# 5

# *Protecting and Recovering Exchange Data*

I have spent a significant amount of time delving into Exchange storage technology in previous chapters. This is because of the critical role the Exchange storage engine plays in Exchange server reliability and data integrity. In the last chapter, I looked at why Exchange servers fail and the dependencies that Exchange has on other services and infrastructure. In this chapter, we will dig a bit deeper into the Exchange database engine in hopes of revealing even more to you about how critical this storage mechanism is to your data. Once again, at the core of the store process (STORE.EXE) and Exchange's ESE lie the keys to backing up and recovering Exchange data.

## 5.1 Exchange backup/restore fundamentals

With the changes and new options implemented for Exchange 2000 (and carried forward to Exchange Server 2003), the disaster-recovery API that Microsoft provides as part of Exchange underwent a bit of a facelift. The ESE database engine in Exchange Server has always made an on-line backup API available via two DLLs called ESECLIB2.DLL and ESEBACK2.DLL. (For Exchange 5.5, the DLLs are ESECLIB1.DLL and ESEBACK.DLL.) This API allows Exchange to stay operational and service users while backup operations are performed. In previous versions of Exchange, since there was only one ESE instance, restore operations were performed off-line (the server is down until the restore is complete). However, since Exchange 2000/2003 provides multiple ESE instances (storage groups), recovery operations can be underway within one storage group while another storage group continues to service users. This means the API must be adapted to allow for concurrent operations and to handle the fact that multiple SGs, MDBs, and log file sets must be managed. In addition, there is new stuff to back up in Exchange 2000/2003 such as the Site Replication Service (SRS) and the Key Management Server (KMS), which has

been removed in Exchange 2003 Server. SRS and KMS were not included in backup operations for previous versions of Exchange. Finally, the ESE recovery API must allow for more granularity. You now are able to back up/restore an entire storage group (best practice) or an individual database (MDB). What's more, while a database was one file (*.EDB) in previous versions, it is now a set that includes both the EDB and the STM file. The API has undergone several changes in Exchange 2000 to accommodate these needs. In this section, we will take a look at how Exchange 2000/2003 performs backup and restore operations. Microsoft Exchange Server provides a specific backup API to allow backup products, including Windows Backup, to access the contents of the Exchange information stores while they are on-line.

### 5.1.1 Exchange backup types

The backup and restore functions of Exchange's ESE provide three types of backup capabilities: full, incremental, and differential. Combined with Windows Server 2003's VSS, Exchange Server 2003 also supports snapshot backup types (more on this type of backup later).

A *full backup* (also called a normal backup) backs up the entire directory or information store and allows you to restore it from a single backup. An *incremental backup* backs up just the changes since the last full or incremental backup. These are simply the transaction logs that have accumulated since the last full backup. Restoring incremental backups requires the original full backup plus all the incremental backups (transaction logs) made since that time. A *differential backup* backs up the changes since the last full backup. Restoring a differential backup requires one differential backup and the original full backup. Appendix A provides pointers to backup and restore API functions and their specific uses.

On-line backup operations are fundamental to Exchange Server and enable you to back up databases without shutting down the entire server to perform a file-by-file type backup (off-line backup). While backup operations are in progress, all services continue to operate, and users can access their data on the Exchange server. Database pages that are cached in memory in the information store buffer pool continue to be updated and flushed to the database on disk. Transactions also continue to be written to the transaction log files, and the checkpoint file continues to advance. All in all, the backup and restore technology for Exchange 2000/2003 is very similar to previous versions of Exchange Server with one notable exception. Exchange 2000's advent of multiple storage groups and multiple databases

**Table 5.1** *Exchange Server 2003 Backup Types*

| Backup Type | Files Included | Logs Truncated? | Restore Method |
|---|---|---|---|
| Normal (Full) | Database (EDB+STM) files, Log files, and Patch files (for Exchange 2000 SP1 and earlier versions) | Yes | Last normal backup |
| Incremental | Log files only | Yes | Last normal + all incremental backups |
| Differential | Log files only | No | Last normal + last differential |
| Copy | Database (EDB+STM) files, Log files, and Patch files (for Exchange 2000 SP1 and earlier versions) | No | Not applicable |
| Snapshot (Windows VSS) | Special (more on this type of backup in later in the chapter) | Special* | Special* |

(MDBs) has a substantial impact on how the Exchange backup API works. Table 5.1 compares the backup types available for Exchange 2003.

### 5.1.2 Normal (full) and copy backups

The normal backup (also referred to as a full backup) is the fundamental unit of operation for most Exchange deployments. Regardless of the strategy you select for backup, the normal backup type will be part of your operational procedures. With a normal backup, both the database files and the log files are copied to tape. In addition, the log files are truncated or deleted once they have been copied to the backup media. The truncation point for the transaction log files is the current database checkpoint location. The normal backup operation is also important to database integrity since only during a normal backup are the 4-KB database pages checked for corruption (they are also checked during copy backups and on-line database maintenance as well). This is accomplished by verifying each page read to make sure that the page number requested is the page the database engine received. Next, each page's CRC information (contained in the page header) is verified to ensure that the data contained in the page is valid. The normal backup is also important to the ESE Page Zeroing feature, which I will discuss later in this chapter. To restore from a normal backup, you only need to restore the complete set and allow the ESE database engine to replay any log files required for the database to be in a consistent state. Sim-

ilar to the normal backup is the copy backup. A copy backup differs in that it does not truncate or purge log files once they have been copied to tape. In addition, the copy backup does not update database backup context information contained in the database file header. Copy backups are very useful for archival purposes or other scenarios in which you want to back up your Exchange databases, but do not want to disrupt the normal backup schedule. A copy backup performs the same functions of integrity checking and page-zeroing (if enabled) as the normal backup.

### 5.1.3 Incremental and differential backups

The backup process in the case of an incremental or differential backup type is somewhat different. Again, as in the case of a normal backup operation, the truncation point marks the beginning of backup. The current E0*n*.LOG file (*n* being the storage group number or designation) is renamed to E0*nnnnnn*.LOG, and a new generation is started in a new E0*n*.LOG. With the incremental and differential backup types, no database files are copied to tape. It is also worth mentioning that, since the databases are not copied, no page verification or checksumming is performed on the database to ensure integrity (during recovery, the log file records are also checksummed to ensure integrity). This is a notable point when selecting which backup strategy you will use for your Exchange deployment. Since an incremental or differential backup only operates on the log files, if circular logging is enabled, neither incremental nor differential backup operations will be capable of providing complete recovery. Like a normal backup, an incremental backup will delete log files up to the truncation point once they have been backed up. This is the key point of difference between incremental and differential backups. To restore from an incremental backup, you will need your last normal (full) backup set plus any incremental backup sets that have been made since. For Exchange 2000/2003, you must indicate when the last backup set has been restored in order for the ESE database instance to recover the database properly.

Like an incremental backup, the differential backup is also only concerned with transaction log files. The point of difference from an incremental backup is that a differential backup does not delete the log files at the truncation point (i.e., the current checkpoint location). While the start of the backup operation is marked by closing and renaming the current E0*n*.LOG to E0*nnnnnn*.LOG and creating a new E0*n*.LOG generation, the log files are left intact. To restore from a differential backup set, as was the case with an incremental backup, the last normal (full) backup set is required. Next, this is combined with the latest differential backup set. This

is due to the fact that, since the differential backups have left log files intact throughout subsequent backup operations, the latest differential backup contains all log files created since the last normal backup set. As was the case with the incremental backup recovery operation, the last backup set must be indicated for the ESE recovery storage instance to properly recover the database or storage group.

### *Brick-level versus normal backup*

Customers have requested the ability to restore a single message, folder, or mailbox since the earliest days of Exchange. This is not possible using the normal backup API because data is read from the database in 4-KB pages and is written to tape in that manner. This means that the physical structures of the database are backed up devoid of the logical structure that is meaningful to individual mailboxes and public folders. There is no contextual information written along with the data, so a full restore is necessary before the data is reordered into a proper database structure. Several backup software vendors have attempted to provide the necessary features to support single message restore (like what folder it is in, whether it contains attachments, and so on) in their Exchange-compliant backup products. This mode requires that data be written to backup media with all its contextual information intact, so the normal backup API cannot be used. Instead, a connection is made using the MAPI protocol in much the same way as a normal MAPI client, and data is read out in mailbox order. This is referred to as a "brick" backup. A brick restore is one in which a single item (mailbox, folder, or other item) is extracted from the backup media and inserted into the information store. Restoring a single item is much easier with this approach, but backup times are significantly longer due to the requirement to write out additional information—brick backups do not use single-instance storage, and they have to expand the MAPI properties of each message, bloating the backup data. Typically, brick backups take four to five times longer than a normal backup. I do not recommend that you use a brick backup as the basis for your daily backup routine. Instead, if you use a product that supports brick backups, consider using this feature once a week and use a normal backup every other day. Another possibility is to use such a product feature for key personnel in the enterprise. Microsoft may never provide brick-level backup for Exchange, but many third-party products can provide some solution to this problem. With the ability to partition the information store into smaller units of manageability in Exchange 2000/2003 and to recover individual deleted items and mailboxes, brick-level backups may become less important.

### 5.1.4 Individual item recovery

Microsoft partly addressed the issue of single-item recovery with the deleted items retention feature in Exchange 5.5 that has been carried over to Exchange 2000/2003. The most common reason people ask for single items to be restored is that they made a mistake and deleted something important that they should have kept. Deleted-items retention means that items are "soft" deleted initially and then "hard" deleted after a set period has elapsed. Soft deletion means that the item is marked as deleted in the database and is hidden from view. Hard deletion means that the item is permanently removed from the database. During the deleted item retention period (between the time when the item is soft deleted and its hard deletion), it can be recovered and recalled to view using an option on Outlook 97, Outlook 98, Outlook 2000, and Outlook XP, and Outlook 2003 clients (Outlook 8.0.3 onwards) and OWA 2003. If your organization has not implemented this feature, help desk or administration staff must perform recovery. I believe that the deleted-item recovery feature implemented in Microsoft Exchange 5.5 covers most of the cases in which a brick-level restore might have been required. In addition, Exchange 2000/2003 extends this feature by adding the ability to recover individual mailbox that have been deleted as well (called deleted-mailbox retention). Many companies have set item retention periods of between 7 and 14 days and have found that this eliminates the vast majority of requests for item recovery—after all, users typically figure out that they have deleted something important fairly quickly. However, you should be aware that implementing this feature will cause your database to grow. A conservative estimate is that you should expect an individual information store database to grow by between 10% and 15% for a retention period of 14 days. This percentage will vary from organization to organization and will largely depend on the usage pattern of the messaging server. Of course, deleted item retention does not help recover items that have passed out of the retention period; for that case, you will probably want to utilize a dedicated server (called an Exchange recovery server—discussed later in this chapter) or Exchange 2003's Recovery Storage Group (RSG) for deleted item recovery. This dedicated server must have sufficient disk space to hold the database being restored.

