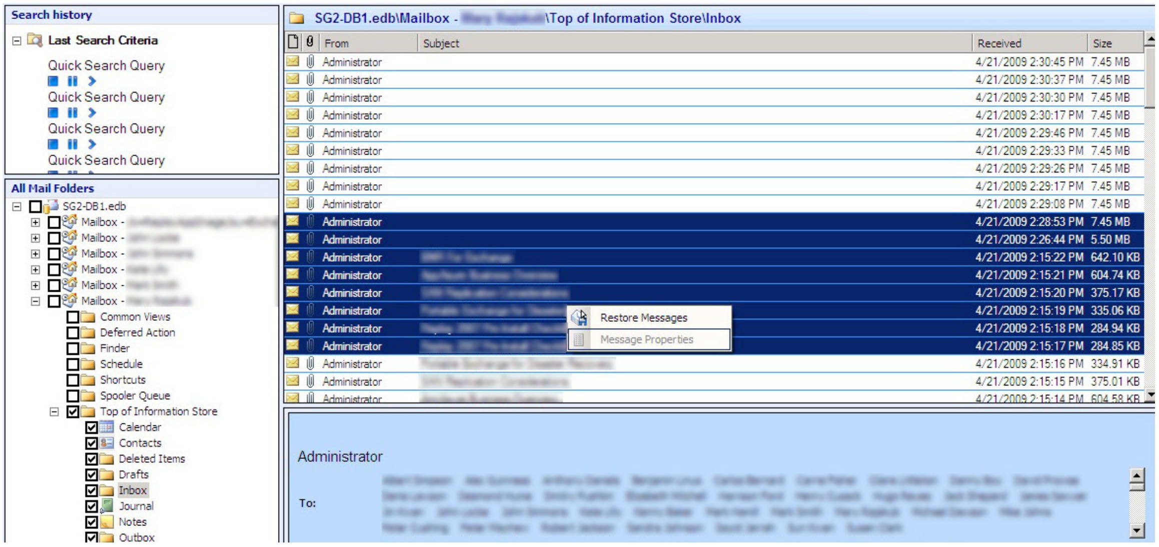
A Backup 2.0 solution needs to be CCR-aware. After all, CCR is still a valuable high-availability tactic, giving you near-instantaneous failover in the event of a complete server failure. CCR even supports geographically-dispersed clustering. So a backup solution needs to understand CCR and work to truncate the active CCR node's log based on the passive CCR node's replication of transactions.

Individual Item Recovery

Recovering mailboxes or individual messages to a PST file may be useful—but you shouldn't be stuck with that as your only option. Honestly, giving a user a PST file and telling them to drag and drop messages in Outlook is insanely primitive. A Backup 2.0 solution should eliminate that overhead and let you restore directly to a live Exchange Server computer.

Further, a Backup 2.0 solution should be able to recover individual messages *from the same backup* that would be used for disaster recovery. In other words, you shouldn't have to extract individual messages out of Exchange—you should be able to recover from the backup image of the actual Exchange database. How?

Practically speaking, it would probably be a two-step process. Assume you have an image-level backup of an Exchange Server computer. That means you've got every block of data from disk, so you can re-construct the entire server. With the right tool set, you would be able to "mount" that backup as a file system—in effect browsing the backed-up file system *from a specific point in time* as if it were a network drive. But you're just looking at the backed-up files—Exchange isn't running and its database files aren't opened and being used; you're looking at a static, point-in-time copy of those files. From there, the right tool would let you mount and browse the Exchange database—giving you access to individual mailboxes, messages, and other data. You wouldn't need to do the usual Exchange escapades of restoring the database file from backup, mounting the database to a Recovery Storage Group, and running ExMerge or other utilities against the mounted database. Instead, you'd just dive into the database using a utility that understands the database structure, and get what you need. It might look something like Figure 4.7.