### 5.1.5 A word to the wise about off-line backup and restore

When Microsoft developed Exchange Server, choices were made about the architecture and operation of the database engine, and an on-line API for backup and restore operations was developed. This ensured that the server could be operational 7×24 and that backup operations would not cause server outage. Microsoft has strongly educated Exchange implementers about the benefits and necessities of performing on-line backup operations. When calling Microsoft PSS, you will be hard-pressed to find a sympathetic ear if your only means of Exchange Server recovery is an off-line backup. Microsoft has specific reasons for enforcing its recommendations for on-line backups. The main reason is that on-line backups, which utilize the ESE APIs, have awareness of the transactional nature of the Exchange database engine. If you simply treat the Exchange database as a file, no transactional integrity of the database is maintained. An on-line backup not only will back up the database, but will also back up log files and provide log file truncation. In addition, on-line backups for Exchange provide management on the restore side as well. When restoring backup sets created using on-line methods, the database engine is able to provide recovery up to the very last transaction recorded by utilizing the log files. All around, on-line backups are a preferable method. I recommend against the practice of off-line methods unless they are used as a mechanism for periodic archival or your databases.

Unfortunately, many have still chosen to use off-line methods to backup their Exchange servers. In an off-line scenario, all Exchange services are shut down in order to perform the backup or restore operation. Backup operations simply treat the Exchange information store databases as individual files in the file system. EDB and STM files would be backed up just like any other file on the server. This can be very problematic for several reasons. First, since an off-line backup method does not utilize the ESE API, no integrity checking of the database is performed (unless you do it manually with ESEUTIL, ESEFILE, or ISINTEG). Remember, using an on-line method and performing a normal (or full) backup will verify each and every page of the database during the backup procedure. Another problem with the off-line backup method for Exchange is that the operator must take responsibility for managing database transaction log files. The transaction log files are required for successful recovery of the database up to the point of the last transaction that occurred. Suppose, for example, you needed to recover your Exchange 2000/2003 server (either an individual database or an entire storage group) from an off-line backup. If your backup was per-

formed at midnight (12:00 A.M.), you would have a consistent copy of the data for that point in time, assuming the services were stopped or an individual database (store) was dismounted. In our example scenario, suppose a failure condition were to occur at 2:00 P.M. (14 hours later) and you were forced to recover the database. If the failure were mild enough (such as database corruption), you would be able to restore the database files (EDB+STM), but would not be able to play through the existing log files that have accumulated (representing real user data) since the database files were backed up. The greatest potential for error with off-line backup methods occurs during the restore of an off-line database—the database engine does not automatically play through the log files as it would normally do in the case of an on-line backup. There are certainly ways to accomplish the task, but this must be accomplished through manual log file management and the use of scripts and "hacks" that attempt to mimic the on-line recovery operations. Realistically, there is no point to this exercise since that is the purpose for which Microsoft designed the on-line backup APIs.

While you may have been able to devise methods of safe recovery using off-line methods in previous versions of Exchange, Exchange 2000/2003 will make it virtually impossible to enjoy success using off-line methods. In previous versions of Exchange (prior to Exchange 2000), the fact that there was a single private and public information store (PUB.EDB and PRIV.EDB) made it possible (although still prone to error) to implement successful disaster-recovery procedures based on off-line methods. There were only two database files and one ESE database engine instance (storage group) in previous versions of Exchange. Consider Exchange 2000/2003, however, in which you can configure up to four storage groups (even more in later releases) on a server—each with five databases. Further consider the fact that the database is now two files (*.EDB and *.STM) instead of one. Putting all of this together, you can see the difficulty in implementing a backup strategy based on off-line methods for Exchange 2000 on a server with multiple storage groups and databases. Here is my bottom line for this discussion: I hope I have convinced you to stay away from an off-line approach and only to use this method for periodic archival or as an added measure of protection that is complementary to on-line, API-based backups. Off-line backups are, arguably, still useful as a last-ditch recovery tool before, for example, a major hardware upgrade—do a full backup to truncate the logs, shut down or dismount, and then do the full off-line backup. Whatever your preference, ensure that you understand the pitfalls of off-line backups with Exchange Server.

## 5.2 In depth: Exchange 2003 backup operation

One of the keys to a solid disaster-recovery strategy for Exchange is thorough understanding of how Exchange's database engine performs backup operations. While there is a tremendous amount of good information on the basics of how Exchange backup operations work, much of this information is inaccurate or not detailed enough for careful planning of your Exchange Server recovery. In this section, we will look closely at how Exchange backup works at the database engine and API level. A similar discussion on restore operations is included in the next section. Also, note that while this discussion is centered around Exchange 2000/2003, previous versions work similarly, even though several minor variances exist between different version and service pack levels of Exchange Server.

### 5.2.1 Starting with the basics

The Exchange database engine is known as the Extensible Storage Engine or ESE. ESE is a transacted storage engine that leverages write-ahead logging technology to perform transacted operations against a database or databases that are structured in a Microsoft-implemented B+Tree format. With a write-ahead logging database engine, write operations to the database are first written to the transaction logs and then to database pages cached in memory (called IS buffers). These write operations are later written to disk asynchronously in an optimal manner designed to batch these operations such that disk I/O performance is maximized. As transactions are committed to the database on disk, a database checkpoint (maintained in the EDB.CHK file) is advanced to reference the current point in the current transaction log file of the last transaction that was committed to the database. ESE uses discrete log files that are 5 MB in size and generationally sequenced. As transactions are written to the log files, once the log file reaches 5 MB in size, it is closed and a new log file is open (the current log file is always E0*n*.LOG—where *n* signifies the storage group instance to which the log file belongs). This implementation provides high performance and maximum reliability expected from a transacted database engine.

ESE was also designed to allow the databases to be backed up while the server was on-line servicing users. These means that Microsoft had to devise a method where the databases could be stored to a backup set while they are mounted and transactions are being committed. For this purpose, the ESE backup API was designed. In Exchange 2000/2003 the backup API is implemented in two DLLs—ESEBACK2.DLL and ESEBCLI2.DLL—that allow backup applications to interface with the database engine to per-

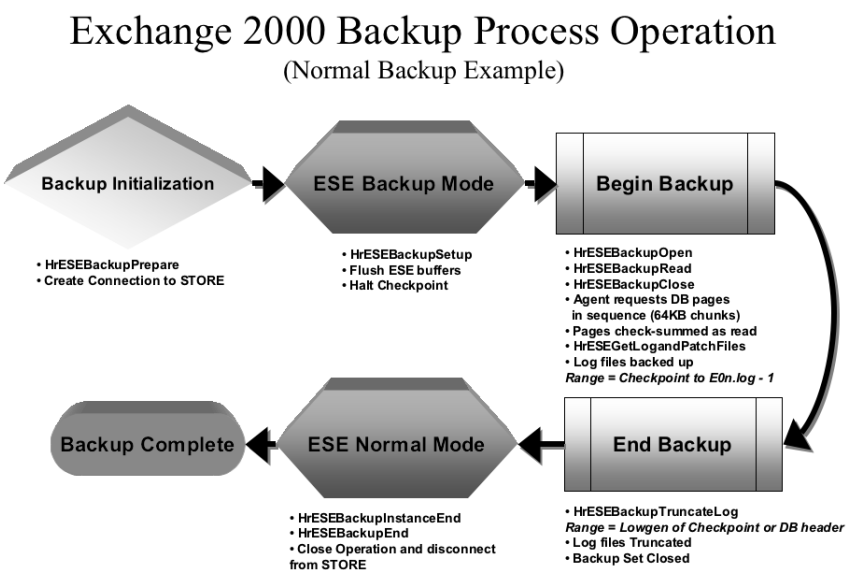


**Figure 5.1**
*Exchange 2003 backup operation.*

form on-line backup and restore operations for Exchange. In the following discussions, the ESE backup API calls will be referenced as we step through the backup operation for Exchange Server. As an interesting side note, Windows AD is also based on ESE, and backup operations and API calls used for backup and restore are very similar to those discussed next.

### 5.2.2 Initializing the backup operation

At the start of Exchange's backup operation (see Figure 5.1), several important operations occur. Two important API calls are initiated by the backup application (this could be Windows/NT Backup or a third-party backup product such as Legato, Veritas, or CA) at this point to begin a backup operation. *HrESEBackupPrepare* is called to establish an RPC connection to the information store process (STORE.EXE). Also, the *HrESEBackupSetup* API call is made and will specify which storage groups (ESE instances) will be involved in the backup operation. ESE responds to these API calls by performing several operations. First, when a full backup operation is initialized, ESE begins by flushing all dirty pages in the ESE cache (IS buffers) to disk and halting the checkpoint. The checkpoint will not advance until the backup operation is complete. If the backup is a not a full backup (i.e., differential, incremental, or copy), the checkpoint is allowed to advance, since the backup operation will not be touching the databases. Next, ESE will create patch files for each of the databases backed up. Patch files are used in special circumstances (transactions that require page splits and/or merges)

during backup operations to ensure database integrity. If you are running Exchange 2000 SP2, patch files are not used, since Microsoft has figured out how to avoid this requirement in the SP2 release.

### 5.2.3 Copying the databases

The next step in the process involves backing up the database files. ESE gets the list of storage groups and databases to be backed up from the backup application via the backup APIs (*HrESEBackupSetup*). The backup application makes three API calls to backup the databases. *HrESEBackupOpenFile* opens the database file for reading, *HrESEBackupReadFile* reads the database, and *HrESEBackupCloseFile* closes the database when all the pages have been read. These databases are not simply copied to the backup set because they are open files (remember Exchange backups allow the databases to be back up on-line). Instead, ESE begins to send the backup application 64-KB chunks (16 4-KB pages at a time) of database pages in sequential order. This is also the crucial step where each page is checksummed and an error results in the backup operation terminating with a –1018 (the *HrESEBackupReadFile* is the API call that initiates checksumming operations for database, log, and patch files during the backup operation). ESE will continue this process until all pages for each database are sent to the backup application. Also remember that Exchange 2000/2003 supports two database files—the Property Store (EDB) and the Streaming Store (STM). Both database files are read page-by-page by ESE and sent to the backup application to store as part of the backup set. ESE will perform these operations for each database being backed up.

### 5.2.4 Backing up the transaction logs

During a full backup operation, the transaction logs must be stored to the backup set next. Remember from above that the checkpoint is halted at the beginning of backup (for a full backup). Even though the checkpoint is halted, transactions continue to be written to the log files, and dirty pages from the database cache in memory are flushed to disk during the backup operation. Note that with Exchange 2000 SP2 and later, dirty pages that cause patching operations are not flushed; this is how Microsoft was able to get rid of the patch files in SP2. In order to back up the log files, the backup application requests a list of log files (and patch files if applicable) from ESE (via the *HrESEBackupGetLogAndPatchFiles* API call). When ESE receives this call, it closes the current log file as the next log generation and opens a new E0*n*.log. In the case of a full backup, ESE then returns a list of log files

to the backup application that starts with the current log generation where the checkpoint is halted and ends with the generation of the log file that was just closed (E0*n*.LOG – 1). In the case of an incremental or differential backup, ESE returns a list beginning with the oldest generation on disk up to the most recent log file closed (E0n.LOG – 1). During incremental, differential, and copy backup operations, only log files are backed up and the checkpoint is not halted. Using this list of files, the backup application can open file handles to the logs and copy them to the backup set (using the *HrESEBackupOpenFile, HrESEBackupReadFile*, and *HrESEBackupCloseFile*). During these operations, ESE also ensures that no log generation is missing from the sequence passed to the backup application. This operation will continue until all relevant log and patch files have been written to the backup set by the backup application.

### 5.2.5 Truncating the logs

Since the log files have been stored to a backup set, they are not needed on disk. ESE will then truncate the log files during full and incremental backup operations. When ESE receives the call from the backup application to truncate the logs (*HrESEBackupTruncateLog* API call), it will truncate the log files on disk. Which log files ESE truncates is determined by the lowest generation of (1) the checkpoint log file generation, or (2) the log generation listed in the database header (you can view this using the ESEUTIL program with the /MH parameter) for the current full backup.

### 5.2.6 Cleaning up

Once the log files are truncated, the backup operation is complete and the backup set is closed. At this point, ESE can return to normal operations and allow the checkpoint to advance forward. To complete the backup operation, the backup application calls *HrESEBackupInstanceEnd* to end operations for that instance of backup. At this point ESE is able to allow the checkpoint for the storage group instance being backed up to advance and normal database operations to recommence. Finally, the backup application calls *HrESEBackupEnd* to disconnect from the information store process allowing it to return to normal operations.

### 5.2.7 Implications for backup planning and procedures

Traditional best practices for backup call for a combination of full and incremental or differential backups to ensure data recoverability. However,

within the Exchange space, the de facto backup best practice is daily full backups with no incremental or differential backups (see Table 5.1 for the different backup types supported by Exchange and their impact on database and log files). For the most part, this strategy has served Exchange administrators well. However, in the event that a particular backup tape media goes bad, there is a potential for data loss.

Let's look at this problem closer. Suppose you are performing daily full backups for your Exchange server. On Day 3 of the rotation, your Exchange database becomes corrupt and you must restore from backup. However, when you go to use the backup tape from Day 2, you discover that the media is damaged, rendering the backup unusable. In this scenario, you would be forced to go to the tape and backup set for Day 1. However, if you use this backup set, it contains a valid database and log files up to the point in time that the backup was performed on Day 1. On the subsequent day (Day 2), the full backup operations have truncated the log files residing on disk that occurred after Day 1. The consequences of this backup strategy provide for recovery of your Exchange data up to the point of backup on Day 1 (in the event that the Day 2 backup set is bad). However, since subsequent full backups have been performed on Days 2 and 3 and log files have been truncated, there are gaps in the generational sequence required to recover to Day 3 using the Day 1 backup set (Day 2's backup is bad). At this point, the Exchange administrator will find himself or herself in a position where he or she can recover to Day 1 and has some log files on disk that were created since the Day 2 backup. There are gaps in the sequence. For example, if the database and log files 001 and 002 were recovered from the Day 1 backup set, Day 2 stored the database and logs 003 and 004, and logs 005 and 006 are on disk on Day 3, the loss of the Day 2 backup set results in a gap in the generational sequence of two log files. (Logs 001, 002, 005, and 006 are available. Logs 003 and 004 are not available.) This means that recovery is only possible to the point of log file 002 since ESE will not allow recovery to proceed beyond the gap in the sequence.

The above scenario exposes a potential flaw in the daily full backup strategy that has become the de facto standard for Exchange Server disaster recovery. To work around this potential problem, we must look at two things. First, we must address the issue of bad media. If frequent media failures are plaguing your disaster-recovery efforts, it is important to take steps to address this issue outside of Exchange management and look at the causes and factors that are contributing to this problem. You may need to look to your hardware vendor or to your procedures and processes to find the cause of frequent media failures. However, even with the most proactive

approach, top-notch hardware, and bulletproof best practices, media failures may still occur. If your SLAs dictate the up-to-the-minute recovery of Exchange data, you may need to consider enhancing your backup strategy to protect from media failures and other issues that would result in missing sequences of log file generations.

One approach to the problem of missing log file sequences as a result of missing or bad backup sets is to augment daily full backups with differential backups at the mid-point during the day. For example, if you perform a full backup at 12:00 A.M., you would perform a differential backup at 12:00 P.M. This would increase the number of log files available on your backup media and potentially lessen the possibility of missing log files in the event of a media failure. However, there would be no guarantees. Another approach would be to perform a copy backup (a copy backup copies the databases and the log files, but does not truncate the log files) at 12:00 A.M. followed by an incremental backup at 12:00 P.M. This approach would preserve all log files during the copy backup and truncate the logs during the 12:00 P.M. incremental backup. Alternately, the 12:00 P.M. incremental backup could be a differential backup, which would not truncate the log files. With this alternative (copy + differential), however, the logs would never be truncated. It is up to the Exchange administrator to understand the implications of the type of backup and schedule chosen along with the potential hazard of bad or lost media. It is important that disaster recovery plans provide the right backup types and schedule to meet the established SLAs.

### 5.2.8 The poor, misunderstood patch file

Although no longer pertinent to Exchange Server 2003 (the need and use of patch files by ESE was eliminated with Exchange 2000 SP2), but important to versions prior, it is important not to avoid skipping this topic for those who are still running versions of Exchange Server prior to Exchange 2000 SP2. Patch files (*.PAT) are special-purpose files used by the Exchange Server database engine on limited occasions. During on-line backup operations, the database file is written out 64 KB at a time in a sequential fashion. In other words, the 4-KB database pages are written sequentially in groups of 16 pages at a time (16 × 4 KB = 64 KB). Since Exchange Server allows backup operations to be performed while the server is running and users are connected, it is possible that changes to pages in the already written (or backed up) section of the database will not be on tape. This case, however, is handled by Exchange since these changes will be stored in the transaction logs, which will be copied to the backup as well. Another case (illustrated in Figure 5.2) exists, however, in which a

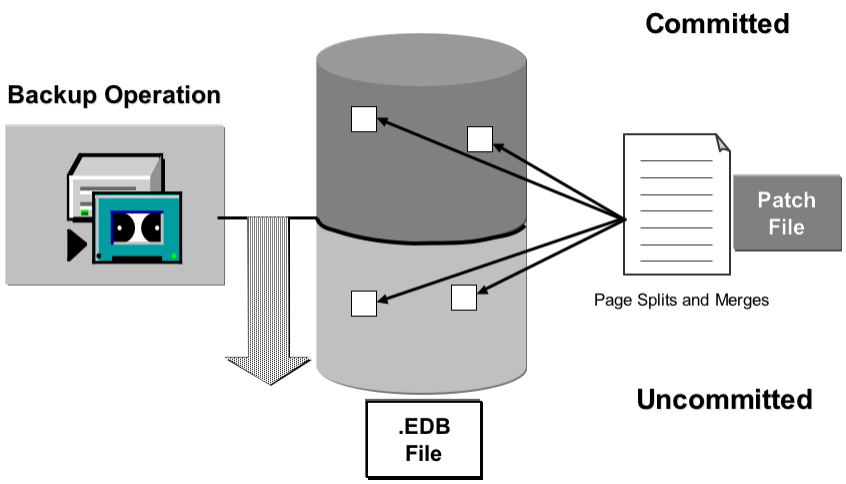


**Figure 5.2** *ESE's patching operation during backup (Exchange 2000 SP1 and earlier).*

single 4-KB page that resides in the portion of the property store (EDB file) already copied in the backup becomes full (i.e., all 4 KB are used) as a result of transactions that have occurred since the backup was begun. In this case, the page must be split and a new page allocated in the B-Tree structure of the database (remember that the streaming store is not a B-Tree database structure and therefore does not require patch file measures). A similar situation occurs if pages must be merged. If the pages are involved in a split or merge operation occurring across the backup committed/uncommitted boundary, they cannot be handled by the transaction logs alone. Therefore, these operations must be written to a separate location in order for the property store database to remain consistent when it is restored from backup. Updates to pages in the portion of the database that has already been copied in the backup are stored in the patch file. There is a patch file created during on-line backup for each database. During recovery, the database engine will update the database with any pages that are stored in the patch file. Starting with Exchange 2000, Microsoft implemented the same level of integrity checking of each page in the patch file as is available for the property store. Each page in the patch file is verified using the checksum stored in the page header. Patch files are an important concept to understand concerning disaster recovery for Exchange server running versions prior to Exchange 2000 SP2. This entire discussion of patch files and their use is purely academic since the important point is that ESE takes care of everything for you.