SG2-DB1.edb\Mailbox - [redacted]\Top of Information Store\Inbox

From	Subject	Received	Size
Administrator		4/21/2009 2:30:45 PM	7.45 MB
Administrator		4/21/2009 2:30:37 PM	7.45 MB
Administrator		4/21/2009 2:30:30 PM	7.45 MB
Administrator		4/21/2009 2:30:17 PM	7.45 MB
Administrator		4/21/2009 2:29:46 PM	7.45 MB
Administrator		4/21/2009 2:29:33 PM	7.45 MB
Administrator		4/21/2009 2:29:26 PM	7.45 MB
Administrator		4/21/2009 2:29:17 PM	7.45 MB
Administrator		4/21/2009 2:29:08 PM	7.45 MB
Administrator		4/21/2009 2:28:53 PM	7.45 MB
Administrator		4/21/2009 2:26:44 PM	5.50 MB
Administrator	[redacted]	4/21/2009 2:15:22 PM	642.10 KB
Administrator	[redacted]	4/21/2009 2:15:21 PM	604.74 KB
Administrator	[redacted]	4/21/2009 2:15:20 PM	375.17 KB
Administrator	[redacted]	4/21/2009 2:15:19 PM	335.06 KB
Administrator	[redacted]	4/21/2009 2:15:18 PM	284.94 KB
Administrator	[redacted]	4/21/2009 2:15:17 PM	284.85 KB
Administrator	[redacted]	4/21/2009 2:15:16 PM	334.91 KB
Administrator	[redacted]	4/21/2009 2:15:15 PM	375.01 KB
Administrator	[redacted]	4/21/2009 2:15:14 PM	604.58 KB

Administrator

To: [redacted]

Figure 4.7: Restoring messages from a mounted backup image.

De-Duplication

Exchange's single-instance storage allows each message to be stored only once in the message store—it's a form of data de-duplication. But it doesn't help when a message is sent to users whose mailboxes are on different servers, or even users whose mailboxes are in different stores on the same server. In those cases, the message will be duplicated at least once per store.

But Backup 2.0 can do a better job of de-duplicating data—at least data from a single server—because it's examining *disk blocks.* The same message stored on disk *looks* the same, even though that message might have to be duplicated across multiple mail databases. So the message will be duplicated in Exchange, but once it's backed up by the Backup 2.0 solution, that solution can detect the duplicate disk blocks and store only a single instance—meaning the backup can be smaller than the original database. Add in compression, something even Backup 1.0 solutions offer, and the backup can be many times smaller than the original data.

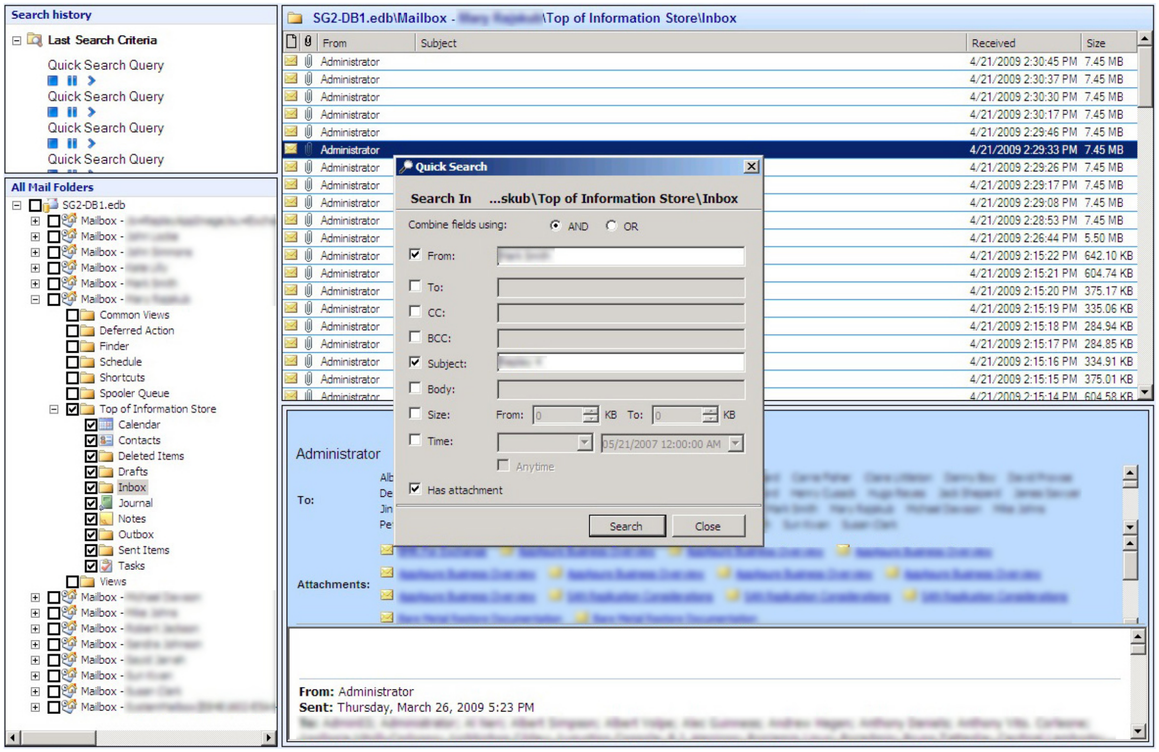
Data Corruption

Because data is being backed up on a block-by-block basis, it's easier for a Backup 2.0 solution to detect and correct errors. A single block of disk data might be as small as 4 kilobytes—detecting a transmission error in that small a piece of data is easy.