So how did Microsoft eliminate the need for patch files in Exchange 2000 SP2? Surprisingly, it was rather simple. Microsoft developers determined that the reasoning behind the original design requirement was a bit

flawed. The patch file was needed because of the possibility that database page modifications that resulted in pages being merged or split could not be handled well during backup operations because, if the location (either before or after the current backup location) of these merged or split, pages would be unknown. They decided to invent a mechanism to handle this—but was it the best way to solve the problem? After many years, the JET/ESE developers came to the conclusion that they could handle the page merge/split scenario and effectively accomplish the same thing as the patch file by simply not advancing the ESE checkpoint and not flushing dirty database pages to disk during backup operations. In this manner, ESE is able to handle page splits and merges during a backup operation. This little thing is a big win since the elimination of the patch file just means there is one less thing to go wrong during hard recovery operations. I wish I could have seen the light bulb go on and heard the resulting "Duh!" when ESE developers figured this one out.

### 5.2.9 ESE's Page Zeroing feature

During an on-line normal backup, another important feature is available. This is ESE' Page Zeroing feature. Page zeroing is the ability to "zero" each deleted page in the database. This is typically implemented as a security measure in which pages of the database that have been logically deleted (i.e., a user deletes a mail message, and it has been aged out of the deleted items cache) are overwritten to ensure that the data is truly deleted and cannot be recovered by would-be spies, hackers, or U.S Department of Justice staff members with too much time on their hands. ESE Page Zeroing became available in Exchange 5.5 Service Pack 2 (released in December 1998) and is an important feature for Exchange deployments desiring the highest levels of data security. ESE Page Zeroing for Exchange 2000/2003 is enabled via an Exchange System Manager option (shown in Figure 5.3) and is available on a per-storage-group level.

Microsoft chose to implement page zeroing as part of the backup process. More specifically, since this operation must touch the database and has nothing to do with the log files, page zeroing is done as part of an on-line normal backup operation. During a normal backup (when ESE Page Zeroing is enabled), as the database engine checks the integrity and copies pages to backup media, it will also zero delete pages in the database by writing a specific byte pattern to the page. Technically, the pages are not zeroed, but contain a byte pattern known to the database as an empty page with no data. Regardless of the technicality, each deleted page no longer contains the original data and is safe from potential security threats. As you might

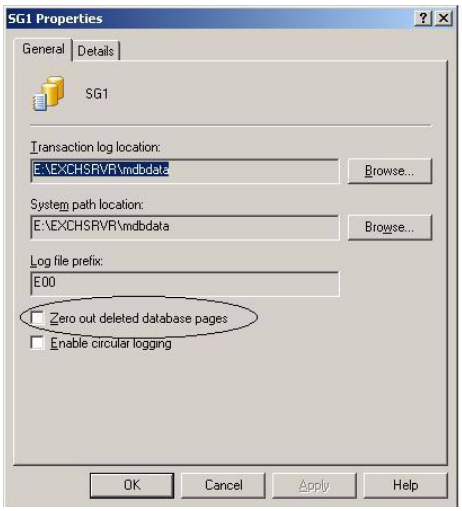


**Figure 5.3** *Enabling ESE Page Zeroing in Exchange system manager.*

guess, ESE Page Zeroing can be a resource-intensive process for your Exchange server. Additional overhead beyond what the backup operation already consumes is required to perform the zeroing operation. When ESE Page Zeroing is enabled, the first normal backup operation will be the most resource intensive because all deleted pages in each database are zeroed. Once the initial operation has been completed by the database engine, only newly deleted pages will need to be zeroed on subsequent normal backups. If you select a strategy of only one normal backup per cycle, all page zeroing operations are only performed at that time. If you are concerned about the additional overhead of ESE Page Zeroing, I recommend that you perform the initial backup (doing a backup to disk is extremely fast) of your databases at a time of low user activity (which may be the most typical case anyway). Subsequent normal backups should not be particularly resource intensive. Also, if you have a large occurrence of deletion, such as when a large number of mailboxes or public folders are moved or deleted, I recommend the same procedure as in the case of the initial backup previously discussed. Overall, page zeroing should not be a significant performance problem on your Exchange server. However, following these recommendations may make your life a bit easier in the long run.

The backup operation for Exchange (ESE) is very intricate, and it can be confusing. However, it is crucial that Exchange administrators understand how this important operation works. As you can see, how you structure your backups and the combination of full and incremental or differential backups you employ will determine the recoverability of your valuable

Exchange data. As an exercise, you should always plan on testing backup operations in a lab environment in order to understand this process better.

## 5.3 In depth: Exchange 2003 restore operation

One of the keys to a solid disaster-recovery strategy for Exchange is a thorough understanding of how Exchange's database engine performs recovery operations. While there is a tremendous amount of information on the basics of how Exchange restore operations work, much of this information is inaccurate or not detailed enough for careful planning of your Exchange server recovery. In this section, I will take a closer look at how Exchange restore operations work at the database engine and API-level. Also, note that while this discussion is centered around Exchange 2000/2003, previous versions work similarly, even though several minor variances exist between different version and service pack levels of Exchange Server.

Obviously, a restore operation is an administrator-initiated activity. This means that before a database can be restored, two important things must happen. First, the database needs to be dismounted using the Exchange System Manager (ESM) MMC snap-in or some other means (such as a script that does this via Windows Management Instrumentation, WMI, using Exchange's Collaboration Data Objects for Exchange Management, or CDOEXM). In addition, the database must be configured (via ESM and so forth) to allow it to be overwritten by a restore operation; by default, the database cannot be overwritten—and important safety measure. Once these preparatory tasks are completed, the database is ready to be recovered.

### 5.3.1 Beginning the restore and copying the databases

By reading the beginning of the backup set, the backup application gets a list of databases that are available. Once the administrator selects the correct database to be recovered, the backup application begins by making ESE API calls to start the restore. First, the backup application asks the administrator for inputs, such as the server to restore to, the location, and a temporary directory for the log, patch (if applicable), and *restore.env* files (see Figure 5.4). The backup application makes the *HrESERestoreOpen* call to gather this information and then the *HrESERestoreAddDatabase* call once for each database that is going to be restored. At this point, ESE leaves it to the backup application to restore the needed database files to the proper locations. ESE does not get involved much in copying the database files to disk from the backup set. ESE allows the backup application to make Win32 file system calls directly to the operating system and copy the files.

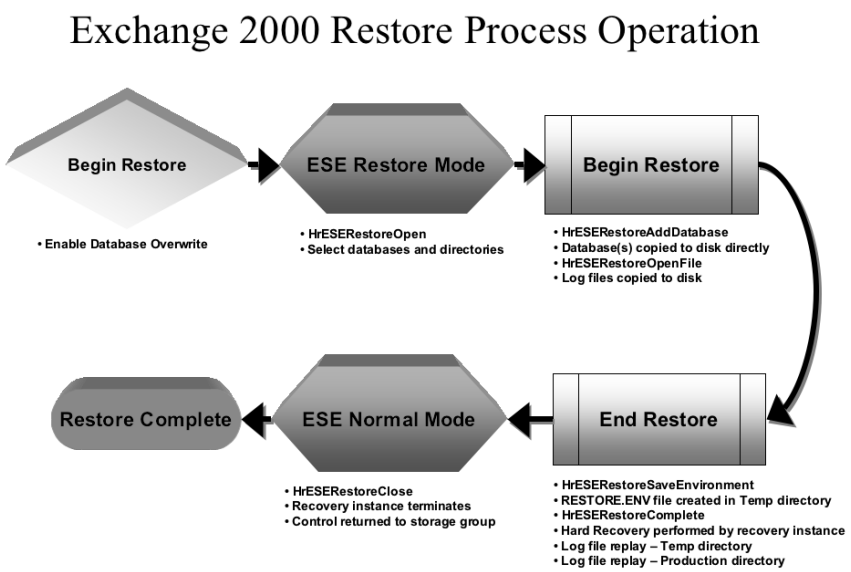


**Figure 5.4**
*Exchange 2003's restore operation.*

The reason for ESE's lack of involvement is based on the reasoning that the database files have already been checksummed when they were backed up, and if the backup set is complete, the databases should be intact. There is no reason for ESE to check the database integrity when it is being restored. Since the databases being restored are dismounted (i.e., not open files), it is much simpler and faster to have the backup application copy these files directly to disk.

### 5.3.2 Restore the log and patch files

Once again, at this point in the operation, the backup application does not need the help of ESE for a while. The backup application simply calls *HrESERestoreOpenFile* (ESE does use this call to create the metadata needed for the *restore.env* file) for each log or patch file to be restored and copies these files to the temporary directory specified at the start of backup by the administrator (note that patch files are no longer used after Exchange 2000 SP2). The log and patch files are copied to the temporary directory because of the requirement to keep them separate from the log files in the production log file directory. This prevents naming conflicts or overlaps between log files in the backup set and the log files on disk. The best course is to copy the log files from the backup set to the temporary directory.

### 5.3.3 The restore environment

Once all log and patch files have been recovered from the backup set, the backup application makes a call that was first introduced in Exchange 2000. If you recall previous versions of Exchange, the Restore_In_Progress key is created in the system registry during a recovery operation. This key contains information about the recovery operation in progress for the database engine (of which there is only one instance in Exchange 5.5 and previous versions). In Exchange 2000/2003, however, there are multiple instances of the database engine (storage groups), as well as concurrent recovery capabilities, and a single key in the registry will not suffice. This led to the advent of the RESTORE.ENV file (which stands for "Restore Environment"). Because a single registry key won't do in the case of concurrent recovery, the RESTORE.ENV file is created during recovery by the backup application when it calls *HrESERestoreSaveEnvironment*. ESE returns the necessary information (similar to that which was stored in the Restore_In_Progress key in previous versions of Exchange) to the backup application, and the RESTORE.ENV file is saved in the temporary location with the log and patch files. You can view the contents of the RESTORE.ENV file using the ESEUTIL program with the /CM switch. The log, patch, and RESTORE.ENV files will be used to complete the recovery operation in the next step.

### 5.3.4 Completing the restore and running hard recovery

Once all the backup sets that are necessary for the current recovery operation are copied, the backup application is ready to complete and terminate its activities and turn control back to the store process. The backup application calls *HrESERestoreComplete* and *HrESERestoreClose* to signal ESE that it is time to take over. At this point, you would think that the storage group that owns the database being recovered would take over and complete the recovery operation. However, this is not the case. The store process instantiates a separate ESE storage group specifically for the purpose and duration of completing the recovery operation. This recovery storage group then takes over and performs the hard recovery operation. Hard recovery is the process of applying patch files to the database, replaying log files from the backup set (located in the temporary directory), and replaying log files from the production log file directory. Once hard recovery completes successfully, the database is ready to be mounted and made available for users. It is important to note here that hard recovery can be performed manually using

the ESEUTIL program (/CC switch) if you are performing simultaneous restores or hard recovery did not complete automatically for some reason (like if you forgot to check the "last backup set" checkbox after restoring the last set of log files). Once automatic hard recovery is completed, the recovery instance deletes the files in the temporary directory, terminates, and turns control over to the storage group that owns the database for normal operations. The owning storage group can then mount the database and begin to apply user transactions.

Too often, Exchange disaster-recovery operations (backup and restore) are trivialized. However, it is of paramount importance that we as Exchange administrators understand exactly how these operations work and the impact they have on our ability to meet service level agreements for our Exchange deployments. This in-depth drill down into the internals of Exchange's ESE and how it exposes these capabilities and performs these operations will serve you in your quest to provide the highest levels of availability for Exchange.

## 5.4 Supercharged disaster recovery for Exchange Server 2003

The increasing importance of messaging and collaboration as a business-critical service has left Exchange administrators and implementers looking for new ways to supercharge the recoverability and availability of their Exchange servers. In addition, hardware and software vendors are now supporting technologies that enable more rapid recovery of server data and applications. One specific enabling technology is volume cloning or volume snapshots. This technology is available in a wide variety of implementations and packages that include both hardware and software solutions and the packages range from full-blown "snapshot manager" products for Exchange to integration kits available to customers and integrators who desire to customize their own solution for Exchange recovery. In the past, regardless of how these technologies were delivered, Microsoft provided no native support for them in either the operating system or Exchange Server. However, with the advent of Windows Server 2003 and Exchange Server 2003, this technology is now available natively to Exchange administrators. In this section, I wanted to give you a taste of the power of these technologies and discuss the Windows VSS and what it means for Exchange Server disaster recovery and availability.

### 5.4.1 Snap/clone technology overview

Before taking another step, I should briefly visit the technologies that enable my discussions. Snapshot and cloning technology is not new to the computer industry, but it is relatively new to the Windows platform. This is partially due to the slow adoption rate of technologies like Storage Area Networks (SANs) and Network Attached Storage (NAS) in the Windows space. Cloning and snapshot technologies provide business continuance volumes (BCVs), thereby providing a much needed capability to the Windows space. BCVs are marketing terminology for snapshots and clones. Simply put, clones and snapshots provide a mechanism for data duplication and point-in-time copies that allows business continuance—thus, the term BCV. On the surface, clones and snapshots may appear to be the same technology. However, they are quite different in actual technical implementation.

#### *Snapshots*

A snapshot is a metadata mapping to volume blocks that represent the "picture" (thus, the word snapshot) of the data at the time the snapshot was created. This means that if you create a snapshot of your Exchange database volume, the snapshot represents the list of the blocks on disk that were used to store your Exchange database (and any other files on the volume) at the time the snapshot was created. Therefore, once a snapshot has been created for a volume, these original volume blocks must be maintained in order for the snapshot to stay intact; snapshot data cannot be moved to other blocks, although it is possible to add data in new blocks that are not on the snapshot list. This requirement forces changes to blocks on the volume to be copied out (snapshots are also known as copy-on-write snapshots) to another location in the storage pool. From an Exchange viewpoint, this means that a change to a page in the Exchange database will require copy-out operations of changed blocks if a snapshot has been created for the volume on which Exchange data resides. On a volume for which a snapshot has been created, when a block of data is changed, the block is actually copied out and another block is allocated from free volume pool space. In this manner, the contents of the original subset of volume blocks that represent the snapshot are preserved. Thus, after snapshot creation, the production data is actually a combination of original unchanged blocks (that are still part of the snapshot) and the changed (copied-out) blocks of data. The snapshot contains the original set of blocks that represent the data view at the time the snapshot was created. From this description, it is obvious that a snapshot is not really a complete redundant copy of the data, but a representation of the data at a point in time. Part of the snapshot is still part of

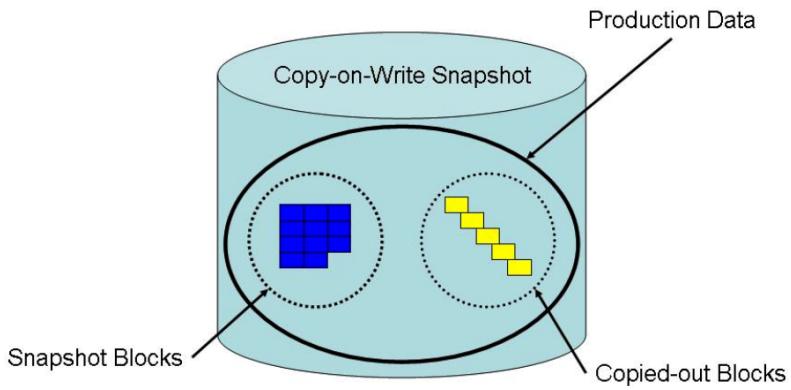


**Figure 5.5** *Snapshot technology illustrated.*

the production data set and part represents out-of-date data (shown in Figure 5.5). Due to the nature of snapshots, creation is relatively quick and simple—the volume block mapping is simply created and the snapshot exists. Based on the particular characteristics of snapshot technology (it is not a complete redundant copy of the data and is subject to disk failures), snapshots are somewhat less desirable than clones.

### *Clones*

Like snapshots, clones are not a recent development. Disk clones come from a foundation of RAID technology—specifically RAID0+1. A clone is, in actuality, just an additional member of a RAID0+1 mirrorset. Typically, we think of mirrored volumes as only having two members. The Windows Logical Disk Manager (LDM) provides two-way mirroring, as do most hardware RAID controllers. However, some software volume managers and high-end storage controllers allow for *N*-way mirrors, where *N* may range from 3 to 32. For example, if you have a RAID0+1 set with 3 disks mirrored to 3 disks, you have a two-member RAID0+1 set. By adding another 3 disks to the existing RAID0+1 set, you would create a three-member mirrorset (a triple mirror). Additional members could be added to the mirrorset as well. By creating multimember mirrorsets and separating members from the set, you gain the ability to create point-in-time clones of the mirrored volume. Unlike snapshots, a clone is a complete standalone copy of the data at a particular point in time.

To create a clone, one or more members of the RAID0+1 mirrorset is simply spilt off from the production set. The result is a production mirrorset that supports the application (two-member RAID0+1 array) and one or more clones (each containing a copy of your data) that have been split off from the production data (as shown in Figure 5.6). Clones can then be used

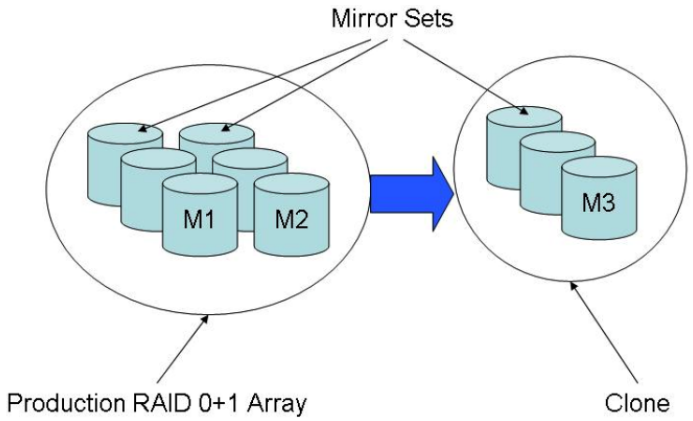


**Figure 5.6**
*Clone technology illustrated.*

in the event of data corruption or loss to recover system data by replacing the production copy with one of the cloned copies. Because clones are a complete redundant copy of the data, they are extremely useful as rapid recovery mechanisms.

### 5.4.2 Building the foundation: Volume Shadow Copy Service

While the clone and snapshot technologies have been somewhat available in the Windows space, Windows and applications have not been able to take full advantage of them. This is mainly due to the lack of native support in the Windows operating system and the inability of applications like Exchange Server to function with such technologies. Third-party hardware and software developers have implemented snapshot and clone technology solutions with little of or no exposure or integration with the operating system and applications. This has lead to the current status quo of varied and noninteroperable solutions from hardware and software vendors that are not supported by Microsoft. While Microsoft has acknowledged that these technologies are available, they have limited support and put the primary support responsibility on the vendors of these solutions. The storage technology investments made by Microsoft in Windows Server 2003 are substantial and VSS is one of those investments. *Shadow copy* is the term that Microsoft uses to describe the snapshot or clone technology discussed above. VSS provides a framework that makes snapshot and clone technologies available to applications and provides some operating system–level support for the synchronization and coordination required to implement them. These services can be used by Windows Server 2003 components (including AD and the Windows Certificate Server), Microsoft and third-party

applications, and third-party backup, data integrity, and SAN product build solutions that leverage snapshot or clone technology. The Windows VSS has three primary goals: (1) to provide application synchronization so that backup programs do not have to be intimately aware of how a particular application stores or recovers its data; (2) to provide a way for tools and utilities to discover and enumerate shadow copies; and (3) to provide a framework where hardware and software vendors can plug-in interoperable shadow copy providers. With these goals in mind, Windows Server 2003 delivers a robust architecture that enables a hardware vendor to supply a shadow copy creation component (called a provider), an application developer to expose shadow copy "packages" called writers that provide XML-based metadata to VSS, and backup vendors that can build applications (called requestors) that can initiate backup and restore operations that leverage these components on a common infrastructure. Figure 5.7 illustrates the VSS architecture included with Windows Server 2003.