Search and e-Discovery

Search and e-Discovery are rapidly becoming key components for many organizations. The US Federal Court System, for example, has imposed strict rules that require pretty rapid responses to e-Discovery requests during court proceedings; failure to meet these requirements can lead to fines and even summary judgments. Knowing that the Exchange database doesn't provide solid search capabilities natively, many companies rely on dedicated message archival and retrieval tools—an additional expense and yet another mass of storage resources to manage. A Backup 2.0 solution, however, can provide solid search and e-Discovery capabilities built right in.

Consider the ability to mount a point-in-time backup image as a browse-able file system, as I've described earlier. Also consider the ability to browse an offline Exchange database from that file system. Given those capabilities—which a Backup 2.0 solution might well offer as a means of performing single-message recovery—you could easily implement a powerful message-search function that makes message searching and e-Discovery possible. Figure 4.8 shows what it might look like.



SG2-DB1.edb\Mailbox - [redacted]\Top of Information Store\Inbox

From	Subject	Received	Size
Administrator		4/21/2009 2:30:45 PM	7.45 MB
Administrator		4/21/2009 2:30:37 PM	7.45 MB
Administrator		4/21/2009 2:30:30 PM	7.45 MB
Administrator		4/21/2009 2:30:17 PM	7.45 MB
Administrator		4/21/2009 2:29:46 PM	7.45 MB
Administrator		4/21/2009 2:29:33 PM	7.45 MB
Administrator		4/21/2009 2:29:26 PM	7.45 MB
Administrator		4/21/2009 2:29:17 PM	7.45 MB
Administrator		4/21/2009 2:29:08 PM	7.45 MB
Administrator		4/21/2009 2:28:53 PM	7.45 MB
Administrator		4/21/2009 2:26:44 PM	5.50 MB
Administrator		4/21/2009 2:15:22 PM	642.10 KB
Administrator		4/21/2009 2:15:21 PM	604.74 KB
Administrator		4/21/2009 2:15:20 PM	375.17 KB
Administrator		4/21/2009 2:15:19 PM	335.06 KB
Administrator		4/21/2009 2:15:18 PM	284.94 KB
Administrator		4/21/2009 2:15:17 PM	284.85 KB
Administrator		4/21/2009 2:15:16 PM	334.91 KB
Administrator		4/21/2009 2:15:15 PM	375.01 KB
Administrator		4/21/2009 2:15:14 PM	604.58 KB

Quick Search

Search In ...skub\Top of Information Store\Inbox

Combine fields using: [✓ AND] OR

- [✓] From: [redacted]
- [ ] To:
- [ ] CC:
- [ ] BCC:
- [✓] Subject: [redacted]
- [ ] Body:
- [ ] Size: From: 0 KB To: 0 KB
- [ ] Time: 05/21/2007 12:00:00 AM
- [ ] Anytime
- [✓] Has attachment

Search Close

Administrator

To: [redacted]