### *VSS providers*

VSS exposes APIs that enable vendors to VSS-enable their solutions. In order for a particular vendor's snapshot/clone technology to function within the VSS framework, each vendor must develop a VSS provider. Providers are the components that manage volumes and create clones and snapshots per a

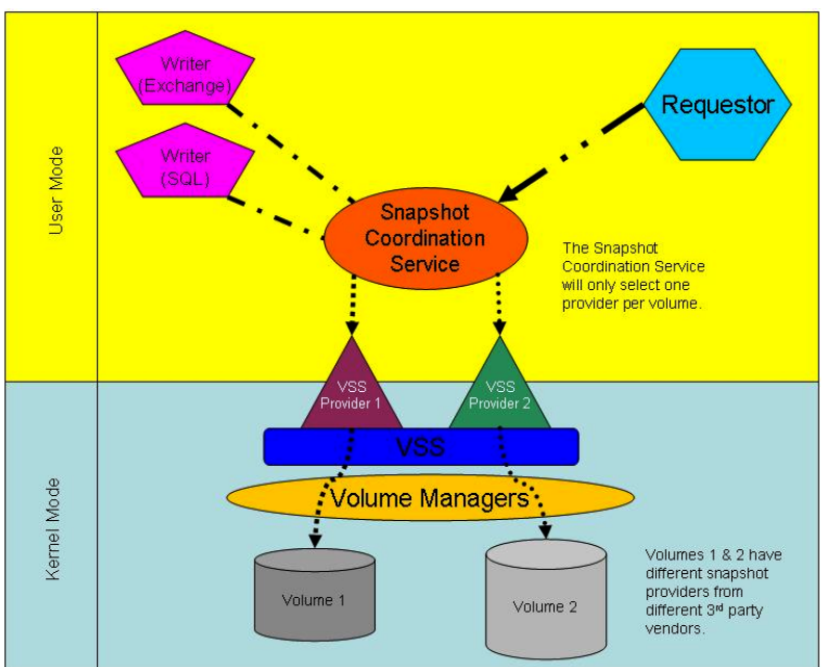


**Figure 5.7** *Windows Volume Shadow Copy Services architecture.*

specific vendor's technology and implementation—think of the provider as the agent that actually writes the shadow copy data on a particular storage platform. Typically, a provider is a process (some kernel-mode and user-mode code) that persists data about a physical shadow copy in order for that shadow copy to be exposed to the operating system and/or applications. Providers must be built regardless of whether the vendor's solution is hardware based or software based. In the case of a software-based provider, the implementation is usually a user-mode process coupled with a kernel-mode device driver. Both types of solutions (hardware and software) and the implementation details of the provider are left to the discretion of the vendor as long as they follow the implementation rules of the VSS framework (this is what makes VSS supportable from a Microsoft perspective). Windows Server 2003 includes a software-based shadow copy provider (implemented as a copy-on-write software snapshot) as part of the operating system; various other vendors have written providers that make their storage products compatible with VSS.

### *VSS writers*

The most important player in the VSS framework is arguably the application. The application must carefully expose recovery "packages" that are specific to an application's technology, implementation, and disaster-recovery requirements and constraints. For example, since Exchange Server is a transacted database engine, it will have requirements that are unique when compared even with applications similar in nature (such as SQL Server or Oracle). VSS writers are code and data that is embedded in applications and components of those applications to enable VSS compatibility. Application writers respond to the shadow copy interface to ensure data integrity and consistency during shadow copy operations. Writers respond to requestors (via the VSS interface) by supplying writer metadata that includes the details of what is required to perform shadow copy operations for the specific application. When a requestor asks for a shadow copy, the writer will prepare its data to be copied, normally by freezing incoming write requests and flushing any cached data that has not been written to disk. After those preparations are complete, the writer signals the VSS framework that it is safe to copy the application data; after the copy is finished, VSS notifies the writer that it is safe to resume normal I/O operations. The goal of this implementation is to ensure that no writes occur on the volume during shadow copy operations (when the shadow copy is created). A backup operation performed using VSS is a systematic and well-orchestrated process that involves the interaction of each of the components

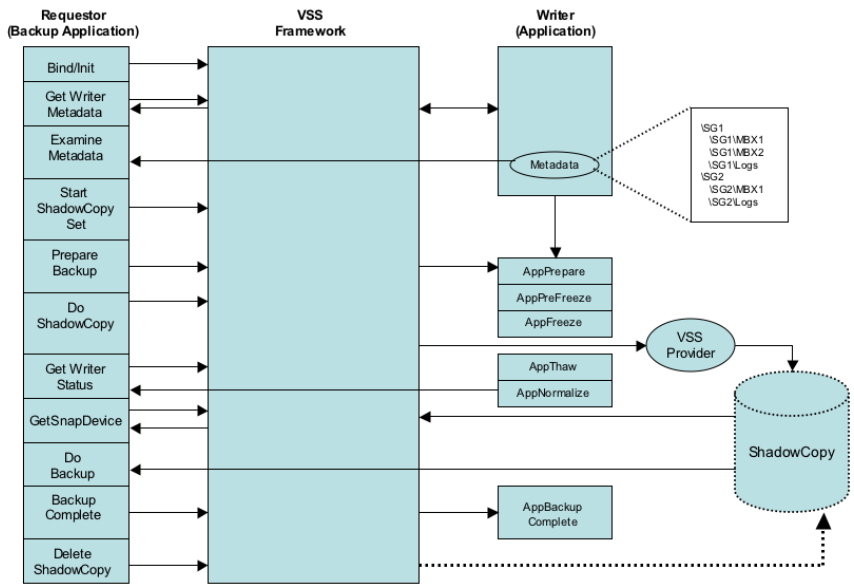


**Figure 5.8** *Typical VSS Shadow Copy operation.*

in the VSS framework. Figure 5.8 provides a generalized flow and interaction diagram of the backup operation using VSS technology.

### *VSS requestors*

Backup and disaster-recovery solution vendors participate in the VSS framework by developing their applications to make use of the VSS architecture, APIs, and implementation rules. These vendors must develop VSS requestors. A requestor is a process or application (automated or GUI-based) that requests that one or more shadow copy sets be taken from one or more volumes. The requestor is the main process that communicates with the VSS interface; VSS coordinates requests by passing them to writers and providers as necessary. The requestor also communicates directly with writers to gather backup components, files, and metadata managed by the writers. This allows a requestor to select the volumes that should be shadow-copied to complete the requirements of the backup operation. With the advent of VSS, I look to the day when all Windows backup solutions are based on VSS—otherwise, they will find it increasingly difficult to be supported by Microsoft.

## 5.4.3 Exchange Server 2003 support for VSS

Based on the earlier discussion, it should be clear that, to support the VSS framework, an application like Exchange Server must provide the VSS

writer component. For previous versions of Exchange (v4.0 — 2000), Microsoft has not and will not provide a writer and, therefore, does not support VSS for these versions. However, the Exchange Server 2003 release does provide VSS support for Exchange information store backup and recovery. In Exchange Server 2003, Microsoft has built the VSS writer functionality into the information store (STORE.EXE) process of Exchange Server. This writer will provide the necessary support for VSS requestors to initiate backup operations for Exchange Server 2003.

### *Exchange Server 2003 backups using VSS*

Traditional Exchange API-based backups focused on four backup types for Exchange databases: full, incremental, differential, and copy. However, the Exchange 2003 VSS writer supports only a full backup at the storage group (SG) level. VSS performs Exchange Full backups at the SG level, even though the Exchange writer treats individual databases as separate components. VSS uses the *AddComponent* call to add each database component to the shadow copy set, which in the case of a Full backup, is the entire SG (i.e., databases or log files). In a Full backup of an SG, VSS creates a complete shadow copy of all volumes that contain Exchange data—the shadow copy contains database and transaction log files associated with that SG. In addition, as is the case with non-VSS full backups, VSS truncates the transaction log files after successfully creating and backing up the shadow copy. To truncate the transaction log files, the shadow copy set must include all databases. For this reason, Microsoft will use the metadata definition for the Exchange writer to force the requestor applications to process only full backups that have all SG components (i.e., databases or log files) in the shadow copy set.

***Exchange VSS full backup***—VSS backups for Exchange will be at the SG level. This is the case even though individual databases are treated as separate components. Each database VSS component is added (with the *AddComponent* call) to the shadow copy set (which, in the case of a full backup, is the entire storage group—databases and log files). In a full backup of a storage group, a complete shadow copy is created of all the volumes (containing database and transaction log files) associated with that storage group. In addition, as is the case with non-VSS full backups, the transaction log files will be truncated after successful shadow copy creation and backup. In order for the transaction log files to be truncated, all databases must be included in the shadow copy set. For this reason, Microsoft will force (via the metadata definition for the Exchange writer) requestor applications to only process full backups that have all storage group components (databases and logs) included in the shadow copy set.

### *Exchange Server 2003 recovery using VSS*

Although VSS backup for Exchange 2003 is at the SG level, you can recover individual databases from the SG snapshot set—each shadow copy has the individual database files, plus the logs needed to reconstitute them. VSS-based restoration of an Exchange 2003 SG is useful when data in one or more databases in the SG is lost or corrupted, but the current log files remain intact on disk; when the current log files on disk are lost or corrupted, but the databases remain intact; or when databases and current log files within an SG are lost or corrupted.

In the context of Exchange 2003 and VSS, only the backup application is responsible for restoring data to disk. The Exchange 2003 database engine, not the requestor, is responsible for recovering the data to a consistent, up-to-date state through playback of the log file. To do so, the database engine activates existing soft- or hard-recovery procedures after the VSS-aware backup application restores the transaction log files and databases. Once the restore is complete, Exchange 2003 remounts and restarts the SG, and then the database engine initiates recovery. The database engine determines that the state of the databases isn't consistent with the end of the log file on disk and begins the recovery procedure.

Three Exchange 2003 data-restoration scenarios exist, but only two procedures for those scenarios exist. The roll-forward recovery and point-in-time recovery procedures for restoring data are the same whether you have lost only the SG's log files or you have lost an SG's log files and databases. You use the same procedure because the loss of the log files is a catastrophic failure in Exchange and requires restoring the entire SG. In either case, these recovery options follow a specific step-by-step process:

1. The affected storage group is taken off-line.
2. VSS-based recovery is initiated. This includes all of the volumes contained in the SG shadow copy set.
   - If one LUN is configured per SG, Exchange recovers all databases except those that are intact.
   - If multiple LUNs per SG are configured, Exchange recovers only the LUNs with the databases needing recovery from the Shadow Copy set.
3. Exchange performs an Extensible Storage Engine (ESE) hard recovery and replays applicable log files for databases being recov-

ered, depending on whether a roll-forward recovery or point-in-time recovery is occurring.

4. The storage group is remounted and resumes normal on-line operations.

***Roll-forward recovery*—**In a roll-forward recovery, one or more databases in the SG are lost, but the log files are intact on the server at the time of the recovery. In this case, you can selectively restore each of the affected databases from a full backup of the SG. Within the context of the VSS framework, you select from the SG backup only those database components that correspond with the databases you want to restore. The VSS-aware backup application restores the databases, and then Exchange recovers the databases and brings them up to date from their state at the time of the snapshot by rolling forward through the transaction logs (Exchange hard recovery). The roll-forward recovery option lets you recover backed up data as well as data that has accumulated (e.g., in transaction logs) since the last backup.

***Point-in-time recovery*—**When the SG's log-file volume has been damaged or lost or the log files have been lost or damaged together with some or all of the SG's databases, you must restore the log files from a previous backup, together with all the databases backed up at the time of the last full backup of the SG. You cannot recover to the point of the failure because the log files and databases since the last backup have been lost or damaged, so you can recover only to the point of the last full backup. This process is known as a point-in-time recovery. Because this option does not provide roll-forward capability, some data will be lost (e.g., data between the point in time of the recovery and the time of the failure). To provide point-in-time recovery, you must restore the databases that you backed up at the time of the full backup as well as the log files from the full backup. In addition, you must recover all databases associated with the SG. You cannot assume that any of the databases were left in a transaction-consistent state at the time the log files were lost and went off-line, because the loss of the transaction log is a fatal error that causes the store to shut down immediately with no guarantee of consistency. Therefore, to ensure that the databases are in a consistent state when you restart the SG, you must return the entire SG to its state at the time of the last Full backup.

### *Implications for Exchange administrators*

As organizations move from previous versions of Windows (NT4 and Win2K) and Exchange (Exchange 4.0 — 2000) to Windows Server 2003 and Exchange Server 2003, the use of VSS-based backup and recovery will become a standard mechanism for Exchange disaster recovery. However, I must concede that VSS solutions are not yet proven or readily available. The non-VSS solutions that exist today allow snapshot and clone technologies to be utilized with Exchange Server. However, these technologies have no native operating system or application support. As a result, support from Microsoft for these solutions is limited [see Table 5.2 for a listing of pertinent Microsoft Product Support Services (PSS) Knowledge Base (Q) articles]. Organizations must therefore rely on the vendors of these solutions for support—both for current non-VSS solutions and for future support and adoption of VSS solutions. At the time of this writing, third-party vendor support for VSS is a bit unknown (for both VSS providers and requestors) but vendors such as EMC, HP, Hitachi, Veritas, and CommVault are leading the way with VSS-enabled solutions released or in beta. Yet it will only be through extensive testing and deployment of these solutions that we will understand the power and utility they bring us. Until Exchange Server 2003 is widely deployed on Windows Server 2003 and vendors have done their part to embrace Windows VSS, the jury is still out as to whether this technology will really help us make our Exchange servers more available. However, as you may imagine, the potential here is huge.

**Table 5.2** *Microsoft Knowledge Base Articles Discussion Snap/Clone Support for Previous Versions of Exchange Server*

| Microsoft Knowledge Base "Q" Article | Subject |
|---|---|
| Q237767 | XADM: Understanding Offline and Snapshot Backups |
| Q311898 | XADM: Hot Split Snapshot Backups of Exchange |
| Q296787 | XADM: Off-line Backup and Restore Procedures for Exchange Server 4.0, 5.0, and 5.5 |
| Q296788 | XADM: Off-line Backup and Restore Procedures for Exchange 2000 Server |

### 5.4.4 Exchange recovery servers

One of the most challenging aspects of an Exchange administrator's job is disaster recovery for Exchange. You must be able to provide recovery for every scenario, including mailbox and message recovery, information store recovery, and complete Exchange server recovery. Providing individual brick-level mailbox and message recovery can be the most challenging of these scenarios. Many third-party solutions exist for mailbox or message-level recovery, but most are far from perfect. In addition, Microsoft does not really provide a solution either—instead leaving this gap open to the third-party developers. However, there is a solution for this problem scenario that should be your best practice regardless of which version of Exchange Server you are running. This solution is called the Exchange recovery server and it can be a very useful tool for every Exchange administrator.

The idea behind the Exchange recovery server is that you maintain a spare server in your environment that is available as a target location to perform recovery operations. Regardless of which version of Exchange you run, you can recover an information store from one server to another server. Once this information store is recovered to the recovery server, you can then perform recovery for complete mailboxes or just individual messages by extracting these items from the store using Outlook or programs like ExMerge. Of course, Exchange 5.5 provides recovery and retention for deleted items, and Exchange 2000 adds mailbox retention to this. However, since even these two mechanisms might not meet all of your disaster recovery requirements, it is nice to know that the recovery server option is available. Figure 5.9 illustrates the idea of how a recovery server is used to provide mailbox and individual item recovery for an Exchange deployment.

#### *Exchange 5.5 recovery servers*

The recovery server capability is available for any version of Exchange. However, the configuration and setup of your recovery server will vary depending on whether you are using Exchange 5.5 (and earlier versions) or Exchange 2000. Let us start our discussion with Exchange 5.5 and earlier versions since this is the version that the majority of you are using. With Exchange 5.5 and earlier versions of Exchange, the recovery server can be any server (member server of domain controller) in the same domain as the original server you are recovering. However, the recovery server must have a different name from that of the server being recovered (you don't want this server to start participating in the Exchange organization and doing directory replication, and so forth). You configure this recovery server by installing it with the same organization and site naming

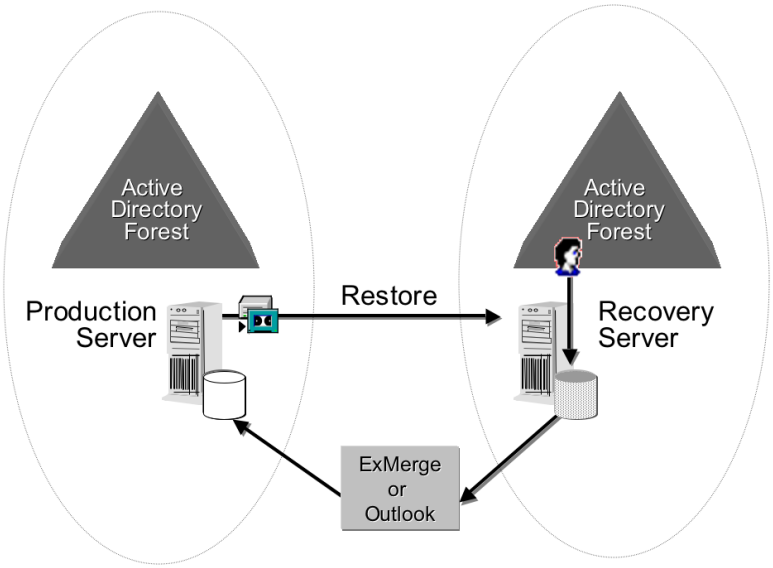


**Figure 5.9** *Exchange 2000 recovery server scenario.*

conventions and hierarchy as the original server (but a different server name) and organization but you do not join this server to the production organization. The result will be a server with the different name, but the same site and organization name as the original. This might seem confusing, but the server will not interfere with your existing Exchange organization because it was not specifically joined to it during installation. This allows the server to function in the environment without causing problems for the production Exchange 5.5 deployment (you can even use the same Exchange Service account).

Once the recovery server is installed and properly configured, you can restore an information store to the server. However, the directory database on this server will not have any objects (remember, you did not join this server to the existing organization). You can easily remedy this by either manually creating mailbox objects in the directory that match the ones you want to recover [the Distinguished Name (DN) is the only attribute that must match]. Alternately, you can run the Exchange 5.5 DS/IS Consistency Adjuster (in the Exchange Administrator program) to automatically generate mailbox objects in the directory for each one found in the information store. These procedures will link information store mailboxes to directory objects on the recovery server. From this point, you can simply install the Outlook client on the server or use tools such as ExMerge to extract mailbox data to personal store files (PSTs) and import the data back to the production Exchange server into the appropriate mailboxes.

### *Exchange 2000 recovery servers*

Deploying a recovery server for Exchange 2000 operates on the same principles as earlier versions. However, several things are different in Exchange 2000. The first issue has to do with Exchange 2000's dependence on the Windows AD. Since you can have only one Exchange organization per AD forest (as things are today—hopefully, someday this will not be the case), we are forced to deploy a completely separate AD forest for our Exchange recovery server(s). This should not be that big a deal since the recovery forest can exist right along side our production forest, and it is a good idea to have a test environment available anyway. However, you will need to determine the administrative impact this has on your organization. The other key differences in Exchange 2000 involve the change from sites (in previous versions of Exchange Server) to administrative groups (in Exchange 2000) and the ability to have more than one information store per server. This adds additional steps to the configuration of an Exchange 2000 recovery server, which I will discuss next.

The first step in deploying and Exchange 2000 recovery server is to deploy your recovery forest. You must have a separate forest when installing your Exchange recovery server or you will be forced to join the existing production Exchange organization. The recovery forest can have any naming convention and need not match naming conventions in the production forest (it can even exist on the same network) as shown in Figure 5.9. Once you have your Exchange 2000 recovery forest deployed, you can install your Exchange 2000 recovery server into the forest with the same organization name as the production server you will be restoring (remember, this is the Exchange organization naming—not the AD naming). The server name can be the same or different as there will be no conflicts with the production server (other than potential DNS issues if it is on the same network). If you are going to keep the recovery forest up and running permanently, I recommend that you install a permanent recovery server into the recovery forest that maintains a permanent name and is the first server in the organization. I also recommend that the naming conventions and administrative group hierarchy match your production Exchange 2000 organization. This will be of huge benefit in the steps I discuss next. When you have a permanent server installed that maintains organization naming and hierarchy, a second recovery server can be installed for each incident to match the naming of the server being recovered. This will alleviate issues with LegacyExchangeDN discussed next.