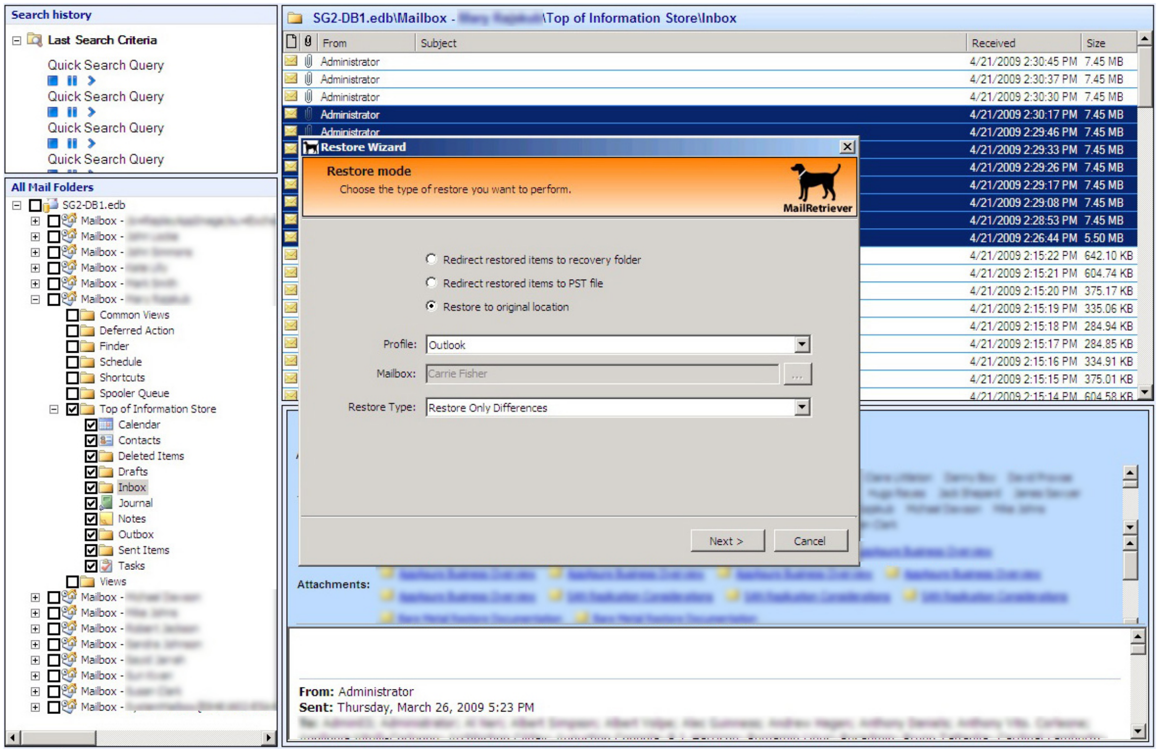
Attachments: [redacted]

From: Administrator
Sent: Thursday, March 26, 2009 5:23 PM
[redacted]

Figure 4.8: Searching for messages in a backup image.

Importantly, this tool would need to be able to attach to *multiple* Exchange databases to search; you can never tell ahead of time which database will contain the messages you're after, and you don't want to have to search each one individually.

In an e-Discovery scenario, you'll typically want to restore messages not to a live Exchange Server computer but rather to a PST—which will often be delivered to legal counsel. Figure 4.9 shows how a Backup 2.0 toolset might implement that export.



```
From            Subject    Received               Size
Administrator              4/21/2009 2:30:45 PM   7.45 MB
Administrator              4/21/2009 2:30:37 PM   7.45 MB
Administrator              4/21/2009 2:30:30 PM   7.45 MB
Administrator              4/21/2009 2:30:17 PM   7.45 MB
Administrator              4/21/2009 2:29:46 PM   7.45 MB
                           4/21/2009 2:29:33 PM   7.45 MB
                           4/21/2009 2:29:26 PM   7.45 MB
                           4/21/2009 2:29:17 PM   7.45 MB
                           4/21/2009 2:29:08 PM   7.45 MB
                           4/21/2009 2:28:53 PM   7.45 MB
                           4/21/2009 2:26:44 PM   5.50 MB
                           4/21/2009 2:15:22 PM   642.10 KB
                           4/21/2009 2:15:21 PM   604.74 KB
                           4/21/2009 2:15:20 PM   375.17 KB
                           4/21/2009 2:15:19 PM   335.06 KB
                           4/21/2009 2:15:18 PM   284.94 KB
                           4/21/2009 2:15:17 PM   284.85 KB
                           4/21/2009 2:15:16 PM   334.91 KB
                           4/21/2009 2:15:15 PM   375.01 KB
                           4/21/2009 2:15:14 PM   604.58 KB
```

Restore mode
Choose the type of restore you want to perform.

- ○ Redirect restored items to recovery folder
- ○ Redirect restored items to PST file
- ● Restore to original location

Profile: Outlook
Mailbox: Carrie Fisher
Restore Type: Restore Only Differences

From: Administrator
Sent: Thursday, March 26, 2009 5:23 PM

Figure 4.9: Exporting search results to a PST file.

It's important that you not over-build your expectations for a Backup 2.0 solution, though. Message search and recovery is only one aspect of e-Discovery; many organizations that routinely deal with legal summons prefer to tag and categorize messages *as they're sent;* this makes it easier to locate messages on-demand (and helps categorize messages for security and monitoring purposes, too), but obviously goes well beyond what you'd expect from a backup solution. So there will still be a market for specific e-Discovery solutions, especially for very large companies who routinely have to perform e-Discovery tasks.

Coming Up Next…

If you thought Exchange Server had some specialize needs, wait until I get into SQL Server in the next chapter. Microsoft's relational database management system is one of the few Microsoft products that has had a well-understood backup and restore system for many years—but once again, I'll try and turn Backup 1.0 on its head and show you where our long-used routines and techniques just don't meet modern business needs.