After the recovery forest and server(s) have been configured, you can perform a restore of the information store database into an administrative

group with the same name as the one where the database was taken from. However, from a database and AD point of view, the LegacyExchangeDN values for the administrative group and the database must match. In addition, the storage group and database names must also match those on the original server. If you have taken steps as outlined earlier to maintain a permanent recovery server in the recovery forest (that matches the naming and hierarchy of the production organization) and have installed a second recovery server to the same administrative group, storage group, and database as the production server, these values will match. However, in the event that you do not wish to maintain this extra recovery configuration, the LegacyExchangeDN value can be updated manually using an LDAP editing utility such as LDP, ADSI Edit (Windows 2000 Support Tools), or LDIFDE (installed by default in Windows 2000) to view and edit this value for the database and administrative group. In addition, Microsoft provides an unsupported tool called LegacyDN.EXE (available with Exchange 2000 SP1 or later) that provides an easy-to-use interface for changing this attribute (note really made for this purpose—which is why it is not supported). Regardless of which tool you choose, the Exchange 2000 organization name, administrative group name, storage group name, database name, and LegacyExchangeDN values for the production environment and the recovery server must all match for the database being restored. For more information on the procedure to modify LegacyExchangeDN values, see Appendix A: Changing the LegacyExchangeDN Attribute Values in the white paper *Exchange Database Recovery* at www.microsoft.com/technet/prodtechnol/exchange/support/dbrecovr.asp.

Once you have restored the information store database to the recovery server, you can proceed to link mailbox objects to mailboxes similar to the Exchange 5.5 recovery scenario. However, for Exchange 2000 and AD, the procedure is different. In earlier versions of Exchange, the mailbox object is linked to a DN and only directory objects with the same DN are required to link mailboxes to directory objects. In Exchange 2000, this is not the case, and you must explicitly connect a mailbox in the database to a directory object. You can do this manually if you are recovering a small number of mailboxes by creating a user object that is not mailbox-enabled (using the AD Users and Computers MMC Snap-in). After creating a user object, it is a good idea to run the mailbox cleanup agent from the Exchange System Manager. Afterward, you should be able to see mailboxes in the restored database that have a red "X" indicating they are orphaned, or not connected to any user object. From this point, you merely right-click on the mailbox object, choose reconnect, and select the user object that you wish to connect to the mailbox. Of course, if you are recovering many

mailboxes, you might want to use something like the Mailbox Reconnect Tool (MBConn—available on the Exchange 2000 CD) to save you some keystrokes and mouse clicks. Once mailboxes are connected to user objects, you can extract data from them in the same manner as we discussed in the Exchange 5.5 recovery server scenario (using Outlook, ExMerge, and so forth).

We have taken a closer look at the concept of a recovery server for your Exchange environment. As an Exchange administrator, you should take this concept to heart and implement Exchange recovery servers as a best practice in your environment—regardless of which version of Exchange you have. In addition to being an important recovery facility for your Exchange environment, the recovery server scenario also provides a great nonproduction testbed that is useful in other situations. If you are not using Exchange recovery servers as part of your everyday life as an Exchange administrator or encouraging your customers to use this best practice, you might consider the benefits this solution can bring to you and your customers. The recovery server concept may seem like extra trouble. However, if you desire to provide this level of disaster recovery for your Exchange users, the recovery server concept can be a lifesaver.

### 5.4.5 Leveraging Exchange Server 2003's Recovery Storage Group feature

These steps in utilizing Exchange recovery servers may seem a bit cumbersome. That's because they are! Microsoft has done a great job of documenting this procedure for recovering mailboxes and data, but it is very complex and somewhat prone to error. In fact, Microsoft Product Support Services (PSS) spend far too many cycles support Exchange recovery server methods. In response, the Exchange development team has heard the cries of Exchange administrators (and PSS support engineers …) and has devised a method for providing this functionality in a much simpler and easier to deploy mechanism. It is my pleasure to introduce you to the Exchange Server 2003 Recovery Storage Group (RSG) (Figure 5.10)!

Bringing greater flexibility to the recovery of databases, mailboxes, and individual items, the Exchange Server 2003 RSG is a powerful new feature. The RSG is a special-purpose fifth storage group available on Exchange Server 2003 servers that exists alongside your production storage groups on the server. This means that even though a server is configured with four production storage groups, you can still add an RSG to the server. You can then use this RSG to recover databases from any Exchange Server 2000

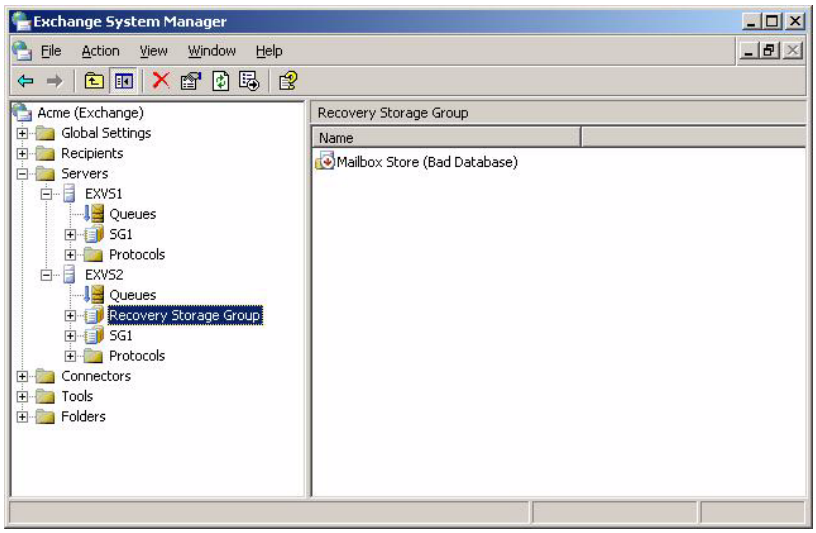


**Figure 5.10** *Exchange Server 2003's Recovery Storage Group feature.*

SP3 and later server that is in the same Administrative Group as the server with the RSG. After you have recovered a database to the RSG, the use and procedure for recovering mailbox data is much the same as the Exchange 2000 recovery server scenario. You can use tools such as ExMerge to move data from the RSG to production storage groups. This allows you to recover an entire database or just a single mailbox. The RSG does have some caveats, however. You can only have one RSG per server, and the RSG does take overhead on the server. In addition, if you want to perform concurrent recovery operations on the server and have the maximum number (four) of storage groups and an RSG configured, you will not be able to do so (since a maximum of five ESE instances are possible per Exchange 2000/2003 server, there are no available instances to perform concurrent operations). Finally, RSGs only support the recovery of mailbox stores—not public folder stores (see Table 5.3 for more information). RSGs are created the same way you create regular storage groups by selecting the server and right-clicking and choosing New … Recovery Storage Group. (Yes, it is that simple!)

The concept of an RSG is new to Exchange Server 2003 and promises to save Exchange administrators and Microsoft PSS hours of work. By avoiding the requirement to set up an entire recovery forest and deploy extra hardware and software (which, by the way, Microsoft required you to pay for …), RSG offer huge wins. However, time will attest to the success of the RSG and whether the efforts the Exchange development team were well invested. If you have ever struggled or been frustrated with the old way

**Table 5.3** *Exchange 2003 Recovery Storage Group Usage Scenarios*

| Scenario | Usage Description |
|---|---|
| Database/mailbox/item recovery | Useful for recovery of lost or deleted data from user mailbox. A database from the same Administrative Group as the RSG can be recovered and data can be extracted via tools such as ExMerge. |
| Rapid recovery | In the event of a catastrophic loss of a mailbox database, a "stub" mailbox database can be created and the logs copied to alternate location.<br>The RSG can then be used to recover the database to the point of failure while the stub database allows continued service for users.<br>Once the database has been recovered, it can be swapped with the stub database and new data from the stub database can be recovered via the RSG to the original production database. |

of doing things (Recovery Servers), Exchange Server 2003's Recovery Storage Group is a welcome relief.

### *Power with responsibility*

With the advent of multiple storage groups and databases in Exchange 2000/2003, the recovery API was stretched to accommodate new scenarios. You must be able to perform backup and restore operations for the entire server, a storage group, or an individual database. In addition, since these operations can be performed concurrently, ESE must be able to handle this as well. Exchange 2000/2003 offers a great deal of flexibility and additional availability that previous versions did not offer. For example, you could have four databases configured that each host 1,000 users (a total of 4,000 users). You begin restore operations for one storage group or database without impacting the other storage groups. In our example, 3,000 users would be on-line accessing their data while the 1,000 users using the database being restored would be the only affected users. This, indeed, gives operators many more options and reduces the overall impact of restore operations. However, it also complicates procedures, requires better training, and has greater potential for error. Gather the knowledge you require to ensure that you are implementing solid disaster-recovery plans for Exchange serv-

ers. Understand the different backup strategies available for Exchange 2000/2003, and select the one that best suits your organization. Also, be sure to investigate how you can leverage Exchange Server 2003's RSG feature. Finally, keep an eye on Exchange 2003's support for Windows VSS and the result vendor solutions that become available. Also, stay tuned because many of the best practices and tricks of the trade for Exchange 2003 have not been discovered yet (although things aren't that different from Exchange 2000). The power of Exchange 2000/2003 storage must not be realized without properly understanding the disaster-recovery implications of your storage design choices.

## 5.5 Additional resources and information

Microsoft Exchange 2000 Disaster Recovery White Paper

```
http://www.microsoft.com/technet/prodtechnol/exchange
/support/e2krecov.asp
```

Microsoft Exchange Server 2000 Support Center

```
http://support.microsoft.com/default.aspx?scid=fh;
EN-US;exch2k
```

Microsoft Active Directory Forest Recovery

```
http://download.microsoft.com/download/win2000srv
/Utility/1.001/NT5/EN-US/forestrecovery.exe
```

Backup and Restoring Connectors on Microsoft Exchange 2000 Server

```
http://go.microsoft.com/fwlink/?LinkId=6272
```

Exchange 2000 Database Recovery

```
http://www.microsoft.com/technet/prodtechnol/exchange
/support/dbrecovr.asp
```

Mailbox Recovery for Microsoft Exchange 2000 Server

```
http://www.microsoft.com/technet/prodtechnol/exchange
/support/mailbox.asp
```